License and Template Access Control for Geospatial Linked Data

Alan Meehan[1], Kaniz Fatema[2], Rob Brennan[1], Éamonn Clinton[3], Lorraine McNerney[3] and Declan O'Sullivan[1]

[1] ADAPT Centre, Knowledge and Data Engineering Group, School of Computer Science and Statistics, Trinity College Dublin, Ireland
{almeehan,rob.brennan,declan.osullivan}@scss.tcd.ie
[2] Department of Electronics, Computing and Mathematics, University of Derby, Kedleston Road, Derby, United Kingdom
k.fatema@derby.ac.uk
[3] Ordnance Survey Ireland, Phoenix Park, Dublin, Ireland
{first.last}@osi.ie

Abstract. As Linked Data technologies mature geospatial data institutions are seeking to exploit these advantages and selectively serve their commercially valuable data to customers on the web. An access control solution is required to ensure customers are not accessing unauthorized data. In this paper we present a fine grained and flexible access control method for geospatial Linked Data. Our declarative approach provides a vocabulary to model data licenses and templates. A license models what a customer is allowed to access at a fine granularity and a template models how data is allowed to be accessed. Each template contains variables and the values that these variables are allowed to contain are specified. This provides great flexibility and allows the creation of templates for a range of different data access use cases. We propose an implementation architecture for our approach which proposes a license and template-based access control web service that sits on top of a triple store, SPARQL processor and a RESTful API for access. We also present a case study where a prototype implementation of our access control approach was applied to Ordnance Survey Ireland's controlled access geospatial Linked Data.

Keywords: Linked Data, Access Control, Geospatial, Ordnance Survey Ireland

1 Introduction

As evidenced by the LOD cloud analyses[1] that have taken place over the past number of years, the growth of the cloud indicates an increased initiative in the adoption of the Linked Data principles [1] for the publication of geospatial data. While the LOD cloud (in theory) contains datasets that are open (openly accessible under a particular license), many institutions have valuable proprietary data they do not wish to make

[1] http://lod-cloud.net/

open, but may wish to serve it to authorized customers as Linked Data. An access control [2] solution is required in this situation to ensure that institutions can still reap the benefits of Linked Data (data enrichment through linking to external datasets, serving fine grain data over the web, etc.) while ensuring that customers are not accessing data they are not authorized to access.

An example of this is Ordnance Survey Ireland (OSi). OSi is Ireland's national mapping agency and is tasked with capturing and recording authoritative geospatial information for the country. OSi holds both open authoritative geospatial data such as the countries regional boundary data, and controlled access proprietary data such as detailed footprint polygons for every building in the country, which has commercial value. OSi publishes its boundary datasets as Linked Data on *data.geohove.ie* [3]. Currently, OSi are trialing ways to serve their controlled access data as Linked Data to customers on the web in a fine grained manner that can accommodate for different customer data access needs, and hence, they are looking for and access control approach to do this.

The research question investigated in this paper is: how can we develop an access control approach that is fine grain enough to capture the details of what a customer is allowed to access and flexible enough to meet the (potential) different data retrieval use cases of each customer, in a geospatial data retrieval scenario?

This paper describes a novel access control approach to address the problem of controlled access to controlled access geospatial Linked Data using licenses and templates. We propose a vocabulary to model these licenses and templates in RDF. In this approach, customers possess an instance of a license which captures the data that they have permission to access. Templates are created by administrators and these are used to model how particular data can be accessed. Each template instance contains a SPARQL query (formulated to retrieve specific data in a specific way) with placeholder variables. A template instance also contains descriptions of the placeholder variables in the query, which specify a range of values that these variables are allowed to contain. Through the creation of a SPARQL query with variable placeholders and then specifying what each placeholder can contain. The template model is enabled to be flexible for changing customer needs and data retrieval use cases. Access to data is proposed through a RESTful API. Customers are distanced from SPARQL so it means that customers do not need to know the language to access data. When a customer makes a call to our API, checks are performed on a customer's license and the template they are using to ensure they do not access restricted data.

