A semantic-based access control mechanism for distributed systems

Mersedeh Sadeghi
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Milano, Italy
mersedeh.sadeghi@polimi.it

Luca Sartor
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Milano, Italy
luca.sartor@mail.polimi.it

Matteo Rossi
Dipartimento di Meccanica
Politecnico di Milano
Milan, Italy
matteo.rossi@polimi.it

ABSTRACT

Access control management in a collaborative environment composed of a multitude of distributed autonomous organizations is a challenging task. To answer the challenge, in this paper we propose a novel approach that incorporates semantic technologies in the Attribute-Based Access Control (ABAC) approach. Building on the basic principles of ABAC, our approach allows for a highly expressive modeling of the context in which access decisions are made, by providing mechanisms to describe rich relationships among entities, which can evolve over time. In addition, our system works in a truly decentralized manner, which makes it suitable for geographically distributed enterprise systems. We show the feasibility in practice of our approach through some experimental results.

KEYWORDS

Semantic-Based Access Control, Attribute-Based Access Control, Distributed Access Control, Distributed Reasoning, Context-aware Access Control

ACM Reference Format:
Mersedeh Sadeghi, Luca Sartor, and Matteo Rossi. 2021. A semantic-based access control mechanism, for distributed systems. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3412841.3442058

1 INTRODUCTION

Large distributed enterprise systems allow companies to consume and offer products, and in particular data and services, through a shared ecosystem, which in turn fosters the creation of new products from the combination of existing ones. This occurs, for example, in the transport sector, where end-to-end travel solutions are built from basic transport services to provide so-called Mobility as a Service (MaaS). These systems are composed of large numbers of independent public and private enterprises that need to share both information and services among them in a carefully controlled manner, to protect their assets and business interests. The sharing occurs in a highly competitive environment in which companies fight for market shares, but they also have complicated business relationships among them that create various conflicts of interest for involved actors. This heightens the security risks, and in particular the need for carefully designed access control policies.

In such an environment, though, Access Control Management poses particularly difficult challenges, since it goes beyond the IT and software engineering aspects of constructing a safe and secure collaborative system, and it is tightly linked to complicated relations and associations involving real organizations (and persons), their business preferences, and their general policies and regulations.

Yet, traditional access control mechanisms often tend to reduce the complexity of the protection strategy to increase the efficiency and intuitiveness of the mechanism. For example the highly popular Role-Based Access Control (RBAC) [24] is built upon a rather modest protection model: a user is associated with a role and a role is associated with a set of permissions, each of which grants a set of operations on a resource. These models of access right designation are very efficient and easy to implement, but they fail to capture, on one side the real-world relations among actors and assets in a large system composed of many independent and distributed agents and, on the other side, their complex, dynamic and often ambiguous protection strategies over their resources.

To address this issue, a higher expressiveness in access control modeling is required; firstly, to enable a more comprehensive description of users and resources and, secondly, to inject additional contextual information in the permission rules. In addition, given the diversity of actors, resources, and protection motivations and strategies in a collaborative environment, an access control model must be flexible enough so organizations can tailor and adapt the required contextual information based on their needs, business models and real-world regulations and agreements. An ontology-based approach, with its associated suite of semantic technologies, seems well suited to tackle the issues concerning the modelling of the wider context in which access control occurs in distributed enterprise systems. It leverages the expressiveness of the model and enables automated inference over data stored in the knowledge base to extract the hidden and implicit relations existing among entities. As a result, the access control system can capture the complex relations among real-world entities with greater precision.

Another major limitation of traditional access control schemas is their centralized nature. Almost all of the most common access models developed in the last two decades assume—implicitly or explicitly—a centralized authorization unit that is in charge of collecting, assessing and asserting the protection policies. Clearly, such

SAC '21, March 22–26, 2021, Virtual Event, Republic of Korea
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8104-8/21/03…$15.00
https://doi.org/10.1145/3412841.3442058

a paradigm does not suit large, collaborative distributed systems. The creation of a fully-distributed authorization mechanism is even more challenging when it comes to incorporating semantic technologies in an access control system. Indeed, the current state of the art in using semantic inference to assess access requests or to derive new knowledge has been entirely relying on reasoning engines that assume the existence of a complete knowledge graph.

To address these issues, this paper presents a new, semantically rich, distributed access control mechanism. The solution proposed in this paper comprises the following parts: (i) An **ontology** to provide an expressive description of users, resources and their relations in an enterprise ecosystem. (ii) A **semantic-based access control mechanism** for systems of systems and collaborative multi-agent environments that enriches the Attribute-Based Access Control (ABAC) paradigm. (iii) A **distributed mechanism for policy assertion, semantic inference, and access right determination** that removes the need for any centralized authorization unit.

The proposed access control system has been developed as an authorization mechanism to be integrated into the Interoperability Framework [23] that is an integral part of the Shift2Rail Innovation Programme 4 (IP4) [1]. The framework aims to enable a collaborative ecosystem of transportation actors (transport operators, service providers, retailers, distributors, public authorities, third-party software developers, etc.) across the European Transport Area, which allows them to offer, share, discover and use a wide range of mobility data and services in an interoperable manner.

The rest of this paper is structured as follows. Section 2 discusses related works. Section 3 briefly overviews the Interoperability Framework. Section 4 presents the proposed access mechanism in detail and Section 5 outlines the prototyped implementation and evaluation. Section 6 concludes the paper.

2 RELATED WORKS

Ever since their emergence, semantic web technologies have been applied to the access control domain. Many solutions have tried to incorporate semantic elements in various aspects of classic access control models. For instance, [22] proposed the concept of semantic authorization that allows semantically equivalent Roles to access semantically equivalent Objects. It is achieved via a semantic mediator that maps to one another the access request formats, role hierarchies and object descriptions defined in different organizations. Unlike our work, [22] and similar approaches [11, 12, 29], have targeted RBAC as the base access model. Hence, the fundamental limitations of RBAC, which often impose crucial constraints on complex distributed systems stemming from the static nature of role-right assignment and role-explosion, remained unsolved.

Attribute-Based Access Control (ABAC) was introduced to addresses the rigidity and limitations imposed by previous access control systems such as RBAC. ABAC generalizes RBAC (since the concept of Role can be modelled as an attribute of a subject), it has higher expressiveness than the latter, and it can eliminate the problem of role explosion that typically occurs in complex systems with a wide variety of users. As showed in [17], ABAC requires up to 2N rules for N attributes to achieve the same level of control obtained in RBAC through the introduction of 2N roles for each possible combination of conditions. Furthermore, ABAC offers, with respect to RBAC, more dynamism and context-awareness, which are essential for an access control schema targeting complex systems. Similarly to our work, [32] and others (see., e.g., [1, 9]) proposed an implementation of ABAC for distributed environments. However, they generally focused on the basic ABAC model and pursued only minimal improvements over it. For example, none of them used semantic technologies to increase the range of expressible policies, nor they tackled challenges such as the assignment of homogeneous and interoperable attributes in a distributed system.

