Social Semantic Network-Based Access Control

Serena Villata, Luca Costabello, Fabien Gandon, Catherine Faron Zucker, Michel Buffa

▶ **To cite this version:**

Serena Villata, Luca Costabello, Fabien Gandon, Catherine Faron Zucker, Michel Buffa. Social Semantic Network-Based Access Control. Security and Privacy Preserving in Social Networks, Springer, 2013, Lecture Notes in Social Networks, 10.1007/978-3-7091-0894-9_6 . hal-01187472

HAL Id: hal-01187472
https://hal.science/hal-01187472

Submitted on 26 Aug 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Social Semantic Network-based Access Control

Serena Villata, Luca Costabello, Fabien Gandon, Catherine Faron-Zucker, Michel Buffa

Abstract Social networks are the basis of the so called Web 2.0, raising many new challenges to the research community. In particular, the ability of these networks to allow the users to share their own personal information with other people opens new issues concerning privacy and access control. Nowadays the Web has further evolved into the Social Semantic Web where social networks are integrated and enhanced by the use of semantic conceptual models, e.g., the ontologies, where the social information and links among the users become semantic information and links. In this paper, we discuss which are the benefits of introducing semantics in social network-based access control. In particular, we analyze and detail two approaches to manage the access rights of the social network users relying on Semantic Web languages only, and we highlight, thanks to these two proposals, what are pros and cons of introducing semantics in social networks access control. Finally, we report on the other existing approaches coupling semantics and access control in the context of social networks.

Serena Villata
INRIA, Sophia Antipolis, France, e-mail: serena.villata@inria.fr

Luca Costabello
INRIA, Sophia Antipolis, France e-mail: luca.costabello@inria.fr

Fabien Gandon
INRIA, Sophia Antipolis, France e-mail: fabien.gandon@inria.fr

Catherine Faron-Zucker
I3S, Université Nice Sophia Antipolis - CNRS, France e-mail: faron@polytech.unice.fr

Michel Buffa
I3S, Université Nice Sophia Antipolis - CNRS, France e-mail: buffa@unice.fr

1 Introduction

One of the key features of the Social Web is the ability to publish, and thus find a lot of personal and professional information about people. With the advent of the Social Semantic Web this is even more evident, as underlined by Breslin et al. [5]. The availability of personal and non personal data of the users has both positive and negative sides. On the one hand, this allows people to share their data, e.g., photos, videos, posts, with their friends and the persons they know. On the other hand, semantic forms of the users' profiles like FOAF profiles and data can be reused elsewhere, e.g., what happened with FOAF search engines and aggregators as Plink, or FoaFSpace. This leads to the need for mechanisms where users can restrict the access to their data by specifying the attributes the accessors must satisfy to have the access granted.

In particular, security, protection, and access control represent a major challenge in content management systems. This issue is central also in collaborative social Web sites, where the collaborative editing and sharing of the documents raises the question of the definition of access rights. Moreover, access control is important to lead to a diffusion of Social Semantic Web platforms to make them able to guarantee the same kind of authorizations as in standard Social Web platforms like Facebook, or Google+. Managing the access to the resources is thus one of the major challenges facing the Social Semantic Web.

In this paper, we address the following research question: *What are the benefits of adopting Semantic Web models and languages for social network-based access control?* Policies, norms and the Semantic Web *Trust Layer*, as shown in Figure 1, are usually presented as a set of rules and constraints that model the intended behaviors of the users. Within W3C, the Policy Languages Interest Group is the forum that coordinates the efforts of the community around the definition of policy languages, frameworks and use cases. Apart from W3C activities, one of the most prominent standard for modelling policies is the eXtensible Access Control Markup Language (XACML). Policies can be defined at community level but they can also be defined at the individual level, e.g., my privacy policies in a social network, e-mail filtering policies, etc. Policies on the Semantic Web build the foundation for privacy and access rights of personal or community data, whereas norms in general establish best practices, e.g., how to publish the data. The definition of both private and community policies is useful for various further applications such as checking compliance or conformance, policies alignment, or checking the internal consistency of policies.

We answer the research question by presenting two approaches for defining the access control policies using Semantic Web languages only. These two approaches are applied to social semantic networks and show pros and cons of introducing semantics in social network-based access control.

First, we consider content management systems based on Semantic Web servers, and we propose an approach for managing access rights to resources based on Semantic Web models and techniques [6]. We present an ontology dedicated to the representation of the access rights given on a document to some users or user classes. We call this ontology AMO, an acronym meaning *Access Management Ontology*.

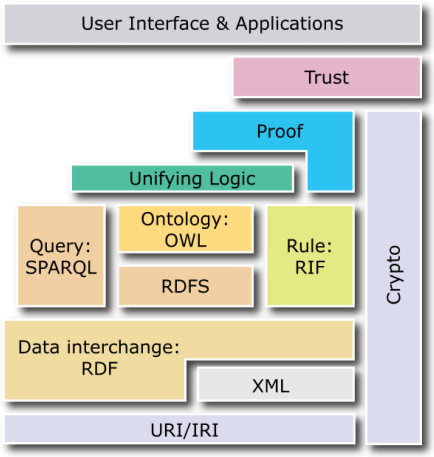

Fig. 1: The Semantic Web stack (from `http://www.w3.org/sw`).

AMO is made of a set of classes and properties for annotating the resources and a base of inference rules modeling the access control policy. When applied to the annotations of resources, these rules enable to control access according to a given strategy. This declarative modeling as a rule base ensures an easy adaptation of the ontology to different access control policies and thus avoids modifying annotations of documents in the case of a change of strategy. In the ISICIL[1] research project, we use the AMO ontology to manage access to resources shared by a network of technical watchers: documents produced by content management tools, wikis or blogs, static HTML documents produced by web scraping (i.e. firefox extensions similar to "Scrapbook"), bookmarks, etc. One of the issues of this project oriented to Web 2.0 and Semantic Web techniques concerns the management of access to the resources shared by the social network of watchers. Among the documents produced by the watchers there are those of a collaborative website run by the semantic wiki SweetWiki that we developed [7] and that is used in this paper to illustrate the use of AMO. SweetWiki integrates Semantic Web technologies to improve structure, search and navigation. More specifically, it associates to wiki pages RDF/S annotations that make the content of these pages processable by the semantic engine CORESE [10].

Second, we describe the Social Semantic SPARQL Security for Access Control vocabulary (`S4AC`[2]), a lightweight ontology which allows to specify fine-grained access control policies for RDF data [44, 45, 15]. We adopt exclusively Semantic Web languages and recycle, when possible, already existing vocabularies. In this model, we avoid the usual access control lists, often maintained by a sole authority, because we cannot specify the access restrictions to any particular user, in a context where the user information is so dynamic. We rely on social tags assigned by the

[1] `http://isicil.inria.fr/`

[2] `http://ns.inria.fr/s4ac/`

users to data and other users. Moreover, contextual information is also considered in this model to grant access to users by considering not only their personal information, but also additional attributes, e.g., time and location constraints. We adopt the PRISSMA[3] vocabulary [14] to model the user context in which the access request takes place. Following the widely adopted formalization of context provided by Dey [21], PRISSMA defines the information context as the sum of the following three dimensions: the *User*, modelling the target mobile user (stereotypes or specific instances), the *Device*, which represents the mobile device in use, and the *Environment*, the dimension dealing with the physical context where the consumption takes place. In particular, the S4AC vocabulary defines an access policy as a tuple composed by an *access privilege*, stating the kind of privilege the data consumer is granted to, an *evaluation context* which is requested to hold when the client query is performed, the *object(s)* to be protected by the policy, i.e., the named graphs [9], and a set of *access conditions* which have to be verified in order to grant the access to the consumer. This vocabulary relies on the SPARQL 1.1 language. In particular, the Access Conditions are expressed through ASK queries, returning true if access is granted to the data consumer, false otherwise. Access privileges are mapped to SPARQL primitives through the SPIN vocabulary[4]. The overall access control framework allows data providers to specify lightweight access policies to protect their data, at *named graph* granularity. The Access Control Manager verifies which named graphs are accessible by the user, so that the user's query is run on those graphs only. The system evaluation shows that access control comes with a cost, and that performance loss is acceptable when dealing with sensitive data. For the time being, our lightweight framework assumes the trustworthiness of the information sent by the data consumer. Moreover, our approach focuses only on SPARQL endpoints. Other access strategies are out of the scope of this work. Despite the amount of proposals of access control models [26, 42, 43, 1, 22, 25, 24, 23, 33, 35, 8], none of them presents a Social Semantic access control model based on Semantic Web languages only, a pluggable and easy-to-integrate filter for generic SPARQL endpoints without modifying the endpoint itself, providing access conditions from triple granularity level up to dataset granularity level, and taking into account the social tags assigned by the users to their data and other users and the contextual information. Moreover, we rely on W3C recommendations only, as we do not introduce any new language or technology.

The two proposals we describe do not deal with access control for the Social Web in general, but we present two frameworks suitable for the Social Semantic Web. Our aim is not to provide a privacy manager or a cryptography system, but we are interested in formalizing, developing and evaluating access control frameworks which authorize or not the access of the users to the data of the other users, without considering personal information only.

The structure of the paper is as follows. Section 2.1 presents the ontology AMO and the use of AMO in SweetWiki, highlighting the adaptability of AMO to different

[3] http://ns.inria.fr/prissma/

[4] http://spinrdf.org/spin.html

access control policies. Section 2.2 presents the S4AC ontology, defined for social access control using the SPARQL 1.1. language, coupled with the PRISSMA ontology for the user context definition, with the aim to propose a contextual access control model for social semantic networks. Section 3 is dedicated to the positioning with respect to the existing work.

2 Semantic approaches to social network-based access control

2.1 Ontology based Access Management

We present in this section an access control model where access control is based on an ontology modeling access rights to resources and access control strategies.

2.1.1 The AMO Ontology

In a file system or in a content management system, roles (administrator, owner, etc.) are associated with users or user groups and different types of access to resources (writing, reading, etc..) are defined, access to resources varying from one user to another depending on its role. This analysis led us to define a set of classes and properties to describe the access rights to resources. This is what we describe in Section 2.1.1.1.

Content management systems share the same general principles for access control to resources, however they adopt strategies that may vary from one system to another. To allow easy adaptation of the ontology supporting the management of access to resources according to the chosen strategy, this latter is declaratively modeled in AMO as a base of inference rules that can be modified at leisure without affecting the annotations of the resources to manage. We describe in Section 2.1.1.2 a rule base that modelizes one strategy for the access control of documents in the semantic wiki SweetWiki.

