Assessing the efficiency and information loss of an RDF graph anonymization framework

Remy Delanaux[1]

Université Lyon 1, LIRIS CNRS, Villeurbanne, France
remy.delanaux@univ-lyon1.fr

Abstract. In this paper, we evaluate the impact of a declarative framework for privacy-preserving publishing relying on SPARQL queries for specifying both privacy policies and anonymization operations.
We focus on the performance and the utility loss on this anonymization framework on RDF data. We first discuss and select utility measures to compare the original graph to its anonymized counterpart. We then define a method to generate new privacy policies from a reference one by inserting incremental modifications. We finally study the behavior of the framework on various real-world and synthetic RDF graphs.
We show that our anonymization technique is effective with reasonable runtime on quite large graphs (several million triples) and is progressive: the more specific the privacy policy is, the lesser its impacts are. We finally discuss the structural graph-based measurement and analyze its relevance.

1 Introduction

RDF is a graph-based data model accepted as the W3C standard for the Linked Open Data (LOD). The LOD cloud is rapidly growing and contains 1,231 RDF graphs connected by 16,132 links (as of June 2018).[1] Since 2007, the number of RDF graphs published in the LOD has grown by two orders of magnitude. Nevertheless, the participation of many organizations and institutions to the LOD movement is hindered by privacy and identity leakage concerns. Personal data are ubiquitous in many of these data sources and recent regulations about personal data, such as the EU GDPR[2], make these organizations reluctant to publish their data in the LOD.

While there has been some effort [5,11] to bring data anonymization techniques from the relational database world to the LOD, such as variations of k-anonymity [13,8,9], most of the state of the art is either based on differential privacy techniques for relational data [3,7], or access control techniques for LOD [6,14,12,10].

We want to study how the query-based, data-independant approach presented in [2] fares in actual, concrete anonymization contexts using real RDF

[1] See https://lod-cloud.net/
[2] See https://eugdpr.org/

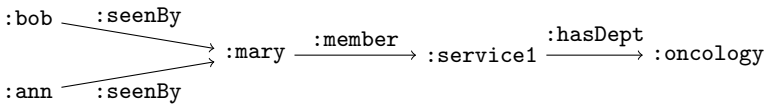


Fig. 1: Graphical representation of the G graph

graphs and tangible privacy constraints. While a static, sound anonymization approach is good in theory, it may raises other issues in practice, notably how to deal performance issues and how big is the utility loss depending on the graph or the privacy constraints provided to the algorithm. To evaluate this, we will focus on designing and performing experiments to evaluate the utility loss of such a static RDF anonymization soluion, and study the various factors affecting the results of such experiments.

The paper is organized as follows: we present in Section 2 the formal background of this anonymization framework and the safety model it is based on. Section 3 details the objectives and means of our evaluation, and Section 4 details the experimental software and hardware setup used for the experiments. Finally, after a detailed study of the results of each experiment in Section 5, we conclude the paper in Section 6 with other possible evaluation directions.

2 Background: the safety model

For the sake of brevity, we do not recall all the formal definitions of RDF graphs and SPARQL queries and we refer the reader to [4] for classical definitions. An *RDF graph* is a finite set of *RDF triples* of the form (s, p, o) where s is called the *subject* (the source of the edge), p the *predicate* (the label of the edge) and o the *object* (the target of the edge). Let us consider for instance the following RDF graph G that is made of four triples, see Figure 1 for a graphical representation:

```
:bob :seenBy :mary.     :mary     :member  :service1.
:ann :seenBy :mary.     :service1 :hasDept :oncology.
```

The declarative query language for RDF graphs is named SPARQL. Intuitively a (basic) SPARQL query is a graph pattern to be found in the database. For instance the following query P looks for people seen by a member of a service in a hospital having an oncology department. The answers to P on G, written $\mathsf{Ans}(P, G)$ are the constants `:bob` and `:ann`.

```
SELECT ?x WHERE {?x :seenBy ?y. ?y :member ?z. ?z :hasDept :oncology.}
```

The SPARQL standard also defines *update operations* which are written `DELETE` $D(\bar{x})$ `INSERT` $I(\bar{y})$ `WHERE` $W(\bar{z})$. Their semantics is to find the answers of W in the database, then from these answers to delete the D pattern and to insert the I pattern. For a sequence of update operations, we write $O(G)$ to denote its application on the G graph.

