Context-Aware Access Control for RDF Graph Stores

Luca Costabello and **Serena Villata** and **Fabien Gandon** [1]

Abstract. We present SHI3LD, an access control framework for RDF stores. Our solution supports access from mobile devices with context-aware policies and is exclusively grounded on standard Semantic Web languages. Designed as a pluggable filter for generic SPARQL endpoints, the module uses RDF named graphs and SPARQL to protect triples. Evaluation shows faster execution time for low-selective queries and less impact on larger datastores.

1 Introduction

The Web is evolving from an information space for sharing textual documents into a medium for publishing structured data. The Linked Data[2] initiative aims at fostering the publication and interlink of data on the Web, giving birth to the *Web of Data*, an interconnected global dataspace where data providers publish their content publicly [13]. The open nature of current Web of Data information and the consumption of web resources from mobile devices 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. Access control is therefore necessary, and mobile context must be part of the access control evaluation.

In this paper we address the problem of defining an access control framework, called SHI3LD[3], for querying Web of Data servers from mobile environments. Three major challenges arise: (i) definition of a fine-grained access control model for graph stores, (ii) modelling of context-aware, mobile consumption of such information, and (iii) integration of mobile context in the access control model.

We protect RDF stores by changing the semantics of the incoming SPARQL queries, whose scope is restricted to triples included in accessible named graphs only. We determine the list of accessible graphs by evaluating pre-defined access policies against the actual mobile context of the requester. Beyond the support for context in control enforcement, our proposal has the advantage of being a pluggable filter for generic SPARQL endpoints, with no need to modify the endpoint itself. We adopt exclusively Semantic Web languages and reuse existing proposals, thus we do not add new policy definition languages, parsers nor validation procedures. We provide protection up to triple level. Our work does not provide yet another context ontology: our model includes base classes and properties only, as we delegate refinements and extensions to domain specialists, in the light of the Web of Data philosophy [13]. We do not deal with mobile context fetch, thus including on-board sensors or server-side inference. 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). We do not provide any privacy-preserving mechanism yet, although aware that sensible data such as current location must be handled appropriately. Our approach focuses only on SPARQL data servers.

The remainder of the paper is organized as follows. Section 2 compares the related work to SHI3LD. Section 3 introduces the context aspects and the access control model. The control enforcement algorithm is detailed in Section 4. Section 5 shows the experimental results.

[1] INRIA Sophia Antipolis, France, email: firstname.lastname@inria.fr

[2] `http://linkeddata.org`

[3] `http://wimmics.inria.fr/projects/shi3ld/`

2 Related Work

We differ from WAC[4] since we go beyond RDF document granularity and we do not rely on access control lists. Sacco and Passant [17] present the PPO vocabulary. The common points of PPO and SHI3LD are the use of the `ASK` queries for representing the access conditions and the use of Semantic Web languages only. Sacco and Passant express access control policies for RDF documents, while we provide an authorization mechanism for RDF stores. Moreover, our framework adopts context-aware policies while in [17] context is not considered. Finally, we provide an evaluation of the experimental results of SHI3LD, differently from [17] where no evaluation is addressed. Flouris et al. [11] provide a fine-grained access control framework on top of RDF repositories coupled with a high level specification language. Finin et al. [10] consider attribute-based access control where, similarly to our proposal, the constraints are based on general attributes of an action. Giunchiglia et al. [12] propose a Relation Based Access Control model, while we specify the attributes the consumer must satisfy. Context information is supported to some extent by Abel et al. [1]. They provide triple-level access control as a layer on top of RDF stores. Contextual conditions are pre-evaluated before expanding the queries. They introduce a high-level syntax for policy definition, while we exclusively rely on RDF. Toninelli et al. [18] adopt context-awareness and semantic technologies for access control but they do not apply their solution to the Web of Data. 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. Covington et al. [7] use the notion of role proposed by Role Based Access Control 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. Cuppens and Cuppens-Boulahia [8] propose an Organization Based Access Control. 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. We support additional contextual dimensions, e.g., the device. Although their policies are expressed in RDF, the system is not designed for the Web of Data.

