A Framework for Policies over Provenance

Tyrone Cadenhead, Murat Kantarcioglu and Bhavani Thuraisingham
The University of Texas at Dallas
800 W. Campbell Road, Richardson, TX 75080
{*thc071000, muratk, bxt043000*}*@utdallas.edu*

Abstract

Provenance captures the history of a data item. This ensures the quality, the trustworthiness and the correctness of shared information, but the provenance may contain sensitive information so we may need to hide it. Sometimes we need access control policies to protect sensitive components and allow access based on certain properties. In other cases, we may need to share provenance but use redaction policies to circumvent the release of sensitive information. In this paper, we formulate an automatic procedure over provenance by combining these policies in an unified framework.

1 Introduction

Provenance is the lineage of a resource (or data item) and is essential for various domains including intelligence, healthcare, legal and industry. A provenance document contains both data items and their relationships [13, 4] formulated as a directed graph. An intermediate node on a path in this graph may contain sensitive information such as the identity of an agent who filed an intelligence report.

Traditionally, we protect documents with access control policies. These policies are used to determine who can access a document and under what conditions access is to be granted. In intelligence, it may be necessary to guard one's methods and sources; hence, an access control policy could limit access to the source of a report to sister agencies. We can also use these policies to permit access to a document if the document has certain properties, and thus we can specify integrity policies. An integrity policy could specify that the data in an intelligence report is valid if it was derived from field agents of a desired agency in a particular country. In other scenarios, however, the shared information may contain identifying or exclusive information; and therefore we need to apply redaction policies that transform the document in order to circumvent any identifying or sensitive information. The traditional access control policies mainly focused on single data items and not the relationships among the data items [4, 5], while the traditional redaction policies focused mainly on files and images [1, 8].

We have already addressed how to effectively apply access control policies over a provenance graph [5] and how to transform a provenance graph to satisfy a set of redaction polices [6]. We can use these approaches separately to apply access control and redaction policies over a provenance graph, but we cannot compare these two policy sets simultaneously or identify redundancies or the superiority of one policy over the other. Our contribution is to provide a unified framework, which extends the traditional policies over a provenance graph, thereby allowing domain users a choice of optimal and compact policies for both, protecting and sharing provenance information.

Section 2 presents our default languages for expressing policies and a corresponding graph framework for applying these policies. Section 3 reviews previous work. In closing, in Section 4 we provide our conclusions and future work.

2 Unified Framework

The problem of securing provenance is first complicated by the fact that provenance contains both data items and their relationships [4]; and secondly by the number of policies that ensure the safety of the released provenance information. Our approach is to provide intermediary policy languages that specify policies over a provenance graph. The idea is to translate these policies into graph operations over a provenance graph by making use of regular expression queries. Our framework can then be used for evaluating different policy sets over a provenance graph and their outcomes graphically. We can also compare the words described by regular expression queries to determine equivalence and subsumption

of policies. Hence, we can write more compact policies as well as eliminate redundancies and inefficiencies.

Our unified framework also presents an interface that accepts a high level policy, which is then translated into the required format for our graph rewriting system; therefore abstracting the details of the framework from a user. For the rest of this section, we will give a brief overview of our policy languages that evaluate over a provenance graph. Then we will describe our graph rewriting system which manipulate an original graph to one that meets the requirements of a set of user-defined high-level policies.

2.1 High Level policy Languages

Figure 1 and Figure 2 provide snippets of two high-level policy languages (see [5, 6] for details), which are sufficient for expressing both access control and redaction policies.

```
<policy ID="1" >
   <target>
    <subject>anyuser</subject>
     <record>Report3</record>
     <restriction>
        Report3 [WasGeneratedBy] process AND
        process [WasTriggeredBy]/country
     </restriction>
     <scope>non-transferable</scope>
  </target>
  <condition>purpose == research</condition>
  <effect>Permit</effect>
</policy>
```

Figure 1: Access Control Policy Language

The description of each element in Figure 1 is as follows: The **subject** element can be the name of a user or any collection of users, e.g. a journalist, or a special user collection *anyuser* which represents all users. The **record** element is the name of a resource. The **restriction** element is an (optional) element which refines the applicability established by the subject or record. The **scope** element is an (optional) element which is used to indicate whether the target applies only to the record or its entire ancestry. The **condition** element is an (optional) element that describes under what conditions access is to be given or denied to a user. The **effect** element indicates the policy author's intended consequence for a true evaluation of a policy.

