Linked Data Access Goes Mobile: Context-Aware Authorization for Graph Stores

Luca Costabello, Serena Villata, Nicolas Delaforge, Fabien Gandon
INRIA Sophia Antipolis, France
firstname.lastname@inria.fr

ABSTRACT

To encourage data providers to publish a maximum of data on the Web, we propose a mechanism to define lightweight access control policies for graph stores. Influenced by the steep growth of the mobile web, our Linked Data access control framework features context-aware control policies. The proposed framework is exclusively grounded on standard Semantic Web languages. The framework architecture is designed as a pluggable filter for generic SPARQL endpoints, and it has been evaluated on a test dataset.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Design, Algorithms

Keywords

Linked Data, Ubiquitous Web, Access Control

1. INTRODUCTION

Denying or allowing access to a set of resources or services is a common problem in a large number of mobile computing fields, from location-based services to personal area networks. As ubiquitous connectivity spreads, access control has been enhanced with location awareness and, to some extent, other contextual dimensions such as the proximity of nearby people or objects. The open nature of current Web of Data information and the consumption of web resources on the go may give providers the impression that their content is not safe, thus preventing further publication of datasets, at the expense of the growth of the Web of Data itself [14]. Access control is therefore necessary, and context must be part of the access control evaluation, given that such *Ubiquitous Web of Data* enables new linked data fruition scenarios.

In this paper we address the problem of defining an access control framework for querying Web of Data servers from mobile environments. Let us consider a content sharing service compliant with the Web of Data: Alice uploads some pictures together with the reviews of a rock concert to the platform. She prefers to share these media to everyone but her boss. Since her colleagues might view the content at work with their smartphones, moving from office to office, she decides that nobody is allowed to access the shared media from a mobile device if the boss is in the same room.
Such application scenario raises three major challenges: (i) how to define a fine-grained access control model for the Web of Data, (ii) how to model context-aware, mobile consumption of such information, and (iii) how to integrate mobile context in the access control model, providing an evaluation of the overall framework. We answer these questions adopting exclusively Web of Data languages and reusing, when possible, already existing proposals, to avoid re-inventing the wheel.
First, we describe the `S4AC`[1] vocabulary, a lightweight ontology which defines fine-grained access control policies for RDF data [23]. We adopt the `PRISSMA`[2] vocabulary to model the mobile context in which linked data consumption takes place. Third, we combine the access control model and the contextual vocabulary into context-aware access conditions defined by data providers. Prototype evaluation shows that contextual access control comes with a cost, but performance still remains acceptable for most Web of Data applications. The main advantage of our proposal is to provide a pluggable and easy-to-integrate filter for generic SPARQL endpoints, without modifying the endpoint itself. We rely on W3C recommendations only, as we do not introduce any new language or technology. For the time being, our framework assumes the trustworthiness of the information sent by the mobile consumer, including data describing context (e.g. location, device features, etc). Our approach focuses only on SPARQL data servers. Other Web of Data access strategies, such as dereferencing resources, are out of the scope of this work.
The reminder of the paper is organized as follows. Section 2 compares the related work to the proposed framework. Section 3 introduces the mobile context aspects. Section 4 describes the access control model, while the access enforcement algorithm is detailed in Section 5. Section 6 shows the experimental results of the prototype implementation of the framework.

LDOW2012 April 16, 2012, Lyon, France.

