Graph Automorphism-Based, Semantics-Preserving Security for the Resource Description Framework (RDF)

Zhiyuan Lin Mahesh Tripunitara

ECE, University of Waterloo, Canada
{z72lin,tripunit}@uwaterloo.ca

ABSTRACT

We address security in the context of the Resource Description Framework (RDF), a graph-like data model for the web. One of RDF's compelling features is a precise, model-theoretic semantics. We first propose a threat model, and under it, observe that the technical challenge is really in hiding information that may be revealed by the structure of an RDF graph. We choose two quantitative, unconditional notions for securing graph-structure from the literature that address the threat model, and adapt them for RDF. We then consider the problem of devising algorithms for achieving a certain level of security while preserving the semantics of the input RDF graph. We observe that there are operations we can perform on an RDF graph that both provide such security and preserve semantics. We observe, further, that there is a natural way to quantify information-loss under these operations, and that there appears to be a natural trade-off between security and information-quality. We study this trade-off and establish fundamental results. We show that the RDF graphs that result from applying the operations induce a lattice that leads to a natural quantification of information-quality. We show also that achieving a certain level of security while retaining a certain level of information-quality is **NP**-complete under polynomial-time Turing reductions. Finally, towards an empirical assessment, we discuss our design and implementation of a reduction to CNF-SAT, and empirical results for two classes of RDF graphs. In summary, our work makes fundamental and practical contributions to semantics-preserving security for RDF.

1. INTRODUCTION

The Resource Description Framework (RDF) is a graph-based data model designed for the web. It is standardized by the World Wide Web Consortium (W3C) [19], and has seen adoption in practice. For example, it is the data model that is used by StarDog [14], the Apache Jena Triple Database [16], DBPedia [3] and GovTrack [7].

CODASPY'17, March 22 - 24, 2017, Scottsdale, AZ, USA

ISBN 978-1-4503-4523-1/17/03…$15.00

DOI: http://dx.doi.org/10.1145/3029806.3029827

As with any database, security is an important consideration in the context of RDF data. What we mean by "security" is that we do not want entities to have access to, or even infer, with high probability, information to which they are not authorized. Such unauthorized entities certainly include those that are to have no access to a database. But it could also include those that have some, but not unqualified access. For example, Alice may be authorized to discover the population of a locality, but not the identities of individuals that reside in it. Such a requirement, in conjunction with the fact that Alice may have access to complementary data from outside the database that helps her compromise security (e.g., identify individuals), makes the task of achieving security, or even characterizing a precise notion of "secure," a technical challenge. Indeed, such attacks have been demonstrated in practice [21], albeit not for RDF data.

Of course, it is easy to provide security if that is the only consideration: we simply release no information. But that is usually considered impractical. Thus, the technical challenge is to provide security while preserving the quality of information that is made available. In this context, characterizing information-quality can be a technical challenge.

Furthermore, a compelling feature of RDF is that it comes with a precise semantics [12]. Therefore, in the context of RDF, it is desirable to preserve the semantics of the original data in responses to queries, while providing security. That is, the preservation of semantics is an important aspect of preserving information-quality while providing security.

Contributions We unify all of the above themes in this work. We start by observing that RDF data can be meaningfully perceived and encoded as a kind of graph, which we call an RDF graph. We then observe that hiding labels on nodes and edges is a basic way to provide security. For example, the blatantly identifying information, "Alice," is removed from a node, and this provides some security from identification to the individual whose name is Alice.

However, the structure of the graph can leak information, particularly in conjunction with complementary information to which an attacker may have access, that is extraneous to the RDF data that we seek to secure. We may not even know exactly what such extraneous information is. Consequently, we enunciate a threat model that captures this rather general scenario (see Section 2). Based on the threat model, we argue for particular, quantitative, unconditional notions of security from prior work that are based on graph-automorphism (see Section 3). We adapt this notion to RDF (see Section 4).

```
@prefix   people: <http://www.people.com/>
people:amy   people:gender   female
people:amy   people:age   22
_:b   people:gender   female
_:b   people:age   30
```

Figure 1: An RDF graph shown as a set of triples.

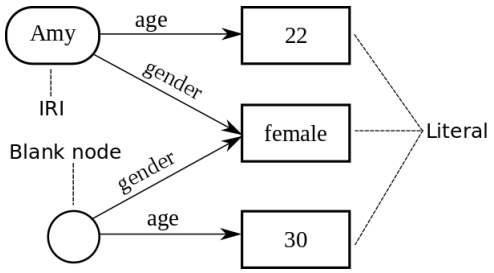

Figure 2: Visualization of Figure 1 as a graph.

We then address the issue of meaningfully characterizing information-quality. As we discuss above, there are two aspects to this. (1) How do we meaningfully characterize information-quality? And, (2) how do we reconcile semantics, which is an appealing feature of RDF? We first address (2), by studying operations that can be applied to an RDF graph G to produce another graph H such that H has the same semantics as G. We then observe that a subset of the operations, each of which corresponds to the removal of some information, can increase security (see Section 4). For that subset of operations, we establish that we induce a lattice of RDF graphs, with the original graph at the top and the empty graph at the bottom. Each edge in the lattice corresponds to an instance of an operation on the graph.

Thus, the number of edges we need to traverse in the lattice, starting at the original RDF graph at the top, to reach an element in the lattice that is an RDF graph that provides the desired level of security, is a natural way to quantify the degradation of information, and therefore information-quality, relative to the original RDF graph (see Section 4).

We then address the problem of devising an algorithm for providing a particular level of security for an RDF graph, while incurring only a certain level of degradation in information-quality. We show that this problem is **NP**-complete, where the **NP**-hardness is under polynomial-time Turing reductions [10] (see Section 4).

With the intent of carrying out an empirical assessment, we reduce the problem efficiently to boolean satisfiability in conjunctive normal form (CNF-SAT) (see Section 4). We have implemented the reduction. We present and analyze empirical results from our implementation together with a constraint solver for two classes of RDF graphs. For example, we look at the manner in which level of security that can be achieved varies with the level of information-degradation we are willing to incur (see Section 5).

As such, our contributions are to both foundations and practice in securing RDF.

Layout Apart from the sections we mention above, We provide necessary background, and present our threat model in the next section. We discuss related work in Section 6. We conclude with Section 7, which discusses also future work.

2. BACKGROUND AND THREAT MODEL

We begin with a background on RDF that suffices for our work. We discuss also semantics. We then present our threat model.

RDF is "a framework for expressing information about resources. Resources can be anything, including documents, people, physical objects, and abstract concepts." [19] Such information is expressed as a set of triples, each of the form $\langle subject, predicate, object\rangle$. Figure 1 is an example of such a set. An example of a triple in the set in the figure is $\langle \text{amy}, \text{age}, 22\rangle$. We discuss other artifacts that appear in the example below. Such a set of triples can be encoded and visualized as a kind of graph. Thus, we call such a set of triples an RDF graph, and define it as follows.

Definition 1. *(RDF Graph) An RDF Graph G is a set of triples of the form $\langle subject, predicate, object\rangle$. Each of subject, predicate and object is of a certain type: subject, predicate are* Internationalized Resource Identifier (IRI) *or* blank symbol*, and object is an* IRI, literal *or* blank symbol.

An *IRI* is a generalization of Universal Resource Identifier (URI) that allows more Unicode characters. It is used in RDF as a globally unique identifier for an entity. It explicitly specifies which entity the vertex represents. In other words every appearance of the same IRI represents the same entity even in different RDF graphs. A *blank symbol* serves the same function as an IRI, except that it is anonymous, i.e. does not reveal what entity it represents. A *Literal* in RDF is similar to that in programming languages. They are syntactic representations of boolean, character, string, or numeric values. Each $\langle subject, predicate, object\rangle$ can be seen as a pair of labelled vertices, the *subject* and *object*, connected by a labelled, directed edge, the *predicate*.