The framework developed in [2] is a follow-up of the one in [1], extended to define *safety* as follows. Intuitively, an RDF graph G is safely anonymized by

operations O according to a privacy policy $\mathcal{P}$ expressed as a set of SPARQL queries if it does not disclose any new query result when it is joined with any external RDF graph G' *even if the latter does not satisfy the privacy policy.* This a quite strong requirement that captures the following idea. If one considers that a SPARQL query $P \in \mathcal{P}$ is to be protected, then (i) all its answers on G must contain some anonymous blank nodes, no answer should be precise; (ii) the addition of external linked data cannot help to obtain precise answers.

Definition 1 (Safe anonymization instance). *Let G be an RDF graph, let O be a sequence of update queries called* anonymization operations *and let $\mathcal{P}$ be a set of queries called* privacy queries. *We say that $(G, O, \mathcal{P})$ is* safe iff *for every RDF graph G', for every $P \in \mathcal{P}$ and for every tuple of constants $\bar{c}$, if $\bar{c} \in \mathsf{Ans}(P, O(G) \cup G')$ then $\bar{c} \in \mathsf{Ans}(P, G')$.*

We first show that deleting triples may guarantee some privacy but not safety as defined above. Example 1 shows that the problem for safety comes from a possible join between an internal and an external constant. This can be avoided by replacing some critical constants by so called *blank nodes*, as shown in Example 2. Blank nodes are part of the RDF standard and are basically anonymous nodes in the graph.

Example 1. Let O_1 be the update query that deletes all the triples instances of the `:seenBy` property. The resulting anonymized RDF graph $O_1(G)$ is the following:

```
:mary :member :service1.        :service1 :hasDept :oncology.
```

However, O_1 is not safe since the union of $O_1(G)$ with an external RDF graph G' containing the triple (`:bob`, `:seenBy`, `:mary`) will provide `:bob` as an answer (which is not an answer of P against G' alone).

Example 2. Consider the following update query O_2:

```
DELETE {?x   :seenBy ?y.   ?y   :member ?z.   ?z   :hasDept :oncology}
INSERT {_:b1 :seenBy _:b2. _:b2 :member _:b3. _:b3 :hasDept :oncology}
WHERE  {?x   :seenBy ?y.   ?y   :member ?z.   ?z   :hasDept :oncology}
```

The result RDF graph $O_2(G)$ is made of the following triples where all nodes' identifiers but `:oncology` are replaced by blank nodes:

```
_:b1 :seenBy _:b2.      _:b2 :member  _:b3.
_:b4 :seenBy _:b2.      _:b3 :hasDept :oncology.
```

$O_2(G)$ is safe. In addition, since all the occurrences of the join variables are replaced by the same blank nodes, its structure is preserved, notably, the number of answers $\mathsf{Ans}(P, G)$ is preserved, i.e., $|\mathsf{Ans}(P, G)| = |\mathsf{Ans}(P, O_2(G))|$.

We now define *data-independent* safety for a sequence of anonymization operations independently of *any* RDF graph and its related computational problem. Given a *set* of SPARQL queries $\mathcal{P}$ to protect, the data-independent Safety problem consists in finding a sequence of SPARQL update operations O ensuring that $(G, O, \mathcal{P})$ is safe for any given G.

Definition 2 (Safe sequence of anonymization operations). *Let O be a sequence of anonymization operations, let $\mathcal{P}$ be a set of privacy queries, O is safe for $\mathcal{P}$ iff $(G, O, \mathcal{P})$ is safe* for every RDF graph G.

Problem 1. The data-independent SAFETY problem.
Input : $\mathcal{P}$ a set of privacy queries
Output: A sequence O of update operations such that O is safe for $\mathcal{P}$.

A solution to the SAFETY problem is to generate operations as the ones in Example 2. In a paper currently under review [2], we have shown that Algorithm 1 presented in the companion appendix computes sound solutions to the SAFETY problem. As exemplified in Example 2, we also proved that counting queries are preserved through the anonymization process.

Our approach therefore creates anonymized versions of any given RDF graph, by generalizing specific parts of the graph which may break some links between entities.

3 Goals and measurements

Our objective is to test this anonymization framework in plausible contexts, and evaluate how it fares in terms of practical usability. To do so, we identify *relevant utility measurements*, as opposed to the theoretical guarantees granted by the framework's algorithm. Indeed, the preservation of the cardinality of answers is guaranteed [2], but to what extent is the structure of the graph preserved once anonymized?

