Undefined 1 (2009) 1–5
IOS Press

Access Control and the Resource Description Framework: A Survey

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Sabrina Kirrane [a,b], Alessandra Mileo [a] and Stefan Decker [a]
[a] *Insight Centre for Data Analytics, National University of Ireland, Galway, Ireland*
E-mail: {firstname.lastname}@insight-centre.org
[b] *Galway-Mayo Institute of Technology, Ireland*

Abstract. In recent years we have seen significant advances in the technology used to both publish and consume structured data using the existing web infrastructure, commonly referred to as the Linked Data Web. However, in order to support the next generation of e-business applications on top of Linked Data suitable forms of access control need to be put in place. This paper provides an overview of the various access control models, standards and policy languages, and the different access control enforcement strategies for the Resource Description Framework (the data model underpinning the Linked Data Web). A set of access control requirements that can be used to categorise existing access control strategies is proposed and a number of challenges that still need to be overcome are identified.

Keywords: Access Control, Policies, Linked Data, Semantic Web, Resource Description Framework

1. Introduction

The term Linked Data Web (LDW) is used to describe a World Wide Web where data is directly linked with other relevant data using machine-readable formats [63,38]. Although the technology underpinning the LDW has been in existence for a number of years, up until now data publishers have primarily focused on exposing and linking public data. With the advent update languages for the Resource Description Framework (RDF) data model, such as SPARQL 1.1 [99], the LDW has the potential to evolve from a medium for publishing and linking public data, to a dynamic read-/write distributed data source. Such an infrastructure will be capable of supporting not only data integration of public and private data, but also the next generation of electronic business applications. However, in order to make the move from simply linking public data to using the Linked Data infrastructure as a global dataspace, suitable security mechanisms need to be put in place.

Security in general and access control in particularity have been extensively studied by both the database and the information system communities, among others. Early work on access control policy specification and enforcement within the Semantic Web community focused on: representing existing access control models and standards using semantic technology; proposing new access control models suitable for open, heterogeneous and distributed environments; and devising languages and frameworks that can be used to facilitate access control specification and maintenance. Later researchers examined access control for the RDF data model in general and access control propagation, based on the semantic relations between policy entities in particular. In recent years the focus has shifted to access control policy specification and enforcement over Linked Data. Although few authors have proposed access control strategies specifically for Linked Data [74,19,51,86], there is a large body of work on general policies languages and ac-

cess control strategies for the RDF data model that could potentially be applied to Linked Data. In order to better understand the potential of existing access control mechanisms, this paper provides an overview of a number of access control models, languages and frameworks. A set of access control requirements are collated from [102,21,101,22,10,73] and categorised according to four different dimensions (specification, enforcement, implementation and infrastructure) that are necessary from an access control perspective. These requirements are used not only to classify existing access control strategies but also to identify challenges that still need to be overcome.

The remainder of the paper is structured as follows: *Section* 2 presents relevant access control models and standardisation efforts and discusses how they have been applied to, or enhanced, using semantic technology. *Section* 3 details several well known policy languages and frameworks that use ontologies, rules or a combination of both to represent policies. *Section* 4 describes the different access control administration and enforcement strategies that have been proposed for RDF data. *Section* 5 presents a set of access control requirements and categorises existing access control languages and frameworks accordingly. Finally *Section* 6 concludes the paper and presents directions for future work.

2. Access Control Models and Standards

Generally speaking the term *access control* is used to refer to the *model*, which is the blueprint that is used to guide the access control process; the *policy language*, which defines both the syntax and the semantics of the access control rules; and the *framework*, which is a combination of the access control model, the language and the enforcement mechanism. At its most basic, an access control rule (otherwise known as an authorisation) can be represented as a tuple $\langle S, R, AR \rangle$ where S denotes the *subject* (entities requesting access to resources), R denotes the *resource* (entities to be protected) and AR represents the *access rights* (permissions and prohibitions often based on actions pertaining to the resource). Sets of access control rules are collectively referred to as an *access control policy*. The decision to grant or deny access is based on two distinct processes, *authentication* and *authorisation*. *Authentication* involves the verification of credentials (you are who you say you are). Whereas, *authorisation* is the process of granting or denying access to system resources based on credentials.

This section describes both well known and emerging access control models, and relevant standardisation efforts. In each instance, a description of the access control model/standard is provided, along with details of how the model/standard has been applied to or enhanced using Semantic Web technologies.

2.1. Access Control Models

Mandatory Access Control (MAC), Discretionary Access Control (DAC) and Role Based Access Control (RBAC) are the most prominent access control models found in the literature, and used in practice. View Based Access Control (VBAC) is a complementary access control model which grants access to sets of entities, logically structured as views. More recently researchers have proposed new access control models, that are deemed more suitable for the open, heterogeneous and distributed architecture of the web. Primary research efforts, involve using properties (relating to the subject, resource or the environment) as opposed to identities, in order to determine if access to resources should be permitted. Attribute Based Access Control (ABAC) and Context Based Access Control (CBAC) are the predominant works in this area.

2.1.1. Mandatory Access Control

MAC limits access to resources using access control policies determined by a central authority [79]. The central authority is responsible for classifying both *subjects* and *resources* according to *security levels*. Resources are assigned labels that represent the security level required to access the resource, and only subjects with the same security level or higher are granted access. MAC was originally developed for military applications and therefore it is best suited to closed environments, where a great deal of control is required [6]. Given the open, heterogeneous and distributed nature of the web, it is not surprising that MAC has not gained much traction among Semantic Web researchers. Primary research efforts focus on:

- defining vocabularies that can be used to support multiple access control models;
- using attributes to represent different credentials (e.g. identities, roles and labels).

Supporting multiple access control models using vocabularies. When it comes to MAC the primary focus has been on demonstrating how MAC can be supported by Semantic Web policy languages. Kodali et al. [54] describe a unifying framework which can be used to enforce a number of different access control models, MAC being one of them. The authors propose a DAML+OIL[1] ontology and demonstrate via example how it can be used to represent authorisations modelled using MAC, DAC and RBAC.

Supporting multiple access control models using attributes. Yagüe et al. [102] tackle the problem of heterogeneity of access control in distributed systems and discuss how their attribute based access control model, can be used to represent MAC, DAC and RBAC. In addition, they demonstrate how Semantic Web concepts and technologies and can be used to specify and enforce the proposed attribute based access control policies.

2.1.2. Discretionary Access Control

DAC policies associate one or more *subjects* with sets of *access rights* pertaining to one or more *resources*. Like MAC, DAC restricts access by means of a central access control policy, however users are allowed to override the central policy and can pass their access rights on to other users, a process known as delegation [81]. According to Weitzner et al. [100], the web needs discretionary, rule based access control. Although the authors describe a motivating scenario and present a potential solution, they focus primarily on the general architecture of the system, as opposed to investigating how discretionary access control can be modelled or enforced. When it comes to intersection between DAC and RDF, research efforts to date have focused on:

- defining vocabularies that can be used to support multiple access control models;
- using attributes to represent identities, roles and labels;
- providing support for the delegated of access rights to others.

Using attributes to supporting multiple access control models. As with MAC, both Kodali et al. [54] and Yagüe et al. [102] demonstrate how the access control models they propose can support multiple access control models, DAC being one.

DAC for the RDF data model. Gabillon and Letouzey [33] describe an enforcement framework whereby users define security policies for RDF graphs and SPARQL views (SPARQL `CONSTRUCT` and `DESCRIBE` queries) that they own. Users may delegate rights to other users by specifying an authorisation which grants `CONSTRUCT` and `DESCRIBE` privileges to their RDF graphs or views.

Kirrane et al. [53] provide a summary of discretionary access control requirements for the RDF data model, based on the different characteristics of the graph data model compared to relational and tree data models. The authors suggest extending the SPARQL query language with two new commands `GRANT` and `REVOKE` that could be used to manage the delegation of access rights in RDF databases. A follow up paper [51] proposes a flexible authorisation framework and demonstrates how authorisations, propagation policies, integrity constraints and conflict resolution policies can be used to enforce DAC over RDF data.

2.1.3. Role Based Access Control

RBAC restricts access to *resources* to groups of *users*, with common responsibilities or tasks, generally referred to as *roles*. In RBAC, users are assigned to appropriate roles and access to resources is granted to one or more roles, as opposed to users directly [82]. The term *session* is commonly used to refer to the set of roles that the user is currently assuming (their active roles). Whereas, *role deactivation* is generally used to refer to the process whereby a user is removed from a role. Depending on the usecase, roles may be organised to form either a hierarchy or a partial order. Such structures are used to simplify access control specification and maintenance. Constraints are commonly used to enforce conditions over access control policies. For RBAC, these constraints take the form of both static and dynamic *separation of duty* (a user cannot be assigned to two roles simultaneously) and *prerequisites* (a user can only be assigned to a role if they have already been assigned another required role). When it comes to RBAC research efforts have primarily focused on:

- modelling RBAC entities as classes;
- examining the pros and cons of *roles as classes* versus *roles as instances*;

[1] DAML+OIL, http://www.w3.org/TR/daml+oil-reference

– adapting the eXtensible Access Control Markup Language to work with roles;
– using the Common Information Model for RBAC.

RBAC entities as classes. Di et al. [24] provide a basic modelling for RBAC concepts and constraints using OWL[2]. `User`, `Role`, `Permission` and `Session` entities are represented as classes. While, the following properties are used to represent relationships:

– `hasRole` (assigns a user to a role);
– `hasPermission` (associates a role with a permission);
– `belongTo` (maps a session to a single user); and
– `hasActiveRole` (maps a set of roles to a session).

Two additional properties are used to model separation of duty and prerequisite constraints:

– `conflictRole` (indicates that there is a conflict between two roles); and
– `prerequesiteRole` (specifies that one role is dependent on another).

Roles as classes versus roles as instances. Finin et al. [29] discuss the advantages and disadvantages of two different approaches for representing RBAC policies in OWL. The first approach which represents *roles as classes*, and the second approach which represents *roles as instances*. In both scenarios `Subject`, `Object` and `Action` entities are represented as classes. The authors propose two disjoint `Action` subclasses, `PermittedAction` and `ProhibitedAction`, which are used to associate permissions and prohibitions with users.

– When ***roles are represented as classes*** a general `Role` is defined as a class and specific roles are defined as subclasses of the `Role` class. The `rdf:type` property is used to associates users with roles and an `activeForm` property is used to indicate if a role is currently active. The role hierarchy is modelled using the `rdfs:subclassOf` property. While, the `owl:disjointWith` property is used to model separation of duty constraints.
– When ***roles are represented as instances*** a general `Role` is defined as a class and specific roles are defined as instances of the `Role` class. The `owl:TransitiveProperty`, which is used to describe a transitive relationship between properties, is used to model role hierarchies. Two properties `role` and `activeRole` are used to assign roles to users and two additional properties `ssod` and `dsod` are used to represent static and dynamic separation of duty.

When *roles are represented as instances* the modelling is simple and more concise. Whereas, when *roles are represented as classes*, it is possible to determine subsumption relationships according to a users active role and the role hierarchy, using standard description logic subsumption. As OWL is monotonic, using either approach, it is not possible to remove assertions from the knowledge base, as such role deactivation cannot be supported.

Using the eXtensible Access Control Markup Language for RBAC. Ferrini and Bertino [28], demonstrate how the eXtensible Access Control Markup Language (XACML)[3], an attribute based access control language and framework, proposed by the Advanced Open Standards for the Information Society (OASIS), and OWL can be used to specify and enforce RBAC. Policies are specified using the XACML vocabulary, whereas role hierarchies and constraints are represented using OWL. A request is composed of one or more attributes pertaining to the *subject*, *action*, *resource* or the *environment*. In order to support RBAC the user submits their role as a subject attribute. On receipt of the request the system extracts the role from the subject attribute and the description logic reasoner is used to retrieve additional roles that can be inferred from the OWL ontology. These roles are subsequently fed into the XACML engine. Finally the XACML engine consults the XACML policy in order to determine if access should be granted. As the policies are not modelled in OWL it is possible to support role deactivation. However, with the existing modelling it is not possible to exploit reasoning over policy resources or access rights.

Using the Common Information Model for RBAC. Alcaraz Calero et al. [2] demonstrate how the Common Information Model (CIM)[4] (a standard vocabulary used to represent in-

[2]OWL, `www.w3.org/TR/owl-ref/`

[3]XACML,`https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`

[4]CIM, `http://www.dmtf.org/standards/cim`

formation technology objects and the relationship between them) can be used to represent RBAC. The authors provide a mapping between RBAC concepts and CIM vocabulary which is represented using OWL. Constraints are represented as rules using the Semantic Web Rule Language (SWRL)[5]. For example, in order to model a separation of duty constraint, a rule is defined which checks if there are any common instances between roles and if so generates an exception. The authors describe how a SPARQL enabled reasoner, such as Pellet, can be used to query the OWL policy, in order to determine if access should be granted or denied.

2.1.4. View Based Access Control

VBAC [37] is used in relational databases to simultaneously grant access to one or more relations, tuples, attributes or values. A similar approach is used in Object Oriented Access Control (OBAC) [27], where access rights are granted to sets of application objects. Primary research efforts with respect to VBAC and RDF focus on:

- using rules to grant and deny access to views of the data;
- using SPARQL `CONSTRUCT` and `DESCRIBE` queries to grant and deny access at multiple levels of granularity.

Using rules to create views. Li and Cheung [57] use rules to generate views of data based on relations between ontology concepts. An access control policy is defined as a tuple $\langle s, v, sign \rangle$, where s denotes the *subject*, v represents a set of *concepts, relations and filters*, and *sign* is used to indicate *permissions and prohibitions*. Both views and queries that are also generated from sets of concepts, relations and filters are represented using rules.

Muhleisen et al. [60] allow access control policies to be specified for *triple patterns*, *resources* or *instances* using SWRL rules. When a requester submits a query, the system uses the access control rules to generate a temporary named graph containing only authorised data. The requesters query is subsequently executed against the temporary named graph and the results are returned to the user.

Using SPARQL to create views. Both Dietzold and Auer [25] and Gabillon and Letouzey [33] describe how RDF data can be logically organised into views using SPARQL `CONSTRUCT` and `DESCRIBE` queries.

Dietzold and Auer [25] propose access control policy specification at multiple levels of granularity (*triples*, *classes* and *properties*). Authorisations are used to associate filters (SPARQL `CONSTRUCT` queries) with users and resources. When a requester submits a query, a virtual model is created based on the matched authorisations. The query is subsequently executed against the virtual model, which only contains data that the requester is authorised to access.

Gabillon and Letouzey [33] also propose an access control model which can be used to grant/deny access to named graphs or SPARQL views. Access control policies are specified using contextual information pertaining to the *user*, *resources* or the *environment*. The body of the rule is a possibly empty condition, or a combination of conditions connected via conjunction or disjunction. The head of the rule is an authorisation.

2.1.5. Attribute Based Access Control

ABAC was designed for distributed systems, where the subject may not be known to the system, prior to the submission of a request. ABAC grants or denies access to *resources*, based on properties of the *subject* and/or the *resource*, known as *attributes*. Primary research efforts to date focused on:

- using rules to grant access based on attributes directly or indirectly using roles;
- using ontologies to grant access based on attributes directly or indirectly using roles;
- proposing a pattern which can be used to guide the development of attribute based access control frameworks.