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

2. RELATED WORK

The Web Access Control vocabulary (WAC[3]) allows data providers to specify access control lists defined at RDF document granularity (we grant access to specific RDF data, e.g. a few named graphs). Sacco and Passant [20] present a Privacy Preference Ontology (PPO[4]) to express fine-grained access control policies to an RDF file. The consumer asks for a particular RDF file, e.g., a FOAF profile and the system selects and returns the accessible part of the file. They do not propose a filter for generic SPARQL endpoints, nor they consider contextual information. Muhleisen et al. [19] present a policy-enabled server for Linked Data called PeLDS, based on SWRL[5]. They deal only with `Read` and `Update` actions and they do not consider contextual information. Giunchiglia et al. [13] propose a Relation Based Access Control model (*RelBAC*). They require to specify who can access the data, while we and [20] specify the attributes the consumer must satisfy. Finin et al. [10] study the relationship between OWL and Role Based Access Control (RBAC). To go beyond RBAC, they consider Attribute Based Access Control where, similarly to our proposal, access constraints are based on general attributes of an action. Hollenbach et al. [15] present a system where providers control the access to RDF documents using WAC, but they do not rely on the consumer's context. Abel et al. [1] present a model of context-dependent access control at triple level. Policies are not expressed using Semantic Web languages, instead they introduce an high-level syntax mapped to existing policy languages, enforcing access control as a layer on top of RDF stores. They pre-evaluate the contextual conditions before expanding the queries sent to the database. Shen and Cheng [21] propose a context-based access control model using Semantic Web technologies, where policies are expressed using SWRL. They consider four types of contexts: subject (our *User* and *Device* dimensions), object, transaction (our *Access Privilege*) and environment (our *Environment* dimension). They do not apply their model to the Web of Data. Covington et al. [7] use the notion of role proposed by RBAC to capture the context of the environment in which the access requests are made. Environmental roles are defined using a prolog-like logical language for expressing policies. Hulsebosch et al. [16] propose context-sensitive verification methods aimed at checking the authenticity of the user's information. Cuppens and Cuppens-Boulahia [8] propose an Organization Based Access Control (OrBAC) model where contextual conditions have to be satisfied to activate a security rule. They introduce a context algebra whereas we rely on Semantic Web languages. Moreover, we deal with a wider range of contextual dimensions. Corradi et al. [5] present UbiCOSM, a security middleware adopting context for policy specification and enforcement. They distinguish between physical and logical contexts while we consider additional contextual dimensions, e.g., the device. Policies are expressed at a high level of abstraction in terms of RDF metadata. Their approach does not apply to the Web of Data. Toninelli et al. [22] follow two design guidelines: 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. Flouris et al. [11] present a fine-grained access control framework on top of RDF repositories. Both their framework and our proposal are repository-independent. On the other hand, their solution does not consider the contextual dimension and they propose a high level specification language to be translated into a SPARQL/SerQL/SQL query to enforce the policy. They focus only on `Read` operations.

[3] `http://www.w3.org/wiki/WebAccessControl`
[4] `http://vocab.deri.ie/ppo`
[5] `http://www.w3.org/Submission/SWRL/`

3. HANDLING CONTEXT WITH PRISSMA

Whenever a mobile application needs to access some resources, the surrounding context (e.g. the physical environment) must take part into the access evaluation procedure. SPARQL queries must be associated with contextual data for access evaluation, according to a proper model.

The choice and the design of a context model necessarily need a context definition first: we agree on the widely-accepted proposal by Dey [9]. More specifically, we rely on the work by Fonseca et al. [6], 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.

Our Web of Data scenario favours the adoption of an ontology-based model. As pointed out by Korpipää and Mäntyjärvi [17], an ontological approach leads to simple and extensible models. This is a common point with the Web of Data rationale: linked data on the Web heavily relies on lightweight vocabularies under the open world assumption (i.e. new ontologies can be added at anytime about anything) and model exchange and re-use are welcomed and promoted at Web scale. A large number of ontology-based context models relying on Dey's definition have been proposed in the latter years, as summarized by Baldauf et al. [2] (e.g. CoOL, SOUPA, COBRA-ONT). These works are grounded on RDF and provide in-depth context expressivity, but for chronological reasons they are far from the Web of Data best practices (e.g. no lightweight approach, limited interlinking with other vocabularies), thus discouraging the adoption and re-use in the Web community.