ABAC hinges on the properties of involved actors and, in addition, the attributes of entities are highly domain-dependent. As a result, there are few generic ABAC frameworks. Rather, many works defined ABAC models—some of them quite feature-rich—tailored for a particular application area, which are not applicable to other domains. For instance, [3, 6, 10, 16, 19, 30] extend the ABAC model in different aspects, focusing on cloud computing, smart spaces and grid computing. However, few contributions have developed an ABAC model for geographically- and administratively-distributed enterprise systems. One of the motivations behind this work was to fill this gap and propose an extended ABAC model that addresses some of the original limitations of ABAC (see Section 4) and which specifically tackles the requirements of this application domain.

Some works, like ours, based access control on semantic technologies. Ontology-Based Access Control (OBAC) has mainly been used in Social Network Systems. [7] proposed an ontology, based on the Friend-of-a-Friend vocabulary [5], to represent the various entities in social networks, and it employed SWRL rules to state permissions. The framework presented in [20] further expanded the work of [7] through the provision of more detailed semantic models and complex authorization mechanisms such as delegated authorization. [26] shares some similarities with our work: it combined ABAC with semantic technologies by using a knowledge graph as attribute supplier; furthermore, it used semantic inference to elaborate on the existing set of attributes and to make implicit data explicit through the help of reasoning engines.

Our work differs from the above contributions in various aspects. Firstly, none of them targets collaborative distributed systems. Secondly, in many cases the role played by reasoning is different than in our work. For example, in OBAC [7, 20] and in the Semantic Access Control (SAC) model [31] inference is used to determine access rules, whereas in our case it is used to derive implicit relations among organizations (see Section 4 for more details). Above all, the most important shortcoming of these–and other—works in the area of semantic-based access control mechanisms [8, 11, 15, 27, 29] is their strict requirement to have a centralized and complete knowledge graph in order to perform reasoning. In this direction, one of the main contributions of our research is the realization of a distributed inference mechanism that lets distributed nodes across the system keep only the portion of the ontology pertinent to them.

Finally, ORDL[2] is a policy expression language that provides an information model for the fine-grained description of policy, permission and prohibition. Our framework currently operates at the logical-based policy expression level, which is a common

[1]shift2rail.org/research-development/ip4

[2]https://www.w3.org/TR/odrl-model

approach in ABAC. The extension of the framework to include the ORDL policy model could be an interesting future work.

3 INTEROPERABILITY FRAMEWORK

The Shift2Rail Interoperability Framework (IF) is a solution designed to leverage the digitalization and servitization in the transportation domain that targets both sides of the system: the consumers (e.g., mobility applications) and the producers (e.g., transport service providers). From the service producer perspective, the IF provides a shared distributed environment that enables the creation of integrated mobility solutions through the publishing, discovery, sharing and use of other parties' assets and products in a collaborative manner. In addition, it offers various services and utilities fostering the automation of processes (e.g., conversion mechanisms) that exploit semantic-based technologies to make heterogeneous services interoperable. From the travel application/consumer perspective, it shields the complexity and distribution of travel services and ICT products by: (i) offering a single logical point of interaction, (ii) publishing a homogeneous abstraction of diverse services provided by a wide range of providers across Europe, and (iii) allowing applications to interact with such services uniformly.

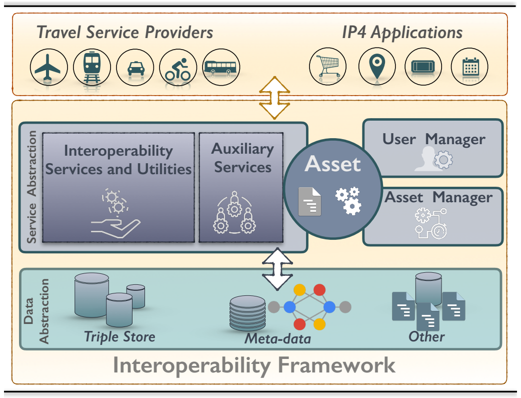

Figure 1: IF Architecture.

The IF is itself a distributed system structured around IF nodes. Figure 1 shows the high-level architecture of a single IF node, which is responsible for handling the users and assets of a specific region. The distribution and granularity of regions could vary from "one IF node per EU country" (which occurs if each IF node acts as National Access Point[3]), to "one IF node per district", to the most fine-grained scenario of "one IF node per major transport operator" (e.g., SNCF). A detailed description of the architecture of the IF is beyond the scope of this paper. Nevertheless, in the following we provide an overview of its main components and entities. In brief, the IF comprises a core component called Asset Manager (AM), a User Manager, two logical layers named Data and Service Abstraction, and two types of entities, User and Asset.

The AM plays an essential role in the IF, since it is the entry point to the framework and it offers the basic functions for interacting with the IF, including those for publishing, discovering and maintaining various kinds of artifacts in the IF. The Data Abstraction mainly encompasses those aspects of the IF that deal with the knowledge bases that are necessary to manage operations such as collection and retrieval of transportation data, ontologies, vocabularies and meta-data provided by different parties. Following a modular and service-oriented approach, the IF includes a Service Abstraction that manages all features related to the publishing, advertising, discovery, interaction with and deployment of a set of self-contained and reusable services.

The User Manager is the component responsible for the operations related to User entities, such as user registration and profile management. Furthermore, it is responsible for the storing of user credentials and for the user authentication process (which is hence outside of the scope of our Access Control system). In principle, a user (i.e., the organization that the user belongs to, since user and organization are equivalent concepts within the IF) of the IF could register themselves as a producer or as a consumer, where only the former can publish and own Assets.

Finally, the notion of Asset is key in the IF: it refers to any resource that a generic actor of the transportation domain may be interested in—to read, share and use. More specifically, an Asset is an artifact that has a unique identifier throughout the IF ecosystem, a description, a definable life-cycle, an owner, and it is discoverable by users and by other Assets. Most importantly, Assets must be protected. In other words, the main role of the Access Control system in the IF is to manage the access of users to Assets.

4 SEMANTIC-AWARE ABAC

In the ABAC model, the values of attributes associated with a user determine the user privileges [13]. The core idea is to evaluate access rights based on the relevant attributes of the requester (*Subject*), the requested resource (*Object*) and any other pertinent conditions (*Environment*). The owner of the *Object* ascertains the relevancy of attributes and conditions by specifying them in a *Policy* script.

In ABAC, access decisions are based on the context of the use and attributes of all involved entities at the time of the access request. Accordingly, permissions in ABAC are not a set of static and predefined privileges allotted to a Subject, but they dynamically evolve as/if the properties of the subject, object and environment change. This is achieved, however, without the owner of the resource needing to monitor the fluctuations in the entities' properties, or even to be aware of them. The owner defines, once and for all, in the policy (which is usually expressed through predicate logic formulae [4]) the set of conditions under which access is granted. It is then the responsibility of the AC system to retrieve and examine the attributes mentioned in the policy to evaluate access rights.