Using rules for ABAC. Stermsek et al. [85] discuss how attributes can be used to specify access control directly using rules and indirectly using roles. In the former, the requesters attributes are compared against a policy, which indicates the attributes necessary to access a resource. Whereas in the latter, permissions are assigned to roles and the requesters attributes are used to determine the access rights or roles the subject should be mapped to. The authors, describe three different mechanisms that can be used to obtain the attributes pertaining to a subject:

(i) subjects can send all of their attributes to the server, with the initial request;

[5] SRWL, `http://www.w3.org/Submission/SWRL/`

(ii) the server can request particular attributes from the subject, once the initial request is submitted; and
(iii) given the subject may be cautious about giving out the requested credentials to an unknown entity, both the server and the client could exchange policies and attributes (a process commonly known as trust negotiation).

Although a high level architecture is described, the formal representation of the model is left to future work.

Combining roles and attributes. Cirio et al. [15] propose an access control model which combines role and the attribute based access control. Rather than associating permissions with attributes directly, like Stermsek et al. [85] the permissions are associated with roles and attributes are added to roles. The authors use `rdfs:domain` and `rdfs:range` properties to assert knowledge instead of using them as constraints. Such an approach simplifies policy specification as concepts can be defined implicitly. Given that it is not possible to specify policies based on relations between instances using description logic, the authors use two predicates `requiresTrue` and `requiresFalse` to specify run time constraints based on user attributes. Constraints are represented using SPARQL `ASK` queries and are evaluated at runtime using a custom policy decision point, which wraps a description logic reasoner.

A Metadata-Based Access Control design pattern. Priebe et al. [66], inspired by software design patterns, present a Metadata-Based Access Control (MBAC) pattern, which aims to encapsulate best practice with respect to attribute based access control. In the presented pattern subjects and objects are modelled as sets of attribute/value pairs. While, authorisation subjects and authorisation resources are described in terms of required attributes and corresponding values. Follow up work, by Priebe et al. [67], describes how attribute based access control policies specified using XACML can be modelled using OWL. The XACML policies together with a reasoning engine, allows for deductive reasoning based on the OWL vocabulary.

2.1.6. Context Based Access Control

CBAC uses properties, pertaining to *users*, *resources* and the *environment*, to grant/deny access to resources. In light of new and emerging human computer interaction paradigms, such as ubiquitous computing and the internet of things, access control based on context has been graining traction in recent years. Existing proposals for CBAC can be summarised as follows:

– proposing a framework which differentiates between physical and logical context;
– using ontologies and rules to cater for access control policies that take context relating to subjects, objects, transactions and the environment into consideration;
– using ontologies and SPARQL `ASK` queries to restrict access to named graphs.

Physical versus Logical Context. Corradi et al. [16] and Montanari et al. [59] propose a context based access control model and framework, called UbiCOSM. The proposed access control model uses context to group policies. The authors distinguish between physical and logical context. The former relates to the physical location denoted by geographical coordinates. Whereas, the latter refers to logical properties pertaining to the users and resources. Access control policies are composed of mappings between *context* and *permissions*. Complex policies are generated using conjunction, disjunction and negation operators. When a user requests access to a resource, the UbiCOSM enforcement framework retrieves the relevant policies and generates a view based on the users permissions.

Context relating to the subject, object, transaction and environment. Shen and Cheng [83] propose a semantic-aware context-based access control model (SCBAC) and demonstrate how together ontologies and rules can be used to generate authorisations. The authors provide a context ontology and a policy ontology, which can be used to specify positive and negative authorisations and obligations. Like Corradi et al. [16] and Montanari et al. [59], context provides a level of indirection between subjects and permissions. The authors propose four different types of context that are associated with subjects, objects, translations and the environment. An authorisation *permits/prohibits* an *action*, based on *sets of contexts* supplied by the user. Rules are used to insert new authorisations, based on contextual information, into a knowledge base. An access request is represented as a tuple $\langle U, P, C, R\rangle$ where *user* U, requests *privilege* P on *resource* R, in light of a given *context* C. Access is enforced by rep-

resenting access requests as SPARQL queries that are executed over the knowledge base. However, given OWL is monotonic, it is not clear how changes to contextual information are handled in the proposed approach.

Using policies to limit access based on contextual conditions. Costabello et al. [17] propose a policy language that can be used to restrict access to named graphs using contextual information supplied by the requester. An access control policy is a tuple $\langle ACS, AP, S, R, AEC \rangle$, where ACS is a set of *access conditions* (specified using SPARQL `ASK` queries); AP is a set of *access privileges* (`CREATE`, `READ`, `UPDATE` or `DELETE`); S denotes the *subjects* to be protected; R represents the *named graphs* to be protected; and AEC is the evaluation context specified using name value pairs (verified using SPARQL `BINDINGS`). In addition to the SPARQL query that the user wishes to execute, the user provides their access credentials, in the form of a SPARQL `UPDATE` query, which contains contextual data. The enforcement framework stores the contextual data in a named graph and retrieves the authorisations that match the query type. In order to determine if access is permitted, the `ASK` query and the `BINDINGS`, that are specified in the authorisation, are executed against the users contextual graph. If the `ASK` query returns true then the query is rewritten to include the corresponding named graph.

2.2. Access Control Standardisations

In recent years, there have been a number of standardisation efforts, by both the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS), in relation to access control for web data. This section provides a high level overview of the relevant standards and details how they have been adapted or enhanced using semantic technology.

2.2.1. eXtensible Access Control Markup Language

The eXtensible Access Control Markup Language (XACML)[3], is an OASIS standard, which is used to represent attribute based access control policies [71]. XML was chosen as the representation formalism for the policy language as it:

- can be used to represent information in a human and machine readable manner;
- can easily be adapted to represent different access control requirements; and
- is widely supported by software vendors.

The specification provides an XML schema which can be used to represent attribute based access control policies. The root of an XACML policy is a `Policy` or a `PolicySet` (used to represent multiple policies). Policies are composed of sets of `Rules` that are in turn composed of sets of `Targets` (conditions relating to `Subjects`, `Resources` and `Actions`) and an `Access Decision` (permit, deny or not applicable). A `Request` is represented as a tuple $\langle subject, resource, action, environment \rangle$. Where *subject* represents the entity requesting access, *resource* denotes the object to be protected, *action* defines the type of access and *environment* represents the requesters attributes. The XACML framework is composed of the following components:

- a *policy decision point*, which evaluates policies and returns a response;
- a *policy enforcement point*, which is responsible for making decision requests and enforcing the decisions;
- a *policy information point*, which obtains attributes pertaining to subjects, resources and the environment; and
- a *policy administration point*, which enables policies and sets of policies to be created and updated;

A number of researchers have used Semantic Web technologies to supplement the existing XACML framework. Primary research efforts include:

- extending XACML to cater for deductive reasoning over attribute ontologies;
- extending XACML policies with context;
- adapting XACML to work with roles.

Deductive reasoning over XACML policies. Priebe et al. [67,68] present an extension to XACML, which enables deductive reasoning over attribute ontologies, specified using OWL. In order to support reasoning, the authors propose two additional architecture components: an *ontology administration point* and an *inference engine*. Attribute ontologies are created and updated using the *ontology administration point*. If the attributes presented by the requester are not explicitly stated in the access control policy, the system attempts to infer the required access rights from the policy attributes, requester attributes and the attributes ontology, using the *inference engine*.

Chen and Stuckenschmidt [14] also extend XACML with OWL deductive reasoning capabilities. XACML policies are specified using OWL and access control is enforced via query rewriting. The authors use SPARQL filters to both permit and deny access to instance data. Reasoning over the data, represented in the filters, is delegated to reasoners which support OWL and SPARQL.

Extending XACML Policies with Context. Franzoni et al. [32] demonstrate how XACML can be extended to consider access control based on contextual properties pertaining to either the user or the application. In addition to standard access control policies, specified using XACML, the authors propose fine grained access control policies, which are used to specify the *instances* of a *concept* that a user is permitted to access. The proposed fine grained access control policies are enforced over RDF data, by expanding a SeRQL query (an alternative to SPARQL), to include triple patterns for the instances that are permitted.

Supporting RBAC using XACML. Ferrini and Bertino [28] describe an extension of XACML, which uses a combination of XACML and OWL, in order to support RBAC constraints, such as static and dynamic separation of duty. Like Franzoni et al. [32], XACML policies are specified using OWL, therefore it is possible to take advantage of OWL's out of the box reasoning capabilities. The authors propose a two layer framework where access control policies are specified using XACML and constraints and role hierarchies are represented using OWL. They further extend the XACML enforcement framework which enhances the XACML engine with Description Logic reasoning capabilities.

2.2.2. Web Identity and Discovery

Web Identity and Discovery (WebID)[6], which is supported by a W3C community group, is a mechanism used to uniquely identify and authenticate a person, company, organisation or other entity, by means of a Uniform Resource Identifier (URI) [80]. Essentially a WebID is a HTTP URI which is used to represent an agent. A description of the agent is provided in an RDF document, known as a WebID profile, which can be dereferenced using 303 redirects or Hash URI's. The WebID-TLS protocol (where TLS stands for Transport Layer Security) specifies how together a WebID profile and a public key certificate can be used to authenticate users [40]. The user places their WebID profile document URI in the *Subject Alternative Names* field of their certificate. Once the certificate has been generated the user adds the public key details to their WebID profile document. A service wishing to authenticate the user, needs to verify that the public key of the certificate it receives matches the public key specified in the WebID profile.

Hollenbach et al. [39] use FOAF[7] and the Secure Sockets Layer (SSL) to determine if the public key in the users FOAF profile matches that of the certificate. When a requester attempts to authenticate, the system extracts the public key from the certificate. The system subsequently verifies the signature and if successful, a SPARQL query is executed against the FOAF profile, in order to verify that it contains a matching public key.

In Berners-Lee et al. [5], the authors describe their vision of a read-write web of data and present a proof of concept. Like Hollenbach et al. [39], the authors discuss how WebID together with FOAF+SSL, can be used for authentication.

Although Stermsek et al. [85] do not use the term WebID, they describe how attributes pertaining to a subject (commonly known as credentials), can be associated with public keys and attached to digital certificates.

2.2.3. Web Access Control

WebAccessControl (WAC)[8] is an RDF vocabulary and an access control framework, which demonstrates how together WebID and access control policies specified using the WAC vocabulary, can be used to enforce distributed access control. WAC authorisations grant *agents*, *access* to *resources*. Agents are specified using the `agent` and `agentClass` properties and resources are specified using the `accessTo` and `accessToClass` properties. Whereas, `Read`, `Write`, `Append` and `Control` access rights are represented as classes. Once the user has been authenticated using WebID, the system checks if a policy exists which grants the user access to the requested resource. If no such policy exists, then the system checks for classes that are granted access.

[6] WebID, `http://www.w3.org/wiki/WebID`

[7] FOAF, http://xmlns.com/foaf/spec/

[8] WAC,`http://www.w3.org/wiki/WebAccessControl`

For each class the system dereferences the URI and checks if the users WedID is a `type` of the given class. If yes, then the user is granted access to the system. When it comes to WAC, primary research efforts focus on:

- demonstrating how WebID and WAC can be used for authentication and authorisation respectively;
- extending the WAC vocabulary to cater for more expressive access control policies.

Using WebID and WAC. In addition to using WebID for authentication, Hollenbach et al. [39] use the WAC vocabulary to specify access control policies. The authors provide a mapping between permissions and HTTP operations and demonstrate how together WebID and WAC can be used to grant/deny access to web resources. In order to verify if the user has the permissions required to perform the requested HTTP operation, a SPARQL query is executed against the access control policy. If access is denied, a 403 response is returned from the server.

Extending the WAC vocabulary. Both Villata et al. [98] and Sacco and Passant [76] extend the WAC to cater for access control over the RDF data model. Using the extended vocabularies, it is possible to associate access control with individual RDF resources (subjects, predicates and objects) and also collections of RDF resources (named graph). In addition, the authors extend the vocabulary to cater for a broader set of access privileges (`create`, `read`, `write`, `update`, `delete`, `append` and `control`).

2.2.4. Platform for Privacy Preferences

Platform for Privacy Preferences (P3P)[9], is a W3C recommendation, which enables websites to express their privacy preferences in a machine readable format. Like *XACML*, the specification provides an XML Schema, which can be used to specify policies. In addition, the specification details how privacy policies can be associated with webpages/websites and describes how P3P policies can be used in conjunction with HTTP. Organisations wishing to specify machine readable privacy policies can publish their privacy policies using the P3P syntax. A reference to the policy can be added to a well known location (for example, `/w3c/p3p.xml`), which can be specified using the HTML `link` tag, or alternatively can form part of the HTTP Response. P3P agents can be built into browsers, plugins or proxy servers. The agent is responsible for fetching the servers privacy preferences and taking some action. This action can vary from simply displaying a symbol, to comparing the servers privacy preferences to those of the client and taking some form of action. A related specification called *A P3P Preference Exchange Language (APPEL)* [20] presents a vocabulary, which is used by individuals (as opposed to websites) to express their privacy preferences. When it comes to P3P and RDF, primary research efforts focus on:

- extending the P3P vocabulary to cater for more expressive privacy preferences;
- representing P3P policies using OWL.

Extending the P3P vocabulary. Kolari et al. [55] propose an extension to P3P, to cater for more expressive privacy preferences. The P3P policies are specified using a policy language based on Semantic Web technologies, known as Rei. As Rei is a general policy language, it can easily be used to represent existing P3P policies in a manner which supports reasoning based on context. As access rights in Rei are based on deontic logic, it is possible to model, not only positive and negative permissions, but also positive and negative obligations.

Representing P3P policies using OWL. Garcia and Toledo [34] demonstrate how P3P policies can be represented in OWL. The authors detail how the Web Services Policy Framework (WS-Policy) can be used by service providers and consumers to specify their privacy policies using OWL ontology concepts based on the P3P vocabulary. Policies are associated with web services and access is determined by a broker by comparing consumer privacy preferences with provider privacy guarantees, taking into account the semantic relationships between ontology concepts.

2.2.5. The Open Digital Rights Language

The Open Digital Rights Language (ODRL)[10] is a general rights language, which is used to define rights to or to limit access to digital resources. The information model and the associated semantics is documented in the core model. Whereas, the common vocabulary provides a dictionary of terms that can be used to express permissions, prohibitions, constraints,

[9] P3P, `http://www.w3.org/TR/P3P/`

[10] ODRL, `https://www.w3.org/community/odrl/`

and duties with respect to digital assets. In addition, a number of serialisations of the core model are provided e.g. XML, JSON and RDF. The core model may be amended/extended in order cater for disparate policy requires (i.e. different communities may define their own ODRL profiles). When it comes to ODRL and the RDF specifically, primary research efforts to date focus on:

- using ODRL vocabularies to specify RDF licenses;
- demonstrating how ODRL can be used to express a variety of access policies.