Our work targets access control in the mobile Web of Data: we need therefore a context model compliant with the Web of Data paradigm. Our context-aware access control framework adopts `PRISSMA`, a lightweight vocabulary originally designed for context-aware adaptation of RDF data [6]. `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 Web of Data vocabularies and recent W3C recommendations are reused (Figure 1). 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:

`prissma:User` represents the target mobile user associated with a `prissma:Context` and consists in a `foaf:Person` sub-

[6] `http://bit.ly/XGR-mbui`

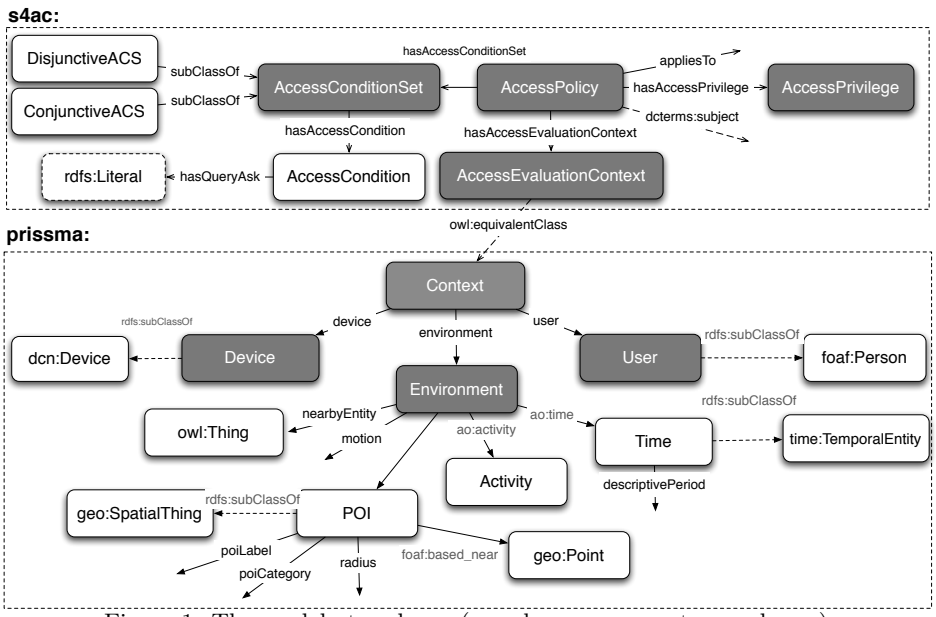


Figure 1: The model at a glance (grey boxes represent core classes).

class. To provide more flexibility, the class can be used to model both user stereotypes and specific users.

`prissma:Device` represents the mobile device on which Web of Data resource consumption takes place, enabling device-specific access control. The class inherits from W3C Delivery Context Ontology [7] `dcn:Device` that provides an extensible and fine-grained model for mobile device features.

`prissma:Environment` models the physical context in which the Web of Data resource consumption takes place. Different dimensions are involved in modelling the surrounding environment, delegating refinements and extensions to domain specialists. Location is modelled with the notion of Point of Interest (POI). The `prissma:POI` class consists in a simplified, RDFized version of the W3C Point of Interest Core specifications[8]. Each `prissma:POI` consists of a `geo:SpatialThing`[9] and can be associated with a given `geo:Point` coupled with a physical radius via the `prissma:radius` property. The properties `prissma:poiCategory` and `prissma:poiLabel` are used to assign a category and a label. Time is modelled extending the `time:TemporalEntity` class[10]. The `prissma:descriptivePeriod` property associates a description to each temporal entity (e.g. `http://dbpedia.org/resource/Evening`). Other dimensions are considered: the `motion` property associates any given high-level representation of motion to a `prissma:Environment`. The environmental proximity of a generic object can trigger different resource representations: nearby objects are associated with the Environment with the `prissma:nearbyEntity` property. The `prissma:Activity` class is a placemark aimed at connecting third-party solutions focused on inferring high-level representations of user actions (e.g.'running', 'driving', 'shopping', etc).

[7] `http://bit.ly/dc-ontology`
[8] `http://www.w3.org/TR/poi-core/`
[9] `http://www.w3.org/2003/01/geo/wgs84_pos`
[10] `http://www.w3.org/TR/owl-time`

Example 1. Figure 2 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 is not moving.