The remainder of this paper is as follows: *Section 2* presents geospatial access control requirements and use cases from OSi. *Section 3* presents related work in the areas of general access control and then access control tailored for Linked Data. *Section 4* presents our access control approach where we detail all aspects of it and provide examples of licenses and templates. *Section 5* presents a case study where we applied our access control approach to OSi's controlled access geospatial Linked Data. In *Section 6* we conclude the paper.

2 Requirements and Use Cases

Ordnance Survey Ireland (OSi), Ireland's national mapping agency has undertaken an initiative to publish more of their authoritative geospatial data as Linked Data. Currently *data.geohive.ie* hosts Ireland's regional boundary datasets and are openly available for use. In more recent work, OSi has been focusing on creating a Linked Data representation of their controlled access data. For example, OSi maintains a building dataset, which contains general and geospatial information about every building and structure in the country. This information is time consuming to identify, collect and organize, thus it holds value for OSi. Commercial organizations have to pay if they wish to avail of access to this rich authoritative data source which would be particularly valuable to architects, construction companies, utility companies etc. To serve their controlled access geospatial data to paying customers on the web, OSi wished to investigate an access control approach to do so.

OSi identified some high level requirements for such an approach which were as follows. (i) Customers wants were to be modelled declaratively in a way that would make them straightforward to manage. (ii) The customer wants were to be valid up to a certain date and be usable to access data a certain number of times. (iii) The approach must allow customers to not only retrieve data but also check the status of their licenses, in terms of what each license permits access to, expiry date and the number of times it can be used to access data. (iv) The approach must support five particular data access use cases but be flexible enough to accommodate additional use cases that customers may specify in the future. These five use cases are:

1. Retrieve the nearest X number of buildings around a point.
2. Retrieve the nearest X number of commercial buildings around a point
3. Retrieve the nearest X number of buildings around another building.
4. Retrieve all buildings of a certain type in a polygon.
5. Retrieve all buildings of a certain type in a county.

As can be seen, these use cases do not just require access control to data of a certain class or data with a certain property. They also require access control for data based on geographical geometric details, such as distance from a point and whether a point exists with a polygon. An access control approach to meet the above requirements and use cases must be able to capture these fine grain details and ensure that data outside of these details cannot be accessed.

3 Related Work

There are a wide variety of approaches spanning many years which tackle the access control problem. There is mandatory access control [4, 5] which is based on a hierarchical classification of levels assigned by a system administrator. Discretionary access control [6] is where individuals assign access to resources they control. Role based access control [7] is where permissions are associated with roles within an organization, and users of a particular role have a specific level of access permission (think of

a university scenario where students, professors and admins all have different access permissions). Attribute based access control [8] can be thought of as an extension of role based access control but access permission is based on attributes possessed by a user. These attributes can be a user's name, age, qualification etc. A comprehensive survey on RDF access control approaches has been published by Kirrane et al [2], we will discuss the approaches more relevant to our work.

Steyskal and Polleres [9] examine the suitability of using a language called Open Digital Rights Language (ODRL) to express access control policies for RDF data. Such policies specify specific access scenarios for users, which include access to datasets, data request limits, data access time windows etc. The authors further developed [10] formal semantics for ODRL, with the objective of enabling reasoning over ORDL-based policies to enforce access control.

Sacco and Passant [11] developed an extension to the Web Access Control [12] vocabulary, called the Privacy Preference Ontology (PPO) which is more expressive and can specify restrictions to specific RDF data. SPARQL ASK queries are posed over the policy to determine if a user has access to the data in question.

The work of Calero et al. [13] describes an approach which proposes access control policies that are modelled in OWL/RDF and data access for specific users is determined through a SPARQL-based reasoning technique.

Reddivari et al. [14] propose an access control policy which specifies actions a user has permission to use against data in a triple store and they propose the RDF Store Access Control Policy framework to analyze and enforce such a policy.

The Social Semantic SPARQL Security for Access Control vocabulary was proposed by Villata et al. [15]. The purpose of this vocabulary is to model access control policies at the named graph level, and specifies actions a user can take to data in a graph they have access to.