Using ODRL vocabularies to specify RDF licenses. Cabrio et al. [13] demonstrate how together the Creative Commons Rights Expression Language Ontology can be used to specify Creative Commons (CC) licenses and how the ODRL Ontology can be used to model all other licenses. In addition, the authors use Natural Language Processing (NLP) techniques to automatically generate machine readable licenses (represented using RDF) from their natural language counterparts.

Using ODRL vocabularies to specify Linked Data access policies. Steyskal and Polleres [86] examine the suitability of ODRL for specifying access policies for Linked Data. The authors present several motivating scenarios and demonstrate via example how ODRL can be used to:

- restrict access to specific datasets;
- limit the number of permissible requests;
- grant or deny access for a specific time window;
- represent common licenses;
- limit data reuse;
- define policies that require the payment of duties.

3. Policy Languages and Frameworks

Policy languages can be categorised as either general or specific. In the former, the syntax caters for a diverse range of functional requirements (access control, query answering, service discovery, negotiation, to name but a few), whereas the latter focuses on just one functional requirement. Two of the most well-known access control languages, KAoS [12,11, 93] and Rei [46,48], are in fact general policy languages. Natural language, programming languages, XML and ontologies can all be used to express policies. XML and ontologies are two popular choices for representing policy languages as they benefit from flexibility, extensibility and runtime adaptability. However, ontologies are better suited to modelling the semantic relationships between entities. Also, the common framework and vocabulary used by ontologies, to represent data structures and schemas, provides greater interpretability and interoperability. Regardless of the language chosen, a logic based underlying formalisation is crucial for automatic reasoning over access control policies.

The work presented in this section is limited to policy languages that use ontologies, rules or a combination of both to represent general policies. As the objective is to provide the reader with an overview of each approach, a detailed description of well known frameworks in each category is presented. For a broader comparison of policy languages, the author is referred to a survey by Bonatti and Olmedilla [10], which evaluates twelve different policy languages against a set of criteria, that are deemed necessary for ensuring security and privacy in a Semantic Web context.

3.1. Ontology Based Approaches

Using ontologies it is possible to specify access control vocabularies that can easily be adopted by others. Additionally, access control policies specified using different vocabularies can easily be merged. Furthermore, it is possible to infer new policies based on relationship between access control entities (deductive reasoning) and determine the access rights required to meet a given policy (abductive reasoning) over access control policies, with standard description logic reasoners. This section examines KAoS [12,11,93] a general policy language which adopts a pure ontological approach.

3.1.1. KAoS

KAoS [12,11,93] is an open distributed architecture, which allows for the specification, management and enforcement of a variety of policies. In initial versions of the language, policies were represented using DAML[11]. However, the authors later moved to OWL, the successor of DAML [94]. As both DAML and OWL are based on description logic, using the KAoS language it is possible to define class

[11]DAML, http://www.daml.org/

and property hierarchies, along with inference rules. Although KAoS was originally designed to enable interoperability between complex web agents (software that acts on behalf of humans) [12,88], it was later applied to web services [96,95,94,97] and grid computing [45,96].

Specification of policies. The authors define a set of core vocabularies, known as KAoS policy ontologies, that is used to specify policies. A policy is used to express either an *authorisation* or an *obligation*, on the part of one or more *actors*, with respect to *actions* relating to *resources*. Policies are represented as instances of the aforementioned policy types. The language is not meant to be exhaustive, but rather to provide a basis, which can be further extended, to cater for use case specific classes, instances and rules. In order to simplify policy administration and enforcement, actors and resources are organised into domains, that can be nested indefinitely. Domains and subdomains are used to represent complex relationships between classes and instances, such as organisation structures.

Enforcement of policies. The authors propose a general policy and domain services framework, which consists of the following components: a *policy administration tool*, *directory services*, *guards*, *enforcers* and a *domain manger*. The *policy administration tool*, known as KPAT, is a user friendly interface that allows administrators, who are unfamiliar with DAML and OWL, to either specify new or maintain existing policies. *Guards* are responsible for enforcing platform independent policies. While, *enforcers* are responsible for enforcing policies that are platform dependent. The *domain manger* is used to manage domain membership and to distribute policies to guards. This component is also responsible for notifying the *guards* of any policy updates. Given that the actual enforcement may depend not only on the action to be performed, but also on the application, it may be necessary for the developer to implement platform specific code. In order to simplify the integration of these custom enforcement mechanisms with the KAoS framework a number of interfaces are provided as a guide for developers.

In a follow up paper [92], the authors discuss how description logic can be used to support policy administration, exploration and disclosure. Administration is primarily concerned with subsumption based reasoning and the determination of disjointness. Using abductive reasoning it is possible to both test constraints, and to return relevant constraints given one or more properties. Using deductive reasoning it is possible to identify and resolve conflicts at design time. The authors propose a general algorithm for conflict resolution and harmonisation, which can be used even when the entities (`actors`, `actions`, `resources` and `policies`) are specified at different levels of abstraction. The proposed conflict resolution strategy is based on policy priorities and timestamps. In the event of a conflict the algorithm takes the policy with the lowest precedence and subdivides it until the conflicting part has been isolated. The conflicting policy is removed, and non conflicting policies are generated and feed into the knowledge base.

3.2. Rule Based Approaches

One of the benefits of a rule based approach is that it is possible to support access control policies that contain instance dependencies or variables. Like ontology based approaches, access control policies are defined over ontology entities. Therefore access control policies specified using different vocabularies can easily be integrated. This section examines two different rule based languages and enforcement frameworks, Rei [46,48] and Protune [8,7,10].

3.2.1. Rei

Rei [46,48] is a Semantic Web policy language and distributed enforcement framework, which is used to reason over policies, that are specified using RDFS or Prolog rules. As OWL has a richer semantics than RDFS, the authors later provided an OWL representation for their policy language [23,49]. Like KAoS, Rei is a general policy language which can be applied to agents and web services [23,47]. Although Rei policies can be represented using RDFS or OWL the authors adopt a rule based enforcement mechanism, in contrast to the description logic enforcement mechanism adopted by KAoS.

Specification of policies. The authors propose a number of core ontologies that are used to describe access control entities. Like KAoS, Rei provides support for four distinct policy types, *permissions*, *prohibitions*, *obligations* and *dispensations*. By choosing to represent access rights as speech acts Rei is able to support not only a wide range of policies but also the del-

egation and revocation of policies. A policy is composed of a set of rules, based on the four policy types, that are used to associate conditions with actions. A `has` predicate is used to associate *permissions* and *obligations* with *entities*. Like KAoS, the core ontologies can be further extended to meet the requirements of specific use cases.

Enforcement of policies. The Rei policy framework, called Rein (Rei and N3) [47,49], consists of the following components:

- *a set of ontologies*, used to represent Rein policy networks (resources, policies, meta-policies and the Rein policy languages) and access requests; and
- *a reasoning engine*, that uses both explicit and derived knowledge to determine if a request should be granted or denied.

The authors propose a distributed enforcement architecture, whereby each entity is responsible for specifying and enforcing their own policies. Rein is capable of acting as a server or a client. In server mode, Rein retrieves the relevant policies; requests the credentials necessary to access the resource; and verifies the credentials against the policies. Whereas in client mode, the server returns a link to a policy which the client must satisfy; the Rein client generates a proof that the requester can satisfy the policy; and forwards the proof to the server. In order to cater for scenarios where part of the policy is private and part is public, the authors propose a hybrid approach, where Rein acts both as a client and a server.

Using Rein it is possible to combine and reason over different access control policies, meta-policies and policy languages. Policies are expressed using either RDFS or OWL, and inference over both data resources and policies is performed using an N3 reasoner, known as Cwm[12]. N3 was originally used as a representation syntax for RDF, however it was later extended to allow for variables and nested graphs. Cwm extends N3 with inference rules and built-in functions, making it possible to express relationships between graphs, specify both existential and universal constraints and to represent implication. Although the authors demonstrate how the Rei vocabulary can be used to specify policies, these policies could be represented using alternative vocabularies.

In Kagal et al. [48], the authors discuss how conflict resolution can be achieved using meta-policies. Priority policies are used to indicate dominance between policies or policy rules. While, precedence policies are used to specify a default grant or deny, for *policies*, sets of *actions* or sets of *entities* satisfying specific conditions. In order to guarantee that a decision can always be reached, the authors propose a partial order between meta-policies. Given Rei allows for *policies* to contain variables, conflicts need to be resolved at run-time, as opposed to design-time, in the case of KAoS.

3.2.2. Protune

Protune [8,7,10] is a policy language which was proposed by the Research Network of Excellence on Reasoning on the Web, known as REWERSE[13]. Like Rei, Protune adopts a rule based approach to policy enforcement. The authors identify *usability* as one of the primary factors for a policy aware web. To this end, Protune was designed to support both trust negotiation and policy explanations. Lightweight ontologies are used to represent concepts, the relationships between these concepts and details of the evidences needed to prove their truth. Protune is an extension of two other well known policy languages, the Portfolio and Service Protection Language (PSPL) [9] and PeerTrust [35]. PSPL is a model and framework, which uses rules to support policy filtering, policy exchange and information disclosure. Whereas, PeerTrust is a language and a framework, which uses semantic annotations and access control rules, in order to cater for automated trust negotiation and access control.

Specification of policies. Protune policies are specified using rules and meta-rules (essentially horn clauses with some syntactic sugar), which provide support for both deductive and abductive reasoning. The former is used to enforce policies, whereas the latter is used to retrieve information about the policy conditions that need to be satisfied. Protune provides three predicate categories (*decision* predicates, *provisional* predicates and *abbreviation* predicates).

- *Decision predicates* are used to specify the outcome of a policy.

[12]Cwm, http://www.w3.org/2000/10/swap/doc/cwm.html [13]REWERSE, http://rewerse.net/

– *Provisional predicates* are used to represent the conditions the requester must satisfy. By default the system supports two conditions: requests for credentials and request for declarations. Both credentials and declarations are used to assert facts about the requester, however credentials are certified by a third party, whereas declarations are not.
– *Abbreviation predicates*, which are composed of one or more provisional predicates, are used to represent abstractions of the conditions listed in the body of the rule, simplifying policy specification and maintenance.

It is also possible to extend the language with custom predicate categories. Ontologies are used to associate evidences (descriptive requirements of what is needed to meet the conditions) with access conditions. Evidences of this nature facilitate negotiation.

Enforcement of policies. The enforcement framework is composed of three separate components, a *negotiation handler*, an *execution handler* and an *inference engine*.

– The *negotiation handler* is responsible for sending conditions to the requester and providing responses to conditions that were requested.
– The *execution handler* is used to interact with external systems and data sources.
– The *inference engine* is tasked with both enforcing policies (deduction) and retrieving evidences (abduction).

Like Rei, Protune can be used as a client, as a server, or both. Protune explanations are provided by a component known as Protune-x, which supports four different types of queries:

– *How-to queries* (provide a description of the policy).
– *What-if queries* (give foresight into potential policy outcomes).
– *Why queries* (give explanations for positive outcomes).
– *Why-not queries* (give explanations for negative outcomes).

Protune is developed in Java with a Prolog reasoning component, which is compiled into Java byte code.

3.3. Hybrid Approaches

A hybrid approach to policy specification and enforcement can be used to exploit the out of the box deductive capabilities, of an ontology based approach, and the runtime inference capabilities, of a rule based approach. This section describes Proteus [90] which uses a combined approach to policy enforcement and examines an alternative approach, presented by Kolovski et al. [56] which demonstrates how description logic based access control policies can be extended with defeasible logic rules.

3.3.1. Proteus

Proteus [90] uses a hybrid approach to access control policy specification. The authors examine early versions of KAoS and Rei, and highlight the strengths and weaknesses of both ontology based and logic based policy languages and frameworks. Like KAoS the authors use ontologies to model both domain information and policies. Such an approach allows for conflict resolution and harmonisation at design time. Like Rei, the authors adopt a rule based approach in order to support dynamic constraints and run time variables. For example, to support access control based on dynamic context pertaining to the requester or the environment. Like Protune, policy descriptions are used to facilitate partial policy disclosure and policy negotiation.

Specification of policies. *Policies* are represented as classes and contextual information, relating to the user, are represented as instances. Description logic deduction is used to determine the policies that are relevant for the instance data supplied. However, using description logic reasoning it is not possible to cater for contextual properties that are based on property paths or that are associated with variables. In order to handle reasoning of this nature, the authors propose context aggregation and context instantiation rules. Such rules are represented as horn clauses, with predicates in the head and ontological classes and properties in the body.

Enforcement of policies. The Proteus policy framework [91] is composed of the following core components: a *policy installation manager*, a *reasoning core*, a *policy enforcement manager* and a *context manager*.

– The *policy installation manager* is responsible for loading ontologies, access control

policies, contextual information and quality constraints.
- The *reasoning core* performs reasoning over policies, context and quality constraints in order to determine which policies are currently active.
- The *policy enforcement manager* intercepts action requests, collects relevant contextual information and interacts with the *reasoning core* in order to determine if access should be granted or denied.
- The *context manager* collects state information pertaining to system entities and forwards this contextual information to the *reasoning core*.

The authors provide details of their prototype which is implemented in Java with a Pellet reasoner. The proposed solution supports incremental reasoning via an OWL application programming interface and SPARQL queries.

3.3.2. Kolovski et al. [56]

Kolovski et al. [56] demonstrate how together description logic and defeasible logic rules, known as defeasible description logic [36], can be used to understand the effect and the consequence of sets of XACML access control policies.

Specification of policies. The XACML policies are represented using description logic and the combination algorithm, which is used to handle conflicts is represented using defeasible description logic rules. The formalism supports both strict rules that cannot be overridden and defeasible rules that may be overridden by a higher priority rule.

Enforcement of policies. Although the actual enforcement framework is not presented, the following subset of policy services are described:

- **Constraints.** Like Finin et al. [30,29] the proposed solution caters for role cardinality and separation of duty;
- **Comparison.** Policies or sets of policies can be compared in order to determine if one is equivalent to or logically contains the other;
- **Verification.** Like Bonatti and Olmedilla [10], this component checks if the policy satisfies a given property;
- **Incompatibility.** This component provides details of policies that cannot be active at the same time;
- **Redundancy.** This component checks hierarchies to ensure that all policies are reachable; and
- **Querying.** Given a set of attributes, this component searches for relevant policies.

The proposed XACML analysis prototype is implemented on top of Pellet (an open source description logic reasoner).

4. Access Control for RDF

This section, presents the different access control mechanisms that have been used to protect RDF data. In particular, it focuses on the specification of access control policies, the simplification of administration using different reasoning strategies and the different techniques used to return partial query results.

4.1. Specification of Access Control for RDF

Over the years several researchers have focused on the modelling and enforcement of access control over RDF data. A number of authors [70,41,1,31] define access control policies based on RDF patterns, that are mapped to one or more RDF triples. Others [57,33,17] propose view based access control strategies for distributed RDF data.

4.1.1. RDF Patterns

Reddivari et al. [70] define a set of actions required to manage an RDF store and demonstrate how access control rules can be used to permit or prohibit the requested actions. The actions are organised into four categories:

- `adding` (the insertion of explicit triples, implicit triples and sets of triples);
- `deleting` (the deletion of explicit triples, implicit triples and sets of triples);
- `updating` (directly replacing one triple with another); and
- `querying` (returning triples or using triples to return answers to queries).