The high flexibility and dynamism of the ABAC model make it suitable for access control management in complex, distributed and multi-agent applications. Indeed, the expressiveness of ABAC is limited only by the expressiveness of the computational language and by the richness of the available attributes [14]. The higher flexibility, however, results in higher complexity of attribute and policy specifications. In particular, the process of defining the relevant attributes and the retrieval of their values for policy evaluation are some of the most difficult parts to realize in ABAC implementations [25]. The latter is especially challenging in distributed systems.

[3]https://www.cestrin.ro/web2014/nap_eueip/

In this direction, we propose to incorporate semantic technologies in ABAC. Firstly, we use an ontology to model the attributes of involved elements. Ontologies are a powerful mean to formalize the contextual information in an area of concern [2]. Accordingly, through an ontology we can model complex enterprise actors, resources and their relations more accurately and comprehensively. Secondly, the formalization of the concepts allows the system to infer new knowledge, automatically derive new (hidden) relationships and expand the knowledge graph, which in turn improves the completeness and precision of the access control mechanism. Finally, in a semantic-based approach attribute retrieval can be done by querying a homogeneous (possibly distributed) knowledge graph, rather than by collecting data from heterogeneous databases.

4.1 Reference Architecture

Although ABAC has been around for several decades [18, 21], there is still no standardized framework for it, or an agreement on its general architecture and components. Among others, the eXtensible Access Control Markup Language (XACML, [28]) has attracted a certain interest. XACML is an XML-based policy language for describing the authorization process and the format of access requests/responses. It introduces a generic authorization schema that is not designed for a particular access control method, but that has been adopted as the core ABAC model by the research community [13, 32]. Hence, our proposed architecture borrows from and extends the XAMCL architecture to include semantic technology elements. Figure 2 shows the main components of the proposed architecture and the flow of information among them.

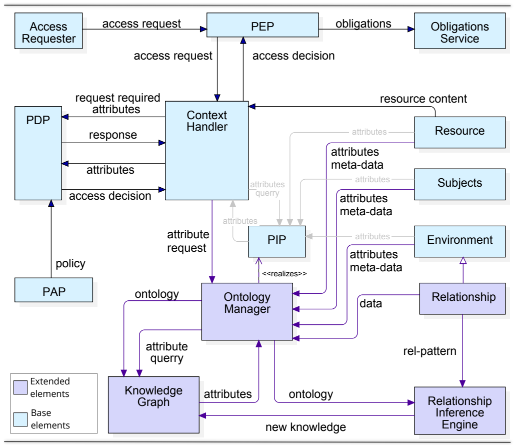

Figure 2: Extended ABAC architecture to incorporate semantic technology elements.

As shown in the figure, the `Policy Administration Point` (`PAP`) provides an interface for the system administrator to manage the policy repository and add, modify or delete policies. The `Policy Enforcement Point (PEP)` receives the request from the *requester*, delivers the access decision (grant/deny) concerning the requested *resource*, and handles any obligations or recommendations that are specified in the policy. The `Policy Decision Point` (`PDP`) determines the final access decision based on the policies retrieved from the repository and the elements provided by the `Policy Information Point (PIP)`, and it resolves any conflicts in policies. In core XACML, the `PIP` retrieves the values of the attributes based on the requests received by the `PDP` by accessing the attribute repository and the systems that provide the *environment* conditions. Finally, the `CH` is an optional component that improves the flexibility of ABAC. It brings the attributes retrieved by the `PIP` in a format compatible with the policy specifications; hence, it decouples the attribute authority from the actual representation of attributes in the policy specifications.

The `CH` allows us to use an ontology to represent attributes and store them as a knowledge graph. To this end, we introduce the `Ontology Manager` component, which is a realization of the `PIP`. The `Ontology Manager` builds and keeps the ontology updated by (i) collecting the attributes' meta-data of the *object*, *subject* and *environment* and adding the necessary individuals, and (ii) adding new information to the knowledge base through the initialization of the relationship inference engine. Finally, the concept of `Relationship` (among enterprise organizations), which is defined through the `Relation Pattern (Rel-Pattern)`, specializes the `Environment` and provides extra information to make access decisions. The next section provides more details on the concept of `Relationship` and its role in our access control mechanism.

4.2 Access Control Ontology

The proposed Access Control ontology of Figure 3 describes the Assets and the Users/Organizations associated with an `IF` node, which are, respectively, the *objects* and the *subjects* in the ABAC terminology. Also, it models *environment* attributes through the concept of relationships among organizations in the real world, and it offers a generic and extensible approach to capture the complex inter-organization associations occurring in the enterprise system.

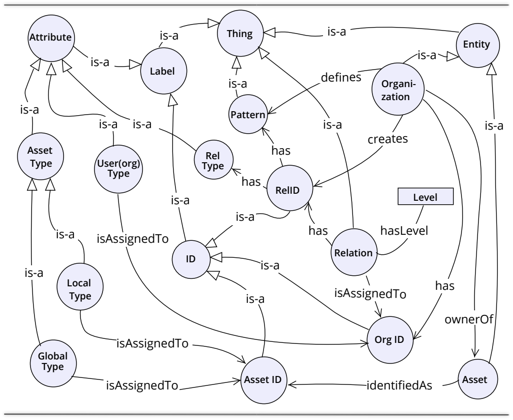

Figure 3: Access Control Ontology

More precisely, every element in the ontology is a member of one of four classes: `Entity`, `Label`, `Relationship` or `Pattern`. The individuals of the `Entity` class constitute the elements of the system, which are further partitioned into two classes: the `Organization` class, which represents `IF` users, and the `Asset` class, which captures the assets published through the `IF`.

The ontology introduces the `Label` class to describe the attributes that characterize `Assets` and `Organizations`. In particular, `Label` could be of type identification (i.e., class `ID`) or description (i.e., class `Attribute`). As the names indicate, the former is a type of attribute to uniquely identify an element throughout the system, while the latter is used to specify any other aspect characterizing an element. As in the basic ABAC model, the ontology includes *subject* and *object* attributes, which, in the context of the `IF`, correspond to features of Users/Organizations and of Assets, and which are captured through classes `User(org) Type` and `Asset Type` in Figure 3. These attributes are automatically derived from the information included in the Users/Organization profile and in the Asset description, which are provided by the users/asset owner when they register to the `IF`.Most ABAC models operate on a fixed and predefined set of attributes, which imposes a major constraint on the system, as users must identify and insert the value of those attributes at design time. This might lead system designers to include in the ABAC model a large set of attributes, to cover a greater number of contexts in which access rights might be evaluated. This, in turn, increases the burden on the system, and users might find many of those attributes irrelevant for their access policies. If, on the other hand, the defined set of attributes is too limited, this might compromise the precision of the access control system.