[4] `http://www.w3.org/wiki/WebAccessControl`

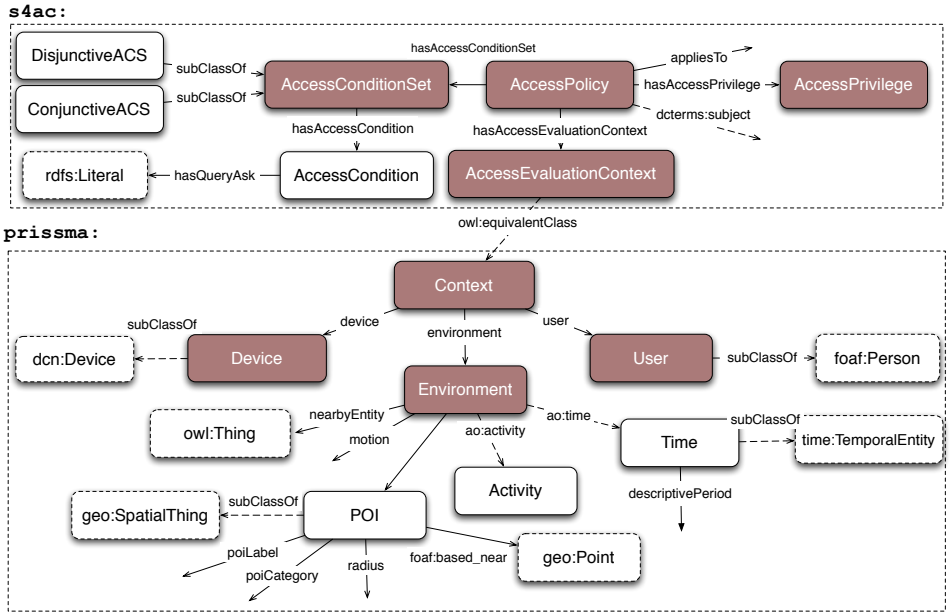

Figure 1: The SHI3LD model at a glance (red boxes represent core classes).

3 The Model

The SHI3LD model is grounded on two ontologies (Figure 1): `S4AC` deals with core access control concepts and `PRISSMA`[5] focuses on the mobile context[6]. The access control model is built over the notion of Named Graph [3], thus supporting fine-grained access control policies, including the triple level. Enforcing permission models is an envisioned use case for RDF named graphs[7]. 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[8]). At conceptual level, our policies can be considered as access control conditions over g-boxes[9] (according to W3C RDF graph terminology), with semantics mirrored in the SPARQL language. The `S4AC` vocabulary [19] reuses concepts from SIOC, SKOS, WAC, SPIN and Dublin Core[10].
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.

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.

An Access Condition, as defined in Definition 2, expresses a constraint which needs to be verified 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 query solution is returned, since SPARQL `ASK` only tests whether 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 $ACS = \{AC_1, AC_2, \ldots, AC_n\}$.

We introduce the ACS to ease the reuse and combination of ACs to dataset administrators lacking deep SPARQL knowledge. We thus avoid the use of more complicated SPARQL `UNION` clauses inside the `ASK`s. Roughly, the verification of an Access Condition Set returns a *true/false* answer and can be provided in a conjunctive or disjunctive fashion.

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 $DACS = AC_1 \vee AC_2 \vee \ldots \vee AC_n$.

[5] Ontologies details at `http://bit.ly/vspecs`
[6] SHI3LD can be adapted to support other definitions of context, stemming from different scenarios.
[7] `http://bit.ly/w3rdfperm`
[8] The discussion about the use of named graphs in RDF 1.1 can be found at `http://www.w3.org/TR/rdf11-concepts`
[9] `http://bit.ly/graphterm`
[10] Reused ontologies details at `http://bit.ly/reusedv`

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

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 future work.

The Access Privilege (Definition 9) 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. 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.
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 to 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`[11]).

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 mobile context in which the SPARQL query has been performed (e.g. `:ctx` in Figure 2b). 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 choice and the design of a context model necessarily need a context definition first. We agree on the widely-accepted proposal by Dey [9]:

[11] `dbpedia.org` is the RDFized porting of Wikipedia.

```
: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)

Figure 2: The Access Policy protecting `:alice_data` (a) and Bob's sample mobile context in TriG notation (b).

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" [9].

More specifically, we rely on the work by Fonseca and colleagues [12], 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 [15], 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

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

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 Bolchini et al. [2] (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 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 [13]. Our context-aware access control framework adopts `PRISSMA`, a lightweight vocabulary originally designed for context-aware adaptation of RDF data [6]. PRISSMA has been originally designed to express the contextual conditions under which activate a given representation for RDF [6]. 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 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:

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 [13] `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 Web of Data 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[14]. Time is modelled extending the `time:TemporalEntity` class[15]. 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[16] could be easily integrated).

[13] `http://bit.ly/dc-ontology`
[14] `http://www.w3.org/TR/poi-core/`
[15] `http://www.w3.org/TR/owl-time`
[16] `http://vocab.deri.ie/rooms`

Example 1. We now present an example of Access Policy with a conjunctive Access Condition Set associated to an `Update` privilege (Figure 2a). The policy protects the 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 2b 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[17] specifically deal with this issue. Hulsebosch et al. [14] 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 [16], 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.

4 Access Control Enforcement

Our Access Control Manager is designed as a pluggable component for SPARQL endpoints. The access control flow is described below (Figure 3):

1. The mobile consumer queries the SPARQL endpoint to access the content. Context data is sent with the query and cached 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 to the datastore (e.g. Figure 5a), the second provides contextual information (e.g. Figure 2b).
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, i.e. those with a matching Access Privilege. This is achieved by mapping the client query to one of the four Access Privileges defined by S4AC with the SPIN vocabulary. The Access Conditions (SPARQL `ASK` queries) included in the selected policies are executed. 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.
4. The client query is sent to the SPARQL endpoint with the addition of the following clauses:

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

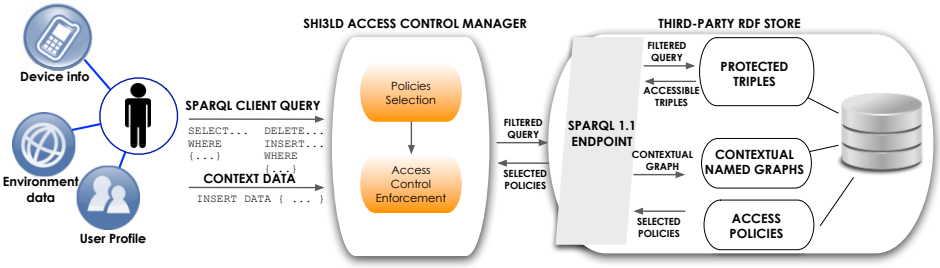

Figure 3: The algorithm of access control enforcement in the SHI3LD architecture.

`FROM/FROM NAMED` clauses for `SELECT` queries, to execute the query only on the accessible named graphs, given the contextual information associated to the consumer. 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/USING NAMED` clauses for `DELETE/INSERT`, `DELETE` and `INSERT` queries. The clauses describe a dataset in the same way as `FROM` and `FROM NAMED`. The keyword `USING` instead of `FROM` in update requests has been chosen to avoid possible ambiguities which could arise from writing `DELETE FROM`[18].

Query execution is therefore performed only on the accessible named graphs, given the consumer contextual information.

```
PREFIX bobCtx: <http://example/contextgraphs/bobCtx>
ASK{?context a prissma:Context.
    ?context prissma:user ?u.
    ?u foaf:knows ex:alice#me.}
  BINDINGS ?context {(bobCtx:ctx)}

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

THE CONSUMER'S CONTEXT

Figure 4: The Access Conditions bound to the actual `prissma:Context` shown in Figure 2

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

THE NAMED GRAPH ACCESSIBLE BY THE CONSUMER

(b)

Figure 5: The SPARQL query issued by Bob's mobile client (a) and the *filtered* version (b).

[18] `http://bit.ly/deleteinsert`

Example 2. An example of client query is shown in Figure 5a, where Bob wants to access and modify the datastore (including Alice data `:alice_data`, protected by the policies in Figure 2a) in such a way that all triples having `dcterms:subject Concert_tours` are changed into `dcterms:subject Music_performance`. Bob wants to perform such operation on the datastore from the context described in Figure 2b. 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 2a). The Access Conditions included in the policies are then coupled with a `BINDINGS` clause, as shown in Figure 4, where the `?context` variable is bound to Bob's actual context. The identification of the named graph(s) accessible by Bob returns, for example, only the graph `:peter_data`. Alice data 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 `USING`, `USING NAMED` clauses to constrain the execution of the client query only on the allowed named graph(s), i.e., `:peter_data`. The *filtered* client query is shown in Figure 5b.

5 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[19] [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[20].

In Figure 6a 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 to the mobile context described in Figure 2b. Each Access Policy contains exactly one Access Condition. In Figure 6a, 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). Our solution is independent from the complexity of the incoming SPARQL query, as the only change we do is adding a list of FROM/FROM NAMED clauses (USING/USING NAMED for updates). Since we do not need to rewrite the query, the overhead is independent from query complexity.