In Figure 1, *people:amy* is a shorthand for *http://www.people.com/amy*, which is an IRI. Every appearance of the same IRI, even in another RDF graph, denotes the same entity. Similarly *people:gender* and *people:age* are IRIs. *_:b* denotes a blank symbol, with a random identifier b that has no explicit meaning. Different blank nodes may have different identifiers associated with them. A graph-like visualization of Figure 1 is Figure 2.

In the RDF standard [12], a predicate of a triple can be an IRI only. However, ter Horst [27] introduced a notion of a *Generalized RDF Graph* which allows blank symbols to be used as the predicate in a triple. In this work, we adopt the generalized notion of RDF graph. Our work is applicable to the more restricted notion as well.

Semantics An appealing feature of RDF is that it has a precise and meaningful semantics [12, 27]. RDF semantics is model-theoretic. We explain model-theoretic semantics, as it pertains to RDF, via a simple example. Consider the assertion, "Bob is a musician and a parent." An aspect of the assertion is whether it is true. Another is other assertions we can draw from it, e.g., "Bob is a parent." RDF semantics

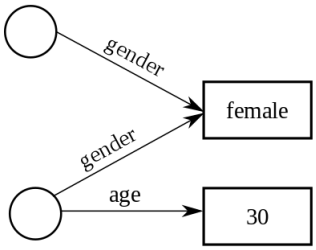

Figure 3: A graph entailed by the graph in Figure 2.

```
SELECT ?x ?y{
  WHERE ?x age ?y .
  FILTER (?y < 25)
}
```

Figure 4: Example SPARQL query for the graph shown in Figure 2. The answer to the query is, "Amy 22"

addresses these two aspects. In RDF semantics, the former is provided by an *interpretation*, and the latter by *entailment*.

An RDF interpretation maps RDF terms such as IRIs to elements of a universe. It also maps triples and RDF graphs to truth-values, which denote whether the statements represented by the triples (or the graph that contains the triples) are true in the real world. When an RDF graph G is true under the interpretation I, we say that I *satisfies* G. We refer the reader to [12, 27] for more details. In our example, for a particular person to whom we refer as "Bob," and particular interpretations of "musician" and "parent," our assertion may be true. For other interpretations, it may be false.

RDF entailment decides whether truth-value is preserved between different graphs. An RDF graph G is said to *entail* an RDF graph H if every interpretation that satisfies G also satisfies H. In other words, the truth of G under *any* interpretation is preserved in H. In this case we write $G \models H$. Entailment helps connect model-theoretic semantics to real-world applications. If A entails B and we assert that A is true in the real world, then we know that B is also true. We can think of the information in B as being contained in A. For example, our assertion, "Bob is a musician and a parent," entails the assertion, "Bob is a parent."

As we discuss further in Section 4, entailment allows us to secure an RDF graph while retaining its semantics. Think of graph G as the original graph and H as the secured version. If G entails H, then H has the same RDF semantics as G. H might not contain as much information as G, but all the information in H is semantically consistent with the information in G.

The *interpolation lemma* [12] allows us to implement this idea syntactically. To explain the interpolation lemma we first define the notion of *instance*: A graph G is an instance of another graph H if there exists a partial mapping, h, from the set of blank vertices in H to a set of IRIs, literals, and blank vertices such that G can be obtained from H by replacing every blank vertex b in the domain of h with $h(b)$. The interpolation lemma then simply states that an RDF graph G entails another RDF graph H if and only if a subset of G is an *instance* of H.

The interpolation lemma translates the semantic definition of entailment into simple syntactic requirements. We present an example here, and discuss the specific operations we consider in Section 4. The graph in Figure 3 is entailed by the graph in Figure 2. An operation that results in a new graph that is entailed by this original graph is blanking. We could, for example, blank the vertex labeled "Amy." Thus, the new graph still has a vertex that represents a person, and in that manner, preserves semantics. Similarly, we can delete Amy's age from the graph and still preserve the graph's semantics. Therefore the graph in Figure 2 entails the graph in Figure 3.

SPARQL The de facto standard query language for RDF is SPARQL[22]. In any endeavor to secure an RDF graph, it is important to consider the language via which queries are issued. Indeed, as we discuss in more detail in Section 6, some prior work proposes rewriting of SPARQL queries as a security-mechanism. In Figure 4 we show an example query in SPARQL, and the result of that query when it is issued against the RDF graph in Figure 2. The meaning of the query is: return all x and y such that *"x age y"* is a triple, and y is less than 25.

For the somewhat strong threat model we adopt below, and the corresponding security property we adopt in the next section, the specifics of SPARQL are rendered irrelevant, except for a particular aspect of semantic entailment. In RDF semantics as specified by Hayes and McBride [12] and ter Horst [27], two blank vertices with distinct identifiers are not assumed to be necessarily distinct. In queries in SPARQL, however, two blank vertices with distinct identifiers represent distinct entities. Arenas et al. [2] give a simple example of two RDF graphs and a SPARQL query, such that under the RDF semantics of [12, 27], we expect the response to the query to be the same for both graphs, but under SPARQL semantics, the responses are different.

In this work we follow the SPARQL semantics, as we expect, as is the case in practice, that SPARQL is the query language via which a user extracts information about an RDF graph. This choice affects the definition of an instance: two blank nodes with distinct identifiers cannot be replaced with the same vertex in a partial mapping from H, to an instance, a subset of G.

Threat model Our threat model is as follows. The "good guy" has an RDF graph G and a subset, A, of vertices in G that he would like to protect. Recall, from our discussions on RDF above, that vertices in an RDF graph are subjects and objects. Thus, protecting the subset of vertices A corresponds to disallowing an adversary from targeting any of those subjects and objects in an attack. This in turn can be achieved by maintaining the anonymity of those vertices.

Let G' be the secure version of G that is released publicly. We model the attacker's prior knowledge with another RDF graph H, which we assume is a subgraph of G. Our threat is: the attacker seeks to deanonymize any of the vertices A in G using the information in G' and H. That is, we deem an

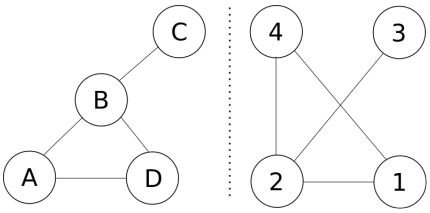

Figure 5: Graph Isomorphism and Automorphism. The graph to the left is non-trivially automorphic, and is isomorphic to the graph to the right.

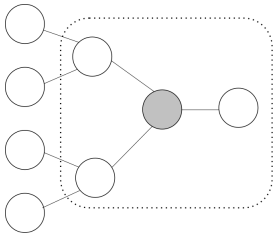

Figure 6: *k-neighbourhood-isomorphism.* The shaded vertex is 3-neighbourhood-isomorphic for a distance-1 neighbourhood. Its distance-1 neighbourhood is shown by the dotted box. The two vertices to its immediate left have isomorphic distance-1 neighbourhoods.

attacker to have succeeded if she is able to correctly identify, within a certain probability, that a vertex in G' is a vertex in H. The probability is a quantification of the desired security. Note that some or all the vertices in A may appear in G' and/or H. We assume also that the attacker has unbounded computational power.

For example, suppose $H \subset G$, the prior knowledge of the attacker, is the graph in Figure 2. And G', the graph that is released publicly, is the graph in Figure 3. We want the anonymity of the node Amy, that appears in G (and H) to be preserved. That is, $A \ni$ Amy. The probability we desire is some value < 1. Then, the attacker succeeds because she is able to identify, with probability 1, that the node with no label, that does not have an edge labelled 'age,' is Amy.