To address this limitation, our ontology enables the user to extend and customize access policies by defining their preferred set of attributes. To this end, `Asset Type` elements are divided into two groups: `Global Types` and `Local Types`. Former includes the generic and default attributes defined by the `IF` that are part of the asset's description (e.g., the *type* of the asset). Then, to let the asset owner tailor the protection policy according to their specific needs, the ontology provides the possibility of adding attributes described through `Local Types`. Such attributes are organization-specific and would not be revealed to other organizations. This greatly increases the flexibility of the access policy, since the asset owner can customize the evaluation parameters at will. For example, an organization might like to limit access to assets created by its R&D department, while allowing for a less restrictive policy for the assets created by the traffic management department. Then, it can introduce the private tag of `Department` as a `Local Type` for each asset, and set the access policy to "if the *Department* attribute of the requested asset is R&D, then access is restricted".

In addition to asset and user attributes, our system takes into account the `Relationship` concept as an additional aspect to determine access rights. Hence, it extends the scope of the ABAC model to include the semantics of complex business associations, which is a driving motivation behind the assertion of policies in industrial and competitive environments.

To this end, ABAC access policies are extended to include the concept of *relative* attribute. A *relative* attribute is a characteristic of a *subject* that is not *absolute* with respect to the *subject* itself, such as its name, or its sector. A *relative* attribute, instead, describes a feature of the *subject* with respect to the *object*, or to the *object's* owner, such as the relation of the requesting organization with the asset owner organization. For example, organization *O1* wants to protect its asset *A1* and allows only specific kinds of organizations (say, train operators) to access it. In this case, organizations *O2* and *O3*, which are both train operators, would have access to asset *A1*. Imagine that *O1* and *O2* are in some sort of partnership, whereas no such partnership exists between *O1* and *O3*. Following general business policy, *O1* managers might prefer to collaborate with partner companies, which would result in preventing *O3* from accessing *A1*, while still allowing access to *O2*. In this case the difference between *O2* and *O3* lies not in their absolute attributes (i.e., their type) but in their characteristics with respect to *O1*. A typical ABAC, however, cannot model a relative attribute of a *subject* because the latter cannot be defined by the *subject* itself, but by the *object*'s owner. For instance, in the example above, when deciding whether to grant *O3* access to *A1*, the access control system could not trust *O3* if it claimed to be partner of *O1* (by defining a corresponding attribute), since only *O1* could declare such a relationship. In our ontology, instead, class `Relationship` contains individuals representing actual relationships/associations between organizations.

4.3 Relationship Pattern and Instance

The concept of `Relationship` included in the model allows organizations to encode in the system their real-world business associations and take them into account in the definition of access policies. A `Relationship` has a `RelID` relating it a `RelType` and a `Pattern` which is a data structure to formally define a relation. The `Pattern` is defined as a tree structure where paths are made of a succession of Node and Arrow structures (Figure 4 shows two examples of `Pattern` definitions). Nodes embody the *organizations* and they are characterized by an identifier and a set of constraints on attributes, specifically the ID, UserType and AssetType classes. Similarly, an Arrow characterizes the connection (i.e., *relationship*) between two Nodes (i.e., *organizations*) in the pattern through properties including name, type and level of relation (either directly assigned or inferred), and an indicator of the direction of the relationship (whether it follows the same direction as the tree, or the opposite). An Arrow must always be followed by one Node, unless if it is a Loop Arrow. The Loop Arrow has been introduced to simplify the computation in the situation where an Arrow should be followed by a Node whose identifier has already appeared along the path. In this case, it can carry a reference to the identifier of the Node.

For example, Figure 4(A) depicts the `Pattern` for the *Secondary Partnership* relation, which corresponds to a "partner of partner" pattern through a composition of atomic *Partnership* relations. More precisely, the pattern defines *Secondary Partnership* between Node X and Node Z as a sequence of two consecutive *Partnership* relations (between Node X and Node Y, and between Node Y and Node Z), each composed of two Nodes which are connected together via an Arrow. The Arrow indicates the direction of the relation and specifies its type (i.e., *Partnership*). Figure 4(B), instead, shows the `Pattern` for the *Weak Partnership* relation between Node X and Node Y. The relation holds when Node Y is partner of Node X, as well as partner of Node Z, while Node Z is competitor of Node X. In other words, the *Weak Partnership* relation holds if a partner (Node Y in Figure 4(B)) of an organization (Node X) is also partner of a competitor (Node Z) of that organization.

4.4 Distributed Relationship Inference

A prominent feature of semantic technology is the ability of inferring new knowledge. By incorporating into our access control

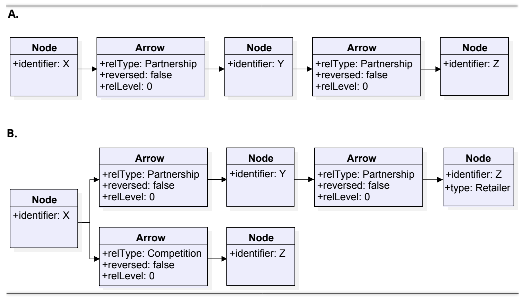

Figure 4: Relationship Pattern for *Secondary Partnership* (A), and for *Weak Partnership* (B).

system the ability to reason on relationship assertions, we can capture the complex relationships that exist in multi-agent enterprise systems more precisely. In particular, we evaluate `Patterns` defined as shown in Section 4.3 to infer new relationships among organizations. To perform the inference, however, one needs to overcome the problem of the distributed knowledge graph. Existing inference engines, such as the reasoning mechanism available in Jena and similar frameworks, operate only on complete knowledge graphs. That is, these reasoners require that the entire ontology be available, which in turn entails that semantic-based access control mechanisms must use a centralized storage for the ontology and the knowledge graph. This, however, is in contrast with the very nature of distributed environments, as it requires to move the resources of federated nodes into a central unit. However, a centralized architecture not only suffers from many short comings such as single-point-of-failure issues, but it goes against the general tendency of enterprise organizations of keeping control of their own assets and sensitive data. To overcome these challenges, we introduce custom *Exploration/Merge* procedures that perform, in a distributed manner, inferences based on `Pattern` definitions.

Reasoning constantly grows the knowledge graph by inferring new relationships based on directly added relations. Each (instance of) a relationship in the system is classified by a so-called `Level` that indicates the level of inference. *Level 0* means that the organization that defined the `Relationship` states its connection with a target organization as an instance of that `Relationship`. *Level 1* indicates that the connection is the result of a derivation over attributes of the involved elements or of *level 0* instances. Similarly, *Level n > 1* means the connection is the result of a derivation in which an instance of *Level n-1* has been considered.

A direct assignment (i.e., *Level 0*) is the strongest and most trustworthy connection since the originating organization has declared it by indicating the target organization ID. It implies that the defining organization is acquainted with the target organization in the real world. Indirect relationships between organizations are assigned by the system through inference by observing the facts and the logic of the connections among the other organizations in the network, to extract relationships that might have been hidden, difficult to grasp or neglected. Hence, it increases the completeness of access decisions and reduces the manual effort needed to explicitly define relations. Finally, to balance the trade-off between the amount of trust in a relationship and the completeness of the access decision, the system relinquishes the full control of relation assignment process to the user. Users can view and remove any indirect relation assignment anytime, tune the policy to consider as trustworthy relations only up to a certain *Level*, and exclude higher-level relations from the access evaluation process.