The approach proposed by Chen and Stuckenschmidt [16] is concerned with restricting access to instance data based on classes and properties. Access control policies specify the classes and properties a user has access to. Their approach uses query rewriting. When a user is attempting to access data that they do not have permission to access, the query is rewritten to filter out that data so they do not gain access to it.

Many existing approaches are concerned with modelling an access control policy for a user and performing an analysis over what a user is attempting to access and the policy to enforce the access control. Our approach models access control policies according to our access control model as licenses. It provides an extra layer of control through templates. Data can only be accessed through templates and administrators create templates which specify exactly which data can be accessed and how it is accessed. Access control is enforced with our approach through analyzing a user input, their license and the template they wish to utilize. Our access control approach is specialized for retrieving fine grain geospatial instance data, which may only be retrieved through geospatial reasoning (such as finding specific features with a 100m radius of a specific point). Our approach does this on the fly and return data to the user without the need for this reasoning to be performed or the data partitioned beforehand (for example partitioned in a named graph).

4 Access Control Approach

In this section we present in detail our access control approach. We describe the license and template model, providing examples of each, and then present the proposed implementation architecture for our approach.

4.1 Access Control Model

The access control model is an OWL/RDFS vocabulary that is used to model both licenses and templates. The access control model vocabulary is available here[2] and will be hosted on OSi's Linked Data Platform[3] in the near future.

License.
The license section of the model is used to capture the details a user will be able to access. It also captures information such as when the license is due to expire and how many times it may be used in conjunction with a template to retrieve data. In addition, a license also captures which existing templates are permissible to be used with it. *Figure 1* displays an overview of the license section of the access control model.

A license instance will be of class *License*. A license is linked to an owner via the *licenseOwner* property. The *queryExecutionNumber* property indicates the remaining number of times a license can be used to execute a template. The *licenseExpiryDate* property indicates the license expiry date. The *templateAllowed* property is used to link the templates that are allowed to be used with a license.

A license can have multiple *LicenseFields* and these can have multiple *licenseFieldValues*. The access control model contains multiple sub-classes of the *LicenseField* class which are used to explicitly capture what a license field specifies. These subclasses are the *FeaturesAllowed* class, the *FeatureURI* class, the *GeographicalPolygon* class, the *GeographicalPoint* class, the *Radius* class and the *FeatureNumber* class.

The *FeaturesAllowed* class is used to specify in a license the types of geospatial features that are permitted to be searched. The *licenseFieldValue* for the *FeaturesAllowed* class would be a feature type from a geospatial vocabulary (i.e. Buildings, Trees, Cities, Towns etc.)

The *FeatureURI* class is used to specify in a license the URI of a (GeoSPARQL) feature that is permitted to be used in a search. The *licenseFieldValue* for the *FeatureURI* class would be a URI of an instance from a geospatial dataset.

The *GeographicalPolygon* and *GeographicalPoint* classes are used to specify a geographical polygon and point geometry respectively. The *licenseFieldValue* for these classes should contain a representation of a polygon or point respectively. Representing polygons and points as WKT literals and assuming the geospatial features within a dataset have associated WKT geometries, allows utilization of GeoSPARQL's geospatial reasoning functions such as *sfWithin*, *distance* etc.

[2] https://www.scss.tcd.ie/~almeehan/accesscontrol
[3] http://ontologies.geohive.ie/osi