Policies are defined using Prolog facts and rules and compiled into Jena rules. Two predicates `permit` and `prohibit` are used to grant and deny access rights based on the aforementioned actions, to one or more triples using triple patterns. Authorisations can be further constrained using conditions relating to policies, triples and agents:

- *policy specific conditions* relate to the access control policies, for example a user can only add instances if they added the class.
- *triple specific conditions* correspond to the triple specified in the authorisation, for example if an authorisation governs a triple then all triples associated with a subProperty relation are governed by the same policy.
- *agent specific conditions* use properties of the user to limit the authorisation, for example it is possible to limit access to users who are managers in a specific company division.

The proposed RDF Store Access Control Policies (RAP) framework checks the policy to ensure that the action is permitted, temporarily allows the action, and afterwards checks the policy to ensure that the inferences are allowed. The authors propose default and conflict preferences that can simply be set to either permit or deny.

Flouris et al. [31] also use RDF triple patterns to expose or hide information represented as RDF. Although the authors go beyond simple graph patterns by allowing the graph pattern to be constrained by a `WHERE` clause, no consideration is given to either data or policy inference. Like Reddivari et al. [70], the authors propose a default policy and a conflict resolution strategy. They formally define the semantics of the individual access control statements and the entire access control policy, and present the different possible interpretations for the default semantics and the conflict resolution. A flexible system architecture that demonstrates how the access control enforcement framework can be used with disparate RDF repositories and query languages is presented. The system is implemented using Jena ARQ, Jena SDB with a Postgresql back-end and Sesame.

Both Jain and Farkas [41] and Abel et al. [1] also use triple patterns to specify access control policies. Information on their approaches is presented under reasoning (*Section* 4.2) and query rewriting (*Section* 4.3), respectively.

4.1.2. Views and Named Graphs

Gabillon and Letouzey [33] highlight the possible administration burden associated with maintaining access control policies that are based on triple patterns. They propose the logical distribution of RDF data into SPARQL views and the subsequent specification of access control policies, based on existing RDF graphs or predefined views. Access control policies are specified using contextual information pertaining to the user, resources or the environment. The body of the rule is a possibly empty condition, or a combination of conditions connected via conjunction or disjunction. The head of the rule is an authorisation. The authors describe an enforcement framework, whereby users define security policies for the RDF graph/views that they own. Users may delegate rights to other users by specifying an authorisation which grants `construct` and `describe` privileges to the RDF graph or view. Although the authors acknowledge the need for conflict resolution, they do not propose a conflict resolution strategy.

4.1.3. Ontology Concepts

Sacco and Passant [76,77] and Sacco et al. [74] demonstrated how an extension of the Web Access Control vocabulary known as the Privacy Preferences Ontology (PPO) can be used to restrict access to an RDF resources, statements and graphs. An access control policy is composed of:

- a restriction in the form of an RDF resource, statement or graph;
- a condition which provides specific details of the restriction, for example `hasProperty`, `hasLiteral`;
- an access privilege, for example `read` and/or `write` access rights; and
- a SPARQL `ASK` query that must be satisfied by the requester.

The authors describe the formal semantics of the PPO and present a detailed description of their Privacy Preferences Manager (PPM), which can be used to enforce access control using SPARQL `ASK` queries. A follow-up paper Sacco and Breslin [75] extends the original PPO and PPM to allow access to be restricted based on a dataset or a particular context. The authors also provide support for conflict resolution, more expressive authorisations (in the form of negation and logical operators), and a broader set of access privileges (`create`, `read`, `write`, `update`, `delete`, `append` and `control`). In Sacco et al. [78], the authors demonstrate how both the PPO and the PPM can be used to cater for fine grained access control on a mobile device. A number of shortcomings of their original enforcement algorithm are identified and a more efficient algorithm which utilises pre-indexing, query analysis and results filtering is presented and evaluated.

An alternative access control vocabulary called Social Semantic SPARQL Security for Access Control (S4AC) is presented in Villata et al. [98]. Like Sacco and Passant [76] they extend the WAC to cater for fine grained access control over RDF data. Their proposal is tightly integrated with several social web and web of data vocabularies. The authors define access control policies for named graphs, which can also be used to grant/deny access to sets of triples. S4AC provides support for logical operators and a broad set of access privileges (`create`, `read`, `write`, `update`, `delete`, `append` and `control`) from the offset. Like Sacco and Passant [76], SPARQL `ASK` queries are used to determine if the requester has the permissions necessary to access a resource. The authors propose the disjunctive evaluation of policies, thus circumventing the need for a conflict resolution mechanism. Follow up work by Costabello et al. [17,18] describes how an access control framework, called Shi3ld, can be used to enforce access control over SPARQL endpoints in a pluggable manner. In [19] the authors extend the Shi3ld framework to cater for access control for a Linked Data Platform (LDP)[14] [84]. Resources refer simply to Linked Data resources that are queries, created, modified and deleted via HTTP requests processed by a LDP. Two alternative frameworks are presented, one which contains an embedded SPARQL engine and a SPARQL-less solution. In the first scenario, Shi3ld remains unchanged. Whereas in the second scenario, authorisations cannot contain embedded SPARQL queries and therefore are evaluated using subgraph matching. In their evaluation the authors compare all three frameworks, using the Billion Triple Challenge 2012 Dataset[15]. Based on their performance evaluation the authors conclude that access control over SPARQL endpoints is marginally slower than access control over LDP resources and that their SPARQL-less solution exhibits a 25% faster response time.

Steyskal and Polleres [86] discuss how the Open Digital Rights Language (ODRL) 2.0 ontology can be used to specify access policies for Linked Data. In addition to standard access control policies the authors demonstrate how ODRL can be used to represent common licenses and policies that require payment. In a follow-up paper [87], the authors propose a formal semantics for ODRL, which can be used as the basis for rule based reasoning over ODRL policies. Both the enforcement framework and the evaluation of the access policies are mentioned in future work.

4.2. Reasoning over RDF Access Control Policies

Inference is a process whereby new data is derived from data which is known or assumed to be true. *Section* 3 discussed how deduction and abduction can be used to simplify both policy specification and maintenance. However, inference can also be used to deduce information, which users should not have access to, commonly known as the *inference problem*. Both Thuraisingham [89] and Nematzadeh and Pournajaf [62] highlight the need for security mechanisms to protect against such unauthorised inference. Although this is not a new problem, the authors argue that with advances in current data integration and mining technologies, the problem is further magnified. According to Qin and Atluri [69], if the semantic relationship between entities is not taken into account it may be possible to infer information which has been restricted, or access control policies may not exist for the inferred information making this information inaccessible.

When it comes to RDF data existing reasoning strategies focus on the propagation of access rights based on authorisation subjects, access rights and resources [69,43,73, 3] or demonstrate how access rights can be inferred for new triples deduced based on RDFS inference rules [41,50,65,58]. Whereas, Bao et al. [4] present a number of use cases where it is desirable to grant access to data which has been inferred from unauthorised data and present a privacy preserving reasoning strategy.

4.2.1. Propagation of Authorisations

Early proposals for the propagation of authorisations focused on reasoning over ontology concepts [69]. Subsequent work by Javanmardi et al. [44,43] focused not only on ontology concepts but also on reasoning over ontology properties and ontology instances. An alternative strategy which is proposed by Ryutov et al. [72,73], takes a more abstract approach by propagating policies based on nodes

[14]LDP, `http://www.w3.org/2012/ldp/wiki/Main_Page`

[15]BTC2012, `http://km.aifb.kit.edu/projects/btc-2012/`

and edges in a semantic network.

Reasoning over ontology concepts. Qin and Atluri [69], extend XML based access control to take into account the semantic relationships between the concepts that need to be protected. The authors propose a Concept Level Access Control (CLAC) model, which allows for reasoning over concepts appearing in authorisation *subjects*, *permissions* and *objects*. Access control policies are represented using an OWL based vocabulary, which they call the Semantic Access Control Language (SACL) and data instances are defined in domain ontologies. Two properties `SACL:higherLevelThan` and `SACL:lowerLevelThan` are used to specify a partial order between authorisation subjects and permissions. The proposed access control propagation is based on six domain independent relationships (`superclass/subclass`, `equivalence`, `partof`, `intersection`, `union` and `complement`). In the case of `equivalence`, `partof`, `union` and `subclass` positive policies are propagated from subject to object and negative policies are propagated from object to subject. Where there is an `intersection` between two concepts, only negative policies are propagated. In the case of `complement` relations neither positive nor negative authorisations are propagated. The authors acknowledge the need for conflict resolution and simply propose a *negation takes precedence* handling mechanism.

Reasoning over concepts, properties and individuals. The Semantic Based Access Control Model (SBAC) proposed by Javanmardi et al. [44,43], builds on the work presented in Qin and Atluri [69], by catering for access control policy propagation, not only based on the semantic relations between ontology *concepts*, but also based on the relations between *concepts*, *properties* and *individuals*. Like Qin and Atluri [69], OWL vocabularies are used to represent the authorisation *subjects*, *permissions* and *objects*, however the authorisations themselves are specified using rules. The authors propose the propagation of access rights based on seven different types of inference, from:

- *concept to concept* (where classes are deemed related based on some vocabulary, for example `rdfs:subClass`);
- *concept to individual* (where an entity is a type of class, for example if employee is a class and JoeBloggs `rdf:type` employee);
- *individual to individual* (using properties such as `owl:sameAs` it is possible to propagate entities that represent the same thing);
- *property to concept* (if access is granted to the property access should be granted to the classes governed by `rdfs:domain` and `rdfs:range`);
- *property to property* (where properties are deemed related based on some vocabulary, for example `rdfs:subProperty`, `owl:equivalentProperty`);
- *property to individual* (where an entity is a type of property, for example if roles is a property and manager is of `rdf:type` role); and
- *concept to property* (where access is granted to a concept it should also be granted to all properties relating to that concept).

The authors describe how the aforementioned semantic relations can be reduced to subsumption relations and propose a general propagation strategy for subsumption relations among *subjects*, *permissions* and *objects*. In the case of subjects and objects, both positive and negative access rights propagate from subsumee to subsumer. However, in the case of *permissions* positive access rights propagate from subsumee to subsumer, while negative access right propagate from subsumer to subsumee. Although an architecture is presented by Javanmardi et al. [43], very little detail on the actual enforcement mechanism is supplied.

Follow-up papers by Ehsan et al. [26] and Amini and Jalili [3] build on previous work, by providing for an access control model with formal semantics and an enforcement framework, which is suitable for distributed semantic aware environments (for example Semantic Web, Semantic Grid and Semantic Cloud Computing). Policy rules, in both the conceptual and individual levels, are specified using a combination of deontic and description logic, which they refer to as $MA(DL)^2$. The prototype consists of a user interface developed using the Google Web Toolkit; a data reasoner implemented in Jena; and a tableaux reasoner implemented in Prolog. The authors present the results of a performance evaluation over increasing policy rules, where the decision to grant or deny access is based on ground policies, inferred policies and the proposed conflict resolution strategy. The authors conclude that real-time reasoning is expensive. Therefore they suggest:

– using the parallelisation facilities of the tableaux system;
– adopting a proof based approach where the requester presents authorisation rules that demonstrate they can access the requested resource; and
– materialisation of inferred relations in advance.

Reasoning based on the semantic network. Ryutov et al. [72,73] propose a policy language which can be used to specify access control in terms of the semantic relationships between the nodes and edges of a graph. In order to cater for policy propagation, two directed acyclic graphs are used to represent the relationship between users and groups (using a `memberOf` property) and between objects and bundles (using a `partOf` relation). Propagation policies are defined to allow for policy propagation based on both the `partOf` and `memberOf` relations. The authors propose a conflict resolution algorithm, which is based on the semantic network. When a policy explicitly refers to a node, the policy distance is zero. Whereas, when a policy is implicitly assigned, based on a propagation policy, the distance is determined by counting the number of nodes in the path from the node with the explicit policy. The smaller the distance, the more specific the policy. If multiple policies exist at different distances, the most specific policy takes precedence. If multiple explicit conflicting policies exist, conflicts are resolved using logical conjunction. In the case of implicit policies, conflicts are resolved using logical disjunction. The authors also propose safety and consistency policies that are used to prevent undesirable access control policy specification and propagation, for example resources that nobody can access.

4.2.2. RDFS Inference

When it comes to RDFS inference, there are two different strands of research. The first infers access rights for triples that are inferred using RDFS entailment rules [41,50]. Whereas the second uses RDFS entailment rules to propagate permissions for triples that already exist [65,58].

Inferring access rights for new triples. Jain and Farkas [41], demonstrate how RDFS entailment rules can be used not only to infer new RDF triples, but also to infer access control annotations for those triples. The authors use RDF triple patterns and associated security classifications, known as security labels, to limit access to RDF statements. They define a subsumption relationship between patterns and stipulate that subsuming patterns must be as restrictive as the subsumed patterns. In addition, they define a partial order between security labels, which is used to determine the security classification of triples inferred via RDFS entailment rules. If more than one pattern maps to a statement the most restrictive or the lowest upper bound takes precedence. The authors provide formal definitions for each of the RDF security objects and define an algorithm to generate security labels for both explicit and inferred triples based on a security policy and a conflict resolution strategy. Limited details of the implementation are supplied and no evaluation is performed.

Kim et al. [50] demonstrate how together authorisations and RDFS inference rules can be used to generate new authorisations. An authorisation is defined as a four tuple $\langle sub, obj, sign, type \rangle$, where *sub* refers to the access control subject; *obj* is represented an RDF triple pattern; *act* is the operation; *sign* indicate if access is granted or denied; and *type* which is either R to indicate that the authorisation should be propagated or L if it should not. The authors discuss how `rdfs:subClass`, `rdfs:subProperty` and `rdf:type` inference can be used to infer new authorisation objects (triple patterns) and consequently new authorisations. In addition, they examine the different scenarios which might result in access to data being both permitted and prohibited. If the authorisation that is prohibited is more specific than the authorisation that is permitted based on the subclass/subproperty hierarchy then access should be denied. In order to determine if there is a conflict, only authorisations with a superclass or superproperty that is negative need to be checked.

Inferring and propagating access rights. Both Lopes et al. [58] and Papakonstantinou et al. [65] propose strategies for deriving access control annotations using RDFS inference policies. In the respective access control models both the triples and the corresponding annotations are represented as quads. Annotations can be:

– directly associated with triples;
– inferred using RDF inference rules;
– propagated using RDF inference rules; or
– assigned a default label.

The authors demonstrate how the RDFS `subClass`, `subProperty` and `type` inference rules can be used to assign annotations to inferred triples. However, they do not dictate how the access control annotations assigned to the premises should be combined, but rather propose an abstract operator which can be adapted to suit particular use cases. In addition, the authors demonstrate how the RDFS `subClass`, `subProperty` and `type` inference rules can be used to propagate permissions to existing triples. As per inference rules existing annotations and propagated annotations are inferred by means of a domain operator.