Of course, G may have other nodes that have been removed to result in G', the graph in Figure 3. In our discussion of this example, we bias in favor of the attacker, that one of the unlabelled nodes to the left in Figure 3 is indeed Amy. In any case, if we remove the edge labelled 'age' from the graph in Figure 3, then the attacker succeeds with probability at most $\frac{1}{2}$.

Our threat model may seem *too* strong. Not only do we not bound an attacker with regards to computational power, but also allow H to be any subgraph of G. This models our uncertainly as to the attacker's extraneous or prior knowledge of G. Of course, to allow $H = G$ is meaningless, because then, the attacker has no need to deanonymize any node in G'. However, it is possible that any strict subgraph of G may be the attacker's prior knowledge.

3. ISOMORPHISM-BASED SECURITY

Given the threat model in the previous section, we now discuss notions of "secure" for graphs, that we then adapt to RDF graphs. An initial mechanism for security is to simply blank vertices and edges in an RDF graph. For example, as we do to the node "Amy" in the graph in Figure 2 so that that node is a blank in Figure 3. Indeed, we expect that we must do this, at the minimum, for every vertex in the set A of vertices we seek to protect.

However, this may not suffice. For example, suppose the attacker's extra knowledge, H, a subgraph of the graph in Figure 2, is the two subjects with their corresponding ages. Then given the graph from Figure 3, the attacker is able to identify that Amy is the blank node that does not have an edge to the age "30." Thus, the structure of G' reveals information. This is the technical challenge we address in this section, and that is the focus of our work. Specifically, we consider three notions from the literature that appear to address exactly the threat model we consider, but for traditional graphs. We then identify relationships between them, which leads us to choose two of the three notions. We adapt these two notions to RDF in the next section.

Common to the notions that we consider, is that they are based on *graph isomorphism* (Iso). Two graphs $G = \langle V_g, E_g \rangle$ and $H = \langle V_h, E_h \rangle$ are said to be *isomorphic* when there exists an invertible mapping $m\colon V_g \to V_h$ such that $\langle u, v \rangle \in E_g$ if and only if $\langle m(u), m(v) \rangle \in E_h$. (Our notation is customary: V_g and V_h are the sets of vertices of G and H respectively, and E_g and E_h are their sets of edges.) Such a mapping is called an isomorphism. A graph is said to be (non-trivially) *automorphic* if it is isomorphic to itself via a mapping that is not the identity. Such a mapping is called an automorphism. For example, in Figure 5, the graph to the left is non-trivially automorphic. An isomorphism is one in which we map A and D to one another, and B and C to themselves. Also, the graph to the left is isomorphic to the graph to the right. An isomorphism is: $A \to 1, \ldots, D \to 4$.

Isomorphism-based security provides anonymity by creating groups of indistinguishable vertices, i.e. for every vertex v there are several other ones that are indistinguishable from v. This lowers the probability that v is correctly re-identified from 1 unconditionally, because the attacker would have to make a blind guess from amongst the group of indistinguishable vertices.

We focus only on isomorphism-based security notions because they comprehensively secure all aspects of the graph structure, compared to other notions such as k-degree-anonymity [17], which only consider the degrees of vertices. We introduce these isomorphism-based security notions below, starting with *k-neighbourhood-isomorphism* [31].

Definition 2. *(k-neighbourhood-isomorphism [31]) Given a graph $G = (V, E)$ and integer k, we say a vertex $u \in V$ is* k-neighbourhood-isomorphic *if there exist at least $k-1$ other vertices $v_1, ..., v_{k-1} \in G$ such that $neighbour(v_1), \ldots, neighbour(v_{k-1})$ and $neighbour(u)$ are isomorphic, where $neighbour(v)$ is the distance 1 neighbourhood of v, i.e. the subgraph of G induced by the set of vertices within distance 1 to v. The graph G is* k-neighbourhood-isomorphic *if every vertex in G is* k-neighbourhood-isomorphic.

Although the above notion is defined over distance-1 neighbourhood, it can be naturally extended to consider neighbourhood of arbitrary distance d. The work of Zhou and Pei [31] is focused on distance-1 neighbourhood only, therefore we do the same. What we call *k-neighbourhood-isomorphism* is called *k-anonymity* in their work. We adopt our nomenclature to more clearly identify the notion, and because the *k-anonymity* is used extensively in the altogether different context of relational data.

A problem with *k-neighbourhood-isomorphism* is that it considers only limited neighbourhoods. See Figure 6 for example. The shaded vertex is 3-neighbourhood-isomorphic: the two vertices immediately to its left have distance-1 neighbourhoods that are isomorphic to the shaded vertex's. However if we consider the distance-2 neighbourhoods of vertices, the shaded vertex becomes unique. The two other security notions we consider below, *k-symmetry* [29] and *k-automorphism* [32], do not share the same problem because they are defined over the entire graph, and not on limited neighbourhoods.

DEFINITION 3. *(k-automorphism [32]): Given $\langle G, k\rangle$ where G is a graph and k an integer, a graph G is k-automorphic if and only if there exist $k-1$ different automorphisms $f_1, ..., f_{k-1}$ for G such that the following two properties are satisfied by all $v \in G$: (a) $f_i(v) \neq f_j(v)$ for all distinct pairs i, j, and, (b) $f_i(v) \neq v$ for all i.*

DEFINITION 4. *(k-symmetry [29]) Given $\langle G, k\rangle$ where G is a graph and k an integer, G is k-symmetric if and only if for each vertex v_i in G there exist $k-1$ different automorphisms $f_{i,1}, ..., f_{i,k-1}$ such that (a) $f_{i,m}(v_i) \neq f_{i,n}(v_i)$ if $m \neq n$, and (b) $f_{i,j}(v_i) \neq v_i$.*

Some clarification of the terms is needed when referring to *k-automorphism*. The *k-automorphism* notion, to our knowledge, was first proposed by Zou et al. [32]. In that work, after proposing a notion of security that they call *k-automorphism*, they attach to it an additional condition that they call the "different match principle." Their work deals exclusively with *k-automorphism* with the different match principle. The property of *k-symmetry* is from the work of Wu et al. [29]. That work simply drops the qualification "with the different match principle" in referring to the work of Zou et al. [32]. We adopt the approach of Wu et al. [29], and simply call the property *k-automorphism*, without qualification.

Both *k-symmetry* and *k-automorphism* are based on the notion of automorphism, and their difference is somewhat nuanced. In the former, it suffices that for each vertex, k different automorphisms exist. Thus, the k automorphisms may be different for each vertex. In *k-automorphism*, on the other hand, the same set of k automorphisms must apply to all vertices. Certainly, if a graph is *k-automorphic*, it is *k-symmetric*. Wu et al. [29] articulate the question as to whether the converse is true, and leave it open. Addressing that question is beyond the scope of our work.

The fact that *k-automorphism* implies *k-symmetry* is interesting from a security perspective: we only have to enforce *k-automorphism* to achieve both. Moreover, we can also show that *k-symmetry* implies *k-neighbourhood-isomorphism*. If a graph $G = (V, E)$ is *k-symmetric*, then for each $v \in V$ there are at least $k-1$ other automorphisms $f_1, f_2, ..., f_{k-1}$ from G to itself such that $f_m(v_i) \neq v_i$ and $f_m(v_i) \neq f_n(v_i)$ when $m \neq n$. In other words, there will be at least $k-1$ vertices $f_1(v_i), ..., f_{k-1}(v_i)$ which have neighbourhoods that are isomorphic to $v_i's$, so v_i is *k-neighbourhood-isomorphic*. Therefore G is also *k-neighbourhood-ismorphic*.