We also want to test the framework on real-world data, as it will provide an idea of how difficult it is to define privacy policies on real-world graphs, and how efficient is the anonymization process. The generation of anonymization operations is independent of the size of the graph, but how long is it to apply these operations on a given graph?

After a careful study of possible and popular measures used in the database literature (RDF databases, relational database, graph databases), we settle on three measurements that compare the original graph G and its anonymized version $G' = O(G)$ where $O = \texttt{find-safe-ops}(\mathcal{P})$ are the operations generated from a privacy policy $\mathcal{P}$. The dimensions are as follows:

performance we measure *the running time of the anonymization process.*
precision loss we measure *the number of blank nodes* added by anonymization operations and a *variant of the RDFprec measurement* [11] which is an RDF-based measure introduced to assess k-anonymity algorithms;
structural loss we compute a *distance between degree distributions* of the original graph and its anonymized version [9], notably the Earth's Mover distance (or Wasserstein distance);

Note that this running time measurement is different from the measures in [1,2], as they were used to measure the duration of the operation *generation*, while we want to measure how long lasts the actual application of these operations on a given RDF graph.

Our precision loss measurement is a $[0, 1]$ value computed by the following formula:

$$Prec = \frac{\text{number of blank nodes added by the anonymization}}{\text{total number of IRIs in the original graph}}$$

It provides a precision value relative to the contents of the original graph, and how destructive was our anonymization in replacing some of the graph's IRIs by blank nodes. The Wasserstein distance between two degree distributions of the same size and sum is the "minimum work" necessary to transform one distribution into the other. Here, this distance represents the amount of nodes moved times the distance by which it is moved. This measurement is useful in the sense that it allows us to see structural differences in the graph, for example if a lot of "core" nodes (with a lot of incoming and outgoing edges) have been modified. This type of changes may not perceptible when computing simple precision values, while computing such a distance captures more abstract changes like this one.

Many RDF graphs amounts to several hundred millions of triples, and performing any editing operation on them could potentially account for a long time, so we need to take this into account in our evaluation. We will therefore measure how long it takes for our hardware and software configuration to run the whole anonymization operation sequence on a given graph, and how the input of the algorithm (and therefore the content of this sequence) affects the running time. Note that the generation of anonymization operations itself is negligible compared to the cost of applying the operations.

Eventually, all these measurements will be performed on various privacy policies to test their influence on the algorithm results. These values will be computed depending on the privacy policy's *specificity*, i.e. how many results its queries return on the original graph. The actual computation of privacy policies with various specificity values will be detailed in the following section.

4 Experimental setup

We selected multiple RDF graphs of various purposes, sizes and structures: smaller and bigger sets of data, real-world data as well as synthetic data for benchmarking. The criteria to select them were (i) their availability as usable data dumps; (ii) the fact that they presented explicit, named entities such as users or products, to mimic sensitive data; (iii) the simplicity and readability of their schema and vocabulary, to put ourselves in the role of a real-world Data Protection Officer designing privacy policies.

For each graph, we have also defined a *reference privacy policy* called *golden privacy policy*. The details regarding each graph and their associated golden privacy policies are reported in Table 1. The indicators are (i) *the cardinality of the graph* (number of triples) with *the number of IRIs* (the number of different identifiers appearing in subject, predicate or object position) as well as the *number of (anonymous) blank nodes* already in the graph; (ii) the *cardinality* (number of queries) and *size* (total number of triples) of each golden privacy policy $\mathcal{P}$.

Table 1: Summary of the various graphs used in our experiments.

Graph	Number of triples	Number of IRIs	Number of blank nodes	Privacy policy cardinality	Privacy policy size
TCL[3]	6,443,256	13,672,913	1,237,805	7	19
Synthetic transportation data based on real graphs					
Drugbank[4]	517,023	1,218,501	0	2	6
Real-world data about approved drugs					
(Swedish) Heritage[5]	4,970,464	12,421,192	0	2	6
Real world Europeana Swedish heritage data					

To provide a numerical value modeling this input, we define the notion of *policy specificity*: a policy is *more specific* (or *less general)* than another if the sum of the results of its queries is *smaller*.

The *golden privacy policy* of each RDF graph is manually written with plausible queries based on the semantics and vocabulary of each graph. Golden privacy policies provide a simple representation of what a data provider could seemingly want to hide from their data. In a real-life scenario, these privacy policies should be defined by the privacy officer in charge of defining the public part of the database to be published. For example, the chosen queries for the TCL graph aim at hiding the given name, family name, postal address, birth date and subscription details of any user existing in the graph. Full privacy policies (as SPARQL queries) are reported in the companion appendix available online[6].