The assignment of an instance based on a relationship pattern consists of traversing knowledge graphs to extract any relation that matches the pattern representation. The distributed nature of the system can be an obstacle to achieve this goal, as the elements matching the pattern might be scattered across multiple graphs stored in separate IF nodes. To this end, we have developed an assignment algorithm that explores the distributed knowledge graph searching for relations that are compatible with the pattern, then merges the partial explorations to build the final complete result. The assignment algorithm is composed of the *Exploration* and *Merging* procedures, and its objective is to find all sets of bindings $\langle NodeID.identifier; OrganizationID \rangle$. That is, the algorithm must find actual organization IDs for the identifier placeholders (e.g., Node X, Node Y and Node Z in Figure 4) to instantiate a new inferred relationship and add it to the knowledge graph.

Exploration and *Merging* are message-driven procedures. At each step, they focus on finding the binding for a single Node of the Pattern by exploring the part of ontology stored in a single (relevant) IF-node. The *Merging* of local exploration results then creates a global and complete view of the ontology, which in turn enables the system to derive an extensive set of relations for each organization.

Exploration. The Exploration procedure is shown in Algorithm 1. Each exploration traverses a single path that leads from the root Node of the Pattern to a leaf one, ignoring Loop Arrows, that are

Algorithm 1 Exploration Procedure

```
1: procedure EXPLORATION
2: Input: message : ExplorationMessage, pattern : Pattern
3: Output: res : Set<ExplorationMessage>
4:     if message.lastArrowReversed then
5:         $candidates$ ← query the KB for organizations with relationship
           compatible with message.lastArrowData in last binding
6:     else
7:         $candidates$ ← message.candidates
8:     $validBindings \leftarrow \emptyset$
9:     $accepted$ ← query the KB for any organization satisfying the conditions
       in the root node of the pattern
10:    for $c$ in $candidates$ do
11:        if $c$ in $accepted$ then
12:            $valid \leftarrow true$
13:            for $la$ : LoopArrow in pattern do
14:                if relationship in graph not compatible with $la$ then
15:                    $valid \leftarrow false$
16:            if $valid$ then
17:                $validBindings \leftarrow validBindings \cup \{c\}$
18:    if $validBindings = \emptyset$ then
19:        $res \leftarrow empty\ message$
20:    else
21:        if pattern has FollowArrow then
22:            for $c$ in $validBindings$ do
23:                for $fa$ : FollowArrow in pattern do
24:                    if $fa$ not reversed then
25:                        $successor$ ← query the KB for targets of relationship
                           of c compatible with $fa$
26:                        $res \leftarrow new\ set\ of\ exploration\ request\ messages$
27:        else
28:            $res \leftarrow answer\ message$
```

treated as additional constraints. If the Node is followed by multiple Arrows, parallel explorations can be carried out for the branches.

By definition, the binding for the root Node is forced to be the author organization. Hence, at the starting point, the exploration process is initiated with the IF-Node hosting the organization that is the author of the relationship. Further explorations are triggered upon receiving the *Exploration Message (ExM)*. An *ExM* that is received by an IF-Node contains information including the bindings found in the previous steps, the conditions and directions of the relations, and a list of candidates for the binding of the current Node under exploration (referred to as root Node)[4].

Upon receiving a message, two checks must be performed. First, the direction of the relationship is checked. For a relation that is not reversed (Follow Arrow), the algorithm (line 7) proceeds to read the organizations in the *candidate list*, to find which of them could be bound to the (current) root Node as the previous step was able to determine the candidates by checking the relationship of the organization bound to the explored node. In case of relation with reversed direction (Reversed Arrow), candidates are organizations with a relationship compatible with the data in the message directed to the last bound organization, so they must be taken from the knowledge graph (line 5). The second check consist of the examination of each element in the *candidate list* to ensure that it respects the conditions specified in the pattern's Nodes (lines 10-17).

After the checks are done, invalid candidates are filtered out and the list of *valid bindings* between organizations and node identifiers is obtained. If the list is empty (line 18), it means that no binding has been found. The procedure then sends to the previous IF-Node an empty answer message (line 19). If the list is not empty and the node is not a leaf, then the Exploration requires additional steps: new *ExM* are created and the same process repeats (line 26).

Merging. The *Merging* follows the *Exploration* and completes the procedure to find the sets of bindings that satisfy the entire subtree that starts from the explored node by using the sets of bindings that satisfy the single paths. The procedure is executed for all the candidates obtained through the *Exploration* for each node, and all the parallel branches. A binding set compatible with the subtree following a node must be compatible with each single path. Consequently, the list of possible solutions contains any set that is the union of one of the provided solutions for each branch, with the constraint that each identifier must be bound to a single value.

The *Merging* procedure starts when each branch has provided an answer that lists the sets of bindings that satisfy a subtree. The solutions are then generated by trying every combination that takes one element for each list and creating the union set, then dropping it if any identifier is bound to more than one value. If the list is not empty, each set is extended with the binding of the current identifier of the node with the considered candidate and the list is added to the result for the merging, the level of the set is compared with the one saved from the exploration and the higher one is assigned. Once all candidates have been merged, the merging process sends back an answer message containing the results.

Finally, the system must always take into account any change in the relations and in the subject and object properties since it might affect the knowledge graph and make the current set of inferences invalid. Accordingly, our system monitors the actions which might produce changes in the ontology and reacts to them. We have defined various categories of operations (e.g., the *insertion*, *elimination* or *modification* of an entity, label or relationship) as well as the respective required course of action for each type. For example, the *insertion* of new elements would only add new data to the graph without impacting on the set of inferred bindings. This happens because the lack of negative constraints ensures that addition of new data can only make an organization satisfy a constraint that it did not satisfy before. So, the system only requires to launch a new exploration process and expand the graph with the new knowledge. However, a change such as a relationship *elimination* forces the system to review those bindings that include any of the two organizations involved in the removed relationship. This is needed because any inference based on that relationship is not valid anymore. Subsequently, the system retrieves the bindings to be examined and recomputes them if necessary.

[4]Note that each time that a binding is resolved, the corresponding Node is removed from the pattern and the next Node is considered as the root.