The observations above connect the three security notions that we have discussed: *k-automorphism* implies *k-symmetry* and *k-symmetry* implies *k-neighbourhood-isomorphism*. Therefore if we ensure a graph is k-automorphic, we know that it is k-symmetric and k-neighbourhood-isomorphic. This means that *k-automorphism* gives us a strong guarantee. We consider both *k-automorphism* and *k-symmetry* for RDF graphs in the next section.

4. SECURING RDF

We now adapt the notions of *k-automorphism* and *k-symmetry* to RDF graphs. As we discuss in Sections 1 2, our intent is to secure RDF graphs in a manner that preserves semantics.

Two issues need to be addressed in adapting *k-automorphism* and *k-symmetry* from Section 3 to RDF. One is that RDF is more general than the kinds of graph for which the two notions have been considered in the literature. Therefore, we need to generalize those notions to suit RDF. We do so in Section 4.1

A second issue regards information-quality. Our objective is to achieve a desired level of security with as little loss of information as possible. We achieve this goal in two ways. One is that in the process of generating the secure version, G', from G, we make creative use of RDF semantics to maintain the consistency of G' and G. The other is that we provide a quantitative measure of the cost of achieving this semantically consistent security, and require that this cost is minimized.

4.1 Adaptation to RDF Graphs

In this section we define the *k-symmetry* and *k-automorphism* for RDF. The original definition *k-symmetry* and *k-automorphism* are not applicable directly to RDF because they are defined on unlabelled, undirected graphs. RDF on the other hand, can be thought of as having explicit labels on both edges and vertices, except for blank vertices and edges. These labels can contain sensitive or identifying information. Therefore a notion of security for RDF must take these labels into consideration. The definition of an RDF graph has been discussed in Section 1. We show below how we redefine automorphism in this context.

We define the following notations to be used in definitions. Given a graph G, let $subj(G)$ be the set of all subjects, $pred(G)$ the set of all predicates, and $obj(G)$ the set of all objects. We define the *terms* of G to be $T(G) = subj(G) \cup pred(G) \cup obj(G)$. Let B be the set of blank vertices in G; then $B \subseteq T(G)$.

DEFINITION 5. *(Automorphism for RDF) An automorphism on an RDF graph is defined as a bijection f: $T(G) \longrightarrow T(G)$, which has the following properties:*

1. *Given $v \in T(G)$, $f(v) = v$ if $v \notin B$; $f(v) \in B$ if $v \in B$; and*
2. *a triple $(a, b, c) \in G$ iff $(f(a), f(b), f(c)) \in G$.*

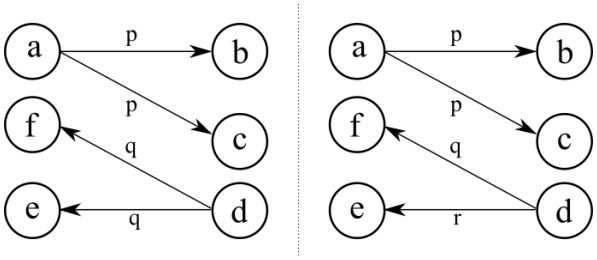

Figure 7: RDF automorphism

Definition 5 is different from the traditional definition of automorphism from Section 3. It recognizes that a non-blank vertex is globally unique, and explicitly identified by its label. Such a vertex under an automorphism cannot be mapped to any other vertex than itself. Blank vertices on the other hand do not have explicit identities. Therefore they can be mapped to other blank vertices.

Moreover, the automorphism applies to the IRIs and blank symbols no matter that they serve as vertices or edges in a graph. IRIs and blank symbols can serve as predicates in triples, in which case they act like edges in a graph. Different appearances of the same IRI as predicates are just instances of the same relationship, therefore an IRI edge can only be mapped to an edge under the same IRI. Different appearances of the same blank symbols as predicates also denote the same relationship, except in this case that the exact identity of the relationship is unspecified. Automorphism for edges can be seen as from edge type to edge type. If we map blank edge type a to blank edge type b, we must map every edge under the type a to an edge under type b. For example, in Figure 7 assuming all vertices and edges are blank, there is an automorphism that maps the vertex a to the vertex d in the left graph, because the blank relationship p can be mapped to q. Such an automorphism however does not exist in the right graph in Figure 7 because the relationship p cannot be mapped to both q and r at the same time. This notion of automorphism is consistent with the RDF semantics we discuss in Section 4.2.

With the notion of automorphism settled, we now define *k-symmetry* and *k-automorphism* for RDF.

Definition 6. *(k-symmetry for RDF) Given an RDF graph G, let the anonymization set A be a set of vertices $A \subseteq subj(G) \cup obj(G)$. G is k-symmetric with respect to A if:*

1. *every vertex in A is blank; and*
2. *for each vertex $v \in A$, there exist k automorphisms $f_1, \ldots, f_k$ such that $f_m(v) \neq f_n(v)$ when $m \neq n$.*

Definition 7. *(k-automorphism for RDF) Given an RDF graph G, let the anonymization set A be a set of vertices $A \subseteq subj(G) \cup obj(G)$. G is k-automorphic with respect to A if:*

1. *every vertex in A is blank; and*
2. *There exist k automorphisms $f_1, \ldots, f_k$ such that $f_m(v) \neq f_n(v)$ for all $v \in A$ when $m \neq n$.*

4.2 RDF Semantics and Security

Given the adaptations, in the above section, of the notions of security we consider for RDF, we consider to the issue of semantics. Specifically, we seek to achieve a level of security, e.g., 4-automorphism, while preserving the semantics of the original graph. A consequence of this requirement is that it limits the mechanisms we are allowed to use to achieve security.

The notion of a blank symbol, to which we refer in the definition of RDF graph is an important one in this context, as it represents an anonymous subject, predicate or object. We propose the use of the following two operations to secure an RDF Graph:

1. Removing RDF triples; and,
2. Replacing an IRI vertex or a Literal vertex with a blank vertex.

We point out, with regards to Operation (1) above, that a vertex in an RDF graph can exist only as part of a triple. For example, if there are no edges to an object, a vertex that corresponds to that object is no longer in the RDF graph.

These are of course not the only operations available. Some other works we discussion in Section 6 employs other techniques such as adding vertices and edges. The reason we choose these two operations only is to preserve RDF semantics. The proof of the following theorem is straightforward.

Theorem 1. *Given an RDF graph G, and another RDF graph H obtained from G by application of some sequence of the above two operations. Then $G \vDash H$.*

Cost of Security Entailment ensures that the information of the RDF graph is semantically preserved. But it does not measure how much information there is in the secured graph relative to the original. For example, the empty graph is entailed by every RDF graph, but there is zero information-quality in publishing it. In the following, we provide a quantitative measure of information based on entailment.

In the following lemma, we assert that entailment induces a lattice on the graphs that result from all possible sequences of applications of the above two operations on the original graph. At the top of the lattice is the original graph, and at the bottom, is the empty graph. An intermediate entry in the lattice is entailed by its ancestors in the lattice, and an entry's descendants are entailed by this entry and thus all of its ancestors.

Lemma 1. *The entailment relationship $\vDash$ is a partial order over the set of all possible anonymizations of G.*

Proof. Let P be the set of all possible anonymizations of G. Note that P contains G itself and the empty graph. We prove that the entailment relationship $\vDash$ is reflexive, transitive, and anti-symmetric.

Reflexivity: An RDF graph always entails itself.

Transitivity: If $G \vDash H, H \vDash I$, then by interpolation lemma a subset of G is an instance of H, also a subset of H is an instance of I. That means a subset of G is an instance of I. Therefore $G \vDash I$.