Lopes et al. [58] demonstrate how AnQL an extension of the SPARQL query language can be used to enforce access control, by rewriting using the requesters credentials to rewrite a SPARQL query to an AnQL query. A follow-up paper [52] demonstrates how custom rules can be used to support multiple access control models and demonstrate how rules can be used to propagate permissions: based on hierarchies of authorisation subjects, access rights and resources; to triples with the same subject; and using resource typing.

Papakonstantinou et al. [65] evaluate their prototype over both ProgresSQL and MonetDB relational databases. Based on their performance evaluation of both the inference and propagation rules, the authors concluded that more efficient storage and indexing schemes are required.

4.2.3. A flexible reasoning framework

Kirrane et al. [53,51] demonstrate how authorisations based on quad patterns together with stratified Datalog rules can be used to enforce DAC over the RDF data model. An *RDF quad pattern* is an RDF quad with optionally a variable V in the subject, predicate, object and/or graph position. A quad pattern is a flexible mechanism which can be used to grant/restrict access to an RDF quad, a collection of RDF quads (multiple quads that share a common subject), a named graph (arbitrary views of the data), specific classes or properties. The authors describe how the hierarchical Flexible Authorisation Framework proposed by Jajodia et al. [42], which is composed of authorisations, propagation policies, conflict resolution rules and integrity constraints, can be extended to cater for the RDF graph data model. They describe how together pattern matching and propagation rules can be used to ease the maintenance of access control policies for linked data sources; and show how conflict resolution policies and integrity constraints can ensure access control policy integrity. Propagation policies can be used to simplify authorisation administration by allowing for the derivation of implicit authorisations from explicit ones. Rather than propose a conflict resolution strategy the authors provide a formal definition for a conflict resolution rule that can be used to determine access given several different conflict resolution strategies. For example conflict resolution policies based on the structure of the graph data system components; the sensitivity of the data requested; or contextual conditions pertaining to the requester. Integrity constraints are used to restrict authorisation creation based on the existing relationships between SPARQL operations and RDF data items. For example, `INSERT` and `DELETE` can only be applied to an RDF quad whereas `DROP`, `CREATE`, `COPY`, `MOVE` and `ADD` can only be associated with a named graph. As per conflict resolution the authors provide a formal definition of an integrity constraint and demonstrate how rules can be used to ensure that only valid authorisation and propagation policies can be specified.

4.2.4. Reasoning over restricted data

When it comes to reasoning over restricted data, there is a general consensus that any information that can be inferred from restricted data should also be restricted. An alternative viewpoint is presented by Bao et al. [4]. The authors focuses on a number of use cases where it is desirable to grant access to information that has been inferred from restricted data:

(i) a calendar showing the existence of an appointment without revealing specifics;
(ii) a booking engine sharing partial hotel details; and
(iii) a pharmacy confirming that the patients drugs are reimbursable without disclosing details.

The open world assumption is used to ensure that users cannot distinguish between information which does not exist and information with is inaccessible. The authors stipulate that, the knowledge base should not lie, the answers given should be independent of any previous answers and it should not be possible to infer any restricted data. The proposed strategy is based on the notion of conservative extension. Essentially the reasoner keeps a history of the answers to all previous queries. For each

subsequent query, the history is consulted, in order to verify that unauthorised information cannot be inferred by the requester.

4.3. Partial Query Results

A number of the access control mechanisms for RDF data that have been presented, demonstrate how their access control can be enforced on top of SPARQL queries. However, the solutions examined thus far either grant or deny access to the entire query. This section examines how data filtering and query rewriting can be used to return partial query results when access to some of the results is restricted by an access control policy.

4.3.1. Data Filtering

Dietzold and Auer [25] examine access control requirements for an RDF store from a semantic wiki perspective. The authors propose access control policy specification at multiple levels of granularity (*triples*, *classes* and *properties*). In addition, they define three atomic actions (`read`, `insert` and `delete`) for both individual triples and sets of triples. Authorisations are used to generate a virtual model of the data, upon which user queries are executed. Authorisations are used to associate filters (SPARQL `CONSTRUCT` queries) with users and resources. When a requester submits a query, a virtual model is created based on the matched authorisations. The query is executed against the virtual model, which only contains data the requester is authorised to access.

Muhleisen et al. [60] describe a Policy-enabled Linked Data Server (PeLDS), which uses WebID to authenticate users. The policy language caters for the specification of access control policies for particular *triple patterns*, *resources* or *instances*, using SWRL rules. An OWL ontology is used to identify the rule types (`single concept` and `triple pattern`) and supported actions `query` and `update`. Negation is not supported in the presented modelling. When a requester submits a query, the system uses their WebID to determine the data instances that the user has been granted access to and generates a temporary named graph containing authorised data. The requesters query is subsequently executed against the temporary named graph and the results are returned to the user.

4.3.2. Query Rewriting

Existing query rewriting strategies involve: creating bindings for variables and adding them to the query `WHERE` clause [1,32]; using `FILTER` expressions to filter out inaccessible data [14,64]; limiting the query to a specific named graph [17] or rewriting a view so that it considers propagation rules and both instance and range restrictions [57].

Bindings added to where or minus clause. Abel et al. [1] propose a combined approach to access control enforcement. Contextual conditions that are not dependent on RDF data are evaluated by a policy engine. Whereas the query is expanded to include the contextual conditions that are dependent on RDF data. Such an approach requires the substitution of variables to ensure uniqueness, however in doing so they are able to leverage the highly optimized query evaluation features of the RDF store. In the presented modelling, both positive and negative authorisations are composed of sets of *contextual predicates*, *path expressions* and *boolean expressions*. Queries are assumed to have the following structure `SELECT/CONSTRUCT` ***RF*** `FROM` ***PE*** `WHERE` ***BE***, where ***RF*** represents the result form (projections in the case of `SELECT` queries and triples in the case of `CONSTRUCT` queries); ***PE*** denotes the path expression; and ***BE*** corresponds to one ore more boolean expressions connected via conjunction or disjunction operators. An authorisation is deemed applicable if the triple pattern the policy is protecting, is part of either the ***PE*** or the ***BE***, and the corresponding contextual predicates, path expressions and boolean expressions are satisfied. The authors propose a query rewriting algorithm, which constructs bindings for authorisation path expressions and contextual predicates. For positive authorisations the bindings are appended to the query `WHERE` clause. Whereas, for negative authorisations the bindings are added to a `MINUS` clause, which in turn is appended to the query. The authors conclude that the proposed rewriting strategy, which was evaluated over a Sesame database, increases linearly with additional `WHERE` clauses.

Binding user attributes to path expressions. Franzoni et al. [32] propose a query rewriting strategy, which is used to grant/deny access to ontology instances. The authors rewrite queries to take into account contextual information, pertaining to the user or the environment. A fine grained access control (FGAC) policy is defined as a tuple $\langle target, \langle property, attribute, operator \rangle \rangle$ where:

- *target* is the resource that the policy relates to;
- *property* is a path expression, which either directly or indirectly relates to the target;
- *attribute* is the user attributes, that are bound to the path expression variables; and
- *operator* is the filter condition.

The authors propose a two tiered approach to access control enforcement. Access control policies are used to determine if access should be granted or denied. FGAC policies are only applied if access is granted. If the query contains one ore more FGAC policy targets, the query is rewritten to include the path expression and a `WHERE` clause, which is composed of an expression generated from the variables in the path expression, the attributes of the requester and the operator.

Optionals and filters. Chen and Stuckenschmidt [14] present a query rewriting strategy, which can be used to restrict access to data represented using ontologies. The authors focus on restricting access to instance data. Access control policies are used to deny access to specific *individuals* or to grant/deny access to instances associated with a given *class* or *property*. When access is prohibited to specific *individuals*, a `FILTER` expression is generated, which ensures that none of the query variables bind to the prohibited individuals. When access is granted to *predicates* or *classes*, a `FILTER` expression is generated, which binds the relevant variables in the query to the specified predicate or class. Whereas, when access is prohibited to *predicates* or *classes*, the matching triple patterns are made `OPTIONAL` and a `FILTER` expression is generated, which ensures that the corresponding variables do not bind to the specified predicate or class, and variables that are `!BOUND` are not returned.

Oulmakhzoune et al. [64] propose a query rewriting strategy for SPARQL queries. In the presented modelling, both positive and negative authorisations are composed of sets of filters that are associated with simple conditions or involved conditions. Given a SPARQL query the algorithm examines each individual *basic graph pattern* (BGP).

In the case of *simple conditions*, when authorisations permit/deny access to a single triple pattern, the following query rewriting strategy is applied: If all authorisations that match the triple pattern, permit access to the triple pattern, no action is required; If all authorisations prohibit access to the triple pattern, the triple pattern is deleted; Otherwise, if the BGP is converted to an `OPTIONAL` BGP, and the authorisation `FILTER` expression is added to the query.

In the case of *involved conditions*, where authorisations permit/deny access for a given *predicate*, the following query rewriting strategy is applied: For positive authorisations, if the query contains a triple pattern which matches the predicate of the authorisation, the `FILTER` condition is added. Alternatively both the triple pattern and the corresponding `FILTER` are added; For negative authorisations, if the query contains a triple pattern which matches the predicate of the authorisation, and the object is a variable, both the `FILTER` condition and a `!BOUND` expression are added; Alternatively the triple pattern, the corresponding `FILTER` condition and a `!BOUND` expression are added.

Named graph added to the query. Costabello et al. [17] restrict access to named graphs using query rewriting. An access control policy is a tuple $\langle ACS, AP, S, R, AEC \rangle$, where ACS is a set of access conditions (specified using SPARQL `ASK` queries); AP is a set of access privileges (`CREATE`, `READ`, `UPDATE` or `DELETE`); S denotes the subjects to be protected; R represents the named graphs to be protected; and AEC is the evaluation context specified using name value pairs (verified using SPARQL `BINDINGS`). In addition to the SPARQL query that the user wishes to execute, the user provides their access credentials, in the form of a SPARQL `UPDATE` query, which contains contextual data. The enforcement framework stores the contextual data in a named graph and retrieves the authorisations that match the query type. In order to determine if access is permitted, the `ASK` query and the `BINDINGS`, that are specified in the authorisation, are executed against the users contextual graph. If the `ASK` query returns true then the query is rewritten to include the corresponding named graph.

Expanding views based on propagation rules and instance and range restrictions. Li and Cheung [57] propose a query rewriting strategy for views generated from ontological relations. An access control policy is defined as a tuple $\langle s, v, sign \rangle$, where s denotes the *subject*, v represents a set of *concepts, relations and*

filters and *sign* is used to indicate *permissions and prohibitions*. Both views and queries that are also generated from sets of concepts, relations and filters are represented using rules. Propagation policies are used to generate implicit authorisations from explicit authorisations, based on subsumption relations between access control subjects and subsumption relations between concepts, appearing in the body of the view. The proposed query rewriting strategy involves: (i) retrieving the policies applicable to the subject, taking into account the subject propagation rules; (ii) expanding each of the concepts in the body of the view based on the concept propagation policies; and (iii) applying the relevant range and instance restrictions to the query based on the expanded view.

5. Access Control Requirements for Linked Data

More recently, the focus has shifted to the specification and enforcement of access control over Linked Data. Costabello et al. [19], Sacco et al. [74] and Kirrane et al. [51] describe how their policy languages and frameworks can be used in conjunction with Linked Data, and Steyskal and Polleres [86] discuss how ODRL can be used to specify access policies for Linked Data. However, any of the access control mechanisms examined thus far could potentially be used (albeit to a lesser or greater extent) to enforce access control over Linked Data.

This section provides a summary of existing requirements for RDF data, and uses these requirements to categorise existing access control strategies that have been proposed for RDF. As the work presented in *Section* 2 focused on extending existing access control models and standards as opposed to enforcing access control over RDF, the analysis is limited to the policy languages presented in *Section* 3 and the different access control strategies described in *Section* 4.

The requirements presented below are derived from several papers that examine access control for RDF from a number of perspectives. Yagüe et al. [102] examine the different layers of the Semantic Web and how the technologies and concepts can be applied to access control. Both Damiani et al. [21] and Weitzner et al. [101] focus on the access control mechanisms that are required to support new access control paradigms where user privacy is a key requirement. De Coi et al. [22] and Bonatti and Olmedilla [10] investigate the interplay between trust, access control and policy languages. While, Ryutov et al. [73] focus more on the data model, investigating access control requirements from a graph perspective, as opposed to the traditional hierarchical approach. To ease referenceability the access control requirements are categorised under the headings specification, enforcement, administration and implementation.

5.1. Specification

Generally speaking, access control policy specification requirements relate to the types of policies that can be expressed, and the interoperability of the chosen representation format. An overview of each of the requirements relating to access control specification is presented below and a summary of existing proposals is depicted in *Table* 1. One requirement, which was not included is ***monotonicity***. According to Bonatti and Olmedilla [10], the addition of new evidences and policies should not negate any of the previous conclusions. However, given the need to support negative access control policies, and also changes in contextual constraints, one could argue that access control should in fact be ***non-monotonic***.

Granularity [73,3]. Ryutov et al. [73] adopt a graph perspective, stating that it should be possible to specify access control rules for nodes (entities) and edges (semantic relationships between entities). Whereas, Amini and Jalili [3] adopt an ontological view, stating that it is necessary to specify policies for both ontology concepts and individuals. Existing access control strategies for RDF, resources are specified at several different levels of granularity. Namely, triples [25,65,76], named graphs [18,33,76], views [57], triple patterns [41,50,60, 70], quad patterns [51], graph patterns with filters [1,14] and graph patterns without filters [31], classes and properties [25], ontology concepts [10,3,26,43,46,56,64,69,76,91,93,86], ontology individuals [3,26,32,43,76] and graph nodes and edges [73]. In the vast majority of cases, access is either granted or denied. However, a number of researchers have investigated returning partial query results to the requester. Such strategies either involve dataset filtering [25,60] or rewriting the query using either filters [1,14,31,32,60,51] or named

graphs [18]. In *Table* 1 the granularity of the authorisations and the different strategies used to cater for partial query results are represented as *Granularity* and *Partial Results* respectively.

Underlying Formalism [22,3,10]. Access control languages should be based on formal semantics, as it decouples the meaning of the policies from the actual implementation. The majority of researchers either adopt formalisms based on logic programming [10,46, 91] or different flavors of description logic [3,4, 14,43,56,91,86]. Kagal and Finin [46] demonstrate how logic programming can be combined with deontic logic, Kirrane et al. [51] adopt a DATALOG formalism, Amini and Jalili [3] demonstrate how description logic can be combined with deontic logic, Kolovski et al. [56] combine description logic and defeasible logic and Ryutov et al. [73] adopt a many sorted first order logic formalism.

Reasoning [21,73,3]. It should be possible to propagate policies based on the semantic relations between authorisation subjects, objects and access rights. Using ontologies, rules or a combination of both, it is possible to perform deductive reasoning and abductive reasoning over access control policies [3,10,14,46, 56,60,91,93]. In addition, a number of authors have proposed propagation strategies based on RDFS entailment [32,41,50,69,70] and hierarchies, partial orders or ontological relations between RDF resources [43,65,73,86]. Kirrane et al. [51] propose a flexible authorisation framework which can be used to specify declarative authorisations, propagation policies, integrity constraints and conflict resolution rules. Unlike the other authors, who use reasoning to either infer or to propagate access control policies, Bao et al. [4] demonstrates how it is possible to reason over restricted data without releasing any restricted information. However, the interplay between the various reasoning strategies proposed, and the access control requirements arising from concrete use cases remains an open issue.