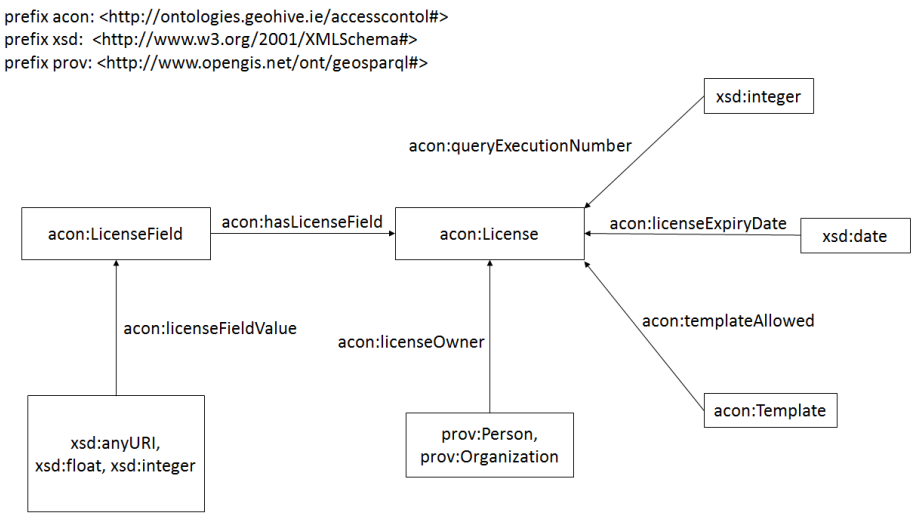

Fig. 1. License Section of the Access Control Model.

The *Radius* class is used to specify a radius in which features can be searched for around a geographical point or feature. The *licenseFieldValue* for this class should be a floating point number of the permitted radius.

The *FeatureNumber* is used to limit the number of features that can be retrieved when searching for features near a geographical point or another geographical feature (i.e. find me the 10 nearest buildings to a specific building). The *licenseFieldValue* for this class should be an integer indicating the permitted number of features allowed to be retrieved.

Listing 1 provides an example of a license. This license has four license fields. The first license field is of type *FeaturesAllowed* and the value of this field is the *Building* class from the geohive building vocabulary (ontologies.geohive.ie/osi/building). The second field is of type *GeographicalPoint* with a value of "POINT(-6.35 53.37)". The third field is of type *Radius* with a value of 100 (meters). The final field is of type *FeatureNumber* with a value of 10. This license would allow the owner to discover the nearest 10 buildings within a 100m radius of the point with latitude -6.05 and longitude 53.37.

Template.

While the license section of the access control model is concerned with modelling *what* data a user is allowed to access, the template section is concerned with modelling *how* data is allowed to be accessed. This is done through attaching a SPARQL query, with placeholder variables, to a template. The template explicitly models each placeholder variable stating the values that a variable is allowed to contain. *Figure 2* displays an overview of the template section of the access control model.


```
01: @prefix acon:
<http://ontologies.geohive.ie/accesscontrol#>.
02: @prefix geohive: <http://ontologies.geohive.ie/osi#>.
03: @prefix geohiveb:
<http://ontologies.geohive.ie/osi/building#>.
04: @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
05: @prefix ex: <http://example.com/test#>.

06: ex:License1 a acon:License ;
07:  acon:hasLicenseField _:b1 ;
08:  acon:hasLicenseField _:b2 ;
09:  acon:hasLicenseField _:b3 ;
10:  acon:hasLicenseField _:b4 ;
11:  acon:licenseOwner ex:user1 ;
12:  acon:queryExecutionNumber "100"^^xsd:integer ;
13:  acon:templateAllowed ex:Template1 ;
14:  acon:licenseExpiryDate "2018-10-10"^^xsd:date .

15: _:b2 a acon:FeaturesAllowed ;
16:  acon:licenseFieldValue geohiveb:Building .

17: _:b2 a acon:GeographicalPoint ;
18:   acon:licenseFieldValue "POINT(-6.35 53.37)" .

19: _:b3 a acon:Radius ;
20:   acon:licenseFieldValue "100" .

21: _:b4 a acon:FeatureNumber ;
22:   acon:licenseFieldValue "10" .
```

Listing. 1. Example License.

A template instance will be of class *Template*. The *templateDescription* property is used to link a description of a templates functionality. The *query* property is used to link a SPARQL query to a template. The SPARQL query is represented as a string and the placeholders within the query take the form of **$variable1**, **$variable2** and so on (see an example in *Listing 2*).

The *TemplateVariable* class is used to model the variables within a query and are linked to a template instance via the *hasVariable* property. Variables have an order number, linked via the *variableOrder* property. The *variableExpression* property is used to link the values that are allowed for a variable placeholder, to a variable.

Listing 2 provides an example of a template instance modelled according to the access control model. This template is concerned with retrieving a specific number of a specific feature type within a certain radius of a specified point. *Line 07* contains the SPARQL query, note the $variable1 to $variable4 placeholders.