To assess the resulting utility after anonymization, we need to create mutations of the golden privacy policies. The type of mutation considered here the replacement of a variable in a policy query's body by a constant, the resulting mutated privacy policy will therefore *be more specific*. The idea is to measure the utility loss on the anonymized graph as a function of the specificity of the privacy policy: a mutated privacy policy should be *less destructive*: indeed, as the mutated applies only to more specific cases, lesser blank nodes should be introduced. In the case of an extremely specific privacy policy, for instance when it hides only a completely instantiated subgraph without variables, no blanks nodes should be introduced. Note that these experiments only make sense if the privacy policy's queries return answers on the original graph.

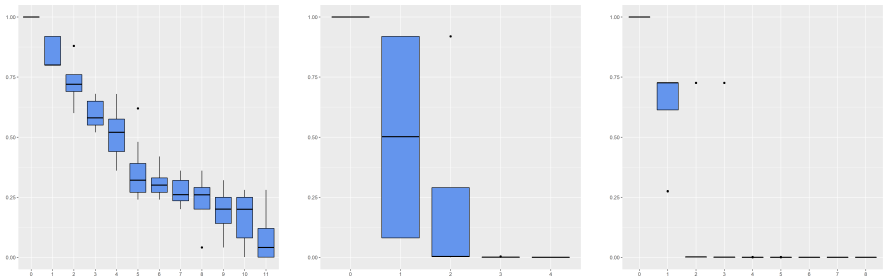
For each graph, we generate a finite set of mutations of the original golden privacy policy by running t times m random successive mutations on this original policy (if possible), creating at most $t \times m$ points where utility can be measured. The parameters are therefore t, the number of parallel copies of the golden privacy policy (decide arbitrarily to have enough values to plot) and a number

[3] Generator program: https://github.com/RdNetwork/DataLyon2RDF/ / Original data portal: https://data.grandlyon.com/

[4] http://wifo5-03.informatik.uni-mannheim.de/drugbank/

[5] General Europeana portal: https://pro.europeana.eu/page/linked-open-data

[6] https://perso.liris.cnrs.fr/remy.delanaux/papers/APVP2019_appx.pdf



(a) TCL ($m = 11$) (b) Drugbank ($m = 4$) (c) Heritage ($m = 8$)

Fig. 2: Specificity depending on mutation depth ($t = 7$)

m of mutations to apply in each copy. This way, we create t different threads of random incremental mutations in order to explore the impact of specificity. Formally, the specificity of a mutated privacy policy $\mathcal{P}'$ of a base golden privacy policy $\mathcal{P}$ on a graph G is defined as $\mathsf{specificity}(\mathsf{P}') = |\,\mathsf{Ans}(\mathsf{P}', \mathsf{G})|/|\,\mathsf{Ans}(\mathsf{P}, \mathsf{G})|$. The boxplots on Figure 2 describes how policy specificity evolves with mutation depth; i.e., how many successive mutations we need to perform on a privacy policy before its queries start having no results on the graph anymore. In the case of the TCL graph, 11 mutations is a fitting threshold, as shows on Figure 2. The figure shows that the mutation process nicely covers the whole spectrum of selectivity, from 0 (a privacy policy with very few or no variables that return almost no results on G) to 1 (the original golden privacy policy).

Each mutated privacy policy has its own specificity, and we compute a normalized value between 0 and 1 by dividing by the specificty of the original, non-mutated privacy policy. This will form the x-axis of our experiments.

The experiments were generated and performed using Python 2.7, using RDF graphs stored on a Virtuoso version 07.20.3230 on a Linux Ubuntu 18.04.1 server. The server is a 16GB RAM with 2 VPCU virtual machine running in Openstack.

5 Results

We present the results for the TCL, Drugbank and Swedish Heritage graphs. Experiments on other graphs, notably the well known LUBM benchmark[7], are still ongoing.