Other context-related issues need to be considered beyond context-model definition, such as context fetch, context trustworthiness and privacy. `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 present paper assumes that context data is fetched and pre-processed beforehand.
The trustworthiness of contextual information sent by mobile consumers should not be taken for granted. The `prissma:User`'s identity needs to be certified: this is an open research area in the Web, and initiatives such as WebID[11] specifically deal with this issue. Hulsebosch et al. [16] 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 [18], where context sensors are authenticated beforehand by a trusted party. We plan to tackle the issue of context-verification in future work.
Context is sent to the data server along with the client query for access evaluation (see Section 5 for details). 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 a previous work, the myCampus experience [12], we deal with access control and obfuscation rules for tracking mobile users. In the present proposition, we do not address this issue, nor the problem of context integrity.

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

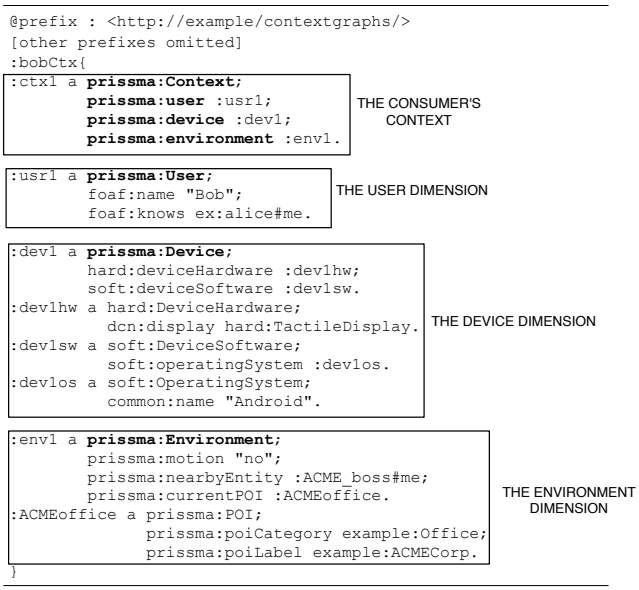


Figure 2: Bob's sample mobile context in TriG notation.

4. WEB OF DATA ACCESS CONTROL

In this section, we present our access control model and we show how it is linked to the PRISSMA context vocabulary presented in Section 3. Our access control model adopts the granularity of named graphs [3], thus supporting fine-grained access control policies, including the triple level. We choose to rely on named graphs to not depend 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). The named graph specification permits to organize the RDF content of a dataset in multiple graphs identified by given URIs[12].

The model is grounded on the S4AC ontology (Figure 1). Our access control model is integrated with lightweight ontologies adopted in the Social Web and the Web of Data. In particular, S4AC reuses concepts from SIOC[13], SKOS[14], WAC, SPIN[15] and Dublin Core[16].

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* the data consumer is allowed to access the data. Otherwise, the access is denied. The constraints specified by the Access Policies may concern the data consumer, the device, the environment, or any given combination of these dimensions (see Section 3).

Definition 1. (Access Policy) An Access Policy (P) is a tuple of the form $P = \langle ACS, AP, S, R, AEC \rangle$ where (i) ACS is a set of Access Conditions to satisfy, (ii) AP is an Access Privilege, (iii) S is the subject of the set of resources to be protected by P, (iv) R is the (set of) resource(s) to be protected by P, and (v) AEC is the Access Evaluation Context of P.

[12]The discussion about the use of named graphs in RDF 1.1 can be found at http://www.w3.org/TR/rdf11-concepts
[13]http://rdfs.org/sioc/spec
[14]http://www.w3.org/TR/skos-reference
[15]http://spinrdf.org
[16]http://dublincore.org/documents/dcmi-terms

An Access Condition, as defined in Definition 2, expresses a constraint which needs to be verified in order to have the Access Policy satisfied.

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 Access Conditions as SPARQL 1.1 ASK queries. Note that no information is returned about the possible query solutions, just whether or not a solution exists.

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*.