2.1.1.1 AMO Classes and Properties

AMO is based on some basic principles shared by all content management systems:

- *Agents* of a content management system are the users, user groups, services that interact with the system.
- These agents have *roles*. In the case of collaborative editing systems such as wikis or CMS, these roles are those of guest (agent not registered in the system), contributor, administrator. Other roles can be modeled depending on the kind of system.

- Each role is associated with a list of authorized *actions*. In the case of collaborative editing systems, the possible actions on a resource are creation, reading, modification and destruction of content, modification of access rights, modification of the list of agents allowed on a resource, change of the access type defined for a resource. Other actions can be modeled for other kinds of systems.
- There are different *types* of access to resources. We choosed to implement a strategy popular in some collaborative editing systems: a resource can be public (all users have reading and writing access), private (only authorized agents have reading and writing access) or semi-private (free reading access, writing access only to authorized agents). Again, other types of access can be added for other types of systems.
- Finally, the actions authorized to an agent on a resource depend on the role of the agent and/or the type of access defined for the resource.

The AMO ontology presented in Figure 2 provides the concepts necessary to represent this knowledge. The three classes `Role`, `Action` and `AccessType` are central to AMO. `Role` is the meta-class of classes `Administrator`, `Contributor` and `Guest`. `Action` is the meta-class of classes `ReadContent`, `ModifyContent`, `DeleteContents`, `ModifyUserRights`, `ModifyAccessType` and `ModifyAuthorizedAgents`. Finally, `AccessType` is the meta-class of class `Private`, `Public` and `SemiPublic`.

Three classes of the FOAF vocabulary – the standard for social web discussed in section 3 – are also central in AMO: `Agent` and its sub-class `Group` and `Document`. They are used as domain or range of properties of AMO and also in the rules of AMO.

Properties `creator` and `hasAuthorizedAgent` associate an agent to a document (they have for domain the class `Document` and for range the class `Agent`); `hasRole` associates a role to an agent and `hasActionOnResource` an action to a role; property `hasAccessType` associates an access type to a document.

In addition, to represent into a model of binary properties the ternary relation which states that an agent is authorized to perform an action on a resource, we have reified this relationship by introducing the subclass `AuthorizedActionOnResource` specializing the class `Action`, a property `hasAuthorizedActionOnResource` that associates an instance of `AuthorizedActionOnResource` to an agent, and the properties `hasDocument` and `hasAction` that associate to an instance of `AuthorizedActionOnResource` respectively a document and an action.

AMO is a RDFS vocabulary which can be used to annotate the RDF resources whose access we want to control.

2.1.1.2 AMO Inference Rules

Content management systems adopt access control strategies to resources that can vary from one system to another. Rather than varying the annotations of resources depending on the control strategies, we propose to model declaratively the control

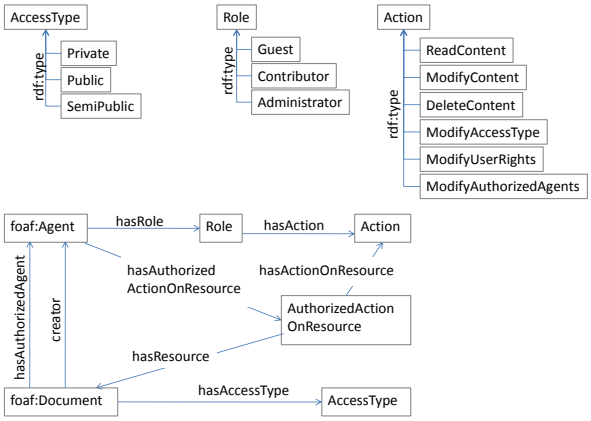

Fig. 2: AMO classes and properties

	Public	Semi-Public	Private
Guest	ReadContent	ReadContent	
Contributor	ReadContent	ReadContent ModifyContent DeleteContent	
AuthorizedAgent	ReadContent ModifyContent DeleteContent ModifyAuthorizedAgents ModifyAccessType		
Administrator	ReadContent ModifyContent DeleteContent ModifyAuthorizedAgents ModifyAccessType ModifyUserRights		

Fig. 3: An access control policy modeled in AMO

strategy in the AMO ontology, as a base of inference rules. Some rules may vary depending on the strategy modeled while the annotations remain unchanged. The rule base presented here is that of SweetWiki whose strategy of access control is similar to that of the widely used open source wiki Mindtouch Deki [5].

By default, administrators have all rights on all resources. The contributors have all rights relative to the content of resources, those reported as agents of a resource by the author thereof also have some administrative rights on it. Guests are only allowed to read the content of resources. Figure 3 below summarizes the access rights to a resource depending on the type of access and the role of the user who tries to access the resource (horizontally are the types of access resources, vertically user roles).

[5] http://www.mindtouch.com/

We model this strategy in AMO *declaratively* by six inference rules, each corresponding to a situation described in Figure 3. For example, Rule 1 below specifies the rights granted to agents of a given resource. Other rules describe general laws such as *a member of a group inherits the roles assigned to her group* (Rule 2) or *creator of a resource is an agent of this resource* (Rule 3).

These rules are expressed in the SPARQL language, by using the query pattern CONSTRUCT/WHERE: such a query enables to *construct* RDF graphs by replacing the variables of its clause CONSTRUCT by the values that satisfy the clause WHERE (they are retrieved by searching for potential matches to clause WHERE with the RDF data available in the content management system). A query CONSTRUCT/WHERE can therefore be seen as a rule applied in forward chaining, with clause WHERE the premise and clause CONSTRUCT the conclusion. These rules however can also be used in backward chaining, as is the case in the semantic engine Corese.

Rule 1:

```
CONSTRUCT {
   ?agent amo:hasAuthorizedActionOnResource ?a
   ?a amo:hasResource ?resource
   ?a amo:hasActionOnResource amo:ReadContent.
   ?a amo:hasActionOnResource amo:ModifyContent.
   ?a amo:hasActionOnResource amo:DeleteContent.
   ?a amo:hasActionOnResource amo:ModifyAccessType.
   ?a amo:hasActionOnResource amo:ModifyAuthorizedAgents }
WHERE {
   ?resource rdf:type foaf:Document.
   ?resource amo:hasAuthorizedAgent ?agent }
```

Rule 2:

```
CONSTRUCT {
  ?agent amo:hasRole ?role }
WHERE {
  ?group amo:hasRole ?role
  ?group foaf:member ?agent }
```

Rule 3:

```
CONSTRUCT ?resource amo:hasAuthorizedAgent ?agent
WHERE ?resource amo:creator ?agent
```

This *declarative* modeling of the strategy of access rights management ensures easy maintenance. Changing rights of a class of users – and this for all resources involved – will only require the addition or deletion of triples statements in the conclusion of a rule. Similarly, the addition of new roles will only require the addition of a class representing this role and the rules representing the access rights associated with that role.

2.1.2 Access Rights Management in SweetWiki

The AMO ontology has been used in the ISICIL project to annotate resources shared by a social network of business watchers. The management of access to these resources in the engine SweetWiki was based on (1) the exploitation of these semantic

annotations, (2) inferences on these annotations based on AMO rules and (3) the formulation of SPARQL queries to retrieve knowledge about the authorized access to a specific user on a given resource. In SweetWiki, annotations of resources are based on FOAF, SIOC and AMO ontologies and SPARQL queries are used in most of the features implemented: RDF annotations feed the semantic engine CORESE embedded in SweetWiki. In particular, by using the approximate search possibilities of Corese[11, 12] and a system of semantic tagging of documents, SweetWiki offers an "intelligent" browsing mechanism enhanced by suggestions.

2.1.2.1 Annotation of Ressources with AMO

When creating a wiki page, the identity of its creator is registered and also the type of access to the page that is decided by her and possibly one or more agents authorized on the page, also designated by the creator. In SweetWiki this knowledge is represented into RDF annotations associated with the created pages. For example, Annotation 1 below results from the creation of a private wiki page by the user `AnnaKolomoiska` who stated that agent `MichelBuffa` is authorized on this page. This annotation uses the AMO properties `creator`, `hasAuthorizedAgent` and `hasAccessType` (and the class `WikiArticle` of the SIOC vocabulary).

Annotation 1:

```
<rdf:RDF xmlns="http://seetwiki.i3s.unice.fr/AMO.rdfs#" ... >
<sioc:WikiArticle rdf:about="#TestPage">
    ...
    <creator rdf:resource="#AnnaKolomoiska"/>
    <hasAuthorizedAgent rdf:resource="#MichelBuffa"/>
    <hasAccessType rdf:resource="#Private"/>
  </sioc:WikiArticle>
</rdf:RDF>
```

When registering a user in SweetWiki, this information is represented in an RDF annotation. For example, Annotation 2 below states that `MichelBuffa` is a contributor to the wiki. It uses the AMO class `Contributor` and AMO property `hasRole` (and the class `Agent` of the FOAF vocabulary discussed in section 3).

Other annotations express knowledge relative to the user groups of the wiki. For example, Annotation 3 states that `AnnaKolomoiska` and `CatherineFaron` are members of the administrator group of the wiki. It uses for that the AMO property `hasRole` (and the FOAF classes `Group` and `Agent` and the FOAF property `member`).

Annotation 2:

```
<rdf:RDF xmlns="http://seetwiki.i3s.unice.fr/AMO.rdfs#" ... >
  <foaf:Agent rdf:about="#MichelBuffa">
    ...
```

```
    <hasRole rdf:resource="#Contributor"/>
   </foaf:Agent>
  </rdf:RDF>
```

Annotation 3:

```
  <rdf:RDF xmlns="http://seetwiki.i3s.unice.fr/AMO.rdfs#" ... >
   <foaf:Group rdf:about="#AdminGroup">
    <foaf:member>
     <foaf:Agent rdf:about="#AnnaKolomoiska"/>
    </foaf:member>
    <foaf:member>
     <foaf:Agent rdf:about="#CatherineFaron"/>
    </foaf:member>
    <hasRole rdf:resource="#Admin"/>
   </foaf:Group>
  </rdf:RDF>
```

2.1.2.2 Inferences with the Rule Base of AMO

Applied to the annotations of resources, AMO rules enable to infer the rights of the wiki users on these resources. For example, consider again Rule 1. Its premise matches with Annotation 1 that illustrates Section 2.1.2.1: the resource `TestPage` is of type `WikiArticle` – a class of the SIOC vocabulary, subclass of the class `Document` of the FOAF vocabulary – and `TestPage` is related to the user `MichelBuffa` with the `hasAuthorizedAgent` property. Applied on Annotation 1, Rule 1 allows to conclude that `MichelBuffa` has the *read*, *modify* and *delete* permissions on the content of the annotated resource `TestPage` and the *modify* permission on its type of access and its list of agents.