Antisymmetry: Given two RDF graphs G and H, If $G \vDash H$ and $H \vDash G$, then G and H are instance of each other. For

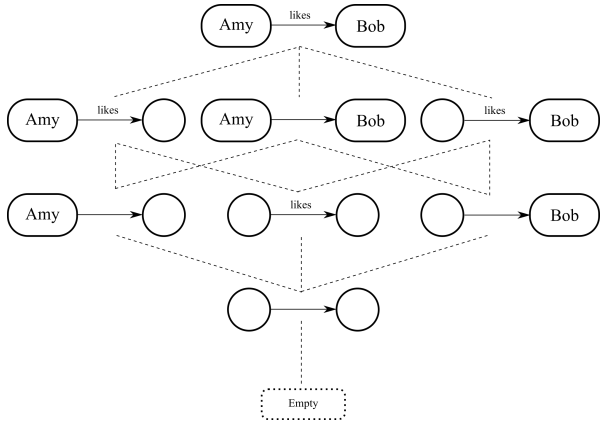

Figure 8: Lattice induced by entailment for one triple.

blank vertices, let h be the partial function used to derive instance relationship. We define $f : T(G) \to T(H)$ where $f(v) = h(v)$ for $v \in domain(h)$, and $f(v) = v$ otherwise. The function f is a bijection because h is an injection and $h(b) \in T(G)$, as discussed in Section 4.2. Therefore the relationship $\vDash$ is antisymmetric. □

It should be noted that the above result is based on the SPARQL notion of entailment. It is unclear whether this result applies to the original RDF simple entailment. As we discuss in Section 2, we adopt the SPARQL notion of entailment for this work. Figure 8 shows a lattice induced from a single triple.

The lattice provides a quantitative approach to measuring the cost of achieving security, which in turn corresponds to (degradation in) information-quality: we simply count the distance from original graph (the top of lattice) to the graph in the lattice that we choose as our secure graph. The longer this distance, the more information we lose, and the higher the cost to achieve security is.

Note that the relationship is not necessarily monotonic. That is, more degradation in information-quality does not necessarily mean we get more security. An easy way to intuit this is to observe in Figure 8, that none of the nodes in the lattice except the bottom one, "Empty," provides any security for Amy or Bob. Indeed, security can even decrease as we descend a lattice. A comprehensive discussion of this non-monotonicity is beyond the scope of this work.

Securing an RDF graph Now that we have notions of "secure," and a measure of information loss, we address the problem of devising an algorithm that transforms an RDF graph G into another graph G' such that G' preserves the semantics of G but also has a desired level of security. We define the decision problem in the form of a language below.

MINIMAL-k-AUTOMORPHISM $= \{\langle G, k, r, A\rangle$ where G is an RDF graph, $A \subseteq subj(G) \cup obj(G)$, and k, r are integers : there exists an RDF graph G' such that G' is k-automorphic with respect to A and G' is reachable from G in no more than r steps in the lattice induced by entailment.$\}$

It is easy to intuit an upper-bound hardness for MINIMAL-k-AUTOMORPHISM: it lies in **NP**. To inuit a lower-bound hardness of the above problem, we first define k-AUTOMORPHISM for *k-automorphism* of traditional graphs, that we introduce in Section 3. k-AUTOMORPHISM $= \{\langle G, k\rangle$ where G is a graph and k is an integer, such that G is *k-automorphic*$\}$. That is, k-AUTOMORPHISM is the problem of determining, given $\langle G, k\rangle$, whether G is *k-automorphic*.

LEMMA 2. *k-AUTOMORPHISM is **NP**-hard under polynomial-time Turing reductions.*

The proof for the above lemma is that for the special case that $k = 2$, k-AUTOMORPHISM corresponds exactly to a fixed-point free automorphism, which is known to be **NP**-hard under polynomial-time Turing reductions [18]. We omit the details here.

We now return to MINIMAL-k-AUTOMORPHISM, which we pose above for RDF graphs. We assert, via the following theorem that it, too, is **NP**-hard under polynomial-time Turing reductions. The proof relies on a reduction from k-AUTOMORPHISM.

THEOREM 2. *MINIMAL-k-AUTOMORPHISM is **NP**-hard under polynomial-time Turing reductions.*

PROOF. We can reduce from the k-AUTOMORPHISM problem defined above. Given $\langle G, k\rangle$ as the input for k-AUTOMORPHISM, we construct an RDF graph G' from G. For every vertex in G, we create a blank vertex in G'. We use only one blank edge type in G' for all the edges we create and for every edge in G, we create two edges in G' that points to opposite directions. Let set A include all the vertices from G'. We then solve MINIMAL-k-AUTOMORPHISM for the input $\langle G', k, 0, A\rangle$. It is easy to see that $\langle G', k, 0, A\rangle \in$ MINIMAL-k-AUTOMORPHISM if and only if $\langle G, k\rangle \in k$-AUTOMORPHISM. □

We can also define a problem MINIMAL-k-SYMMETRY based on k-SYMMETRY. A similar reduction as above establishes that MINIMAL-k-SYMMETRY is **ISO**-hard. **ISO** is Graph Isomorphism: the problem of determining, given two graphs, whether they are isomorphic to one another [8]. **ISO** is thought to be a strict superset of **P**, and a strict subset of **NP**. We omit further details here on account of space.

CNF-SAT Encoding Given the above complexity results, there is no known algorithm for solving these problems efficiently. Therefore we provide reductions from MINIMAL-k-SYMMETRY and MINIMAL-k-AUTOMORPHISM to CNF-SAT, with the intent of using a SAT solver in practice. We now discuss our reduction from MINIMAL-k-SYMMETRY to CNF-SAT.

Input: $\langle G, k, r, A\rangle$ where G is an RDF graph, k, r are integers, $k < |subj(G) \cup pred(G)|$, $A \subseteq subj(G) \cup obj(G)$.

Additional notations:

We define the following notations to be used in the encoding.

- $V = T(G)$. Let $n = |T(G)|$. $V = [1, n]$.
- V comprises two partitions: B (blank symbols) and $\overline{B}$. Assume $B = [1, b], \overline{B} = [b + 1, n]$.
- Let $f_{i,j} : V \longrightarrow V$ be an automorphism. We define a mapping $g_{i,j} : G' \longrightarrow G'$ for each $f_{i,j}$ such that for triples $t_1, t_2 \in G'$ such that $t_1 = (s_1, p_1, o_1), t_2 = (s_2, p_2, o_2)$,

  $g_{i,j}(t_1) = t_2 \iff f(s_1) = s_2 \wedge f(p_1) = p_2 \wedge f(o_1) = o_2$.

Boolean variables:

We create the following boolean variables in our CNF-SAT encoding.

1. $\forall v \in V$, a variable x_v such that $x_v = 1$ iff v is blank(ed).
2. $\forall t \in G$, a variable y_t such that $y_t = 1$ iff t is deleted.
3. $\forall v \in V$, a variable z_v such that $z_v = 1$ iff all tuples in which v appears are deleted.
4. $\forall \alpha, \beta \in V, \gamma \in A, \forall \delta \in [1,k]$, a variable $f_{\alpha,\beta,\gamma,\delta}$ such that $f_{\alpha,\beta,\gamma,\delta} = 1$ iff $f_{\gamma,\delta}(\alpha) = \beta$.
5. $\forall \alpha, \beta, \in V, \gamma \in A, \forall \delta \in [1,k]$, an intermediate variable $b_{\alpha,\beta,\gamma,\delta} \iff f_{\alpha,\beta,\gamma,\delta} \wedge x_\beta \wedge \neg z_\beta$.
6. $\forall t_1, t_2 \in G, \forall \gamma \in A, \forall \delta \in [1,k]$, a variable $g_{t_1,t_2,\gamma,\delta}$ such that $g_{t_1,t_2,\gamma,\delta} = 1$ iff $g_{\gamma,\delta}(t_1) = t_2$.
7. $\forall t_1, t_2 \in G, \forall \gamma \in A, \forall \delta \in [1,k]$, an intermediate variable $h_{t_1,t_2,\gamma,\delta} \iff g_{t_1,t_2,\gamma,\delta} \wedge \neg y_{t_1}$.