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 a *true*/*false* answer. 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 $CACS = AC_1 \wedge AC_2 \wedge \ldots \wedge AC_n$.

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.

We introduce the ACS, instead of using for instance the SPARQL UNION clause inside the ASK, because the idea is to define basic ACs with a simple and focused goal to allow their reuse by users without a SPARQL background.

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(s) 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 to keep a close relationship with CRUD-oriented access control systems, allowing a finer-grained access control beyond

simple read/write privileges. Moreover, we relate the four privilege classes to SPARQL 1.1 query and update language primitives through the SPIN ontology, which models the SPARQL primitives as SPIN classes. We show how this matching is actually used in Section 5.

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 specific named graphs, or a set of named graphs associated with a common subject. The former is achieved by providing the URI(s) of the named graph(s) to protect using the `s4ac:appliesTo` property. The latter is implemented by listing the subjects of the named graphs to protect using the property `dcterms:subject`. The assumption here is that named graphs have been previously annotated with such metadata. Summarizing, both S and R represent the data to protect, but R specifies the URI(s) of the named graphs, while S specifies the subject of the graphs (e.g., the policy protects the named graphs whose subject is *Concert*, `http://dbpedia.org/resource/Concert`).

Finally, the Access Policy is associated with 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 mobile context in which the SPARQL query has been performed. More specifically, we choose to implement the Access Evaluation Context as a SPARQL 1.1 `BINDINGS` clause to constrain the `ASK` evaluation, i.e. `BINDINGS ?ctx {(`URI_{ctx}`)}`. However, the same result can be obtained by binding directly the variable `?ctx` to the URI of the contextual graph.

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 provides 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.

Conflicts among policies might occur if the data provider uses Access Conditions with contrasting `FILTER` clauses. For instance, it is possible to define positive and negative statements such as `ASK{FILTER(?u=<http://example#bob>)}` and `ASK{FILTER(!(?u=<http://example#bob>))}`. If these two Access Conditions are applied to the same data, a logical conflict arises. This issue is handled in the framework by evaluating policies applied to a resource in a disjunctive way. We expect to add a mechanism to prevent the insertion of conflicting policies as a future work.

Example 2. Let us consider the named graph `:alice_reviews` whose content is shown in Figure 3. We now present an example of Access Policy with a conjunctive Access Condition Set associated with a `Read` privilege (Figure 4). The policy protects the named graph `:alice_reviews` and allows the access to the named graph only if the consumer (i) knows Alice, and (ii) is not located near Alice's boss.

```
:alice_reviews {
ex:29900 a bibo:Article;
       dcterms:title "A great festival";
       dcterms:date "2011";
       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".
}
```

Figure 3: The named graph `:alice_reviews`, in TriG syntax. The graph contains the concert reviews authored by Alice.

```
:policy1 a s4ac:AccessPolicy;                          ACCESS POLICY
         s4ac:appliesTo :alice_reviews;                RESOURCE TO PROTECT
         s4ac:hasAccessPrivilege [a s4ac:Read];        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))}""".
```

Figure 4: The Access Policy protecting `:alice_reviews`

Policy validation can be addressed in two different ways. First, the SPIN vocabulary can be used to express the literal representing the `ASK` query as RDF statements. On the other hand, we can perform a two-step validation, combining RDF validation for the policy and SPARQL validation for the literals of `s4ac:hasQueryAsk`, i.e. the `ASK` queries.

5. CONTROL ENFORCEMENT

Our Access Control Manager is designed as a pluggable component for SPARQL endpoints (Figure 5). The access control flow is described below:

1. the mobile consumer queries the SPARQL endpoint to access the content. At the same time, contextual information is sent with the query and saved as a named graph using SPARQL 1.1 update language statements. Each time a context element is added we use an `INSERT DATA`, while we rely on a `DELETE/INSERT` when the contextual information is already stored and has to be updated. Summarizing, the mobile client sends two SPARQL queries: the first is the client query aimed at