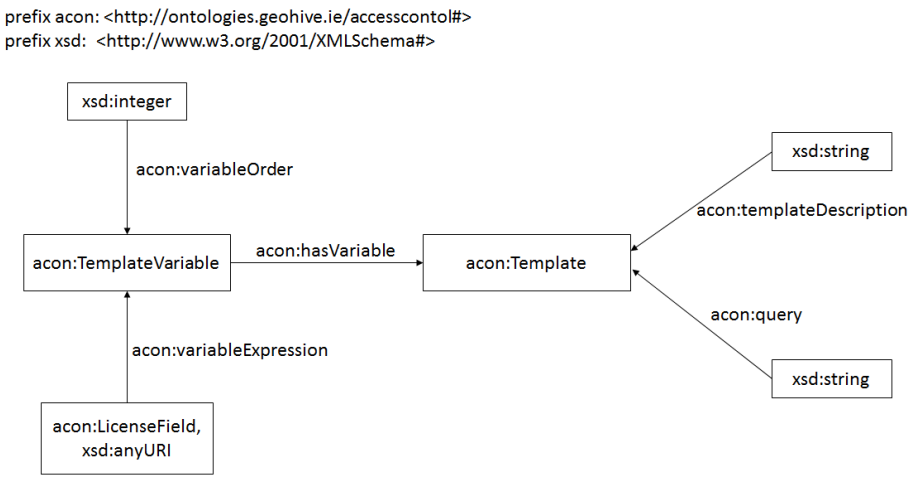

Fig. 2. Template Section of the Access Control Model.

Lines 08-10 model variable 1. It has a variable expression of *acon:*FeatureNumber. *Lines 11-13* model variable 2. It has a variable expression of geohiveb*:Building*. *Lines 14-16* model variable 3. It has a variable expression of *acon:Radius*. *Lines 17-19* model variable 4. It has a variable expression of *acon:GeographicalPoint*.

4.2 Proposed Implementation Architecture

This section describes the proposed architecture of how our access control approach should be implemented. This proposed implementation consists of five parts: the access control model, a RESTful API, a Template Selector Component, a Template Analyzer Component and a Query Processor Component. *Figure 3* shows a high level architecture diagram of the proposed implementation for our approach.

Access Control Model.
The licenses and templates will be created as desired and stored in separate named graphs in a triple store.

RESTful API.
Access to our access control approach is proposed through a RESTful API call. We propose two main calls - a "status" and a "query" call. For each call, a user should be able to specify the format that the data will be returned e.g. JSON, CSV etc. (specified in a HTTP header). The purpose of the status call is so a user can find out details about their licenses (expiry date, number of query executions remaining) and what templates they can use with their current licenses. The status call will invoke the template processor component (described below). In a status call, a GET method is used and a user must specify their user ID. An example of a status call is as follows:

- /acon/status/{userID}

The purpose of a query call is to access data. The query call will invoke the template analyzer and query processor components. In a query call a GET method is used and a user must specify their user ID, the license they wish to use, the template they wish to use and the variable values for the template. This way the user accessing the data does not need to know the SPARQL query language to interact with our approach. An example is as follows:

- /acon/query/{userID}/{LicenseID}/{TemplateID}?variable1={variable_1_value} &variable2={variable_2_value}&variableN={variable_N_value}

```
01: ex:Template2 a acon:Template ;
02:  acon:hasVariable _:b1 ;
03:  acon:hasVariable _:b2 ;
04:  acon:hasVariable _:b3 ;
05:  acon:hasVariable _:b4 ;
06:  acon:templateDescription "This template will…" ;
07:  acon:query """
PREFIX geo:<http://www.opengis.net/ont/geosparql#> PREFIX
geof:<http://www.opengis.net/def/function/geosparql/>
PREFIX units:<http://www.opengis.net/def/uom/OGC/1.0/>
SELECT ?feature WHERE { ?feature a <$variable2> ;
geo:hasGeometry ?g1. ?g1 geo:asWKT ?g1_wkt .
BIND(geof:distance("$variable4"^^geo:wktLiteral, ?g1_wkt,
units:metre) as ?distance) FILTER( ?distance <= "$varia-
ble3"^^xsd:double) } LIMIT $variable1"""^^xsd:string .

08: _:b1 a acon:TemplateVariable ;
09:   acon:variableOrder "1"^^xsd:integer ;
10:   acon:variableExpression acon:FeatureNumber .

11: _:b2 a acon:TemplateVariable ;
12:   acon:variableOrder "2"^^xsd:integer ;
13:   acon:variableExpression geohiveb:Building .

14: _:b3 a acon:TemplateVariable ;
15:   acon:variableOrder "3"^^xsd:integer ;
16:   acon:variableExpression acon:Radius .

17: _:b4 a acon:TemplateVariable ;
18:   acon:variableOrder "4"^^xsd:integer ;
19:   acon:variableExpression acon:GeographicalPoint .
```