5 ACCESS CONTROL PROTOTYPE AND EVALUATION

Figure 5 shows the proposed access control system incorporated in the IF. As mentioned in Section 3, the AM component of the IF handles the authentication process, hence the mechanism proposed in this paper focuses on the evaluation of access requests of authenticated users concerning a given Asset. The access control mechanism is launched after the discovery process by the AM, which returns a generic description of an Asset along with its ID, and upon direct request from a user to view a selected Asset. The system follows the generic ABAC architecture and procedure, but it employs an ontology to provide attributes for the evaluation of the policies. *Policies* concerning the access to a particular asset are written by the owner of the Asset following the XACML standard. A policy is composed of one or more rules defining the general conditions which are associated with two possible effects: grant or deny. To solve and conflicts between rules, policies have a combination algorithm to decide which decision the policy must return given the results of the single rules. The system also implements a default deny policy to cover the situations in which either no permission is applicable, or the algorithm to solve conflicts is unable to reach a decision. In addition, Asset owners can define new `Patterns` in the form of XML files through the provided interfaces of the system.

The `PDP` and `Attribute Handler(AH)` modules are tied to the evaluation of access requests and serve the same purpose as the `PDP` and `Context Handler(CH)` components in core ABAC (Figure 2). In particular, `AH` is tasked with retrieving the relationships, subject and object attributes from the knowledge graph through the `Local Ontology Manger` and `Remote Ontology Manager` components. It then wraps and presents the set of attributes required for evaluation of an access request in a format comprehensible to the `PDP`. As specified in the XACML standard, the `PDP` module should be able to recover the provided data by means of XPath expressions and the module must interpret them and return the correct attributes.

The second set of modules are the ones tied to the ontology and its management, which include the `Local` and, `Remote Ontology`

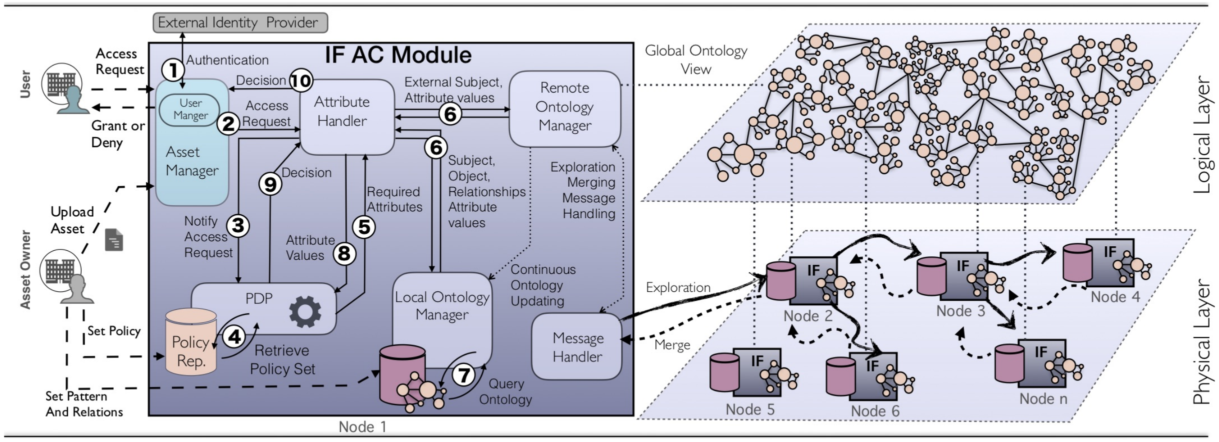

Figure 5: Distributed semantic-based access control architecture

Manager and the Message Handler. The latter is in charge of handling the *Exploration* and *Merging* procedures for the indirect assignment, and exploiting the parallelization opportunities provided by the structure of the algorithm. Ontology managers, instead, include all the elements that build and keep up-to-date the ontology graph. The interface offered allows external elements to add individuals to the ontology, define new relationships, start the relationship mining process and answer queries from other modules.

5.1 An application Scenario

Figure 6 depicts an example topology of the system. Organizations *Org*1, to 7 are all various transportation corporations registered in the IF and willing to use its services and benefits. *Org1* operates as a Transport Service Provider (TSP) and publishes through the IF its Assets, including timetables and paths of owned trains. Data provided by *Org1* are interesting Assets for organizations, like *Org4*, that develop and manage Travel Expert (TrEx) systems to build travel plans for clients. *Org1* and *Org4* hence have the Partnership relation to facilitate the collaborations and sharing different types of data. Furthermore, *Org1* is investing in the development of ticket retailing software for its trains. Recently *Org1* found out about two new companies, *Org6* and *Org7*, which are entirely focused

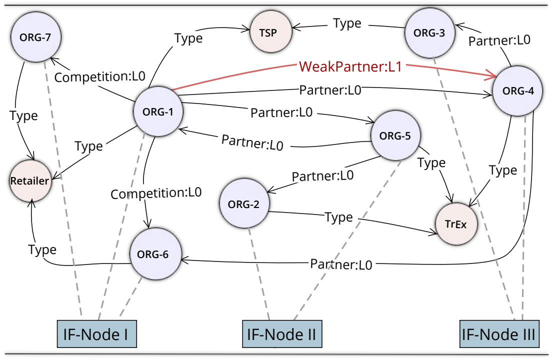

Figure 6: An example topology

on online transportation ticketing and retailing mainly in collaboration with Travel Experts. *Org1* hence considers both of them as its rivals and decides to restrict their access to its data. So, it directly adds the Competition relation with *Org6* and *Org7* and sets the corresponding policies to protect its Assets against any of its competitions (*Rule 1* in Table 1). Yet, *Org1* knows that if any of its partners are collaborating with *Org6/7*, then it is probable that *Org6/7* gains access to some of its Assets that are already shared with such partners. *Org1* then considers those of its partners which are also partners of *Org6/7* less trustworthy and decides to impose on them a more restrictive sharing policy. Therefore, it defines a Pattern for the Weak Partner relation (as explained in Section 4.3 and Figure 4(B)) and then adds *Rule 2* in the *Policy* shown in Table 1.

The rest of the process would be executed by the system automatically. It constantly monitors the entire network of organizations to identify the Weak Partners of *Org1* and subsequently protects the travel data in case of an access request from them. To this end, the system undergoes several *Exploration* and *Merging* steps to derive new relationship instances and keep the knowledge graph updated.

Given the Pattern in Figure 4(B) and the current topology of the organizations and their relations shown in Figure 6, the process starts at IF-Node I, which is the host of *Org1*. By definition, variable X is forced to bind to *Org1*, since it is the author of the Pattern. Then, the process must explore two branches: one for assigning a partner of X and its partner—i.e., respectively, Y and Z in the top branch (***Branch 1***) of Figure 4(B); and another one for the competitor—i.e., Z in the bottom branch (***Branch 2***). The *Merging* of the two branches then completes the whole bindings forX-Y-Z and states the Weak Partner relation between X and Y (if any).