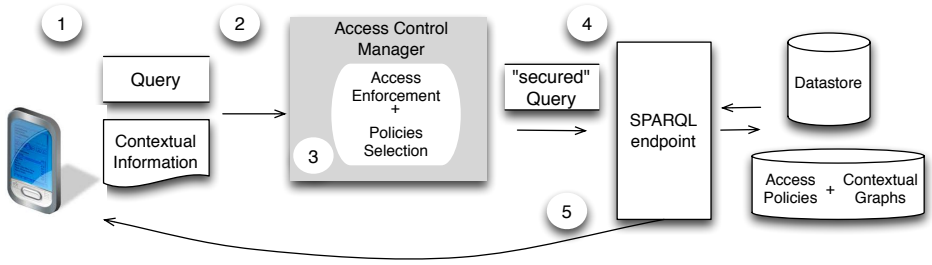


Figure 5: The access control framework architecture.

the datastore, the second provides contextual information (like the one visualized in Figure 2).

2. the client query is filtered by the Access Control Manager instead of being directly executed on the SPARQL endpoint.
3. the Access Control Manager selects the set of policies affecting the client query and after their evaluation returns the set of named graphs the consumer is granted access to.
4. the client query is executed only on the accessible named graphs.
5. the result of the query is returned to the consumer.

The aim of the Access Control Manager is twofold: it first selects the Access Policies to assess and it verifies the set of Access Conditions included in the selected policies to grant or not the access. We describe the two algorithms to protect the access to the data (Figure 8).

Algorithm 1 is the main procedure for the execution of a query with access enforcement. The input of the algorithm is the client query Q and the RDF graph G_{ctx} modeling the client mobile context. It assumes the existence of a repository of access policies APS. The algorithm starts by saving the contextual graph in a local cache (line 1). At the beginning, the set of accessible named graph NGS is empty (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). According to the type of Access Condition Set (i.e., conjunctive or disjunctive), for each verified policy, the associated named graph is added to the set of accessible named graphs (lines 11-12). Finally, after the execution of all Access Conditions, the client query is sent to the SPARQL endpoint with the addition of the `FROM` clause (line 16). Query execution is therefore performed only on the accessible named graphs, given the consumer contextual information. Line 18 outputs the triples resulting from Q.

Algorithm 2 is the Access Policies Selection routine. It selects the Access Policies concerned by the client query. The input of the algorithm is the query Q and the repository of the policies APS. 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

```
PREFIX ctxgraphs: <http://example/contextgraphs/>

ASK{?context a prissma:Context.
    ?context prissma:user ?u.
    ?u foaf:knows ex:alice#me.}
  BINDINGS ?context {(ctxgraphs:bobCtx)}

ASK {?context a prissma:Context.
     ?context prissma:environment ?env.
     ?env prissma:based_near ?p.
     FILTER (!(?p=ex:ACME_boss#me))}
  BINDINGS ?context {(ctxgraphs:bobCtx)}
```

THE CONSUMER'S CONTEXT

Figure 6: The Access Conditions bound to the actual `prissma:Context` shown in Figure 2

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
SELECT *
WHERE {?review a bibo:Article}
```

(a)

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
SELECT *
FROM :peter_reviews
WHERE {?review a bibo:Article}
```

NAMED GRAPH ACCESSIBLE BY THE CONSUMER

(b)

Figure 7: The SPARQL query issued by Bob's mobile client (a) and the constrained version (b).

execute. In particular, the algorithm maps the client query to one of the four access privileges `S4AC` defines using the SPIN vocabulary (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 Access Enforcement algorithm (Algorithm 1).

Example 3. An example of client query is shown in Figure 7.a, where Bob wants to access all rock festival's reviews from the context described in Figure 2. When the query is received by the Access Control Manager, the latter selects the Access Policies concerning this query (for instance the policy shown in Figure 4). The Access Conditions included in the policies are then coupled with a `BINDINGS` clause, as shown in Figure 6, where the `?context` variable is bound to Bob's actual context. The identification of the named graph(s) accessible by Bob returns only the graph `:peter_reviews`. The named graph `:alice_reviews` of Figure 3 is forbidden because Access Conditions evaluation leads to a `false` answer with Bob's context (Bob is near Alice's boss). The Access Control Manager adds the `FROM` clause to constrain the execution of the client query only on the allowed named graph. The "secured" client query is shown in Figure 7.b.