Listing. 2. Example Template.

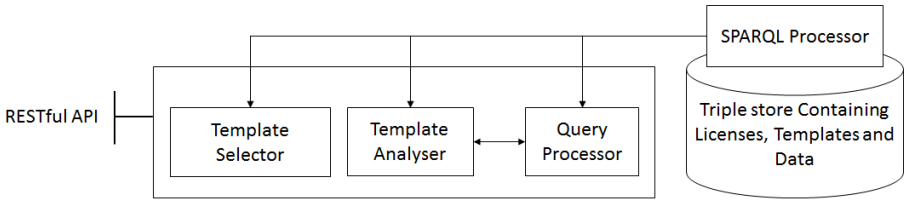

Fig. 3. High Level Architecture of Proposed Implementation.

Template Selector Component.
The purpose of the template selector component is to discover which of the existing templates are allowed to be used by a user, based on their license(s). This is done through analyzing a user's licenses and any existing templates. The license fields are checked against each variable of each template to see if a license field is allowable. If a license field is allowable for each variable in a template, then that template can be used with that license. The template selector component should also ensure that a license has not expired and it has at least one query execution left. When the analysis of the template selector is complete, it should create a link between licenses and templates via the *templateAllowed* property. In addition the template selector component should return data to a user indicating which templates can be used for each of a user's license, the description for a template and a description of how to use each template (i.e. what can be specified for the variables of a template). This data should be returned in the format specified in the HTTP header of a status call.

Template Analyzer Component.
The purpose of the template analyzer component is to validate a query call made to the access control approach. This component should ensure that the details specified in a query call are correct according to the values allowable by the variables of the template specified and also allowable according to the user's license specified. If any of the details specified in the query call are not allowable, then an error should be returned to the user and the implementation should proceed no further. If all the details specified in the query call are allowable, then the query processor component should be invoked.

Query Processor Component.
The query processor component should be invoked by the template analyzer component after it has checked and validated all the details specified in a query call are allowable. The query processor component should retrieve the SPARQL query associated with the template specified in the query call. From there it should substitute the variable values (specified in the query call) into the placeholders in the SPARQL query. The query processor component should then send the query for execution to the SPARQL processor, retrieve the results and return them to the user in the format specified in the HTTP header of the query call.

In summary, we have discussed the different aspects of our access control approach - consisting of the access control model and the proposed implementation architecture. We developed a prototype implementation according to our proposed architecture and applied in to an OSi access control scenario which is discussed in the following section.

5 Case Study

We applied our access control approach to OSi's access control scenario. We created a prototype implementation (which is not open source) according to the proposed architecture in *Section 4.2*. The prototype is a web application, implemented in Python, hosted on an Apache web server and utilizes a Parliament triple store which supports GeoSPARQL. Customer wants were modelled as licenses and use cases were modelled as templates. While we did not perform any evaluations, we made observations to (i) ensure that the licenses and templates could model the customer wants and the use cases; (ii) ensure that the implementation would correctly reject a query call when values are specified that are not allowed to according to a particular license and/or template; (iii) ensure the implementation would correctly allowed a query call with values specified that are allowable; (iv) ensure that any results returned for each use case did not contain any data that it should not be accessing.