The description of each element in Figure 2 is as follows:: The **lhs** element describes the left hand side of a rule. The **rhs** element describes the right hand side of a rule. Each path in the **lhs** and **rhs** begins at a starting entity. The **condition** element has two optional sub elements, the **application** defines the conditions that must hold for rule application to proceed, and the **attribute** element describes the annotations in *LHS*. Similarly, the **embedding** element has two optional sub elements, **pre** describes how *LHS* is connected to the provenance

```
<policy ID="2" >
  <lhs> start=Report3
        chain=[WasGeneratedBy] process AND
        .....
        process [Used] report AND
        report [WasGeneratedBy] process. </lhs>
  <rhs> start=Report3
        chain=[WasGeneratedBy] process AND
    process [WasTriggeredBy] _:A1. </rhs>
  <condition>
   <application>null</application>
   <attribute>null</attribute>
  </condition>
  <embedding>
    <pre>null</pre>
   <post>(ProcessJ,Used, Report3)</post>
  </embedding>
</policy>
```

Figure 2: Redaction Policy language

graph and the **post** describes how *RHS* is connected to the provenance graph.

The main advantages of these languages can be summarized as follows:

- XML-based and therefore inherit the features of being extensible and open.
- Support regular expression in the restriction tag (see Figure 1) and the lhs and rhs tags (see Figure 2).
- Specify the operations over a provenance graph by using the lhs, rhs and the embedding tags (see Figure 2).

2.2 Graph rewriting System

A Graph Rewriting System is a three tuple, (G_ℓ, P, q) where, G_ℓ is a labeled directed graph. P is a policy set and q is a request on G_ℓ that returns a subgraph G_q. For every policy $p = (r, e)$ in P, $r = (se, re)$ is a rule, where se is a starting entity and re is a regular expression string; and e is an embedding instruction.

Let $G_q \subseteq G_\ell$ be the result of a path query. A production rule is $r : L \rightarrow R$, where L is a subgraph of G_q and R is a graph. During a rule manipulation, L is replaced by R and we embed R into $G_q - L$. Embedding information, e, specifies how to connect R to $G_q - L$ and also gives special pre- and post-processing instructions. These instructions can be textual or graphical and are useful for specifying conditions to be satisfied in the graph rewriting process. A direct application of these instructions is to specify how R is glued to $G_q - L$ to ensure the final graph is both acyclic and the causal relationships between any two entities in the final graph existed in the original graph G_q. This condition is needed so that our graph rewriting system returns a valid provenance graph.

2.2.1 Graph Models

We apply two graph models: The first one is the Resource Description Framework [11], which is used as a representation and storage for provenance. The second is the Open Provenance Model [13], which specifies an abstract model for provenance.

Resource Description Framework
The Resource Description Framework (RDF) terminology $\mathscr{T}$ is the union of three pairwise disjoint infinite sets of terms: the set $\mathscr{U}$ of urirefs, the set $\mathscr{L}$ of literals (itself partitioned into two sets, the set $\mathscr{L}_p$ of plain literals and the set $\mathscr{L}_t$ of typed literals), and the set $\mathscr{B}$ of variables.

Definition 1 *(RDF Triple) A RDF triple* $(s,\ p,\ o)$ *is an element of* $(\mathscr{U} \cup \mathscr{B}) \times \mathscr{U} \times \mathscr{T}$.