The running time depends on the size of graph and of the privacy policy. Indeed, the bigger the policy and its queries are, the longer the operation sequence will be. Table 2 indicates for each graph the length of anonymization sequence, that is the number of updates to apply, and the maximum running time of the anonymization operations across all mutations. While the number of operations generated can become high, only a few will cost time: the ones dealing with the subgraphs patterns with many occurrences in the graph. Anyway,

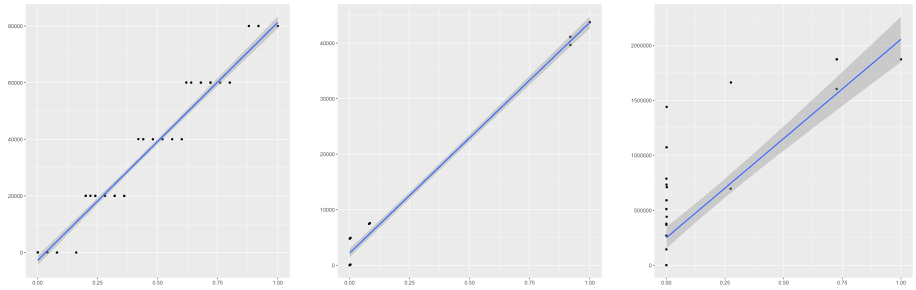
[7] http://swat.cse.lehigh.edu/projects/lubm/

Table 2: Running time of anonymization operations.

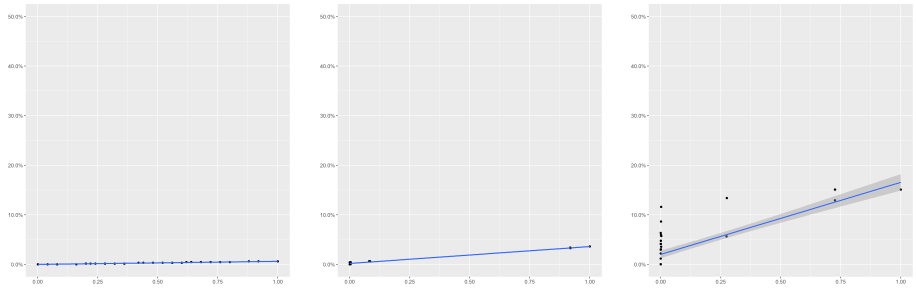
Graph	Number of operations	Algorithm duration (s)	Anonymization duration (s)
TCL	16	0.207	3.5
Drugbank	6	0.012	1.7
Swedish Heritage	14	0.013	53.6

we show that the running time is quite decent. Indeed, our approach uses only *standard SPARQL update queries* which are efficiently processed by optimized RDF databases such as Virtuoso. We also report the performance results of the operation generation algorithm, similarly to the ones computed in [2].

To analyze the precision loss and the progressiveness of our approach, we measure *the number of blank nodes* introduced by the anonymization process. Results displayed on Figures 3a to 3c show that this number grows linearly with the policy specificity: the lesser precise the privacy policy is in its selection of data, the more blank nodes will be inserted in the graph.



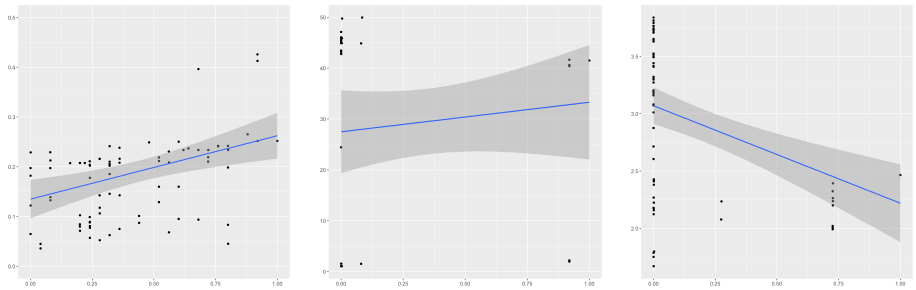
(a) TCL (b) Drugbank (c) Swedish Heritage



(d) TCL (e) Drugbank (f) Swedish Heritage

Fig. 3: Loss of precision depending on the privacy policy specificity for each graph.

As a relative precision measurement, we use the *RDFprec* variant to measure how this addition of blank nodes impacts the graph as a whole. Here, *RDFprec*



(a) TCL graph (b) Drugbank graph (c) Swedish Heritage graph

Fig. 4: Distance between degree distributions of the original graphs and the anonymized graph depending on the privacy policy specificity

counts the fraction of non blank nodes in the graph. We can observe on Figures 3d to 3f that precision is very dependent on the input: if the privacy policy covers only a specific part of the whole data (e.g. only the subscription data in a graph dealing with transportation, and containing data regarding subway lines, bus stops, etc.) its impact is quite small. This explains why in the case of the TCL graph (Figure 3a), this precision value only drops by a very low margin (99.9% to 99.4%). Nevertheless, the trend is identical: precision drops when the privacy policy gets more general. It drops to 85% in the case of the Swedish Heritage graph, and 96% for the Drugbank graph. This confirms that in general, using plausible privacy policy semantics, the number of IRIs lost in the anonymization process is not huge.