Starting from *Branch 1*, the first variable is Y, and the candidates to be bound to it are *Org4* and *Org5* since they both are a partner of *Org1* (See Figure 6). The exploration continues on IF-Node II, and IF-Node III which are hosting *Org5* and *Org4*, respectively. The Nodes in the Pattern present no constraints and the absence of Loop Arrows shows that no further constraint needs to be analysed. That means both possible bindings $\langle Y = \texttt{Org4}\rangle$ and $\langle Y = \texttt{Org5}\rangle$ can be accepted. The fact that it exists a following arrow means that there is the need for a further exploration step. For the first

Table 1: A Policy Definition

Conflict Resolution	deny-override algorithm
Rule 1	**IF** `subject_relation`: Competition **THEN** DENY
Rule 2	**IF** `object_attribute`:TravelData **and** `subject_relation`: WeakPartner **THEN** DENY

group of bindings, then, the candidates for the next variable—i.e., Z—are *Org3* and *Org6*, which are both partners of *Org4*. Similarly, for the second group of bindings the candidates for Z are *Org1* and *Org2*, which are partners of *Org5*. The *Exploration* messages are then sent to the `IF-Nodes` that are the host of each organization and selection of the binding for Z continues. Ultimately, *Org1* and *Org3* could not be accepted as bindings for Z, since *Org1* was already bound to variable X, and *Org3* is an organization of Type TSP that is violating the constraints stated in the `Pattern` for Z in *Branch 1*. In other words, the `Pattern` targets only those partners of Y which are a Retailer (and hence they are potential competitors of X). So the final values for *Branch 1* are as follows:

Branch 1 : $\langle Y = \texttt{Org4},\ \ Z = \texttt{Org6}\rangle$ and $\langle Y = \texttt{Org5},\ \ Z = \texttt{Org2}\rangle$

In parallel, the search for the bindings of Z in *Branch 2* has started and there are two candidates for that: *Org6* and *Org7*, which are organizations with the competition relation with *Org1*. Here, both candidates are acceptable, so the result of this branch is as follows:

Branch 2 : $\langle Z = \texttt{Org6}\rangle$ and $\langle Z = \texttt{Org7}\rangle$

The *Merging* process then starts when both branches have been explored as explained above. The second binding pair of *Branch 1* cannot be merged with any of the two results in *Branch 2* as Node Z appears in both members of the pair and it is bound to different individuals. For the same reason, the first pair of *Branch 1* cannot be merged with the second binding of *Branch 2*, but it can be merged with the first one. The result of the merging happening in Node X is a single path with the bindings: $\langle Y = \texttt{Org4},\ \ Z = \texttt{Org6}\rangle$ of level 0, which is to be extended with the candidate in Node X resulting in the final binding as follows.

Final Binding : $\langle X = \texttt{Org1},\ \ Y = \texttt{Org4},\ \ \texttt{Z} = \texttt{Org6}\rangle$

After the *Exploration* and *Merging* processes are completed, the system updates the knowledge graph with the new relationship assignment (shown as the red arrow in Figure 6). Afterwards, when a request for accessing travel data assets of *Org1* is received, the system queries the graph to retrieve all the organizations related to *Org1* through either the `Competition` relation—i.e., *Org6* and *Org7*—or the `Weak Partner` relation—that is, *Org4*. Then if the subject organization (i.e., the requester) is among them the access is denied, otherwise it is granted!

5.2 Evaluation

The prototype system has been developed in Java and employs Apache Jena as triple-store. Its initial setting runs over an automatically generated ontology containing 100 organizations. Our current prototype implementation focuses on the realization of the access control part. The communications with a network of IF nodes has been simulated by a special-purpose module that mimics the required procedure to interact with other IF nodes. Furthermore, the integration with the IF (or any other enterprise ecosystem and framework) is achieved through various interfaces that are supposed to communicate with the Access Control module for dispatching the access request and other required data and inputs.

The evaluation presented here focuses on the examination of the novel aspect of the proposed the system, i.e., the automatic relation assignment algorithm that continuously grows the knowledge graph by inferring new relations among organizations.[5] In particular, we aimed to verify the *correctness* of the algorithm and to analyze the increase in *computation time* depending on the variation of the size of the ontology and on the complexity of the relationships patterns. Several tests have been carried out on multiple ontologies that have been randomly populated to avoid the insertion of bias, and for the three relationships `Partner`, `Secondary Partner` and `Weak partner`.

The first set of experiments aims to verify the correctness of the algorithm, that is, to check whether all and only the correct sets of bindings ⟨Node-identifier; organizationID⟩ are retrieved from the complete knowledge graph for each pattern. To prove that the algorithm provides every possible set, it is sufficient to, first, show that the exploration step takes into consideration any organization known to the IF node and discards those violating the constraints. Second, to verify that the merging process, during the building of the superset, does not add any binding that is not necessary to satisfy at least one of the branches following the node, except for the binding of the node itself. The correctness tests were executed by searching for the relationship `Weak Partner` and `Secondary Partner` on a small ontology of 20 organizations to allow human verification, and the algorithm has been able to yield the correct results every time.

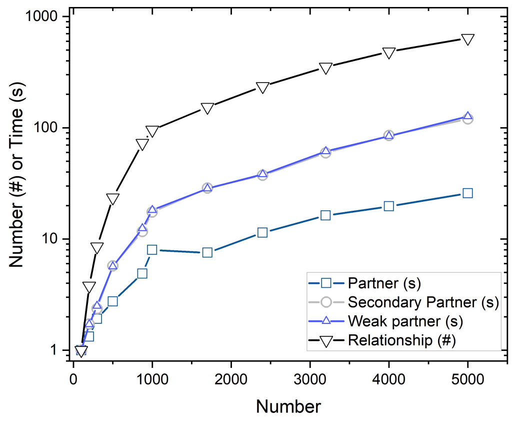

Figure 7: Observation of Number of Relations and Computation time as number of organization grows

The next validation was focused on the examination of the *complexity* of the algorithm that can be determined as a function of

[5]The other aspects of the system have been tested and showed satisfying results as well (e.g., the average computation time for access right determination upon a particular request is below 1 second); for reasons of space, we leave the complete evaluation report to a future extended version of the paper.

the number of successive exploration steps taken during the computation. That is because each step can be a request to a remote node, which makes it the most time-consuming part of the procedure. The message-driven nature of each step allows a massive parallelization for the exploration steps by having each candidate checked in parallel, and for different branches starting from the considered node.

Under the condition that there is a sufficient amount of threads for the elaboration of the requests, the complexity defaults to the length of the longest path leading from the root to a leaf node. To prove these considerations, the three relationships have been submitted to the system on different, randomly populated ontologies. According to the results shown in Figure 7, the computation time for the relationship `Partner` (that is the simplest one) is significantly lower than the other relationships. Yet, interestingly, relationships `Weak Partner` and `Secondary Partner` behaved very similarly, despite the fact that `Weak partner` requires an additional merging procedure. So, a more complicated pattern would not necessarily consume a longer computation time to be resolved. In other words, we do not observe a proportional ratio between computation time and complexity of the pattern. Possibly, the topology of the network and shape of the relations among individuals have more impact on the computation time than the complexity of the pattern. Finally, an interesting observation is that the computation time increases with much lower rate compared to the increase in the size of the knowledge graph and in the number of relationships, which makes the algorithm suitable for large scale applications.