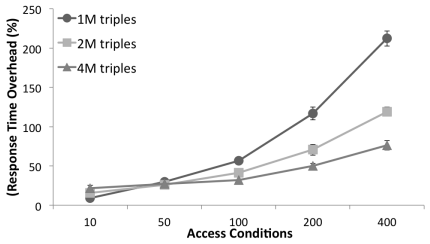
Algorithm 1: Query Execution with Access Enforcement

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`>` to Q;
$R \leftarrow$ run Q;
return R;

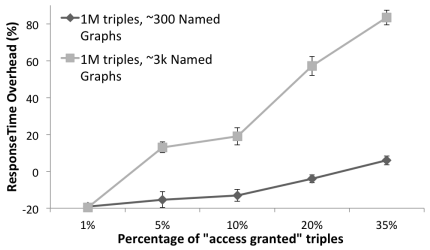
Algorithm 2: Access Policies Selection

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;

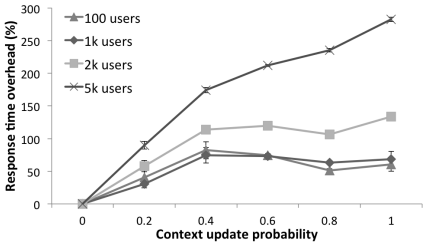
Figure 8: SPARQL Query Execution Procedure



(a)



(b)



(c)

Figure 9: Response time overhead

6. 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 and SPARQL 1.1 query engine[17] [4]. We evaluate 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[18].

In Figure 9 we execute 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 with the mobile context described in Figure 2. Each Access Policy contains exactly one Access Condition. In Figure 9.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).

In a typical scenario, the Access Control Manager restricts the results of a query. In Figure 9.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:RatingSite`s, 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 `FROM` clauses. As shown in Figure 9.b, the cost of this operation grows with the number of named graphs returned by the evaluation of the Access Policies.

In Figure 9.c we analyse the overhead introduced on response time by queries executed in dynamic mobile 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 update (Section 5). 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 mobile users, since each of them is associated with a mobile context graph.

[17] `http://tinyurl.com/corese-engine`
[18] `http://bit.ly/berlin-sparql`

7. CONCLUSIONS

Accessing the Web of Data needs an access control mechanism. Moreover, consumption and production of linked data might origin from mobile devices immersed into pervasive environments. This paper presents an approach towards context-aware access control for the ubiquitous Web of Data. The proposed solution is conceived as an easy-to-integrate pluggable filter for data servers that support the SPARQL query language. Our framework relies only on Web of Data languages and existing vocabularies; no other formalism has been added. The prototype evaluation shows that, despite the overall performance needs to be ameliorated, the delay introduced by our fine-grained, context-based access control is acceptable given that data protection comes with a cost. Future testing campaign will be carried out to provide a thorough evaluation with other SPARQL query engines, such as Virtuoso, Sesame, Jena and AllegroGraph. An effective backend user interface to define Access Policies has to be designed, as user interaction issues should not be underestimated. The trustworthiness of the information sent by the mobile consumer, including data describing context (e.g. location, device features, etc.) should not be taken for granted: future work needs to investigate this issue. 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, and we will therefore focus on this issue in the future.

8. REFERENCES

[1] F. Abel, J. L. De Coi, N. Henze, A. W. Koesling, D. Krause, and D. Olmedilla. Enabling Advanced and Context-Dependent Access Control in RDF Stores. In *Procs of the 6th Int. Semantic Web Conf. (ISWC-2007), LNCS 4825*, pages 1–14, 2007.

[2] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. J. of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[3] J. J. Carroll, C. Bizer, P. J. Hayes, and P. Stickler. Named graphs. *J. Web Sem.*, 3(4):247–267, 2005.