However, we notice that figures 3c and 3f on the Swedish Heritage graph have a quite large spread on the $x = 0$ line. Indeed, the privacy policy forbids the disclosure of quite general pieces of information such as the description and the time of objects in the graph. The anonymization process thus leads to many replacements by blank nodes.

We then compute the Wasserstein distance measurement between degree distributions to assess the effect of the anonymization on the graphs' structures. The results displayed on Figure 4 show that for the TCL graph the distance exhibits a clear raising tendencies: the more stringent the privacy policy is, the farther the anonymized graph is. However, the Swedish Heritage and Drugbank graphs have a very straightforward structure and fairly simplistic policies: this creates very few different specificity values, but where the distance can vary greatly depending on how prevalent are the edited nodes in the graph. The identification of another metric that captures the preservation of structures applicable to RDF graphs is still ongoing, as well as the design of more intricate policies.

6 Conclusion

The results of the various experiments we performed on this anonymization framework exhibit and confirm three main points regarding its output. First,

the use of native SPARQL elements let us achieve a satisfying running time even anonymizing large graphs, with plausible input policies. The fact that the anonymization algorithms only delete graph patterns that are necessary for satisfying the given privacy policy highlights the fact that the anonymized graph still have a decent usability and do not lose much data. Finally, our exploration on how to measure the structural impact of such an anonymization confirm that more thorough work is necessary to find an adequate measurement, as the distance between degree distribution quickly show its limits with simple policies and on graphs where highly frequent patterns are impacted.

Possible extensions include testing with other graphs with different characteristics (density, degree distribution, size...), experimenting other types of policy mutation (such as finer generalizations or specifications or deleting triples from policy queries), or developing other utility loss measurements.

References

1. Delanaux, R., Bonifati, A., Rousset, M., Thion, R.: Query-based linked data anonymization. In: ISWC (1). LNCS, vol. 11136, pp. 530–546. Springer (2018)
2. Delanaux, R., Bonifati, A., Rousset, M., Thion, R.: RDF graph anonymization robust to data linkage (2019 Submission under review)
3. Dwork, C.: Differential privacy. In: ICALP (2). LNCS, vol. 4052, pp. 1–12. Springer (2006)
4. Gutiérrez, C., Hurtado, C.A., Mendelzon, A.O.: Foundations of semantic web databases. In: PODS. pp. 95–106. ACM (2004)
5. Heitmann, B., Hermsen, F., Decker, S.: k–RDF-neighbourhood anonymity: Combining structural and attribute-based anonymisation for linked data. In: PrivOn@ISWC. vol. 1951. CEUR-WS.org (2017)
6. Kirrane, S., Mileo, A., Decker, S.: Access control and the resource description framework: A survey. Semantic Web **8**(2), 311–352 (2017)
7. Machanavajjhala, A., He, X., Hay, M.: Differential privacy in the wild: A tutorial on current practices & open challenges. PVLDB **9**(13), 1611–1614 (2016)
8. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: *L*-diversity: Privacy beyond *k*-anonymity. TKDD **1**(1), 3 (2007)
9. Nobari, S., Karras, P., Pang, H., Bressan, S.: L-opacity: Linkage-aware graph anonymization. In: EDBT. pp. 583–594. OpenProceedings.org (2014)
10. Oulmakhzoune, S., Cuppens-Boulahia, N., Cuppens, F., Morucci, S.: Privacy policy preferences enforced by SPARQL query rewriting. In: ARES. pp. 335–342. IEEE (2012)
11. Radulovic, F., García-Castro, R., Gómez-Pérez, A.: Towards the anonymisation of RDF data. In: SEKE. pp. 646–651. KSI Research Inc. (2015)
12. Sacco, O., Breslin, J.G.: *PPO & PPM 2.0*: extending the privacy preference framework to provide finer-grained access control for the web of data. In: I-SEMANTICS. pp. 80–87. ACM (2012)
13. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(5), 557–570 (2002)
14. Villata, S., Delaforge, N., Gandon, F., Gyrard, A.: An access control model for linked data. In: OTM Workshops. LNCS, vol. 7046, pp. 454–463. Springer (2011)