From our observations, our access control approach worked adequately for OSi's access control scenario. We foresee no issues in regards to customers with new data access use cases as our approach allows great freedom with template creation - it is only bounded by the capabilities of a (Geo) SPARQL query. The access control model can be updated to accommodate any additional access scenarios and updates to the model will have little to no effect on the implementation of our approach.

6 Conclusions and Future Work

In this paper we have presented a new approach for access control of geospatial Linked Data. We set out to investigate an approach that can capture, at a fine granularity, the details that a user is allowed to access and is flexible enough to cater for the data retrieval needs of different users. Our approach provides a model for capturing what a user is allowed to access as a license. Our model also allows the specification of templates, which provide a highly flexible means for data retrieval use cases and are limited only by the capabilities of a (Geo) SPARQL query. We also applied our approach in an OSi access control case study. This involved the creation of a prototype implementation of or approach and it being used in a situation with specific data retrieval use cases over OSi's building Linked Data. From observations of this case study, our approach worked adequately.

Future work will involve improving upon the prototype implementation (used in the case study) and investigating the efficiency aspect of our approach as it is unknown how well it would fare in situations where there are large numbers of licenses and templates.

Acknowledgements. This research is supported by the ADAPT Centre at Trinity College Dublin which is funded by the SFI Research Centers Program (Grant 13/RC/2106).

References

1. Bizer, C., Heath, T. and Berners-Lee, T., 2009. Linked data-the story so far. *International journal on semantic web and information systems*, *5*(3), pp.1-22.
2. Kirrane, S., Mileo, A. and Decker, S., 2017. Access control and the resource description framework: A survey. *Semantic Web*, *8*(2), pp.311-352.
3. Debruyne, C., Meehan, A., Clinton, É., McNerney, L., Nautiyal, A., Lavin, P. and O'Sullivan, D., 2017, October. Ireland's Authoritative Geospatial Linked Data. In *International Semantic Web Conference* (pp. 66-74). Springer, Cham.
4. Bell, D.E. and La Padula, L.J., 1976. *Secure computer system: Unified exposition and multics interpretation* (No. MTR-2997-REV-1). MITRE CORP BEDFORD MA.
5. Biba, K.J., 1977. *Integrity considerations for secure computer systems* (No. MTR-3153-REV-1). MITRE CORP BEDFORD MA.
6. Osborn, S., Sandhu, R. and Munawer, Q., 2000. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security (TISSEC)*, *3*(2), pp.85-106.
7. Sandhu, R.S., Coyne, E.J., Feinstein, H.L. and Youman, C.E., 1996. Role-based access control models. *Computer*, *29*(2), pp.38-47.
8. Yuan, E. and Tong, J., 2005, July. Attributed based access control (ABAC) for web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE.
9. Steyskal, S. and Polleres, A., 2014, September. Defining expressive access policies for linked data using the ODRL ontology 2.0. In *Proceedings of the 10th International Conference on Semantic Systems* (pp. 20-23). ACM.
10. Steyskal, S. and Polleres, A., 2015, August. Towards formal semantics for ODRL policies. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web* (pp. 360-375). Springer, Cham.
11. Sacco, O. and Passant, A., 2011, October. A Privacy Preference Manager for the Social Semantic Web. In *SPIM* (pp. 42-53).
12. W3C. "WebAccessControl" [Online]. Available: https://www.w3.org/wiki/WebAccessControl [Accessed: 21-February-2018].
13. Calero, J.A., Perez, G.M. and Skarmeta, A.G., 2010. Towards an authorisation model for distributed systems based on the Semantic Web. *IET information security*, *4*(4), pp.411-421.
14. Reddivari, P., Finin, T. and Joshi, A., 2005, May. Policy-based access control for an RDF store. In *Proceedings of the Policy Management for the Web workshop* (Vol. 120, No. 5, pp. 78-83).
15. Villata, S., Delaforge, N., Gandon, F. and Gyrard, A., 2011, October. An access control model for linked data. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 454-463). Springer, Berlin, Heidelberg.
16. Chen, W. and Stuckenschmidt, H., 2009. A Model-driven Approach to enable Access Control for Ontologies. In *Wirtschaftsinformatik (1)* (pp. 663-672).