Condition Expressiveness [22,10]. When it comes to access control, it should be feasible to specify conditions under which a request will be permitted or prohibited. The majority of the access control strategies that were examined support both authorisations and obligations. However, ODRL [86] and the general policy languages [3,46,91,93] also catered for obligation and dispensation policies.

Attributes, Context & Evidences [21,22,10,3]. As the requester many be unknown to the system prior to submitting a request, access should be based on properties pertaining to the requester, commonly know as attributes, instead of traditional identities [3,10,18,32,33, 46,51,56,70,73,76,91]. It should also be feasible to dynamically activate policies based on context [1,3,10,18,32,33,46,65,76,91,93,86]. Context can relate to the requester, the system or the environment. Attributes and context should be communicated by means of digital certificates, known as evidences. The de facto standard for submitting evidences is WebID [60,18,51,76]. However, a number of researchers have also proposed using OpenID [10,60,76] .

Heterogeneity & Interoperability [102]. Access control for open distributed environments, such as the web, needs to be able to support a wide variety of disparate policies, resources and users. One of the primary goals of standardisation is to maximize interoperability. As each of the access control strategies examined use open standards, such as RDF, RDFS, OWL and SPARQL, regardless of the specific use case they are suitable for access control over Linked Data. Using ontologies it is possible to specify access control vocabularies that can easily be adopted by others. In addition, OWL predicates such as `owl:sameAs` and `owl:disjointFrom` can be used to merge different access control vocabularies.

Table 1: Specification requirements

	Granularity	Partial Results	Underlying Formalism	Reasoning	Condition Expressiveness	Attributes, Context & Evidences	Interoperability
Abel et al. [1]	graph patterns & filters	bindings & filters	-	-	permissions±	context	RDF & SPARQL & SeRQL
Amini and Jalili [3] Ehsan et al. [26]	ontology concepts & individuals	-	DL & deontic logic	deduction & abduction	permissions± obligation±	attributes & context	OWL
Bao et al. [4]	ontology concepts	-	DL *SHIQ*	privacy preserving	-	-	OWL
Bonatti et al. [8] Bonatti and Olmedilla [7,10]	ontology concepts	-	LP	deduction & abduction	permissions±	attributes & context & OpenID	RDF
Chen and Stuckenschmidt [14]	graph patterns & filters	bindings & filters	DL	deduction & abduction	permissions±	-	OWL & SPARQL
Villata et al. [98] Costabello et al. [17,18,19]	named graphs	named graphs	-	-	permissions±	attributes & context & WebID	RDF & SPARQL
Dietzold and Auer [25]	triples, classes & properties	data filtering	-	-	permissions	-	RDF
Flouris et al. [31]	graph patterns	bindings & filters	-	-	permissions±	-	RDF & SPARQL
Franzoni et al. [32]	ontology individuals	bindings & filters	-	RDFS entailment	permissions±	attributes & context	RDFS & SPARQL
Gabillon and Letouzey [33]	named graph & views	-	-	-	permissions±	attributes & context	RDF & SPARQL
Jain and Farkas [41]	triple patterns	-	-	RDFS entailment	permissions±	-	RDFS
Javanmardi et al. [43]	ontology concepts & individuals	-	DL *SHOIN*	subjects, predicates & objects subsumption	permissions±	-	OWL
Kagal and Finin [46] Kagal et al. [48,49] Kagal and Berners-lee [47] Denker et al. [23] Kodali et al. [54]	ontology concepts	-	LP	deduction & abduction	permissions± obligation±	attributes & context	OWL

continued overleaf

Table 1 – continued from previous page

	Granularity	Partial Results	Underlying Formalism	Reasoning	Condition Expressiveness	Attributes, Context & Evidences	Heterogeneity & Interoperability
Kim et al. [50]	triple pattern	-	-	RDFS entailment	permissions±	-	RDFS
Kirrane et al. [52,53,51] Lopes et al. [58]	quad patterns	SPARQL 1.1 queries & updates	DATALOG	flexible framework	permissions±	attributes & WebID	RDF & RDFS & SPARQL & RDB2RDF
Kolovski et al. [56]	ontology concepts	-	DL *SHOIN* & defeasible logic	deduction & abduction	permissions±	attributes	OWL
Li and Cheung [57]	views	propagation	-	-	permissions±	-	RDF
Muhleisen et al. [60]	triple patterns	data filtering	-	deduction & abduction	permissions+	WebID & OpenID	OWL
Oulmakhzoune et al. [64]	ontology concepts	bindings & filters	-	-	permissions±	-	RDF & SPARQL
Papakonstantinou et al. [65]	triples	-	-	RDFS entailment	permissions±	context	RDFS & SPARQL
Qin and Atluri [69]	concept	-	-	ontology concept relations	permissions±	-	RDF
Reddivari et al. [70]	triple patterns	-	-	RDFS entailment	permissions±	attributes	RDFS
Ryutov et al. [72,73]	nodes & edges	-	many sorted first order logic	subject & object subsumption	permissions±	attributes	RDF
Sacco et al. [74,78] Sacco and Passant [77] Sacco and Breslin [75]	resource, triple & graph	-	-	-	permissions±	attributes & context & WebID & OpenID	RDF & SPARQL
Steyskal and Polleres [86,87]	ontology concepts	-	LP	-	action subsumption & composition	attributes	ODRL & RDF & SPARQL
Toninelli et al. [90,91]	ontology concepts	-	DL & LP	deduction & abduction	permissions± obligation±	attributes & context	OWL
Uszok et al. [93,96,95,94,97,92] Bradshaw et al. [12,11] Johnson et al. [45] Suri et al. [88]	ontology concepts	-	DL	deduction & abduction	permissions± obligation±	context	OWL

5.2. Enforcement

Access control enforcement requirements refer to constraints that are placed on the policy language or mechanisms that assist the requester to complete their request. An overview of the requirements is presented below and a snapshot of existing support for said requirements is presented in *Table* 2.

Negotiation [21,22,10]. In other to protect user privacy, it should be possible for both the service provider and the requester to define polices and exchange credentials until and agreement has been reached. The process is commonly known as negotiation. The P3P recommendation and the APPEL vocabulary have been designed to support automatic negotiation between clients and servers. The access control mechanisms proposed by Amini and Jalili [3], Bonatti and Olmedilla [10] and Toninelli et al. [91] all cater for access control negotiation.

Explanations [22,10]. Rather than simply granting or denying access, the policy should also provide details of how the decision was reached. Such explanations would be of benefit to both the requester and the policy owner, making it easier for the requester to understand what is required of them and for the policy owner to troubleshoot potential problems. Bonatti and Olmedilla [10], Ryutov et al. [73] and Toninelli et al. [91] all provide policy explanations. However, both Ryutov et al. [73] and Bonatti and Olmedilla [10] provide a means to execute queries over policies in order to obtain additional information.

Conflict Resolution [3]. Conflicts between both explicit and implicit policies should be resolved automatically. A number of different conflict resolution strategies have been proposed. In the event of a conflict some authors simply default to grant/deny [1,18,31,33,65,69,86], use priorities to determine dominance [56] or use metapolicies as a flexible means to resolve conflicts [46,70]. A number of authors propose conflict resolution algorithms based on several different measures [3,4,41,43,50,73]. Kirrane et al. [51] suggest a general syntax, which can be used to specify declarative conflict resolution policies. Whereas, others try to isolate individual data items that are in conflict and propose harmonisation strategies [57,91,93].

5.3. Administration

This section presents a number of access control requirements that are necessary to simplify the specification and maintenance of access control policies. An overview of the requirements is presented below and a summary of current support is presented in *Table* 3. Although a number of researchers indicate that they provide some level of support for these requirements, generally speaking research efforts seem to focus more on the specification and enforcement mechanisms, and very little detail is supplied.

Delegation [10]. It should be feasible to temporarily transfer access rights to other users. In relational databases, users are granted sole ownership of the tables and views that they create. They can subsequently grant access rights to other database users. Amini and Jalili [3], Bonatti and Olmedilla [10], Gabillon and Letouzey [33] and Kagal and Finin [46] indicate that they support the delegation of access rights. Kirrane et al. [51] demonstrate how the discretionary access control model can be used to guide access control specification and administration. However, the suitability of existing revocations strategies, for the RDF graph model, warrants further research.

Consistency & Safety [73]. In order to ensure the access control system is complete and accurate, insertion and deletion of policies should be controlled. It should not be possible to elevate your own privileges or to assign permissions that would make data inaccessible to everyone. Although a number of researchers indicate that their frameworks support consistency and safety constraints, very little information is provided. Kolovski et al. [56], Amini and Jalili [3], Bao et al. [4] and Jain and Farkas [41] ensure consistency and safety as part of their administration algorithms. Chen and Stuckenschmidt [14] and Ryutov et al. [73] suggest that meta policies can be used to ensure consistency and safety. Kirrane et al. [51] propose a general syntax for integrity constraints, which can be used to specify declarative constraints. Given the diversity of access control models, policies and reasoning strategies that have been proposed for RDF, additional research is required in order to determine potential issues with access control policies and propose suitable handling mechanisms.

Table 2
Enforcement requirements

	Negotiation support	Explanation	Conflict Resolution
Abel et al. [1]	-	-	default
Amini and Jalili [3] Ehsan et al. [26]	bidirectional policies	-	algorithm
Bao et al. [4]	-	-	algorithm
Bonatti et al. [8] Bonatti and Olmedilla [7,10]	bidirectional policies	queries	-
Chen and Stuckenschmidt [14]	-	-	-
Villata et al. [98] Costabello et al. [17,18,19]	-	-	default
Dietzold and Auer [25]	-	-	-
Flouris et al. [31]	-	-	default
Franzoni et al. [32]	-	-	-
Gabillon and Letouzey [33]	-	-	default
Jain and Farkas [41]	-	-	algorithm
Javanmardi et al. [43]	-	-	algorithm
Kagal and Finin [46] Kagal et al. [48,49] Kagal and Berners-lee [47] Denker et al. [23]	-	-	meta policies
Kim et al. [50]	-	-	algorithm
Kirrane et al. [52,53,51] Lopes et al. [58]	-	-	flexible framework
Kolovski et al. [56]	-	-	priorities
Li and Cheung [57]	-	-	harmonisation
Muhleisen et al. [60]	-	-	-
Oulmakhzoune et al. [64]	-	-	-
Papakonstantinou et al. [65]	-	-	default
Qin and Atluri [69]	-	-	default
Reddivari et al. [70]	-	-	meta policies
Ryutov et al. [72,73]	-	user interface	algorithm
Sacco et al. [74,78] Sacco and Passant [77] Sacco and Breslin [75]	-	-	-
Toninelli et al. [90,91]	bidirectional policies	descriptions	harmonisation
Steyskal and Polleres [86,87]	-	-	default
Uszok et al. [93,96,95,94,97,92] Bradshaw et al. [12,11] Johnson et al. [45] Suri et al. [88]	-	-	harmonisation

Usability [21,3]. The specification and the maintenance of access control policies should be as simple as possible. Administration facilities that support ease of both specification and maintenance of policies have been provided by a number of researchers [41,73,76,93]. Given the complexity associated with reasoning over graph data, advanced data analytics and visualisation techniques are needed to highlight the effects of advanced policies, constraints and deduction rules.

Understandability [73,3]. It should be easy to understand the interplay between policies. Only a handful of researchers associate policy explanations with policies. Similarly, only a select few provide systems that enable administrators to verify the interplay between policies [10,56,73,93,56]. Given the dynamic nature of context based access control and the various deduction and propagation strategies, further research on automating the explana-

Table 3
Administration requirements

	Delegation	Consistency & Safety	Usability	Understandability
Abel et al. [1]	-	-	-	-
Amini and Jalili [3] Ehsan et al. [26]	case study	algorithm	-	-
Bao et al. [4]	-	algorithm	-	-
Bonatti et al. [8] Bonatti and Olmedilla [7,10]	language	-	-	ProtuneX explanation
Chen and Stuckenschmidt [14]	-	meta policies	-	-
Villata et al. [98] Costabello et al. [17,18,19]	-	-	-	-
Dietzold and Auer [25]	-	-	-	-
Flouris et al. [31]	-	-	-	-
Franzoni et al. [32]	-	-	-	-
Gabillon and Letouzey [33]	construct & describe	-	-	-
Jain and Farkas [41]	-	algorithm	RACL admin module	-
Javanmardi et al. [43]	-	-	-	-
Kagal and Finin [46] Kagal et al. [48,49] Kagal and Berners-lee [47] Denker et al. [23]	speech acts	-	-	-
Kim et al. [50]	-	-	-	-
Kirrane et al. [52,53,51] Lopes et al. [58]	DAC	flexible framework	-	-
Kolovski et al. [56]	-	analysis services	-	analysis services
Li and Cheung [57]	-	-	-	-
Muhleisen et al. [60]	-	-	-	-
Oulmakhzoune et al. [64]	-	-	-	-
Papakonstantinou et al. [65]	-	-	-	-
Qin and Atluri [69]	-	-	-	-
Reddivari et al. [70]	-	-	-	-
Ryutov et al. [72,73]	-	meta policies	RAW policy editor	RAW permission check
Sacco et al. [74,78] Sacco and Passant [77] Sacco and Breslin [75]	-	-	privacy preference manager	-
Steyskal and Polleres [86,87]	-	-	-	-
Toninelli et al. [90,91]	-	-	-	-
Bradshaw et al. [12,11] Uszok et al. [93,96,95,94,97,92] Johnson et al. [45] Suri et al. [88]	-	-	KAoS Policy Admin Tool	policy disclosure

tions and presenting the results in a manner which is digestible by humans is necessary.

5.4. Implementation

Implementation requirements generally refer to non-functional requirements. As with any software system, non-functional requirements hold the key to the adoption of a tool or technology. Although a number of authors indicate that the solutions they propose are flexible or extensible, seldom do researchers evaluate these claims. *Table* 4 provides an overview of the technologies adopted and indicates the evaluations performed.