Clauses, i.e., constraints:

Below we list the CNF-SAT clauses that we generate from the input. For the sake of brevity and clarity we write logical implications of the form $a \implies b$ where necessary. Clauses of these types are transformed to $\neg a \wedge b$ in the actual implementation.

1. $\forall v \in B$, a clause: x_v.

   This encodes vertices already known to be blank.
2. $\forall t = \langle s,p,o \rangle \in G$: $y_t \implies x_s \wedge x_p \wedge x_o$.

   In the lattice, deletion of a triple occurs only after all vertices in the triple are blank(ed).
3. $\forall v \in V$, let $T_v = \{t : t = \langle s,p,o \rangle \in G \wedge (v = s \vee v = p \vee v = o)\}$. $\forall v$, a clause: $y_{t_{v_1}} \wedge \ldots \wedge y_{t_{v_{|T_v|}}} \iff z_v$.

   Vertex considered deleted if and only if every triple in which it appears is deleted.
4. clauses that correspond to: $x_{b+1} + \ldots x_n + y_{t_1} + \ldots + y_{t_{|T|}} \leq r$.

   Number of triples from G that are deleted + Number of vertices that are originally not blank being blanked $\leq r$.
5. $\forall i \in V, \forall \gamma, \delta$: $\neg x_i \wedge \neg z_i \implies f_{i,i,\gamma,\delta}$.

   Vertex not blank and not deleted only if it is mapped to itself.
6. $\forall i \in V, \forall \gamma, \delta$: $x_i \wedge \neg z_i \implies (f_{i,1,\gamma,\delta} \wedge x_1 \wedge \neg z_1) \vee \ldots \vee (f_{i,n,\gamma,\delta} \wedge x_n \wedge \neg z_n)$.

   Vertex is blank and not deleted only if it is mapped to at least one blank vertex that has not been deleted.

   - This is transformed to $x_i \wedge \neg z_i \implies b_{i,1,\gamma,\delta} \vee \ldots \vee b_{i,n,\gamma,\delta}$, and $b_{\alpha,\beta,\gamma,\delta} \iff f_{\alpha,\beta,\gamma,\delta} \wedge x_\beta \wedge \neg z_\beta$
7. $\forall i, j, p \in V, i \neq j, \gamma \in A, \delta \in [1,k]$, $\neg f_{i,p,\gamma,\delta} \vee \neg f_{j,p,\gamma,\delta}$.

   Two vertices cannot both be mapped to the same vertex.
8. $\forall t_1 = (s_1,p_1,o_1), t_2 = (s_2,p_2,o_2) \in G$, $\forall \gamma \in A, \delta \in [1,k]$,

   - $\neg y_{t_1} \wedge \neg y_{t_2} \wedge g_{t_1,t_2,\gamma,\delta} \implies f_{s_1,s_2,\gamma,\delta} \wedge f_{p_1,p_2,\gamma,\delta} \wedge f_{o_1,o_2,\gamma,\delta}$
   - $\neg y_{t_1} \wedge \neg y_{t_2} \wedge f_{s_1,s_2,\gamma,\delta} \wedge f_{p_1,p_2,\gamma,\delta} \wedge f_{o_1,o_2,\gamma,\delta} \implies g_{t_1,t_2,\gamma,\delta}$

   We perceive triples as edges in automorphism. This constraint encodes the definition of function $g_{\gamma,\delta}$.
9. $\forall t \in G$, $\forall \gamma \in A, \delta \in [1,k]$, $\neg y_t \implies (g_{t,t_1,\gamma,\delta} \wedge \neg y_{t_1}) \vee \ldots \vee (g_{t,t_{|G|},\gamma,\delta} \wedge \neg y_{t_{|G|}})$.

   - This is transformed to $\neg y_t \implies h_{t,t_1,\gamma,\delta} \wedge \ldots h_{t,t_{|G|},\gamma,\delta}$, and $h_{t,t_i,\gamma,\delta} \iff g_{t,t_i,\gamma,\delta} \wedge \neg y_{t_i}$.
10. $\forall t, t_1, t_2 \in G$ such that $t_1 \neq t_2$, $\forall \gamma \in A, \delta \in [1,k]$, $\neg g_{t,t_1,\gamma,\delta} \vee \neg g_{t,t2,\gamma,\delta}$

    The three constraints above ensures that natural constraint of the automorphism: there is a one-to-one mapping between the triples that are not deleted.
11. $\forall i \in A, \forall j \in V, \forall$ distinct δ, ψ: $\neg z_i \implies \neg d_{i,j,i,\delta} \vee \neg d_{i,j,i,\psi}$.

    The "k" different mappings in k-SYMMETRY, defined only for vertices that have not been deleted.

A similar reduction can be constructed from MINIMAL-k-AUTOMORPHISM to CNF-SAT. The only change is that where a variable $f_{\alpha,\beta,\gamma,\delta}$ is used, we use $f_{\alpha,\beta,\delta}$ to encode only k automorphisms in total, instead of k automorphisms for each $v \in A$.

Efficiency Our reduction runs in time polynomial in the size of its input. However, its output size, as suggested by our subscripts for the boolean variables above, is quartic, i.e., degree 4, in the size of the input. Certainly, a more efficient reduction may exist. An investigation is beyond the scope of this work.

5. EMPIRICAL RESULTS

We have conducted a limited empirical assessment of *k-symmetry* and *k-automorphism* on RDF with the encoding provided in Section 4. Our intent with the empirical assessment is to investigate the relationship between security and information-quality (or cost of achieving security) under realistic graphs of two different classes. More specifically we would like to understand, in practice, how the various parameters such as the input graph G, the level of security k, the anonymization set A and the minimum cost r interact.

For instance, given G and A, it seems reasonable to expect that the minimum cost r required grows with k, but how fast does it grow? Is it harder to anonymize a larger graph? These are some of the questions we investigate in this section. We intentionally keep our RDF graph inputs small so it is easier to intuit relationships between the parameters.

It should be noted that although we have investigated both *k-symmetry* and *k-automorphism*, these two notions exhibit the same behaviour during our experiments. In other words, achieving these two security notions for the same k

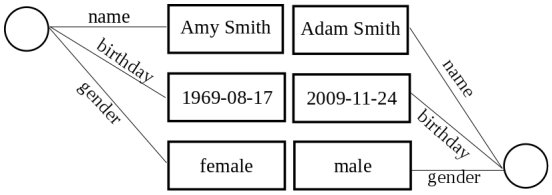

Figure 9: Attribute graph.

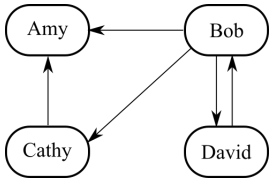

Figure 10: Network graph

has the same cost for all the inputs that were tested. Therefore we do not differentiate them in the rest of this section, and just refer to them as *k-security*.

We conduct our empirical assessment over two classes of graphs. One, which we call *network* graphs, are based on examples used in social network anonymization such as [31]. The other, which we call *attribute* graphs are information of US senators retrieved from [7]. The attribute graphs are real life examples of RDF data published online. It also closely resembles other online RDF data sources such as DBpedia in structure. Figure 9 is an example of attribute graphs, and Figure 10 of network graphs. In Figure 10 an edge denotes the *knows* relationship.