In a typical scenario, the Access Control Manager restricts the results

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

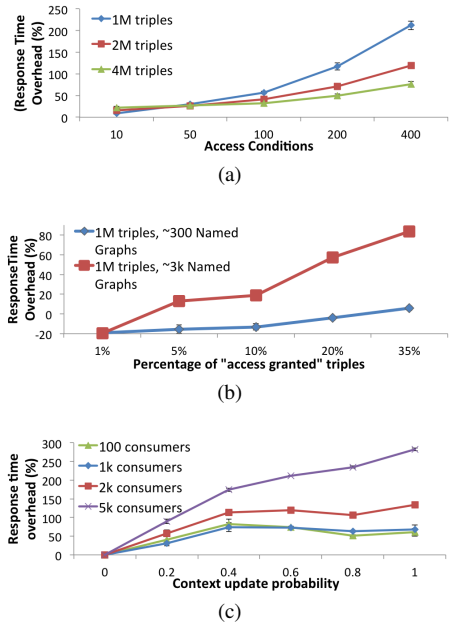

Figure 6: Response time overhead

of a query. In Figure 6b 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 `FROM` clauses. As shown in Figure 6b, the cost of this operation grows with the number of named graphs returned by the evaluation of the Access Policies.

In Figure 6c 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 4). 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 to a mobile context graph.

6 Conclusions

This paper presents an access control manager for RDF stores, designed as a pluggable filter for generic SPARQL endpoints. Our solution features context-aware control policies and relies only on Semantic Web languages, thus we do not add ad-hoc policy definition languages, parsers nor validation procedures. We protect triples by (i) relying on named graphs and (ii) by changing the semantics of incoming SPARQL queries, whose scope is restricted to triples included in accessible named graphs only. We add support for mobile context in control enforcement and we deliver fine-grained protection, up to triple level. Prototype evaluation shows that when the access is granted to a small fraction of named graphs, the query is executed faster than the case without access control. The delay introduced by our Access Control Manager grows with the number of Access Conditions in the system but has less impact on larger datasets while depending on the number of requesters. An effective backend user interface to define Access Policies has to be designed as user interaction issues should not be underestimated. Future work includes supporting the trustworthiness of context data sent by the mobile consumer and a privacy-preserving mechanism to handle mobile user context appropriately.

ACKNOWLEDGEMENTS

The second author acknowledges support of the DataLift Project ANR-10-CORD-09 founded by the French National Research Agency.

REFERENCES

[1] F. Abel, J. L. D. Coi, N. Henze, A. W. Koesling, D. Krause, and D. Olmedilla. Enabling Advanced and Context-Dependent Access Control in RDF Stores. In *Proc. of ISWC*, pages 1–14, 2007.

[2] C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. A data-oriented survey of context models. *SIGMOD Record*, 36(4):19–26, 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 *Proc. of Web Intelligence*, pages 338–341. IEEE, 2010.

[5] A. Corradi, R. Montanari, and D. Tibaldi. Context-based access control management in ubiquitous environments. In *Proc. of NCA*, pages 253–260, 2004.

[6] L. Costabello. DC Proposal: PRISSMA, Towards Mobile Adaptive Presentation of the Web of Data. In *Proc. of ISWC*, pages 269–276, 2011.

[7] L. Costabello, S. Villata, N. Delaforge, and F. Gandon. Linked Data Access Goes Mobile: Context-Aware Authorization for Graph Stores. In *Proc. of LDOW*, 2012.

[8] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. In *Proc. of SACMAT*, pages 10–20, 2001.

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

[10] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[11] 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 *Proc. of SACMAT*, pages 73–82, 2008.

[12] G. Flouris, I. Fundulaki, M. Michou, and G. Antoniou. Controlling Access to RDF Graphs. In *Proc. of FIS*, pages 107–117, 2010.

[13] F. Giunchiglia, R. Zhang, and B. Crispo. Ontology driven community access control. In *Proc. of SPOT*, 2009.

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

[15] R. J. Hulsebosch, A. H. Salden, M. S. Bargh, P. W. G. Ebben, and J. Reitsma. Context sensitive access control. In *Proc. of SACMAT*, pages 111–119, 2005.

[16] P. Korpipää and J. Mäntyjärvi. An Ontology for Mobile Device Sensor-Based Context Awareness. In *Proc. of CONTEXT*, pages 451–458, 2003.

[17] D. Kulkarni and A. Tripathi. Context-aware role-based access control in pervasive computing systems. In *Proc. of SACMAT*, pages 113–122, 2008.

[18] O. Sacco, A. Passant, and S. Decker. An access control framework for the web of data. In *Proc. of TrustCom, IEEE*, pages 456–463, 2011.

[19] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In *Proc. of ISWC*, pages 473–486, 2006.