Effectiveness [73]. In order to work in practice, access control enforcement and administration

Table 4
Implementation requirements

	Effectiveness	Distributed Use case	Flexibility & Extensibility
Abel et al. [1]	query rewriting performance	-	SeRQL, Protune
Amini and Jalili [3] Ehsan et al. [26]	enforcement performance	case study	Prolog, GWT, Jena, JIP, Protege
Bao et al. [4]	-	-	-
Bonatti et al. [8] Bonatti and Olmedilla [7,10]	explanation performance	demo	Java, TuProlog
Chen and Stuckenschmidt [14]			Jena
Villata et al. [98] Costabello et al. [17,18,19]	enforcement performance BSBM & BTC datasets	mobile use case	Java, Corese-KGRAM
Dietzold and Auer [25]	-	-	RDF, SPARQL
Flouris et al. [31]	annotation performance	-	Java, Jena, Sesame, Progress
Franzoni et al. [32]	-	-	Java, SeRQL, Sesame
Gabillon and Letouzey [33]	enforcement performance	-	Java, Tomcat, Sesame
Jain and Farkas [41]	-	-	Java, Jena, Jess
Javanmardi et al. [43]	policy reasoning	-	PELLET, SWRL
Kagal and Finin [46] Kagal et al. [48,49] Kagal and Berners-lee [47] Denker et al. [23]	-	use cases	Java, Prolog
Kim et al. [50]	-	-	-
Kirrane et al. [52,53,51] Lopes et al. [58]	performance	Linked Data use case	Java, Jena
Kolovski et al. [56]	policy reasoning Continue dataset	-	Pellet
Li and Cheung [57]	-	-	-
Muhleisen et al. [60]	enforcement performance BSBM dataset	demo	Joseki, Jena, Pellet
Oulmakhzoune et al. [64]	- discussion	-	-
Papakonstantinou et al. [65]	enforcement reasoning performance	-	PostgreSQL, MonetDB
Qin and Atluri [69]	-	-	-
Reddivari et al. [70]	query performance	-	Java, Jena, RDQL
Ryutov et al. [72,73]	-	-	Java
Sacco et al. [74,78] Sacco and Passant [77] Sacco and Breslin [75]	enforcement performance	mobile use case	Java
Steyskal and Polleres [86]	-	Linked Data use case	-
Toninelli et al. [90,91]	-	-	-
Uszok et al. [93,96,95,94,97,92] Bradshaw et al. [12,11] Johnson et al. [45] Suri et al. [88]	-	use cases	Java

needs to be efficient. A number of authors have presented performance evaluations of their access control enforcement [3,18,33,51,60,70,76], query rewriting [1], annotation [31] , explanation [10] or reasoning [43,51,56,65] algorithms. However, there is still no clear access control benchmark, that can be used to compare different approaches. One suggestion would be to build a set of access control scenarios and extend the BSBM dataset generator to cater for solution benchmarking.

Distributed [3]. In order to ensure scalability, it should be possible to cater for the distributed specification and enforcement of access control policies. A number of researchers have examined how their proposed solution can be applied to use cases requiring distributed access control mechanisms. Amini and Jalili [3] describe a case study on distributed semantic digital library. Bonatti and Olmedilla [10] and Muhleisen et al. [60] developed demos in order to demonstrate how their policy languages can be used in a distributed setting. Whereas, the access control languages and enforcement frameworks proposed by Costabello et al. [18], Kagal and Finin [46], Sacco and Passant [76] and Uszok et al. [93] are motivated by distributed uses cases. However, the adaptation of current distributed query processing techniques to cater for access control over Linked Data has not be explored to date.

Flexibility & Extensibility [102,21,10]. The system should be capable of handling frequent changes to policies, user, access rights and resources. In addition, in order to provide support for different scenarios and future enhancements, the enforcement frameworks should be flexible and extensible. As each of the access control strategies examined use one or more open standards (*see Table* 1), they are by design flexible and extensible. An overview of the technologies used in each of the access control proposals examined is presented in *Table* 4.

6. Conclusions and Future Work

This paper provided an overview of relevant access control models (MAC, DAC, RBAC, VBAC, ABAC, CBAC) and standardisation efforts (XACML, WebID, WAC, P3P, APPEL, ODRL), and described how they have been either enhanced by/applied to RDF. A number of well known policy languages, that adopt ontology based, rule based and combined ontology and rule based access control enforcement mechanisms were examined in detail. Several different strategies that have be used to specify access control over RDF (triple patterns, views, named graphs and ontologies) and various reasoning, filtering and query rewriting strategies were presented. Finally, a set of requirements for Linked Data, based on several papers that examine access control for RDF from a number of perspectives, were derived. These requirements were subsequently used to classify the various access control specification, enforcement and administration strategies that have been proposed for RDF data.

Based on this analysis a number of gaps with respect to access control for Linked Data, which still need to be addressed were identified:

Usability & Understandability. Access control administration in general, and over large datasets in particular, can become extremely difficult to manage. Access control policies may be composed of authorisations specified at multiple levels of granularity. In addition, permissions may be inferred or propagated using different inferencing mechanisms, making the task of administration is even more cumbersome. An interesting avenue for future work, would be to investigate if graph based data clustering and visualisation techniques, such as those proposed by [61], can be used to assist systems administrators to examine the interplay between authorisations and rules, and also determine the impact of new authorisations.

Explanations & Negotiation. The benefits associated with explanations are two fold: (i) they allow the requester to understand what is required of them and (ii) they enable the policy owner to troubleshoot potential issues with existing policies. However, when it comes to explanations in particular and negotiation in general, there is a fine line between usability and security. As such, different levels of detail may need to be relayed to the requester depending on the context. In order to devise guidelines for access control explanations, it would be beneficial to examine the different reasons for access denial and the potential security impact associated with both single and multiple explanations.

Effectiveness. In order to work in practice, both access control enforcement and administration need to be effective from both a performances and a correctness perspective. Although a number of authors have conducted access control performances evaluations using the BSBM dataset, when it comes to access control for RDF data there is currently no general access control benchmark. In addition, there is a pressing need for general mechanisms that can be used to verify the correctness of proposed access control strategies.

References

[1] Fabian Abel, JuriLuca De Coi, Nicola Henze, ArneWolf Koesling, Daniel Krause, and Daniel Olmedilla. Enabling advanced and context-dependent access control in rdf stores. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg, 2007. URL `http://dx.doi.org/10.1007/978-3-540-76298-0_1`.

[2] J.M. Alcaraz Calero, G. Martinez Perez, and AF. Gomez Skarmeta. Towards an authorisation model for distributed systems based on the semantic web. *Information Security, IET*, 4(4): 411–421, December 2010.

[3] M. Amini and R. Jalili. Multi-level authorisation model and framework for distributed semantic-aware environments. *Information Security, IET*, 4(4):301–321, December 2010.

[4] Jie Bao, Giora Slutzki, and Vasant Honavar. Privacy-preserving reasoning on the semanticweb. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 791–797. IEEE Computer Society, 2007. URL `http://dx.doi.org/10.1109/WI.2007.147`.

[5] Tim Berners-Lee, Richard Cyganiak, Michael Hausenblas, Joe Presbrey, Oshani Seneviratne, and Oana-Elena Ureche. Realising a read-write web of data. Technical Report, available at `http://web.mit.edu/presbrey/Public/rw-wod.pdf`, 2009.

[6] E. Bertino and R. Sandhu. Database security - concepts, approaches, and challenges. *Dependable and Secure Computing, IEEE Transactions on*, 2(1):2–19, Jan 2005.

[7] P. Bonatti and D. Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*, pages 14–23, June 2005.

[8] PA Bonatti, JL De Coi, D Olmedilla, and L Sauro. Protune: A rule-based provisional trust negotiation framework. n.d.

[9] Piero Bonatti and Pierangela Samarati. Regulating service access and information release on the web. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 134–143, New York, NY, USA, 2000. ACM. URL `http://doi.acm.org/10.1145/352600.352620`.

[10] Piero A. Bonatti and Daniel Olmedilla. Rule-based policy representation and reasoning for the semantic web. In *Proceedings of the Third International Summer School Conference on Reasoning Web*, RW'07, pages 240–268. Springer-Verlag, 2007. URL `http://dl.acm.org/citation.cfm?id=2391482.2391488`.

[11] J. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho, D. Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, and R. Van Hoof. Representation and reasoning for daml-based policy and domain services in kaos and nomads. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 835–842, New York, NY, USA, 2003. ACM. URL `http://doi.acm.org/10.1145/860575.860709`.

[12] Jeffrey M. Bradshaw, Stewart Dutfield, Pete Benoit, and John D. Woolley. Software agents. chapter KAoS: Toward an Industrial-strength Open Agent Architecture, pages 375–418. MIT Press, 1997. ISBN 0-262-52234-9. URL `http://dl.acm.org/citation.cfm?id=267985.268008`.

[13] Elena Cabrio, Alessio Palmero Aprosio, and Serena Villata. These are your rights a natural language processing approach to automated rdf licenses generation. In *The Semantic Web: Trends and Challenges*, pages 255–269. Springer, 2014.

[14] W. Chen and H. Stuckenschmidt. A model-driven approach to enable access control for ontologies. *Business Services: Konzepte, Technologien, Anwendungen: Wirtschaftsinformatik 2009*, pages 663–672, 2010. URL `http://www.chenwilly.de/publications/wi2009.pdf`.

[15] Lorenzo Cirio, Isabel F. Cruz, and Roberto Tamassia. A role and attribute based access control system using semantic web technologies. In *Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems - Volume Part II*, OTM'07, pages 1256–1266, Berlin, Heidelberg, 2007. Springer-Verlag. URL `http://dl.acm.org/citation.cfm?id=1780453.1780518`.

[16] A Corradi, R. Montanari, and D. Tibaldi. Context-based access control for ubiquitous service provisioning. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pages 444–451 vol.1, Sept 2004.

[17] Luca Costabello, Serena Villata, Nicolas Delaforge, and Fabien Gandon. Linked data access goes mobile: Context-aware authorization for graph stores. In *LDOW - 5th WWW Workshop on Linked Data on the Web*, 2012. URL `http://hal.archives-ouvertes.fr/hal-00691256/`.

[18] Luca Costabello, Serena Villata, and Fabien Gandon. Context-aware access control for rdf graph stores. In *ECAI*, pages 282–287, 2012.

[19] Luca Costabello, Serena Villata, Oscar Rodriguez Rocha, and Fabien Gandon. Access control for http operations on linked data. In *The Semantic Web: Semantics and Big Data*, vol-

ume 7882 of *Lecture Notes in Computer Science*, pages 185–199. Springer Berlin Heidelberg, 2013. URL http://dx.doi.org/10.1007/978-3-642-38288-8_13.

[20] Lorrie Cranor, Marc Langheinrich, and Massimo Marchiori. A P3P Preference Exchange Language (APPEL). W3C Working Draft, available at http://www.w3.org/TR/P3P-preferences/, W3C, April 2002.

[21] E. Damiani, S.D.C. di Vimercati, and P. Samarati. New paradigms for access control in open environments. In *Signal Processing and Information Technology, 2005. Proceedings of the Fifth IEEE International Symposium on*, pages 540–545, Dec 2005.

[22] J.L. De Coi, P. Karger, AW. Koesling, and D. Olmedilla. Control your elearning environment: Exploiting policies in an open infrastructure for lifelong learning. *Learning Technologies, IEEE Transactions on*, 1(1):88–102, Jan 2008.

[23] Grit Denker, Lalana Kagal, and Tim Finin. Security in the semantic web using owl. *Inf. Secur. Tech. Rep.*, 10(1):51–58, January 2005. URL http://dx.doi.org/10.1016/j.istr.2004.11.002.

[24] Wu Di, Lin Jian, Dong Yabo, and Zhu Miaoliang. Using semantic web technologies to specify constraints of rbac. In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, PDCAT '05, pages 543–545, Washington, DC, USA, 2005. IEEE Computer Society. URL http://dx.doi.org/10.1109/PDCAT.2005.247.

[25] Sebastian Dietzold and Soren Auer. Access control on rdf triple stores from a semantic wiki perspective. In *Proceedings of the ESWC'06 Workshop on Scripting for the Semantic Web*, 2006.

[26] Moussa Amir Ehsan, Morteza Amini, and Rasool Jalili. A semantic-based access control mechanism using semantic technologies. In *Proceedings of the 2nd International Conference on Security of Information and Networks*, SIN '09, pages 258–267. ACM, 2009. ISBN 978-1-60558-412-6. URL http://doi.acm.org/10.1145/1626195.1626259.

[27] Mark Evered and Serge Bögeholz. A case study in access control requirements for a health information system. In *Proceedings of the Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation - Volume 32*, ACSW Frontiers '04, pages 53–61, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc. URL http://dl.acm.org/citation.cfm?id=976440.976447.

[28] Rodolfo Ferrini and Elisa Bertino. Supporting rbac with xacml+owl. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, SACMAT '09, pages 145–154, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-537-6. URL http://doi.acm.org/10.1145/1542207.1542231.

[29] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham. Rowlbac: Representing role based access control in owl. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, pages 73–82, New York, NY, USA, 2008. ACM. URL http://doi.acm.org/10.1145/1377836.1377849.

[30] Tim Finin, Anupam Joshi, Lalana Kagal, Jianwei Niu, Ravi Sandhu, William H Winsborough, and Bhavani Thuraisingham. Using owl to model role based access control. Technical report, University of Maryland, Baltimore County, February 2008.

[31] Giorgos Flouris, Irini Fundulaki, Maria Michou, and Grigoris Antoniou. Controlling access to rdf graphs. In *Proceedings of the Third Future Internet Conference on Future Internet*, pages 107–117. Springer-Verlag, 2010. URL http://dl.acm.org/citation.cfm?id=1929268.1929280.

[32] S. Franzoni, P. Mazzoleni, S. Valtolina, and E. Bertino. Towards a fine-grained access control model and mechanisms for semantic databases. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 993–1000, July 2007.

[33] A Gabillon and L. Letouzey. A view based access control model for sparql. In *Network and System Security (NSS), 2010 4th International Conference on*, pages 105–112, Sept 2010.

[34] D.Z.G. Garcia and M. Toledo. A web service privacy framework based on a policy approach enhanced with ontologies. In *Computational Science and Engineering Workshops, 2008. CSEWORKSHOPS '08. 11th IEEE International Conference on*, pages 209–214, July 2008.

[35] Rita Gavriloaie, Wolfgang Nejdl, Daniel Olmedilla, KentE. Seamons, and Marianne Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *The Semantic Web: Research and Applications*, volume 3053 of *Lecture Notes in Computer Science*, pages 342–356. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-21999-6. URL http://dx.doi.org/10.1007/978-3-540-25956-5_24.

[36] Guido Governatori. Defeasible description logics. In *Rules and Rule Markup Languages for the Semantic Web*, pages 98–112. Springer, 2004.

[37] Patricia P. Griffiths and Bradford W. Wade. An authorization mechanism for a relational database system. *ACM Trans. Database Syst.*, 1 (3):242–255, September 1976. URL http://doi.acm.org/10.1145/320473.320482.

[38] Tom Heath and Christian Bizer. *Linked data: Evolving the web into a global data space*, volume 1. Morgan & Claypool Publishers, 2011. URL http://linkeddatabook.com/.

[39] James Hollenbach, Joe Presbrey, and Tim Berners-Lee. Using rdf metadata to enable access control on the social semantic web. In *Proceedings of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK2009)*, volume 514, 2009.

[40] Toby Inkster, Henry Story, and Bruno Harbulot. WebID Authentication over TLS. W3C Editor's Draft, available at http://www.w3.org/2005/Incubator/webid/spec/tls/, W3C, March 2014.

[41] Amit Jain and Csilla Farkas. Secure resource description framework: An access control model. In *Proceedings of the Eleventh*

ACM Symposium on Access Control Models and Technologies, SACMAT '06, pages 121–129. ACM, 2006. URL http://doi.acm.org/10.1145/1133058.1133076.

[42] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, June 2001. URL http://doi.acm.org/10.1145/383891.383894.

[43] Sara Javanmardi, Morteza Amini, and Rasool Jalili. An Access Control Model for Protecting Semantic Web Resources. In *2nd Semantic Web Policy Workshop*. CEUR-WS, 2006.