The reason we have chosen these two types of graphs for experiments is that they highlight two typical, yet distinct kinds of data that are encoded as RDF graphs. The *network* graphs highlights the relationship between vertices. All the vertices in a *network* graph represent the same type of entities (people in this example) and the graph describes relationship between the entities. The *attribute* graphs, on the other hand, describes attributes of entities. Some of the vertices in the graphs represent people while others represent attributes of these people such as name, gender, and birthday. The *attribute* graph does not contain relationship between entities of the same type. These two characteristics can be mixed of course in one graph. However in practical RDF graphs they are often separated, and from a theoretical standpoint it would be interesting to investigate how their differences would affect security.

Note that our intent is not to assess the performance of our reduction from the previous section. In keeping with this mindset, and to more easily understand how different parameters are related, we keep our inputs small: graphs of at most 25 vertices.

As shown in Figure 9, the attribute graphs have a lot of small components of similar structures. In fact our empirical result shows that this property of the attribute graphs makes it easier to achieve higher-level of security. As we can see from Figure 11, for the input attribute graph, achieving 3-security and 5-security has the same cost. In other words, when 3-security is achieved, 4-security and 5-security is also achieved. This is not a coincidence, but a result of the structure of the attribute graph.

The network graph, in contrast, does not possess this property. As is shown in Figure 12, the cost r increases steadily with the security parameter k. In this case, the cost only stops increasing after the entire graph is removed, in which case it can be seen as infinitely secure. This suggests that achieving the same level of security in an attribute graph is easier than in a network graph of similar size. The structural randomness of the network graph makes it harder to secure.

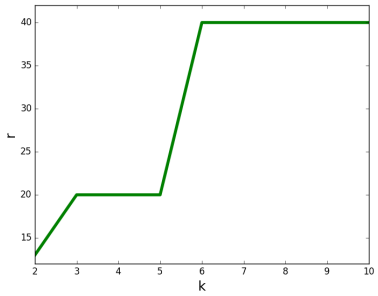

Figure 11: Relationship between k and r for attribute graphs

We also looked at the relationship between the size of the graph and the cost of security. Figure 15 and Figure 14 shows that when the network graphs becomes bigger (with more vertices and proportionally more edges), even if the number of vertices to be secured remains the same, it becomes more costly to achieve a certain level of security. Figure 15 shows the case where we fix the number of vertices to be anonymized and the security parameter k, and investigate the size of the graph (characterized by $|V|$) and the minimum cost r required to achieve k-security. The result shows that the cost r increases as the input graph becomes larger. Figure 14 supports the above observation from another perspective: when r and the number of vertices to be secured are fixed, the smaller a graph is, the easier it is to achieve a high level of security.

Similarly, the set A of vertices to be secured affects the cost of security positively. Figure 13 and Figure 16 together show the intuitive result that for the same graph, achieving a higher-level of security becomes more difficult when more vertices are added to A. Related to the this observation, Figure 17 demonstrates that to achieve the same level of security with the same cost, as the graph becomes larger, the set A needs to be smaller. We can see that at first the size of A is equal to the size of V because the fixed cost r is sufficient to reach k-security for all vertices in V. However as the size of graph grows, k-security can only be reached with cost r for a smaller set A.

6. RELATED WORK

There are several pieces of prior work on RDF security. Most such work [1, 9, 23] studies RDF from the perspective of database access control. Abel et al. [1] for example designed a system that enforces complex security policies by means of re-writing queries. Other works adopt a similar framework, and only propose different policy language

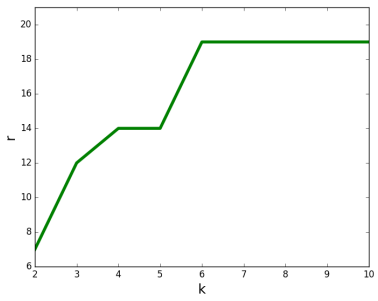

Figure 12: Relationship between k and r for network graphs

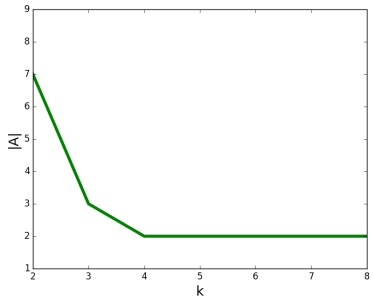

Figure 13: Relationship between k and $|A|$ for network graphs

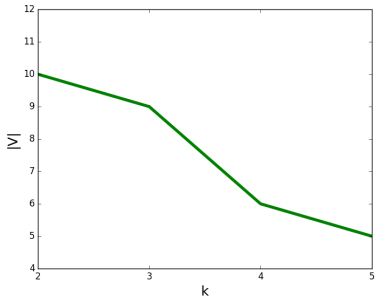

Figure 14: Relationship between k and $|V|$ for network graphs

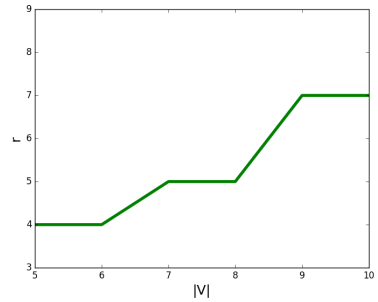

Figure 15: Relationship between $|V|$ and r for network graphs

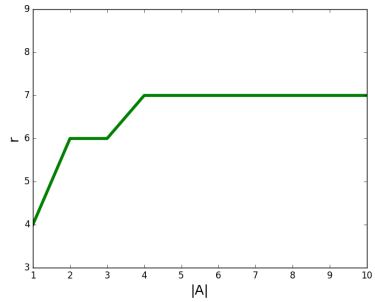

Figure 16: Relationship between $|A|$ and r for network graphs

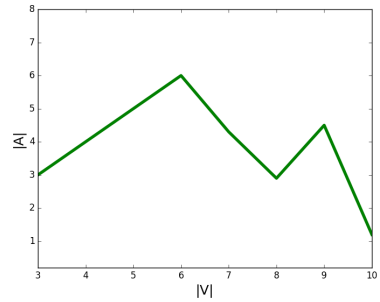

Figure 17: Relationship between $|V|$ and $|A|$ for network graphs

and enforcement mechanisms. A more recent piece of work on RDF access control by Rachapalli et al. [23] proposed three operations for sanitizing vertices, edges and paths in an RDF graph. Their work complements that of Bishop et al. [5], which discusses RDF sanitization and anonymization at a high level.

Some studies [24, 15, 4, 20, 25] deal with the case where RDF data is combined with inference rules to generate more triples. Jain and Farkas [15] for example attach security labels to each triple in RDF and compute labels automatically for inferred triples.

An issue with such work in access control is that they fail to provide a clear notion of security. We believe a clear notion of security is necessary in order to evaluate the proposed techniques: if an operation is designed to achieve security, it should be clear what security guarantee it provides.

Indeed, if we look at the work of Rachapalli et al. [23] in greater detail, the system can be compromised by issuing multiple queries and combining the results. At a high level, that work proposes "sanitizing" the graph that is the result of a query. Suppose we have the following RDF graph, which comprises two triples only.

Student_1 lives_in House_1

Student_2 lives_in House_2

Suppose that the policy states that the House data should be sanitized. An attacker first issues the following query, q_1, that selects everything.

```
SELECT ?a ?b ?c
WHERE {?a ?b ?c}
```

We expect that the response for q_1 is:

Student_1 lives_in XXX

Student_2 lives_in YYY

An attacker can then issue the following query, q_2. The intent of the "MINUS" part of the query is so all triples except those with House_1 as object are to be selected. It is possible to carry out this attack without using "MINUS."

```
SELECT ?a ?b ?c
WHERE {?a ?b ?c
    MINUS {sameTerm(?c, House_1)}}
```

We expect that the response to query q_2 is:

Student_2 lives in YYY

The attacker now immediately infers that Student_1 lives in House_1.