[4] O. Corby and C. Faron-Zucker. The KGRAM Abstract Machine for Knowledge Graph Querying. In *Web Intelligence*, pages 338–341. IEEE, 2010.

[5] A. Corradi, R. Montanari, and D. Tibaldi. Context-based access control management in ubiquitous environments. In *Procs of the 3rd IEEE Int. Symposium on Network Computing and Applications (NCA-2004)*, pages 253–260, 2004.

[6] L. Costabello. DC Proposal: PRISSMA, Towards Mobile Adaptive Presentation of the Web of Data. In *Procs of the 10th Int. Semantic Web Conf. (ISWC-2011)*, LNCS 7032, pages 269–276, 2011.

[7] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. In *Procs of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT-2001)*, pages 10–20, 2001.

[8] F. Cuppens and N. Cuppens-Boulahia. Modeling contextual security policies. *Int. J. Inf. Sec.*, 7(4):285–305, 2008.

[9] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5:4–7, 2001.

[10] T. W. Finin, A. Joshi, L. Kagal, J. Niu, R. S. Sandhu, W. H. Winsborough, and B. M. Thuraisingham. R*OWL*BAC: representing role based access control in *OWL*. In *Procs of 13th ACM Symposium on Access Control Models and Technologies (SACMAT-2008)*, pages 73–82, 2008.

[11] G. Flouris, I. Fundulaki, M. Michou, and G. Antoniou. Controlling Access to RDF Graphs. In *Procs of the 3rd Future Internet Symposium (FIS-2010)*, LNCS 6369, pages 107–117, 2010.

[12] F. Gandon and N. M. Sadeh. A semantic e-wallet to reconcile privacy and context awareness. In *Procs of the 2nd Int. Semantic Web Conf. (ISWC-2003)*, LNCS 2870, pages 385–401, 2003.

[13] F. Giunchiglia, R. Zhang, and B. Crispo. Ontology driven community access control. In *Procs of the 1st Workshop on Trust and Privacy on the Social and Semantic Web (SPOT-2009)*, 2009.

[14] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.

[15] J. Hollenbach, J. Presbrey, and T. Berners-Lee. Using RDF Metadata To Enable Access Control on the Social Semantic Web. In *Procs of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK-2009)*, 2009.

[16] R. J. Hulsebosch, A. H. Salden, M. S. Bargh, P. W. G. Ebben, and J. Reitsma. Context sensitive access control. In *Procs of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT-2005)*, pages 111–119, 2005.

[17] P. Korpipää and J. Mäntyjärvi. An ontology for mobile device sensor-based context awareness. In *Procs of the 4th Int. and Interdisciplinary Conf. Modeling and Using Context (CONTEXT-2003)*, LNCS 2680, pages 451–458, 2003.

[18] D. Kulkarni and A. Tripathi. Context-aware role-based access control in pervasive computing systems. In *Procs of 13th ACM Symposium on Access Control Models and Technologies (SACMAT-2008)*, pages 113–122, 2008.

[19] H. Muhleisen, M. Kost, and J.-C. Freytag. SWRL-based Access Policies for Linked Data. In *Procs of the 2nd Workshop on Trust and Privacy on the Social and Semantic Web (SPOT-2010)*, 2010.

[20] O. Sacco and A. Passant. A Privacy Preference Ontology (PPO) for Linked Data. In *Procs of the 4th Workshop about Linked Data on the Web (LDOW-2011)*, 2011.

[21] H. Shen and Y. Cheng. A semantic context-based model for mobile web services access control. *Int. J. Computer Network and Information Security*, 2011.

[22] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. A semantic context-aware access control framework for secure collaborations in pervasive computing environments. In *Procs of the 5th Int. Semantic Web Conf. (ISWC-2006)*, LNCS 4273, pages 473–486, 2006.

[23] S. Villata, N. Delaforge, F. Gandon, and A. Gyrard. An access control model for linked data. In *Procs of the 7th Int. IFIP Workshop on Semantic Web & Web Semantics (SWWS-2011)*, LNCS 7046, pages 454–463, 2011.