[44] Sara Javanmardi, Morteza Amini, Rasool Jalili, and Yaser GanjiSaffar. Sbac: A semantic based access control model. In *11th Nordic Workshop on Secure IT-systems (NordSec'06), Linkping, Sweden*, volume 22, 2006.

[45] M Johnson, P Chang, R Jeffers, JM Bradshaw, VW Soo, MR Breedy, L Bunch, S Kulkarni, J Lott, N Suri, et al. Kaos semantic policy and domain services: An application of daml to web services-based grid architectures. In *Proceedings of the AAMAS*, volume 3, 2003.

[46] L. Kagal and T. Finin. A policy language for a pervasive computing environment. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 63–74. IEEE Comput. Soc, 2003.

[47] Lalana Kagal and Tim Berners-lee. Rein : Where policies meet rules in the semantic web. Technical report, Laboratory, Massachusetts Institute of Technology, 2005.

[48] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy based approach to security for the semantic web. In *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 402–418. Springer Berlin Heidelberg, 2003. URL http://dx.doi.org/10.1007/978-3-540-39718-2_26.

[49] Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel Weitzner. Using semantic web technologies for policy management on the web. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1337–1344. AAAI Press, 2006. URL http://dl.acm.org/citation.cfm?id=1597348.1597401.

[50] Jaehoon Kim, Kangsoo Jung, and Seog Park. An introduction to authorization conflict problem in rdf access control. In *Proceedings of the 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Part II*, KES '08, pages 583–592, Berlin, Heidelberg, 2008. Springer-Verlag. URL http://dx.doi.org/10.1007/978-3-540-85565-1_72.

[51] Sabrina Kirrane, Ahmed Abdelrahman, Alessandra Mileo, and Stefan Decker. Secure manipulation of linked data. In *The Semantic Web - ISWC 2013*, volume 8218 of *Lecture Notes in Computer Science*, pages 248–263. Springer Berlin Heidelberg, 2013. URL http://dx.doi.org/10.1007/978-3-642-41335-3_16.

[52] Sabrina Kirrane, Nuno Lopes, Alessandra Mileo, and Stefan Decker. Protect your rdf data! In *Semantic Technology*, volume 7774 of *Lecture Notes in Computer Science*, pages 81–96. Springer Berlin Heidelberg, 2013. URL http://dx.doi.org/10.1007/978-3-642-37996-3_6.

[53] Sabrina Kirrane, Alessandra Mileo, and Stefan Decker. Applying dac principles to the rdf graph data model. In *Security and Privacy Protection in Information Processing Systems*, volume 405 of *IFIP Advances in Information and Communication Technology*, pages 69–82. Springer Berlin Heidelberg, 2013. URL http://dx.doi.org/10.1007/978-3-642-39218-4_6.

[54] Naren Kodali, Csilla Farkas, and Duminda Wijesekera. An authorization model for multimedia digital libraries. *International Journal on Digital Libraries*, 4(3):139–155, 2004. URL http://dx.doi.org/10.1007/s00799-004-0080-1.

[55] P. Kolari, Li Ding, G. Shashidhara, A Joshi, T. Finin, and L. Kagal. Enhancing web privacy protection through declarative policies. In *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*, pages 57–66, June 2005.

[56] Vladimir Kolovski, James Hendler, and Bijan Parsia. Analyzing web access control policies. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 677–686, New York, NY, USA, 2007. ACM. URL http://doi.acm.org/10.1145/1242572.1242664.

[57] Jian Li and William K Cheung. Query rewriting for access control on semantic web. In *Secure Data Management*, volume 5159 of *Lecture Notes in Computer Science*, pages 151–168. Springer Berlin Heidelberg, 2008.

[58] Nuno Lopes, Sabrina Kirrane, Antoine Zimmermann, Axel Polleres, and Alessandra Mileo. A Logic Programming approach for Access Control over RDF. In *Technical Communications of ICLP'12*, volume 17, pages 381–392. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012.

[59] R. Montanari, Alessandra Toninelli, and J.M. Bradshaw. Context-based security management for multi-agent systems. In *Multi-Agent Security and Survivability, 2005 IEEE 2nd Symposium on*, pages 75–84, Aug 2005.

[60] Hannes Muhleisen, Martin Kost, and Johann-Christoph Freytag. SWRL-based Access Policies for Linked Data. In *Proceedings of the Second Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2010)*, 2010. URL http://CEUR-WS.org/Vol-576/paper1.pdf.

[61] P. Mutton and J. Golbeck. Visualization of semantic metadata and ontologies. In *Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on*, pages 300–305, July 2003.

[62] A Nematzadeh and L. Pournajaf. Privacy concerns of semantic web. In *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, pages 1272–1273, April 2008.

[63] Kieron O'Hara, Noshir S Contractor, Wendy Hall, James A Hendler, and Nigel Shadbolt. Web science: understanding the emergence of macro-level features on the world wide web. *Foundations and Trends in Web Science*, 4(2-3):103–

267, 2013.

[64] Said Oulmakhzoune, Nora Cuppens-Boulahia, Frédéric Cuppens, and Stephane Morucci. fquery: Sparql query rewriting to enforce data confidentiality. In Sara Foresti and Sushil Jajodia, editors, *Data and Applications Security and Privacy XXIV*, volume 6166 of *Lecture Notes in Computer Science*, pages 146–161. Springer Berlin Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-13739-6_10.

[65] Vassilis Papakonstantinou, Maria Michou, Irini Fundulaki, Giorgos Flouris, and Grigoris Antoniou. Access control for rdf graphs using abstract models. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, SACMAT '12, pages 103–112. ACM, 2012. URL http://doi.acm.org/10.1145/2295136.2295155.

[66] Torsten Priebe, EduardoB. Fernandez, JensI. Mehlau, and Günther Pernul. A pattern system for access control. In Csilla Farkas and Pierangela Samarati, editors, *Research Directions in Data and Applications Security XVIII*, volume 144 of *IFIP International Federation for Information Processing*, pages 235–249. Springer US, 2004. ISBN 978-1-4020-8127-9. URL http://dx.doi.org/10.1007/1-4020-8128-6_16.

[67] Torsten Priebe, Wolfgang Dobmeier, and Nora Kamprath. Supporting attribute-based access control with ontologies. In *Proceedings of the First International Conference on Availability, Reliability and Security*, ARES '06, pages 465–472. IEEE Computer Society, 2006. URL http://dx.doi.org/10.1109/ARES.2006.127.

[68] Torsten Priebe, Wolfgang Dobmeier, Christian Schläger, and Nora Kamprath. Supporting attribute-based access control in authorization and authentication infrastructures with ontologies. *Journal of Software*, 2(1):27–38, 2007. URL http://dblp.uni-trier.de/db/journals/jsw/jsw2.html#PriebeDSK07.

[69] Li Qin and Vijayalakshmi Atluri. Concept-level access control for the semantic web. In *Proceedings of the 2003 ACM Workshop on XML Security*, pages 94–103. ACM, 2003. ISBN 1-58113-777-X. URL http://doi.acm.org/10.1145/968559.968575.

[70] Pavan Reddivari, Tim Finin, and Anupam Joshi. Policy-based access control for an rdf store. In *Proceedings of the Policy Management for the Web workshop*, volume 120, pages 78–83, 2005.

[71] Erik Rissanen. Extensible access control markup language (xacml) version 3.0. OASIS Standard, available at http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html, OASIS Committee Specification, January 2013.

[72] T. Ryutov, T. Kichkaylo, Robert Neches, and M. Orosz. Sfinks: Secure focused information, news, and knowledge sharing. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 404–409, May 2008.

[73] Tatyana Ryutov, Tatiana Kichkaylo, and Robert Neches. Access control policies for semantic networks. In *Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks*, POLICY '09, pages 150–157. IEEE Computer Society, 2009. URL http://dx.doi.org/10.1109/POLICY.2009.11.

[74] O. Sacco, A Passant, and S. Decker. An access control framework for the web of data. In *Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE 10th International Conference on*, pages 456–463, Nov 2011.

[75] Owen Sacco and John G. Breslin. PPO & PPM 2.0: Extending the privacy preference framework to provide finer-grained access control for the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 80–87. ACM, 2012. ISBN 978-1-4503-1112-0. URL http://doi.acm.org/10.1145/2362499.2362511.

[76] Owen Sacco and Alexandre Passant. A privacy preference ontology (ppo) for linked data. In *Linked Data on the Web*. CEUR-WS, 2011. URL http://ceur-ws.org/Vol-813/ldow2011-paper01.pdf.

[77] Owen Sacco and Alexandre Passant. A privacy preference manager for the social semantic web. In *Semantic Personalized Information Management: Retrieval and Recommendation*. CEUR-WS, 2011. URL http://ceur-ws.org/Vol-781/paper6.pdf.

[78] Owen Sacco, Matteo Collina, Gregor Schiele, Giovanni Emanuele Corazza, John G Breslin, and Manfred Hauswirth. Fine-grained access control for rdf data on mobile devices. In *Web Information Systems Engineering – WISE 2013*, volume 8180 of *Lecture Notes in Computer Science*, pages 478–487. Springer Berlin Heidelberg, 2013. URL http://dx.doi.org/10.1007/978-3-642-41230-1_40.

[79] Pierangela Samarati and SabrinaCapitani de Vimercati. Access control: Policies, models, and mechanisms. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, pages 137–196. Springer Berlin Heidelberg, 2001. URL http://dx.doi.org/10.1007/3-540-45608-2_3.

[80] Andrei Sambra, Henry Story, and Tim Berners-Lee. Web Identity and Discovery. W3C Editor's Draft, available at http://www.w3.org/2005/Incubator/webid/spec/identity/, W3C, March 2014.

[81] R.S. Sandhu and P. Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, Sept 1994.

[82] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control: a multi-dimensional view. In *10th Annual Computer Security Applications Conference*, pages 54–62, Dec 1994.

[83] Haibo Shen and Yu Cheng. A semantic context-based model for mobile web services access control. *International Journal of Computer Network and Information Security, MECS Publisher*, 1: 18–25, 2011.

[84] Steve Speicher, John Arwe, and Ashok Malhotra. Linked Data Platform 1.0. W3C Candidate Recommendation, available at http://www.w3.org/TR/ldp/, W3C, June 2014.

[85] Gerald Stermsek, Mark Strembeck, and Gustaf Neumann. Using subject-and object-specific attributes for access control in web-based knowl-

edge management systems. In *Workshop on Secure Knowledge Management (SKM)*, 2004.

[86] Simon Steyskal and Axel Polleres. Defining expressive access policies for linked data using the odrl ontology 2.0. In *Proceedings of the 10th International Conference on Semantic Systems*, pages 20–23. ACM, 2014.

[87] Simon Steyskal and Axel Polleres. Towards formal semantics for odrl policies. In *Proceedings of the 9th International Web Rule Symposium (RuleML)*, 2015.

[88] Niranjan Suri, JeffreyM. Bradshaw, Mark Burstein, Andrzej Uszok, Brett Benyo, Maggie Breedy, Marco Carvalho, David Diller, Renia Jeffers, Matt Johnson, Shri Kulkarni, and James Lott. Daml-based policy enforcement for semantic data transformation and filtering in multi-agent systems. In *Multi-Agent Systems and Applications III*, volume 2691 of *Lecture Notes in Computer Science*, pages 122–136. Springer Berlin Heidelberg, 2003. URL `http://dx.doi.org/10.1007/3-540-45023-8_13`.

[89] Bhavani Thuraisingham. Security and privacy for multimedia database management systems. *Multimedia Tools and Applications*, 33(1):13–29, 2007. URL `http://dx.doi.org/10.1007/s11042-006-0096-1`.

[90] Alessandra Toninelli, Jeffrey Bradshaw, Lalana Kagal, and Rebecca Montanari. Rule-based and ontology-based policies: Toward a hybrid approach to control agents in pervasive environments. In *Proceedings of the Semantic Web and Policy Workshop*, November 2005.

[91] Alessandra Toninelli, Antonio Corradi, and Rebecca Montanari. A quality of context-aware approach to access control in pervasive environments. In Jean-Marie Bonnin, Carlo Giannelli, and Thomas Magedanz, editors, *MobileWireless Middleware, Operating Systems, and Applications*, volume 7 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 236–251. Springer Berlin Heidelberg, 2009. URL `http://dx.doi.org/10.1007/978-3-642-01802-2_18`.

[92] A Uszok, J Bradshaw, P Hayes, R Jeffers, M Johnson, S Kulkarni, M Breedy, J Lott, and L Bunch. Daml reality check: A case study of kaos domain and policy services. In *The International Semantic Web Conference (ISWC 03)*, 2003.

[93] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY '03, pages 93–. IEEE Computer Society, 2003. URL `http://dl.acm.org/citation.cfm?id=826036.826850`.

[94] A Uszok, J.M. Bradshaw, M. Johnson, R. Jeffers, A Tate, J. Dalton, and S. Aitken. Kaos policy management for semantic web services. *Intelligent Systems, IEEE*, 19(4):32–41, Jul 2004.

[95] Andrzej Uszok, Jeffrey M. Bradshaw, Renia Jeffers, Matthew Johnson, and Austin Tate Jeff Dalton. Policy and contract management for semantic web services. In *Systems. Stanford University*. AAAI Press, 2004.

[96] Andrzej Uszok, JeffreyM. Bradshaw, and Renia Jeffers. Kaos: A policy and domain services framework for grid computing and semantic web services. In *Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 16–26. Springer Berlin Heidelberg, 2004. URL `http://dx.doi.org/10.1007/978-3-540-24747-0_2`.

[97] Andrzej Uszok, JeffreyM. Bradshaw, Renia Jeffers, Austin Tate, and Jeff Dalton. Applying kaos services to ensure policy compliance for semantic web services workflow composition and enactment. In SheilaA. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *The Semantic Web - ISWC 2004*, volume 3298 of *Lecture Notes in Computer Science*, pages 425–440. Springer Berlin Heidelberg, 2004. URL `http://dx.doi.org/10.1007/978-3-540-30475-3_30`.

[98] Serena Villata, Nicolas Delaforge, Fabien Gandon, and Amelie Gyrard. An access control model for linked data. In *On the Move to Meaningful Internet Systems: OTM 2011 Workshops*, pages 454–463, 2011.

[99] W3C SPARQL Working Group. SPARQL 1.1 Overview. W3C recommendation, available at `http://www.w3.org/TR/sparql11-overview/`, W3C, Jan 2013.

[100] Daniel J Weitzner, Jim Hendler, Tim Berners-Lee, and Dan Connolly. Creating a policy-aware web: Discretionary, rule-based access. *Web and Information Security*, page 1, 2006.

[101] Daniel J Weitzner, Jim Hendler, Tim Berners-Lee, and Dan Connolly. Creating a policy-aware web: Discretionary, rule-based access. page 1. IGI Global, 2006.

[102] Mariemma I. Yagüe, Antonio Maña, Javier López, and José M. Troya. Applying the semantic web layers to access control. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, DEXA '03, Washington, DC, USA, 2003. IEEE Computer Society. URL `http://dl.acm.org/citation.cfm?id=942790.942921`.