Another important aspect of our work study regards information-quality, which has also been largely ignored in prior work. We base this part of our work on existing work on RDF semantics and more specifically RDF entailment [12, 27, 2]. RDF entailment is originally a notion used to ensure the consistency between an inferred RDF graph and the original graph. In this work we use entailment creatively to ensure that the sanitized graph is consistent with the original with regards to semantics.

Separate from work in RDF security, many security notions have been proposed for traditional graphs under different attack models over the years. Most of these notions were

defined in the context of social networks, as it is a prominent use case for graphs. Their results however do apply to graphs in other contexts. The notions are mostly inspired by *k-anonymity* [26], that was defined for relational data. The definitions and techniques however differ in the context of graphs.

Liu and Terzi [17], for example, have proposed *k-degree-anonymity* in order to to prevent degree attacks. They assume that the attacker has prior knowledge of the degrees of some vertices in the graph. Addition and deletion of edges are used in this case to achieve security.

Hay et al. [11] have considered the *vertex refinement attack*, which is a generalization of the degree attack. That work proposes an anonymization approach utilizing random sampling, which retains integrity of summary data of the graph. Thompson and Yao [28] also define security against the degree attack, but in their work the attacker is assumed to know about the degrees of all neighbours within distance i of some vertices. Zheleva and Getoor [30], on the other hand, have considered security for social graphs whose edges are labelled but nodes are not. They used edge anonymization to prevent sensitive relationships from being revealed.

As Narayanan and Shmatikov [21] have demonstrated, it is possible to conduct re-identification attack to a graph with only structural information. Thereafter, more recent notions consider attacks on structural information.

Zhou and Pei [31] have defined the neighbourhood attack in which the adversary tries to re-identify vertices in a graph with prior knowledge of their complete neighbourhood structure. They propose a notion of *k-anonymity* by means of isomorphic neighbourhoods. The technique used to achieve security is inserting edges.

A more recent study by Wu et al. [29] proposes *k-symmetry*, which promises security against all structural attacks. The security notion proposed by [29] is very similar to that in [32], which is termed *k-automorphism*. These two notions are both based on the graph isomorphism problem, and the similarity was acknowledged by the authors. However the relationship between these two notions is unclear.

Imeson et al. [13] studies graph security in the context of integrated circuits. Their security notion is based on the problem of subgraph isomorphism. There, the attacker is explicitly provided two graph inputs, as opposed to only one (the secure graph) as is the case with us. In our work, the attacker's second input graph is some a priori knowledge that the attacker has. As we allow this second graph to be any subgraph of the original graph, it is as though the attacker chooses, and is not given, the second graph.

Other works such as Blocki et al. [6] study graph security from the perspective of differential privacy. Such pieces of work focus on providing summary statistics of the graph instead of the graph itself. The isomorphism-based security notions we discuss in this work focus on securing structural information and do not guarantee differential privacy.

7. CONCLUSION

In this work we make contributions to both theory and practice in the context of securing RDF. We observe that RDF data can be meaningfully encoded as a kind of graph that we call an RDF graph. We enunciate a rather strong threat model, and consider three notions for quantitative, unconditional, isomorphism-based approaches for graph-security from the literature. We pick the strongest two, and adapt those for RDF. Then, we show that under SPARQL semantics, entailment induces a lattice among RDF graphs under two operations that provide security, while preserving semantics.

The lattice naturally provides us a notion of information-quality, and equivalently, the cost of achieving security. We then consider the problem of computing a secure graph G' from a given RDF graph G such that G' yields some threshold k of security, and does not exceed some threshold r of cost. We show that the problem is **NP**-complete. We then discuss our design and implementation of a reduction to CNF-SAT. We have implemented our reduction and present empirical results for two classes of RDF graphs.

There is considerable scope for future work. One is to establish clear relationships between the various notions for graph-security in traditional graphs related, for example, to their relative strength, and computational complexity. A second topic for future work is consideration of large, realistic RDF databases, and attempting to sanitize them. A third topic is making the reduction to CNF-SAT more efficient so it is practical in realistic settings.

Acknowledgements

We thank the reviewers for their thoughful and constructive reviews. The first author was supported by a scholarship funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

8. REFERENCES

[1] F. Abel, J. L. De Coi, N. Henze, A. W. Koesling, D. Krause, and D. Olmedilla. Enabling advanced and context-dependent access control in rdf stores. In *The Semantic Web*, pages 1–14. Springer, 2007.

[2] M. Arenas, M. Consens, and A. Mallea. Revisiting blank nodes in rdf to avoid the semantic mismatch with sparql. In *RDF Next Steps Workshop*, pages 26–27, 2010.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[4] J. Bao, G. Slutzki, and V. Honavar. Privacy-preserving reasoning on the semanticweb. In *Web Intelligence, IEEE/WIC/ACM International Conference on*, pages 791–797. IEEE, 2007.

[5] M. Bishop, J. Cummins, S. Peisert, A. Singh, B. Bhumiratana, D. Agarwal, D. Frincke, and M. Hogarth. Relationships and data sanitization: a study in scarlet. In *Proceedings of the 2010 workshop on New security paradigms*, pages 151–164. ACM, 2010.

[6] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96. ACM, 2013.

[7] L. Civic Impulse. Govtrack. https://www.govtrack.us/. Accessed: 2016-07-01.

[8] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[9] G. Flouris, I. Fundulaki, M. Michou, and G. Antoniou. Controlling access to rdf graphs. In *Future Internet Symposium*, pages 107–117. Springer, 2010.

[10] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[11] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *Proceedings of the VLDB Endowment*, 1(1):102–114, 2008.

[12] P. Hayes and B. McBride. Rdf semantics, 2014.

[13] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara. Securing computer hardware using 3d integrated circuit (ic) technology and split manufacturing for obfuscation. In *USENIX Security*, volume 13, 2013.

[14] C. Inc. Stardog: Enterprise data unification with smart graphs. http://stardog.com/. Accessed: 2016-07-01.

[15] A. Jain and C. Farkas. Secure resource description framework: an access control model. In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 121–129. ACM, 2006.

[16] A. Jena. Apache jena. *jena. apache. org [Online]. Available: http://jena. apache. org [Accessed: Mar. 20, 2014]*, 2013.

[17] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 93–106. ACM, 2008.

[18] A. Lubiw. Some np-complete problems similar to graph isomorphism. *SIAM Journal on Computing*, 10(1):11–21, 1981.

[19] F. Manola, E. Miller, and B. McBride. Rdf 1.1 Primer, 2014. Available from https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/

[20] A. Mileo, N. Lopes, and S. Kirrane. A logic programming approach for access control over rdf. In *International Conference on Logic Programming (ICLP), Technical Communications*. International Conference on Logic Programming (ICLP), 2012.

[21] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187. IEEE, 2009.

[22] E. Prud Hommeaux, A. Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[23] J. Rachapalli, V. Khadilkar, M. Kantarcioglu, and B. Thuraisingham. Towards fine grained rdf access control. In *Proceedings of the 19th ACM symposium on Access control models and technologies*, pages 165–176. ACM, 2014.

[24] P. Reddivari, T. Finin, and A. 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.

[25] T. Sayah, E. Coquery, R. Thion, and M.-S. Hacid. Inference leakage detection for authorization policies over rdf data. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 346–361. Springer, 2015.

[26] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[27] H. J. ter Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):79–115, 2005.

[28] B. Thompson and D. Yao. The union-split algorithm and cluster-based anonymization of social networks. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 218–227. ACM, 2009.

[29] W. Wu, Y. Xiao, W. Wang, Z. He, and Z. Wang. K-symmetry model for identity anonymization in social networks. In *Proceedings of the 13th international conference on extending database technology*, pages 111–122. ACM, 2010.

[30] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *Privacy, security, and trust in KDD*, pages 153–171. Springer, 2008.

[31] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 506–515. IEEE, 2008.

[32] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.