6 CONCLUSIONS

This paper presented a semantic-based and distributed access control mechanism to address the authorization challenge in distributed enterprise systems. The key features of the proposed mechanism are, firstly, the ontology-based modeling of the complex concepts and relations of organizations in a collaborative ecosystem and, secondly, a reasoning procedure to infer implicit contexts. Furthermore, our proposed solution follows a federated access control mechanism and an exploration procedure for each replica of the system that traverses and carries the search of a pattern in the partial graphs and incrementally harvests the results.

To demonstrate the applicability of the proposed system a prototype implementation has been developed. In particular, we have implemented the system as the access control module of the Shift2Rail Interoperability Framework for the transportation domain. The experiments proved that the procedure is lightweight and highly parallelizable, and it allows multithreaded implementations.

Our future work will concentrate on extending the prototype implementation and deliver the actual integration with the Interoperability Framework. In addition, user interactions with the system will be improved mainly by the development of more sophisticated GUIs for pattern and policy insertion, and provision of more default relationship patterns to reduce the need for manual pattern creation.

Acknowledgments. This work was supported by Shift2Rail and the EU Horizon 2020 research and innovation programme under grant agreement No: 826172 (SPRINT).

REFERENCES

[1] Claudio Agostino Ardagna, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. 2006. A web service architecture for enforcing access control policies. *Electronic Notes in TCS* 142 (2006), 47–62.

[2] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. 2007. A survey on context-aware systems. *Int. Journal of Ad Hoc and Ubiquitous Computing* (2007).

[3] Luciano Baresi and Mersedeh Sadeghi. 2018. Fine-grained context-aware access control for smart devices. In *8th Int. CSIT Cont.* IEEE, 55–61.

[4] Prosunjit Biswas, Ravi Sandhu, and Ram Krishnan. 2016. Label-based access control: An ABAC model with enumerated authorization policy. In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control.* 1–12.

[5] Dan Brickley and Libby Miller. 2007. FOAF vocabulary specification 0.91. (2007).

[6] Daniel J Buehrer and Chun-Yao Wang. 2012. CA-ABAC: Class algebra attribute-based access control. In *Int. Conf. WI-IAT.* IEEE.

[7] Barbara Carminati, Elena Ferrari, and et al. 2009. A semantic web based framework for social network access control. In *Proc. of ACM sym. on Access control models and technologies.*

[8] Lorenzo Cirio, Isabel F Cruz, and Roberto Tamassia. 2007. A role and attribute based access control system using semantic web technologies. In *OTM Confederated Int. Conf." On the Move to Meaningful Internet Systems".* Springer, 1256–1266.

[9] Ni Dan, Shi Hua-Ji, Chen Yuan, and Guo Jia-Hu. 2012. Attribute based access control (ABAC)-based cross-domain access control in service-oriented architecture (SOA). In *Int. Conf. on Computer Science and Service System.* IEEE, 1405–1408.

[10] Marwah Hemdi and Ralph Deters. 2016. Using REST based protocol to enable ABAC within IoT systems. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).* IEEE, 1–7.

[11] Shohreh Hosseinzadeh, Seppo Virtanen, Natalia Díaz-Rodríguez, and Johan Lilius. 2016. A semantic security framework and context-aware role-based access control ontology for smart spaces. In *Proc.of the Int. Workshop on Semantic Big Data.* 1–6.

[12] Luokai Hu and Ying at al. 2009. Towards an approach of semantic access control for cloud computing. In *Int. Con. on Cloud Computing.* Springer.

[13] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. 2013. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication* 800, 162 (2013).

[14] Vincent C Hu, D Richard Kuhn, David F Ferraiolo, and Jeffrey Voas. 2015. Attribute-based access control. *Computer* 48, 2 (2015), 85–88.

[15] ASM Kayes, Jun Han, and Alan Colman. 2013. An ontology-based approach to context-aware access control for software services. In *International Conference on Web Information Systems Engineering.* Springer, 410–420.

[16] Florian Kerschbaum. 2010. An access control model for mobile physical objects. In *Proc. of the ACM symp. on Access control models and technologies.* 193–202.

[17] D Richard Kuhn, Edward J Coyne, and Timothy R Weil. 2010. Adding attributes to role-based access control. *Computer* 43, 6 (2010), 79–81.

[18] Bo Lang, Ian Foster, and et al. 2009. A flexible attribute based access control method for grid computing. *Journal of Grid Computing* 7, 2 (2009).

[19] Bo Lang, Hangyu Li, and Wenting Ni. 2010. Attribute-based access control for layered grid resources. In *Int. Conf. on FGCN.* Springer.

[20] Amirreza Masoumzadeh and James Joshi. 2010. Osnac: An ontology-based access control model for social networking systems. In *2010 IEEE Second International Conference on Social Computing.* IEEE, 751–759.

[21] Catherine Jensen McCollum, Judith R Messing, and L Notargiacomo. 1990. Beyond the pale of MAC and DAC-defining new forms of access control. In *Proc. on Research in Security and Privacy.* IEEE, 190–200.

[22] Chi-Chun Pan, Prasenjit Mitra, and Peng Liu. 2006. Semantic access control for information interoperation. In *Proceedings of the eleventh ACM symposium on Access control models and technologies.* 237–246.

[23] Mersedeh Sadeghi, Petr Buchníček, and et al. 2020. SPRINT: Semantics for PerfoRmant and scalable INteroperability of multimodal Transport. In *TRA 2020.*

[24] Ravi S Sandhu. 1998. Role-based Access Control. In *Adv. in comp.* Vol. 46. Elsevier.

[25] Daniel Servos and Sylvia L Osborn. 2017. Current research and open problems in attribute-based access control. *ACM Computing Surveys (CSUR)* 49, 4 (2017).

[26] Haibo Shen. 2009. A semantic-aware attribute-based access control model for web services. In *Int. Conf. on Alg. and Arch. for Parallel Processing.* Springer.

[27] Alexander Smirnov, Alexey Kashevnik, Nikolay Shilov, and Nikolay Teslya. 2013. Context-based access control model for smart space. In *Int. Conf. CYCON.* IEEE.

[28] OASIS Standard. 203. extensible access control markup language (xacml) V. 3.0. (203).

[29] Lili Sun, Hua Wang, Jianming Yong, and Guoxin Wu. 2012. Semantic access control for cloud computing based on e-Healthcare. In *Proc. of Int. Conf. CSCWD.*

[30] Zhiguo Wan, Robert H Deng, et al. 2011. HASBE: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing. *Transactions on information forensics and security* (2011).

[31] Wang Xiaopeng, Luo Junzhou, Song Aibo, and Ma Teng. 2005. Semantic access control in grid computing. In *Int. Conf. ICPADS'05)*, Vol. 1. IEEE.

[32] Eric Yuan and Jin Tong. 2005. Attributed based access control (ABAC) for web services. In *Proc.of Int. Conf. on Web Services (ICWS).*