Similarly, Rule 2 applied on Annotation 3 allows to conclude that user `CatherineFaron` has the administrator role. Another rule of AMO (not provided here) describes general rights of an agent having the administrator role on any resource. It enables to conclude that `CatherineFaron` owns all the rights on the specific resource `TestPage`.

Finally, Rule 1 and Rule 3 applied on Annotation 1 enable to conclude that user `AnnaKolomoiska`, creator of resource `TestPage`, has the rights of an agent of that resource: *read*, *modify* and *delete* rights on its content and *modify* right on its type of access and its list of agents.

2.1.2.3 SPARQL Requests for Access Rights Management

Access to a particular resource by a given user depends, as all the actions in SweetWiki, on the answers to a SPARQL query provided by the Corese engine launched on the base of resource annotations. For this, Corese combines backward chaining on the AMO rule base and matching of queries with the annotation base. For ex-

ample, the answer to the following SPARQL query will indicate whether the user `CatherineFaron` is allowed to modify the content of the resource `TestPage`:

Query 1:

```
prefix amo: <http://sweetwiki.unice.fr/AMO.rdfs#>
ASK {
 <http://sweetwiki.unice.fr#CatherineFaron>
                    amo:hasAuthorizedAccessOnResource ?x
 ?x amo:hasActionOnResource amo:ModifyContent
 ?x amo:hasResource <http://sweetwiki.unice.fr#TestPage> }
```

Other SPARQL queries are formulated to support all the functionnalities of SweetWiki. For instance, the processing of the following query will provide the list of all the users having some rights on resource `TestPage` and for each of them it will state the list of her authorized actions on`TestPage`:

Query 2:

```
prefix amo: <http://sweetwiki.unice.fr/AMO.rdfs#>
SELECT ?agent ?action {
 ?agent amo:hasAuthorizedAccessOnResource ?x
 ?x amo:hasActionOnResource ?action
 ?x amo:hasResource <http://sweetwiki.unice.fr#TestPage> }
order by ?agent
```

2.2 Context-Aware Access Control for Semantic Social Networks

We present in this section a new access control model where access control is based on the features of the user accessing the protected data, and context has an important role in determining whether the user is granted access or not. This new model called *S4AC-PRISSMA* enhances the expressive power of the AMO model, and allows the data provider to protect in a finer-grained way her resources.

2.2.1 The Context-aware Access Control Model

In this section, we present our access control model. The access control model is built over the notion of Named Graph [9], thus supporting fine-grained access control policies, including the triple level (Enforcing permission models is an envisioned use case for RDF named graphs[6]). We rely on named graphs to avoid depending on documents (one document can serialize several named graphs, one named graph can be split over several documents, and not all graphs come from documents[7]). At conceptual level, our policies can be considered as access control conditions over

[6] `http://bit.ly/w3rdfperm`

[7] The discussion about the use of named graphs in RDF 1.1 can be found at `http://www.w3.org/TR/rdf11-concepts`

g-boxes[8] (according to W3C RDF graph terminology), with semantics mirrored in the SPARQL language.

The model is grounded on the Social Semantic SPARQL Security for Access Control Ontology (S4AC). An overview of S4AC lightweight vocabulary is provided in Figure 4. Our access control model is integrated with the models adopted in the Social Semantic Web. In particular, S4AC reuses concepts from SIOC[9], SKOS[10], WAC[11], NiceTag[12], SPIN[13], Dublin Core[14], and the access control model as a whole is grounded on further existing ontologies, such as FOAF[15] and RELATIONSHIP[16].

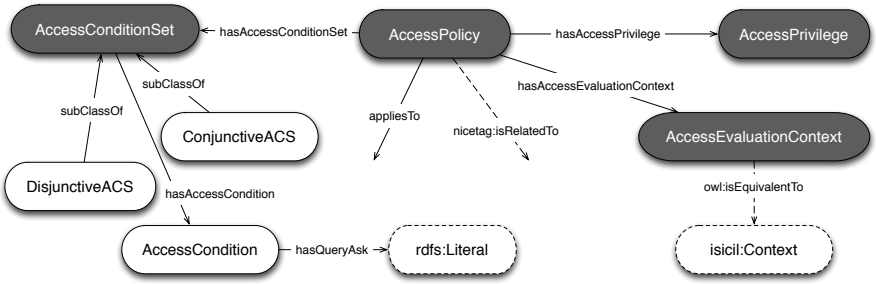

Fig. 4: An overview of the S4AC ontology. Core classes are in grey.

The main component of the S4AC model is the Access Policy, as presented in Definition 1. Roughly, an Access Policy defines the constraints that must be satisfied to access a given named graph or a set of named graphs. If the Access Policy is *satisfied* then the data consumer is allowed to access the data. Otherwise, the access is not granted. The constraints specified by the Access Policies may concern the data consumer, i.e., the user or the environment in which the user is querying the SPARQL endpoint, or any given combination of these dimensions.

Definition 1. (Access Policy) An Access Policy (P) is a tuple of the form

$$P = \langle ACS, AP, T, R, AEC \rangle$$

[8] http://bit.ly/graphterm
[9] http://rdfs.org/sioc/spec/
[10] http://www.w3.org/TR/skos-reference/
[11] http://www.w3.org/wiki/WebAccessControl
[12] http://ns.inria.fr/nicetag/2010/09/09/voc.html
[13] http://spinrdf.org/
[14] http://dublincore.org/documents/dcmi-terms/
[15] http://xmlns.com/foaf/spec/
[16] http://vocab.org/relationship/

where (i) *ACS* is a set of Access Conditions to satisfy, (ii) *AP* is an Access Privilege, (iii) *T* is the tag of the set of resources to be protected by *P*, (iv) *R* is the resource(s) to be protected by *P*, and (v) *AEC* is the Access Evaluation Context of *P*.

An Access Condition, as defined in Definition 2, expresses a constraint which needs to be verified in order to have the Access Policy satisfied. Notice that both *T* and *R* represent the data to protect. The difference is that *T* refers to the set of named graphs associated with a certain tag, and *R* refers to the URI(s) of the specific named graph(s).

Definition 2. (Access Condition) An Access Condition (*AC*) is a condition which tests whether or not a query pattern has a solution.

In the `S4AC` model, we express an Access Condition as a SPARQL 1.1 `ASK` query[17].

Definition 3. (Access Condition verification) If the query pattern has a solution (i.e., the `ASK` query returns *true*), then the Access Condition is said to be *verified*. If the query pattern has no solution (i.e., the `ASK` query returns *false*), then the Access Condition is said *not* to be *verified*.

Example 1. An example of Access Condition, which is verified only if the data consumer is a *collaborator* of the provider of the resource to protect, is the following:

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rel: <http://purl.org/vocab/relationship/>

ASK {?resource dcterms:creator ?provider.
    ?provider rel:collaboratesWith ?consumer.}
```

Each Access Policy *P* is composed by a set of Access Conditions, as defined in Definition 4.

Definition 4. (Access Condition Set) An Access Condition Set (*ACS*) is a set of access conditions of the form $ACS = \{AC_1, AC_2, \ldots, AC_n\}$.

Roughly, the verification of an Access Condition Set returns an answer of the kind *true*/*false*. We consider two standard ways to provide such an evaluation: conjunctively and disjunctively.

Definition 5. (Conjunctive Access Condition Set) A Conjunctive Access Condition Set (*CACS*) is a logical conjunction of Access Conditions of the form $CACS = AC_1 \wedge AC_2 \wedge \ldots \wedge AC_n$.

[17] `http://www.w3.org/TR/sparql11-query/#ask`

Definition 6. (Conjunctive ACS evaluation) A *CACS* is verified if and only if every contained Access Condition is verified.

Definition 7. (Disjunctive Access Condition Set) A Disjunctive Access Condition Set (*DACS*) is a logical disjunction of Access Conditions of the form $DACS = AC_1 \vee AC_2 \vee \ldots \vee AC_n$.

Definition 8. (Disjunctive ACS evaluation) A *DACS* is verified if and only if at least one of the contained Access Conditions is verified.

The second component of the Access Policy is the Access Privilege. The privilege specifies the kind of operation the data consumer is allowed to perform on the resource protected by the Access Policy.

Definition 9. (Access Privilege) An Access Privilege (*AP*) is a set of allowed operations on the protected resources of the form $AP = \{Create, Read, Update, Delete\}$.

We model the Access Privileges as four classes of operations in order to maintain a close relationship with CRUD-oriented access control systems. This relationship allows a finer-grained access control than simple read/write privileges as in WAC, and it suggests to the data providers how to specify the access privileges, following the example of CRUD-oriented systems, as we will discuss in relation to the user interface. The idea is that in the Social Semantic Web, there is a difference in allowing the users who ask to access my data to update my data or to delete my data. We distinguish the Update, Create and Delete operation to let the user to specify with a deeper degree of detail what the consumers are allowed to perform on her data. Moreover, we relate the four privilege classes to the SPARQL 1.1 query and update language. This matching is realized with the `skos:related` property through the SPIN ontology. The latter models the primitives of the SPARQL query and update languages (e.g., `SELECT`, `INSERT DATA`, etc.) as SPIN classes. We show how this matching is actually used by our framework in Section 2.2.2.

As previously explained, policies protect data at named graph level. We offer two different ways of specifying the protected object: the provider may target one or more given named graphs, or it may target a set of named graphs with a common tag. The former is achieved by providing the URI(s) of the named graph(s) to protect using the `s4ac:appliesTo` property. The latter is accomplished by listing the tags of the named graphs to protect with the property `nicetag:isRelatedTo`. In this case, the assumption is that the named graphs have been annotated with such metadata.

The Access Policy is associated to an Access Evaluation Context. The latter provides an explicit link between the policy and the actual context data (in the case of the mobile context it is modelled with `PRISSMA`) that will be used to evaluate the Access Policy.

Definition 10. (Access Evaluation Context) An Access Evaluation Context (*AEC*) is a list of predetermined bound variables of the form $AEC = (\langle var_1, val_1 \rangle, \langle var_2, val_2 \rangle, \ldots, \langle var_n, val_n \rangle)$.

In this paper, we focus on the mobile context, thus the Access Evaluation Context list is composed only by a couple $AEC = (\langle ctx, URI_{ctx} \rangle)$. We map therefore the variable *ctx*, used in the policy's Access Conditions, to the URI identifying the actual user context in which the SPARQL query has been performed. More specifically, the Access Evaluation Context is implemented as a SPARQL 1.1 `BINDINGS` Clause[18] to constrain the `ASK` evaluation, i.e. "`BINDINGS ?ctx {(`URI_{ctx}`)}`".

The choice and the design of a context model necessarily need a context definition first. We agree on the widely-accepted proposal by Dey [21]:

Definition 11. (Context) "*Context* is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [21].

More specifically, we rely on the work by Fonseca and colleagues [19], that we adopt as a foundation for our proposal. The mobile context is seen as an encompassing term, an information space defined as the sum of three different dimensions: the mobile *User* model, the *Device* features and the *Environment* in which the action is performed.
The social semantic web scenario favours the adoption of an ontology-based model. As pointed out by Korpipää and Mäntyjärvi [28], an ontological approach leads to simple and extensible models. A large number of ontology-based context models relying on Dey's definition have been proposed in the latter years, as summarized by Bolchini et al. [4] (e.g. CoBrA, CoDaMoS, SOCAM). These works are grounded on RDF and provide in-depth context expressivity, but for chronological reasons they are far from the best practices common on the social semantic web (e.g. lightweight approach, heavy interlinking with other vocabularies), thus discouraging the adoption and re-use in the Web community. Our context-aware access control framework adopts `PRISSMA`, a lightweight vocabulary originally designed for context-aware adaptation of RDF data [14]. PRISSMA has been originally designed to express the contextual conditions under which activate a given representation for RDF [14]. In this paper we propose context-based access policies, and we therefore need a vocabulary to model mobile context. We thus re-use classes and properties of the PRISSMA vocabulary for a different purpose, i.e. to represent contextual conditions for accessing RDF graphs. `PRISSMA` provides classes and properties to model core mobile context concepts, but is not meant to deliver yet another mobile contextual model: instead, well-known lightweight vocabularies and recent W3C recommendations are reused (Figure 6). Moreover, it does not provide a comprehensive, exhaustive context representation: the approach is to delegate refinements and extensions to domain specialists. The overall context is modelled by the class `prissma:Context` and is determined by the following dimensions:

[18] `http://www.w3.org/TR/sparql11-federated-query/#update`
[19] `http://bit.ly/XGR-mbui`

```
:policy1 a s4ac:AccessPolicy;                          ACCESS POLICY
         s4ac:appliesTo :alice_data;                   RESOURCE TO PROTECT
         s4ac:hasAccessPrivilege [a s4ac:Update];      ACCESS PRIVILEGE
         s4ac:hasAccessConditionSet :acs1.

:acs1 a s4ac:AccessConditionSet;
        s4ac:ConjunctiveAccessConditionSet;
        s4ac:hasAccessCondition :ac1,:ac2.             ACCESS CONDITIONS TO VERIFY

:ac1 a s4ac:AccessCondition;
       s4ac:hasQueryAsk
       """ASK {?context a prissma:Context.
               ?context prissma:user ?u.
               ?u foaf:knows ex:alice#me.}""".

:ac2 a s4ac:AccessCondition;
       s4ac:hasQueryAsk
       """ASK {?context a prissma:Context.
               ?context prissma:environment ?env.
               ?env prissma:based_near ?p.
               FILTER (!(?p=ex:ACME_boss#me))}""".
```

(a)

```
@prefix : <http://example/contextgraphs/bobCtx>
[other prefixes omitted]
<http://example/contextgraphs/bobCtx>{
:ctx a prissma:Context;                                THE CONSUMER'S CONTEXT
       prissma:user :usr;
       prissma:device :dev;
       prissma:environment :env.

:usr a prissma:User;                                   THE USER DIMENSION
       foaf:name "Bob";
       foaf:knows ex:alice#me.

:dev a prissma:Device;                                 THE DEVICE DIMENSION
       hard:deviceHardware :devhw;
       soft:deviceSoftware :devsw.
:devhw a hard:DeviceHardware;
         dcn:display hard:TactileDisplay.
:devsw a soft:DeviceSoftware;
         soft:operatingSystem :devos.
:devos a soft:OperatingSystem;
         common:name "Android".

:env a prissma:Environment;                            THE ENVIRONMENT DIMENSION
       prissma:motion "no";
       prissma:nearbyEntity :ACME_boss#me;
       prissma:currentPOI :ACMEoffice.
:ACMEoffice a prissma:POI;
             prissma:poiCategory example:Office;
             prissma:poiLabel example:ACMECorp.
}
```

(b)

Fig. 5: The Access Policy protecting `:alice_data` (a) and Bob's sample mobile context in TriG notation (b).

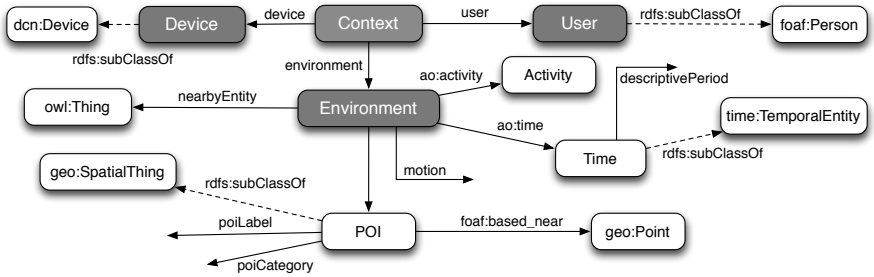

Fig. 6: The PRISSMA vocabulary at a glance.

Definition 12. (User Dimension) The *User* represents the mobile requester associated to a *Context* and consists in a `foaf:Person` sub-class. It can model both stereotypes and specific users.

Definition 13. (Device Dimension) The *Device* consists in a structured representation of the mobile device used to access the RDF store.

The `Device` class inherits from W3C Delivery Context Ontology [20] `dcn:Device`, providing an extensible and fine-grained model for mobile device features and enabling device-specific access control.

Definition 14. (Environment Dimension) The *Environment* is the model of the physical context in which the resource consumption takes place.

Different dimensions are involved in modelling the surrounding environment. Location is modelled with the notion of Point of Interest (POI). The `POI` class consists in a simplified, RDFized version of the W3C Point of Interest Core specifications[21]. Time is modelled extending the `time:TemporalEntity` class[22]. Other dimensions are considered: the `motion` property associates any given high-level representation of motion to a `Environment`. The proximity of an object might determine access restrictions: nearby objects are associated to the Environment with the `nearbyEntity` property. The `Activity` class consists in a placemark aimed at connecting third-party solutions focused on inferring high-level representations of user actions (e.g.'running', 'driving', 'shopping', etc). Further refinements and extensions are delegated to domain specialists (e.g. if dealing with indoor location, the `room` vocabulary[23] could be easily integrated).

Example 2. We now present an example of Access Policy with a conjunctive Access Condition Set associated to an `Update` privilege (Figure 5a). The policy protects the

[20] `http://bit.ly/dc-ontology`
[21] `http://www.w3.org/TR/poi-core/`
[22] `http://www.w3.org/TR/owl-time`
[23] `http://vocab.deri.ie/rooms`

named graph :alice_data and allows the access and modification of the named graph only if the consumer (i) knows Alice, and (ii) is not located near Alice's boss. Figure 5b visualizes a sample mobile context featuring all the dimensions described above. The user, Bob, knows Alice and is currently at work, near his and Alice's boss. Bob is using an Android tablet with touch display and he is not moving.

When dealing with mobile context, other issues need to be considered beyond context-model definition, such as context fetch, context trustworthiness and privacy. The present paper assumes that context data is fetched and pre-processed beforehand. PRISSMA supports both raw context data fetched directly from mobile sensors (e.g. GPS location, mobile features) and refined information processed on board or by third-party, server-side services (e.g. POI resolution or user activity detection). The trustworthiness of contextual information sent by mobile consumers should not be taken for granted. The User's identity needs to be certified: this is an open research area in the Web, and initiatives such as WebID[24] specifically deal with this issue. Hulsebosch et al. [26] provide a survey of context verification techniques (e.g. heuristics relying on context history, collaborative authenticity checks). A promising approach is mentioned in Kulkarni and Tripathi [31], where context sensors are authenticated beforehand by a trusted party. We plan to tackle the issue of context-verification in future work. Privacy concerns arise while dealing with mobile user context. We are aware that sensible data such as current location must be handled with a privacy-preserving mechanism. In the present proposition, we do not address this issue, nor the problem of context integrity.

Further details of the S4AC model include, among others: the specification of the creator of the policy (sioc:hasCreator) to keep track of this information in the Social Semantic platform, the creation date (dcterms:created), the specification of the variables used in the access conditions of the policies and their description in natural language adopted in the user interface of the framework to help the provider reusing others' policies, and a skos:prefLabel property associated to the Access Conditions to provide a sort of "explanation" to the consumer in case she cannot access the data (following the example of AIR [27]). Moreover, we are able to manage the fact that only a maximum number of accesses is granted, as in Giunchiglia et al. [24], by means of an Access Condition, and we can grant random access to a resource (e.g. ASK{FILTER(rand()>0.5)}.

The semantics of our Access Control Policies is mirrored in the semantics of the SPARQL language, in particular concerning the ASK query and the BINDINGS clause. The result of the verification of each access condition is composed, in case of multiple conditions, conjunctively or disjunctively, and this combination is the overall result of the policy evaluation. The Access Privilege and the resource to protect are components of the policy which do not concur to its verification. All the semantics of our Access Policies relies on the semantics of the ASK queries combined with the contextual BINDINGS.

Conflicts among policies might occur if data provider adds Access Conditions with contrasting FILTER clauses. For instance, it is possible to define positive

[24] http://www.w3.org/2005/Incubator/webid/spec/

and negative statements such as `ASK{FILTER(?user=<http://example#bob>)}` and `ASK{FILTER(!(?user=<http://example#bob>))}`. If these two Access Conditions are applied to the same data, a logical conflict arises. This issue is handled in our framework by evaluating policies applied to a resource in a disjunctive way. This means that in the example above, if the consumer satisfies one of the two access conditions, then the access is granted to her. This is not satisfactory in many situations, thus we expect to add a mechanism, following the example of [20], to avoid the insertion of conflicting policies as a future work.

Example 3. Consider the following scenario. Alice is attending a music festival and she uploads some content to the social platform. She prefers to share these contents to all the people knowing her but not to those who are friends of her boss. The policy (Figure 5a) protects the named graph containing Alice's reviews of concerts (`:alice_reviews` visualized in Figure 7).

```
ex:29900 a bibo:Article;
      dcterms:title "Great concert with Bob!";
      dcterms:date "2010";
      dcterms:creator example:alice#me;
      bibo:abstract "Really enjoyed Coldplay".

ex:29655 a bibo:Article;
      dcterms:title "Disappointed";
      dcterms:date "2010";
      dcterms:creator example:alice#me;
      bibo:abstract "Not up to the standards".
```

Fig. 7: The content of the named graph `:alice_reviews` containing the reviews authored by Alice.

FIgure 8 presents some examples of the `ASK` queries which may be associated to the access conditions. *Cond1* grants the access to those users who have a relationship of kind "colleagues" with the provider. *Cond2* grants the access to the friends of the provider, and *Cond3* extends this access condition also to the friends of friends. *Cond4* is more complicated[25]. It grants the access to those users that are marked with a specified tag. To specify the tag, we use again the NiceTag ontology which allows to define the relationship among the resources and the tags for each tagging action. Negative access conditions are allowed, where we specify which user cannot access the data. This is expressed, as shown in *Cond5*, by means of the `FILTER` clause, and access is granted to every user except *bob*. *Cond6* expresses an access condition where the user can access the data only if he is a minimum lucky, e.g., one chance out of two. *Cond7* provides a positive exception where only a specific user can access the data, it is the contrary of *Cond5*. *Cond8* grants the access to users who are members of a particular group, to which the provides belongs too.

[25] The GRAPH keyword is used to match patterns against named graphs.

cond1	`ASK { ?resource dcterms:creator ?provider . ?provider rel:hasColleague ?user }`
cond2	`ASK { ?resource dcterms:creator ?provider . ?provider rel:hasFriend ?user }`
cond3	`ASK { ?resource dcterms:creator ?provider . ?provider rel:hasFriend{1,2} ?user }`
cond4	`ASK { ?resource dcterms:creator ?provider . ?provider dcterms:creator ?g . GRAPH ?g { ?user nicetag:hasCommunitySign ?tag }}`
cond5	`ASK { FILTER(! (?user= <http://MyExample.net#sery>))}`
cond6	`ASK { FILTER(random()>0.5) }`
cond7	`ASK { FILTER(?user= <http://MyExample.net#bob>)}`
cond8	`ASK { ?resource dcterms:creator ?provider . ?provider sioc:member_of ?g . ?user sioc:member_of ?g }`

Fig. 8: Examples of access conditions.

2.2.2 The Access Control Manager

The Access Control Manager (ACM), visualized in Figure 9, is the core module which allows the data providers to define and check the Access Conditions.

The framework is developed in the following way:

1. the data consumer queries the SPARQL endpoint to access the content, and at the same time, the social platform sends the user information coupled with the query. This data is sent as an `INSERT DATA` statement to build the named graph representing the user's data. Summarizing, the user sends two SPARQL queries to the endpoint, the first one for accessing the datastore, and the second one for providing her personal information. A caching mechanism can be introduced here to avoid sending the personal information every time a query is performed.
2. the query of the consumer is not directly processed by the SPARQL endpoint, but it is filtered by the Access Control Manager.
3. the Access Control Manager selects the policies concerning the consumer's query, and after their evaluation, it returns the set of named graphs the consumer is granted access to.
4. the query of the consumer is processed only on the accessible named graphs.

5. the result of the query is sent to the consumer.

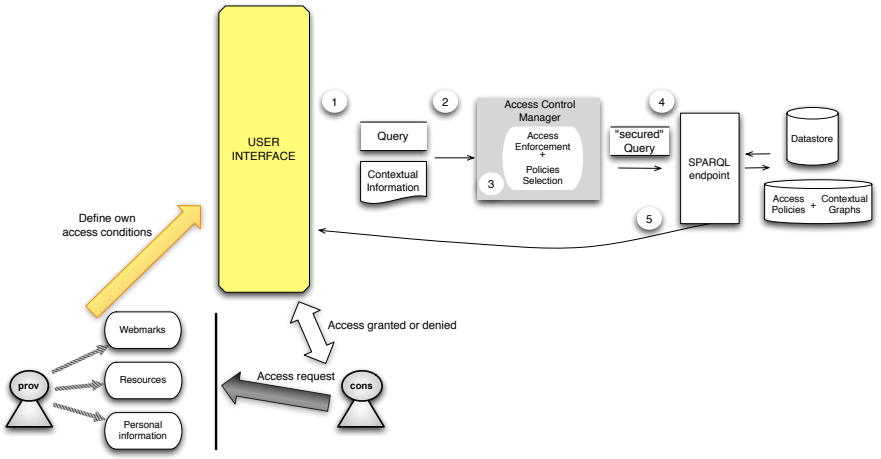

Fig. 9: The Access Control Manager.

The core of our framework is the Access Enforcement Module. The aim of this component is twofold: first, the module selects the Access Policies to assess, and second, it verifies the set of Access Conditions included in the selected policies to allow or not the access. In the following, we describe the two algorithms used to decide whether the consumer is allowed to access or not the data.

```
ASK{?context a isicil:Context.
      ?context isicil:user ?u.
      ?u foaf:knows ex:alice#me.}
      BINDINGS ?context {(example:actualCtx1)}

ASK {?context a isicil:Context.
      ?context isicil:user ?u.
      ?u rel:hasFriend ?f.
      FILTER (!(?f=ex:ACME_boss#me))}
      BINDINGS ?context {(example:actualCtx1)}
```

Fig. 10: The Access Conditions bound to the actual user context with the `BINDINGS` clause.

Algorithm 1 is the overall algorithm for the query execution with the access enforcement. The input of the algorithm is the consumer's query Q and the RDF graph G_{ctx} modeling the information of the consumer. We assume the existence of a repository of access policies *APS*. The algorithm starts by saving the consumer's

```
DELETE {ex:article dcterms:subject
        <http://dbpedia.org/page/Category: Concert_tours>. }
INSERT {ex:article dcterms:subject
        <http://dbpedia.org/page/Category: Music_performance>. }
WHERE {ex:article a bibo:Article}
```

(a)

```
DELETE {ex:article dcterms:subject
        <http://dbpedia.org/page/Category: Concert_tours>. }
INSERT {ex:article dcterms:subject
        <http://dbpedia.org/page/Category: Music_performance>. }
USING :peter_data
USING NAMED :peter_data
```

THE NAMED GRAPH ACCESSIBLE BY THE CONSUMER

```
WHERE {ex:article a bibo:Article}
```

(b)

Fig. 11: The SPARQL query issued by Bob's mobile client (a) and the *filtered* version (b).

graph in a local cache (line 1). The set of selected accessible named graph *NGS* is empty at the beginning of the algorithm execution (line 3). The selection of the Access Policies is addressed by the sub-routine Access Policies Selection (line 4), which returns the set of Access Policies the query is concerned by. Then, the algorithm runs all the Access Conditions composing the selected policies (lines 7-10). For each policy, depending on the kind of Access Conditions Set, i.e., conjunctive or disjunctive, if the policy is verified then the named graph to which the policy allows the access is added to the set of accessible named graphs (lines 11-12). Finally, after the execution of all the policies, the query of the consumer is sent to the protected SPARQL endpoint with the addition of the `FROM` and `FROM NAMED` clauses (line 16). This allows the enforcement module to execute the query only on those named graphs which are accessible, given the user information. Adding the `FROM` clause is not enough because, in case the client query includes a `GRAPH` clause, we need to specify the set of named graphs to be queried in a `FROM NAMED` clause, otherwise the query will be executed on all the named graphs of the store. `USING` and `USING NAMED` describe a dataset in the same way as `FROM` and `FROM NAMED` clauses. The keyword `USING` instead of `FROM` in update requests has been chosen to avoid possible ambiguities which could arise from writing "`DELETE FROM`". The algorithm outputs the triples resulting from Q (line 18).

Example 4. An example of client query is shown in Figure 11a, where Bob wants to update the rock festival's reviews[26]. When the query is received by the Access Control Manager, it selects the Access Policies concerning this query (Figure 5a).

[26] Notice that the client query can be every kind of query defined by the SPARQL 1.1 Query and Update language, e.g., `CONSTRUCT`, `SELECT`.

The Access Conditions composing the policy are then coupled with a `BINDINGS` clause, as shown in Figure 10, where the *?context* variable is bound to the actual Bob's information. Suppose Bob knows Alice, but it is also a friend of Alice's boss (note that these information are retrieved from Bob's FOAF profile): the Access Policy protecting Alice's named graph does not grant access to Bob. After the identification of the named graph(s) accessible by Bob (for instance, the named graph `:peter_reviews`), the Access Control Manager adds the `USING` and `USING NAMED` clauses[27] to constrain the execution of the client query only on the allowed named graphs. The "secured" client query is shown in Figure 11b.

Algorithm 2 is the Access Policies Selection routine. The aim of this algorithm is to select, starting from the consumer's query, what are the Access Policies the query is concerned with. The input of the algorithm is the query Q and the repository of the policies *APS*. The idea is that we do not want to verify all the Access Policies every time a query is run. Thus, we adopt a selection mechanism to obtain only a subset of Access Policies to execute. In particular, the algorithm maps the consumer's query to one of the four access privileges `S4AC` defines (line 1). Then, the algorithm selects all the Access Policies which have the identified Access Privilege (lines 3-7). The selected policies are returned to the main algorithm of access enforcement.

2.2.3 Evaluation

To assess the impact on response time, we implemented the Access Control Manager as a Java EE component and we plugged it to the Corese-KGRAM RDF store[28] and SPARQL 1.1 query engine[29] [12]. We evaluated the prototype on an Intel Xeon E5540, Quad Core 2.53 GHz machine with 48GB of memory, using the Berlin SPARQL Benchmark (BSBM) dataset 3.1[30].

In Figure 13, we show the execution of 10 independent runs of a test query batch consisting in 50 identical queries of a simple `SELECT` over `bsbm:Review` instances (tests are preceded by a warmup run). We measure the response time with and without access control. When executed against the Access Control Manager, the test SPARQL query is associated to the static user context. Each Access Policy contains exactly one Access Condition. In Figure 13.a, to simulate a worst-case scenario, access is granted to all named graphs defined in the base (i.e. all Access Conditions return true), so that query execution does not benefit from cardinality reduction. Larger datasets are less affected by the delay introduced by our prototype, as datastore size plays a predominant role in query execution time (e.g. for 4M triples and 100 always-true Access Policies we obtain a 32.6% response time delay).

[27] `http://www.w3.org/TR/sparql11-update/#deleteInsert`

[28] Concerning accessing inferred statements, Corese-KGRAM allows to know where are the inferred triples. In this way, we can apply to these inferred triples the same access policies that regulate the access to the triples from which these triples have been inferred.

[29] `http://www-sop.inria.fr/edelweiss/software/corese/`

[30] `http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/Dataset/`

Algorithm 1: Query Execution with Access Enforcement

Input: a SPARQL query Q, an RDF graph G_{ctx}, Access Policy Set APS
Output: the SPARQL query result R
save G_{ctx} in local contextual cache;
if G_{ctx} *has changed* **then**
  $NGS = \emptyset$;
  $APS \leftarrow$ `APSelection(`Q,APS`)`;
  forall the $AP_i \in APS$ **do**
    $ACcount_{false} = 0$;
    forall the $AC_j \in ACS_i$ **do**
      append G_{ctx} to AC_j as `BINDINGS` clause;
      if `ASK`$_{AC_j}$ *execution returns false* **then**
        $ACcount_{false}++$;
    if (ACS_{AP_i} *is DACS and* $ACcount_{false} < |ACS_{AP_i}|$)||($ACS_{AP_i}$ *is CACS and* $ACcount_{false} = 0$ **then**
      $NGS \leftarrow NGS \cup NG_{AP_i}$;
else
  $NGS \leftarrow NGS_{cached}$;
forall the $NG_i \in NGS$ **do**
  append `FROM <`NG_i`>`, `FROM NAMED<`NG_i`>` to Q;
  append `USING <`NG_i`>`, `USING NAMED<`NG_i`>` to Q;
$R \leftarrow$ run Q;
return R;

Algorithm 2: Access Policies Selection

Input: SPARQL client query Q, APS
Output: a reduced set of Access Policies APS_r
$AccPrv_Q \leftarrow$ map Q type to CRUD operation;
$APS_r = \emptyset$;
forall the $AP_i \in APS$ **do**
  if $AccPrv_{AP_i} \equiv AccPrv_Q$ **then**
    $APS_r \leftarrow APS_r \cup AP_i$;
return APS_r;

Fig. 12: SPARQL Query Execution Procedure

In a typical scenario, the Access Control Manager restricts the results of a query. In Figure 13.b, we assess the impact on performance for various levels of cardinality reduction, using modified versions of the BSBM dataset featuring a larger amount of named graphs (we define a higher number of `bsbm:RatingSites`, thus obtaining more named graphs). When access is granted to a small fraction of named graphs, the query is executed faster than the case without access control (e.g. if access is granted to only 1% of named graphs, the query is executed 19% faster on the 1M triple test dataset). As more named graphs and triples are accessible, performance decreases. In particular, response time is affected by the construction of the active

graph, determined by the merge of graphs in the `FROM` clauses. As shown in Figure 13.b, the cost of this operation grows with the number of named graphs returned by the evaluation of the Access Policies.

In Figure 13.c, we analyze the overhead introduced on response time by queries executed in dynamic user environments. We execute independent runs of 100 identical `SELECT` queries, dealing with a range of context change probabilities. In case of a context update, the query is coupled with a SPARQL 1.1 `DELETE/INSERT` (i.e. update) of the context graph. Not surprisingly, with higher chances of updating the context, the response time of the query grows, since more SPARQL queries need to be executed. The delay of `INSERT DATA` or `DELETE/INSERT` operations depends on the size of the triple store and on the number of named graphs (e.g. after a `DELETE` query, the adopted triple store refreshes internal structures to satisfy RDFS entailment). Performance is therefore affected by the number of active consumers, since each of them is associated to a user context graph.

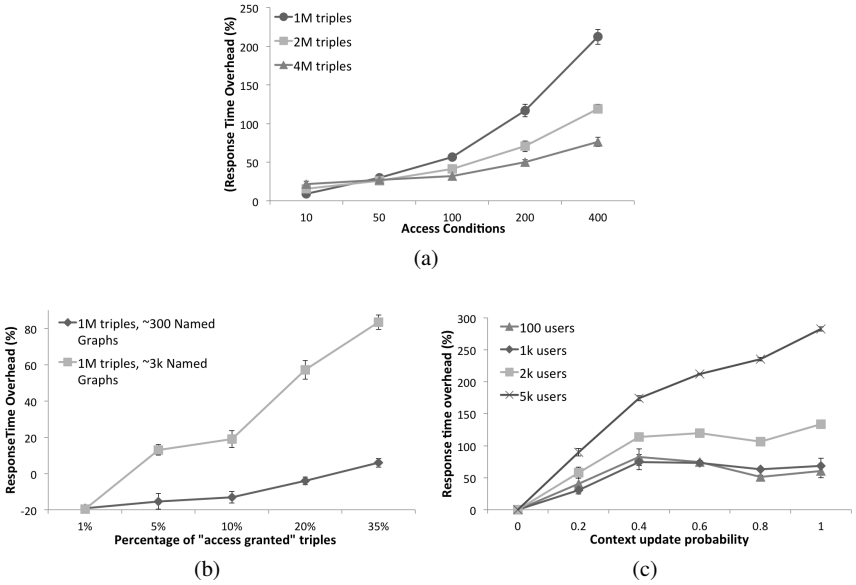

Fig. 13: Response time overhead

3 Related Work

3.1 XML Languages for Access Control and Digital Rights

Most of the mechanisms of access control implemented in content management systems are based on XML languages dedicated to the description of policies of access control and digital rights management (DRM). These systems exploit the metadata associated with resources to which access must be controlled and these metadata comply with the XML schemas of these dedicated languages. Among these languages, the most famous are XrML[31] (Right eXtensible Markup Language) used as the basic language of expression rights of MPEG-21[32], ODRL[33](Open Digital Right Language) implemented by the Open Mobile Alliance (OMA) and XACML[34] (Extensible Access Control Markup Language) developed by OASIS. The ODRL model is based on the concepts of *Asset*, *Party*, *Permission*, *Constraint*, *Requirement*, *Condition*, *Rights holder*, *Context*, *Offer*, *Agreement* and *Revoking rights*. The XACML model allows to represent access control policies by rules. It is based on the concepts of *Rule*, *Policy* and *Policy Set* and these concepts can be refined with those of *Subject*, *Resource*, *Action*, *Environment*. A *Rule* comprises *Conditions* and *Effects* and a *Policy* embeds *Rules* and *Obligations*.

3.2 Semantic Approaches to Access Control

With the emergence of the Web of data and people, new approaches to manage access to content have emerged based on semantic Web models and technologies. Notably [2] shows the limitations of solutions using non-semantic description languages for managing access rights. They propose an OWL ontology to describe the access to web services inspired from the XACML model. More generally, in the few existing semantic models for managing access to content, we recognize some concepts that were already present in the older XML languages.

The W3C initiative is also noticeable: it uses since 2001 an RDF-based system to control access to the files of its servers: W3C ACL System[35]. [25] proposed an evolution of this system to a scalable system that allows for decentralized user authorization via an RDF metadata file containing an access control list (ACL). The ontology used in this system is called Basic Access Control Ontology[36]. It is presented as a basis to develop more sophisticated models. Our models extend it for allowing the construction of more fine-grained access control policies. With the

[31] http://www.xrml.org/

[32] http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm

[33] http://www.w3.org/TR/odrl/

[34] http://www.oasis-open.org/committees/xacml/

[35] http://www.w3.org/2001/04/20-ACLs

[36] http://www.w3.org/ns/auth/acl

AMO ontology we propose a rule-based access control, at document level (rather than directory level). With S4AC-PRISSMA, we provide a fine-grained access control model which grants access to specific RDF data, i.e. the data provider may want to restrict the access to a few named graphs. Moreover, we let consumers submit any SPARQL query, and the user information takes part into the evaluation.

Requirements in terms of access rights management in social platforms like ISICIL are similar to those of digital libraries which [18] propose an overview. However, one of the key issues for digital libraries is not relevant in the context of ISICIL: the respect of the copyrights of available documents and for this purpose the protection of documents by DRM. Indeed the documents handled by the watchers remain in the corporate intranet or are public documents on the web. Among the work on access management in digital libraries, we notice those of [32] on the Fedora architecture for managing digital resources and those of [30] on the semantic Digital Library JeromeDL.

The Fedora authors propose a model called DARS (acronym for Distributed Active Relationships) for associating metadata to objects in a digital library, especially for managing access rights. However, although part of the model of access management is thus in an ontology, the Fedora system also uses XACML metadata associated with resources the it handles.

Access management in JeromeDL is based on the EAC ontology[37] (acronym for Extensible Access Control) [29]. EAC enables to associate licenses to resources, for each license corresponding to an access policy. For example, a license can specify that only people of a given organization can access some resources of the library. The purpose of EAC is to filter access to resources while that of AMO is to define access rights associated to user roles.

Approaches of access control based on annotations of resources are particularly well suited for social platforms. For example, in [34] end users are able to annotate by tagging both resources and members of their social network. Access control policies are then based on these annotations. For example, a basic policy states that if a resource shares the same tag as a member of the social network, this member has access to the resource. This user-centric approach is more flexible than role-centric access control since no real role nor actions need to be defined. It does not require an administrator user having global maitenance access rights to the system. Therefore, it seems more dedicated to the mananagement of personnal data rather than public shared data with many contributors (like in wikis). However, we plan to investigate how we could combine such a user-centric approach with our role-centric approach.

Sacco and Passant [35, 36] present a Privacy Preference Ontology (PPO[38]), built on top of WAC, to express fine-grained access control policies to an RDF file. In their approach, the consumer asks to access a particular RDF file, e.g., a FOAF profile. Their access control manager selects the part of the file the consumer can access, and returns it to the consumer. They do not propose an access control filter for generic SPARQL endpoints, mapping the queries with the access policies. They

[37] http://www.jeromedl.org/eac/1.0/spec/index.html

[38] `http://vocab.deri.ie/ppo`

also specify access queries with SPARQL `ASK`s, but the PPO vocabulary does not consider contextual information. They rely entirely on the WAC vocabulary without distinguishing different kinds of `Write` actions, and they cannot specify conjunctive and disjunctive sets of privacy preferences.

Muhleisen et al. [33] present a policy-enabled server for Linked Data called PeLDS, where the access policies are expressed using a descriptive language called PsSF, based on SWRL[39]. They distinguish only `Read` and `Update` actions, and they do not consider contextual information. The system is based on an ontology of the actions that can be performed on the datasets, but no further description is provided.

Giunchiglia et al. [24] propose a Relation Based Access Control model (*RelBAC*), a formal model of permissions based on description logic. They require to specify who can access the data, while in our framework and in [36] the provider specifies the attributes the consumer must satisfy.

Finin at al. [22] study how to represent RBAC using the OWL language. The authors show also the representation of policies based on general attributes of an action, similarly to what we present in this paper. The difference is that we specify the policies using SPARQL 1.1 ASK queries, where the Bindings clause is used to specify the values of the variables, and temporal and spatial constraints may be expressed too.

Abel et al. [1] present a model of context-dependent access control at triple level, where also contextual predicates are allowed, e.g., related to time, location, credentials. The policies are not expressed using Semantic Web languages, but they introduce a high level syntax then mapped to existing policy languages. They enforce access control as a layer on top of RDF stores. They pre-evaluate the contextual conditions, then the queries are expanded, and sent to the database.

Flouris et al. [23] present a fine-grained access control framework on top of RDF repositories. As in our approach, the authors underline the need of a fine-grained access control framework while being repository independent. Differently from our framework, they do not consider the contextual dimension, and they propose a high level specification language which has to be translated into a SPARQL/SerQL/SQL query to enforce the policy, while we use directly SPARQL 1.1 to specify the policies. Moreover, they focus only on read operations.

Carminati et al. [8] propose a fine-grained on-line social network access control model based on semantic web technologies. Their main idea is to encode social network-related information by means of an ontology. By constructing such an ontology, the authors model the Social Network Knowledge Base. They assume that a centralized reference monitor hosted by the social network manager will enforce the required policies. The access control policies are encoded as SWRL[40] rules. This approach is also based on the specification of who can access the resources, i.e., the access request is a triple (u, p, URI), where the user u requests to execute privilege p on the resource located at URI.

[39] `http://www.w3.org/Submission/SWRL/`

[40] http://www.w3.org/Submission/SWRL/

Stroka et al. [40] present a preliminary proposal about securing the collaborative content on the platform KiWi. They consider global permissions, individual content item permissions, and RDF type based permission management. They do not specify the kind of access polices they can define.

3.3 Social Semantic Web Standards

A key specificity of the Semantic Web based approaches we adopt is to be interoperable with the models of the social Web and semantic Web. Specifically, SweetWiki uses FOAF and SIOC to annotate resources and AMO complements these ontologies to manage access to content. FOAF[41] (acronym for Friend Of A Friend) is an RDF vocabulary used in social networks to describe people and and the relations among them. SIOC[42] (acronym for Semantically-Interlinked Online Communities) is another RDF vocabulary that models the concepts of social web applications: forums, blogs, wikis. It reuses some concepts from FOAF and other popular ontologies (Dublin Core, SKOS, etc.) and it has established itself as the standard. It is now integrated into numerous applications such as the WordPress blog engine and its adoption within the *Linked Data*[43] project confirms its popularity.

FOAFRealm is an extension of FOAF proposed to collaboratively filter access to resources based on on user profiles and their relationships in a social network. This vocabulary is used for example in JeromeDL for filtering based on measures of trust in a social network. Such filtering may be complementary to access control allowed by AMO, based on user roles and types of access to resources.

Finally, the problem of authorizing access to resources which AMO adresses is related to the problem of authentication of agents Our approaches should be compliant with the FOAF-SSL protocol[39].

3.4 Context-oriented Access Control

A significant number of works in various research areas deal with context-aware access control. We describe first a list of proposals not related to the Web, followed by works targeting the Web and its evolution, the Web of Data.

Hulsebosch et al. [26] propose a context-sensitive access control infrastructure enriched by the verification of user-provided context information. Their work shows that context sensitive access control improves classic access control by imitating real-world authorization procedures. They provide a comprehensive overview of context verification techniques.

---

[41] http://xmlns.com/foaf/spec/

[42] http://sioc-project.org/ontology

[43] http://linkeddata.org/

Bertino et al. [3] limit on location awareness and discuss the motivations behind enriching access control with position data. The location verification problem is presented, along with solutions (e.g. authenticator devices for physical location of users). Their approach relies on geographical role-based access control (GEO-RBAC), where users are assigned roles with given spatial validity.

Another contextual role-based approach is presented in Kulkarni and Tripathi [31]. The authors propose a context management layer in charge of authenticating the sensors that produce context data used to evaluate access.

Shen and Cheng [38] propose a model called SCBAC which combines Semantic Web technologies with a context-based access control mechanism. Policies are expressed using SWRL[44]. The authors consider four types of contexts: subject contexts (our User and Device dimensions), object contexts, transaction contexts (our access privilege) and environment contexts (our Environment dimension). They do not apply their model to the social semantic web.

Covington et al. [16] present an approach where the notion of role proposed by RBAC [37] is used to capture the context of the environment in which the access requests are made. The environmental roles are defined using a prolog-like logical language for expressing policies. In a subsequent work, Covington proposes a context-aware attribute-based access control model called CABAC [17]. They heavily rely on contextual attributes.

Cuppens and Cuppens-Boulahia [19] propose an Organization Based Access Control (OrBAC) model where also contextual conditions are expressed. Contextual conditions are considered as extra statements to be satisfied to activate a security constraint and are based on Datalog rules. A context algebra is introduced. The main difference is that we entirely rely on Semantic Web languages. Moreover, we consider additional context data, beyond the temporal and spatial dimensions.

Corradi et al. [13] present UbiCOSM, a security middleware adopting context as a basic concept for policy specification and enforcement. As we do, the authors consider context as a first-class design principle to control access to resources. They distinguish among physical (i.e., physical spaces) and logical contexts (e.g., temporal conditions, user activities). We add further contextual dimensions, e.g., the device. Policies are expressed at a high level of abstraction in terms of RDF metadata. Their approach is not applied to the Web of Data.

Toninelli et al. [42, 43] follow two design guidelines, which inspire also the framework proposed in this paper: context-awareness to control resource access and semantic technologies for context and policy specification. They adopt spontaneous coalitions as an application scenario, while we deal with the Web of Data. Moreover, the semantic technology adopted differs, i.e., rule-based approach with description logic in their case and SPARQL 1.1 in our proposal. Their contextual information does not include the device dimension. Finally, their solution is not meant to be a pluggable framework for SPARQL endpoints. In a separated work, Toninelli et al. [41] present the Proteus policy framework and discuss the role of the quality of

[44] `http://www.w3.org/Submission/SWRL/`

context in access control systems. In this paper, we do not take into account this problem that is left as future work.

	Adopted languages	CRUD	Context awareness	Granularity	Performance evaluation
WAC	RDF	Read/Write	NO	RDF document	NO
Sacco and Passant	RDF/SPARQL	Read/Write	NO	Part of RDF document	NO
Muhleisen et al.	SWRL	Read/Write	NO	RDF document	NO
Giunchiglia et al.	DL		NO		NO
Finin et al.	OWL/RDF		NO		
Hollenbach et al.	RDF		NO	RDF document	NO
Abel et al.	High level syntax		YES	Triples	YES
Cuppens and Cuppens-Boulahia	Datalog		YES but only temporal and spatial		NO
Corradi et al.	RDF		YES	Resources	NO
Toninelli et al.	DL		YES	Resources	YES
Flouris et al.	High level syntax	Read	NO	Triples	YES
Carminati et al.	SWRL		NO	Resources	NO
Stroka et al.	RDF		NO	Resources	NO

Fig. 14: A summarizing table about the related work.

4 Summary

Access control is a fundamental issue in the field of the Social Semantic Web. The users aim at protecting their resources to grant the access to their data only to authorized users. In this paper, we present two semantic access control frameworks we introduced.

First, we have presented an ontology based model for access control. This model is grounded on the AMO vocabulary for representing access rights to resources and on a base of inference rules using this vocabulary to represent access control

strategies. It has been implemented and experienced to manage access to resources in the SweetWiki engine.

Second, we have introduced a fine-grained access control model for the Social Semantic Web. This model is grounded on the `S4AC` vocabulary which allows the users of social networks to define Access Conditions for their data. In particular, these Access Conditions are implemented as SPARQL 1.1 `ASK` queries, and they can be either conjunctively or disjunctively evaluated. Moreover, Access Policies can be constrained w.r.t. the set of tags the resources are tagged with. The Access Evaluation Context provides the mapping, implemented as a `BINDINGS` clause, between the information about the consumer and the Access Conditions. We have presented our Access Control Manager, realized in the context of the ISICIL social platform. The manager grants or denies access to the users. Through an interface which allows also non-experts to interact with the system, users can specify the Access Policies to protect their data. The manager looks for the policies which apply to the resource, and after checking the contextual constraints, if present, and the features of the consumer, it states whether the access is granted or not.

There are several lines to follow for future work. First of all, in this paper we assume that the user's information is trustworthy. The trustworthiness of the information sent by consumers should not be taken for granted. The `User`'s identity needs to be certified: this is an open research area in the Web, and initiatives such as WebID[45] specifically deal with this issue. Hulsebosch et al. [26] provide a survey of context verification techniques (e.g. heuristics relying on context history, collaborative authenticity checks). At the same time, also the other contextual information like the time of the query or the location of the consumer when she is querying the SPARQL endpoint have to be checked. We plan to follow the example of Toninelli et al. [41]. Second, a user evaluation campaign is needed to evaluate our access control framework. The campaign aims at establishing the understandability of the framework, the definition of the access policies from non-expert users, the explanation of the result after the attempt to access the data, and many other features of our framework. Third, as ubiquitous connectivity grows, access control in the Social Semantic Web must not ignore the mobile context in which data consumption takes place. We are currently working on a mobile access control framework which will introduce also the *device* dimension to our user's information.

The two proposals presented in this paper highlight the expressive power a semantic-based approach to access control adds to the usual approaches. However, our frameworks represent the first step towards a fully semantic approach to network-based access control. Several issues still need to be addressed in this context, as mentioned above, like privacy preservation, consumer's trustworthiness assessment, and user-friendly interfaces definition.

[45] `http://www.w3.org/2005/Incubator/webid/spec/`

References

1. Fabian Abel, Juri Luca De Coi, Nicola Henze, Arne Wolf Koesling, Daniel Krause, and Daniel Olmedilla. Enabling Advanced and Context-Dependent Access Control in RDF Stores. In *6th Int. Semantic Web Conference (ISWC)*, volume 4825 of *LNCS*, pages 1–14. Springer, 2007.
2. A. Alam, G. Subbiah, B.M. Thuraisingham, and L. Khan. Reasoning with Semantics-aware Access Control Policies for Geospatial Web Services. In *3rd ACM Workshop On Secure Web Services (SWS)*, pages 69–76. ACM, 2006.
3. Elisa Bertino and Michael S. Kirkpatrick. Location-Aware Authentication and Access Control. In *IEEE 23rd Int. Conf. on Advanced Information Networking and Applications (AINA)*, pages 10–15. IEEE Computer Society, 2009.
4. Cristiana Bolchini, Carlo Curino, Elisa Quintarelli, Fabio A. Schreiber, and Letizia Tanca. A Data-Oriented Survey of Context Models. *SIGMOD Record*, 36(4):19–26, 2007.
5. John Breslin, Alexandre Passant, and Stefan Decker. *The Social Semantic Web*. Springer, 2009.
6. Michel Buffa and Catherine Faron-Zucker. Ontology-based access rights management. In *Advances in Knowledge Discovery and Management*, volume 398 of *Studies in Computational Intelligence*, pages 49–61. Springer, 2012.
7. Michel Buffa, Fabien L. Gandon, Guillaume Erétéo, Peter Sander, and Catherine Faron. Sweet-Wiki: A Semantic Wiki. *J. of Web Semantics*, 6(1):84–97, 2008.
8. Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani M. Thuraisingham. Semantic web-based social network access control. *Computers & Security*, 30(2-3):108–115, 2011.
9. Jeremy J. Carroll, Christian Bizer, Patrick J. Hayes, and Patrick Stickler. Named Graphs. *J. of Web Semantics*, 3(4):247–267, 2005.
10. Olivier Corby, Rose Dieng-Kuntz, and Catherine Faron-Zucker. Querying the Semantic Web with Corese Search Engine. In *16th Eureopean Conference on Artificial Intelligence (ECAI)*, pages 705–709. IOS Press, 2004.
11. Olivier Corby, Rose Dieng-Kuntz, Catherine Faron-Zucker, and Fabien L. Gandon. Searching the Semantic Web: Approximate Query Processing Based on Ontologies. *IEEE Intelligent Systems*, 21(1):20–27, 2006.
12. Olivier Corby and Catherine Faron-Zucker. The KGRAM Abstract Machine for Knowledge Graph Querying. In *Web Intelligence*, pages 338–341. IEEE, 2010.
13. Antonio Corradi, Rebecca Montanari, and Daniela Tibaldi. Context-Based Access Control Management in Ubiquitous Environments. In *3rd IEEE International Symposium on Network Computing and Applications (NCA)*, pages 253–260. IEEE Computer Society, 2004.
14. Luca Costabello. DC Proposal: PRISSMA, Towards Mobile Adaptive Presentation of the Web of Data. In *Doctoral Consortium, 10th Int. Semantic Web Conference (ISWC)*, volume 7032 of *LNCS*, pages 269–276. Springer, 2011.
15. Luca Costabello, Serena Villata, Nicolas Delaforge, and Fabien L. Gandon. Ubiquitous Access Control for SPARQL Endpoints: Lessons Learned and Future Challenges. In *WWW (Companion Volume)*, pages 487–488, 2012.
16. Michael J. Covington, Wende Long, Srividhya Srinivasan, Anind K. Dey, Mustaque Ahamad, and Gregory D. Abowd. Securing context-aware applications using environment roles. In *6th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 10–20, 2001.
17. Michael J. Covington and Manoj R. Sastry. A Contextual Attribute-Based Access Control Model. In *Workshops On the Move to Meaningful Internet Systems (OTM)*, volume 4278 of *LNCS*, pages 1996–2006, 2006.
18. K. Coyle. Rights Management and Digital Library Requirements. *Ariadne*, 40, 2004.
19. Frédéric Cuppens and Nora Cuppens-Boulahia. Modeling Contextual Security Policies. *Int. J. of Information Security*, 7(4):285–305, 2008.
20. Frédéric Cuppens, Nora Cuppens-Boulahia, and Meriam Ben Ghorbel. High Level Conflict Management Strategies in Advanced Access Control Models. *Electr. Notes Theor. Comput. Sci.*, 186:3–26, 2007.

21. Anind K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.
22. Timothy W. Finin, Anupam Joshi, Lalana Kagal, Jianwei Niu, Ravi S. Sandhu, William H. Winsborough, and Bhavani M. Thuraisingham. ROWLBAC: Representing Role based Access Control in OWL. In *13th ACM Symposium on Access Control Models and Technologies*, pages 73–82. ACM, 2008.
23. Giorgos Flouris, Irini Fundulaki, Maria Michou, and Grigoris Antoniou. Controlling Access to RDF Graphs. In *3rd Future Internet Symposium (FIS)*, volume 6369 of *LNCS*, pages 107–117. Springer, 2010.
24. Fausto Giunchiglia, Rui Zhang, and Bruno Crispo. Ontology Driven Community Access Control. In *1st Workshop on Trust and Privacy on the Social and Semantic Web (SPOT)*, 2009.
25. James Hollenbach, Joe Presbrey, and Tim Berners-Lee. Using RDF Metadata to Enable Access Control on the Social Semantic Web. In *Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK)*. CEUR-WS.org, 2009.
26. R. J. Hulsebosch, Alfons H. Salden, Mortaza S. Bargh, P. W. G. Ebben, and J. Reitsma. Context Sensitive Access Control. In *10th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 111–119. ACM, 2005.
27. Ankesh Khandelwal, Jie Bao, Lalana Kagal, Ian Jacobi, Li Ding, and James A. Hendler. Analyzing the AIR Language: A Semantic Web (Production) Rule Language. In *Web Reasoning and Rule Systems, 4th Int. Conference (RR)*, volume 6333 of *LNCS*, pages 58–72. Springer, 2010.
28. Panu Korpipää and Jani Mäntyjärvi. An Ontology for Mobile Device Sensor-Based Context Awareness. In *Modeling and Using Context, 4th Int. and Interdisciplinary Conference (CONTEXT)*, volume 2680 of *LNCS*, pages 451–458. Springer, 2003.
29. S. R. Kruk. *Extensible Access Control (EAC) Ontology Specification*, 2008. DERI, http://www.jeromedl.org/eac/1.0/spec/index.html/.
30. S. R. Kruk, M. Cygan, and A. Gzella. JeromeDL - Semantic and Social Technologies for Improving User Experience in Digital Libraries. In *World Wide Web Conference, WWW 2008*. ACM, 2008.
31. Devdatta Kulkarni and Anand Tripathi. Context-Aware Role-based Access Control in Pervasive Computing Systems. In *13th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 113–122. ACM, 2008.
32. C. Lagoze, S. Payette, E. Shin, and C. Wilper. Fedora: an Architecture for Complex Objects and their Relationships. *Int. J. on Digital Libraries*, 6(2):124–138, 2006.
33. Hannes Muhleisen, Martin Kost, and Johann-Christoph Freytag. SWRL-based Access Policies for Linked Data. In *2nd Workshop on Trust and Privacy on the Social and Semantic Web (SPOT)*. CEUR-WS.org, 2010.
34. P. Nasirifard, V. Peristeras, C. Hayes, and S. Decker. Extracting and Utilizing Social Networks from Log Files of Shared Workspaces. In *10th IFIP Working Conference on Virtual Enterprises, (PRO-VE)*, pages 643–650, 2009.
35. Owen Sacco and Alexandre Passant. A Privacy Preference Manager for the Social Semantic Web. In *2nd Workshop on Semantic Personalized Information Management: Retrieval and Recommendation (SPIM)*, 2011.
36. Owen Sacco and Alexandre Passant. A Privacy Preference Ontology (PPO) for Linked Data. In *Linked Data on the Web Workshop (LDOW)*. CEUR-WS.org, 2011.
37. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.
38. Haibo Shen and Yu Cheng. A Semantic Context-Based Model for Mobile Web Services Access Control. *Int. J. Computer Network and Information Security*, 2011.
39. Henry Story, Bruno Harbulot, Ian Jacobi, and Mike Jones. FOAF+TLS: RESTful Authentication for Distributed Social Networks. In *1st Workshop on Trust and Privacy on the Social and Semantic Web (SPOT)*. CEUR-WS.org, 2009.
40. Stephanie Stroka, Sebastian Schaffert, and Tobias Burger. Access Control in the Social Semantic Web - Extending the idea of FOAF+SSL in KiWi. In *2nd Workshop on Trust and Privacy on the Social and Semantic Web (SPOT)*. CEUR-WS.org, 2010.

41. Alessandra Toninelli, Antonio Corradi, and Rebecca Montanari. A Quality of Context-Aware Approach to Access Control in Pervasive Environments. In *2nd Int. Conf. on Mobile Wireless Middleware, Operating Systems, and Applications (MOBILWARE)*, volume 7 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 236–251. Springer, 2009.
42. Alessandra Toninelli, Rebecca Montanari, Lalana Kagal, and Ora Lassila. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In *5th International Semantic Web Conference (ISWC)*, volume 4273 of *LNCS*, pages 473–486. Springer, 2006.
43. Alessandra Toninelli, Rebecca Montanari, Lalana Kagal, and Ora Lassila. Proteus: A Semantic Context-Aware Adaptive Policy Model. In *8th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 129–140. IEEE Computer Society, 2007.
44. Serena Villata, Nicolas Delaforge, Fabien Gandon, and Amelie Gyrard. An Access Control Model for Linked Data. In *7th Int. IFIP Workshop on Semantic Web & Web Semantics (SWWS)*, volume 7046 of *LNCS*, pages 454–463. Springer, 2011.
45. Serena Villata, Nicolas Delaforge, Fabien Gandon, and Amelie Gyrard. Social Semantic Web Access Control. In *4th Int. Workshop Social Data on the Web (SDoW)*, pages 48–59, 2011.