A RDF graph is a finite set of RDF statements, i.e. subject-predicate-object triples; subjects and objects of triples are viewed as nodes, linked by predicates (predicates are usually called properties). A triple $(s,\ p,\ o)$ is depicted as an edge $s \xrightarrow{p} o$, that is, s and o are represented as nodes and p is represented as an edge label.

The Open Provenance Model
The Open Provenance Model (OPM) recognizes provenance as a directed acyclic graph (DAG) and identifies three entities, namely artifacts, processes and agents [13]. The OPM model also describes a set of abstract predicates, indicating causal relationships among the entities.

The nomenclature in [13] is used to define the nodes and edges in our provenance graph; therefore we can refer to a node as being an artifact, a process or an agent. We also restrict the set of RDF graphs to those that are acyclic in order to represent provenance as a RDF graph. We then use RDF to describe and represent the entities and relationships of a provenance graph. For example, with the abstract OPM predicate labels, we have the following RDF triples.

```
<opm:Process> <opm:WasControlledBy> <opm:Agent>
<opm:Process> <opm:Used> <opm:Artifact>
<opm:Artifact> <opm:WasDerivedFrom> <opm:Artifact>
<opm:Artifact> <opm:WasGeneratedBy> <opm:Process>
<opm:Process> <opm:WasTriggeredBy> <opm:Process>
```

Definition 2 *(Provenance Graph) Let* $H = (V,E)$ *be a RDF graph where* V *is a set of nodes with* $|V| = n$*, and* $E \subseteq (V \times V)$ *is a set of ordered pairs called edges. A provenance graph* $G = (V_G, E_G)$ *with n entities is defined as* $G \subseteq H$*,* $V_G = V$ *and* $E_G \subseteq E$ *such that G is a directed graph with no directed cycles.*

We speak of a valid OPM graph as one that is a provenance graph that conforms to the OPM nomenclature convention.

Evaluating a policy and locating a resource in a provenance graph are done by graph pattern matching. For example, the notion of integrity could be specified with a query for a constraint on a path in the provenance graph. These patterns depend on the notion of reachability, and therefore the norm is to locate a provenance subgraph with a path query. A path query is basically a query extended with regular expressions, where the edges in the query are used to match paths in a graph. To this end, we define our policies with a query language for RDF.

SPARQL
SPARQL Protocol and RDF Query Language (SPARQL) is a RDF query language and a World Wide Web Consortium (W3C) initiative that is based around graph pattern matching [15].

Definition 3 *(Graph pattern) a SPARQL graph pattern expression is defined recursively as follows:*

1. *A triple pattern is a graph pattern.*
2. *If P1 and P2 are graph patterns, then expressions (P1 AND P2), (P1 OPT P2), and (P1 UNION P2) are graph patterns.*
3. *If P is a graph pattern and R is a built-in SPARQL condition, then the expression (P FILTER R) is a graph pattern.*
4. *If P is a graph pattern, V a set of variables and* $X \in \mathscr{U} \cup V$ *then (X GRAPH P) is a graph pattern.*

Regular Expressions
A subset of $\mathscr{U}$, namely the labels of RDF predicates, describes the terms of an alphabet Σ. A language over Σ defines the subgraphs accepted by a SPARQL Query.

Definition 4 *(Regular Expressions) Let* Σ *be an alphabet of labels on RDF predicates, then the set* $RE(\Sigma)$ *of regular expressions is inductively defined by:*

- $\forall x \in \Sigma, x \in RE(\Sigma)$*;*
- $\Sigma \in RE(\Sigma)$*;*
- $\varepsilon \in RE(\Sigma)$*;*
- *If* $A \in RE(\Sigma)$ *and* $B \in RE(\Sigma)$ *then:* $A|B, A/B, A^*, A^+, A? \in RE(\Sigma)$.

The symbols | and / are interpreted as logical OR and composition respectively.

The path queries we consider are navigational and are evaluated relative to some designated source vertex. Given a symbol x in Σ, the answer to a path query q is the set of all nodes x' reachable from x by some path whose labels spell a word in q.

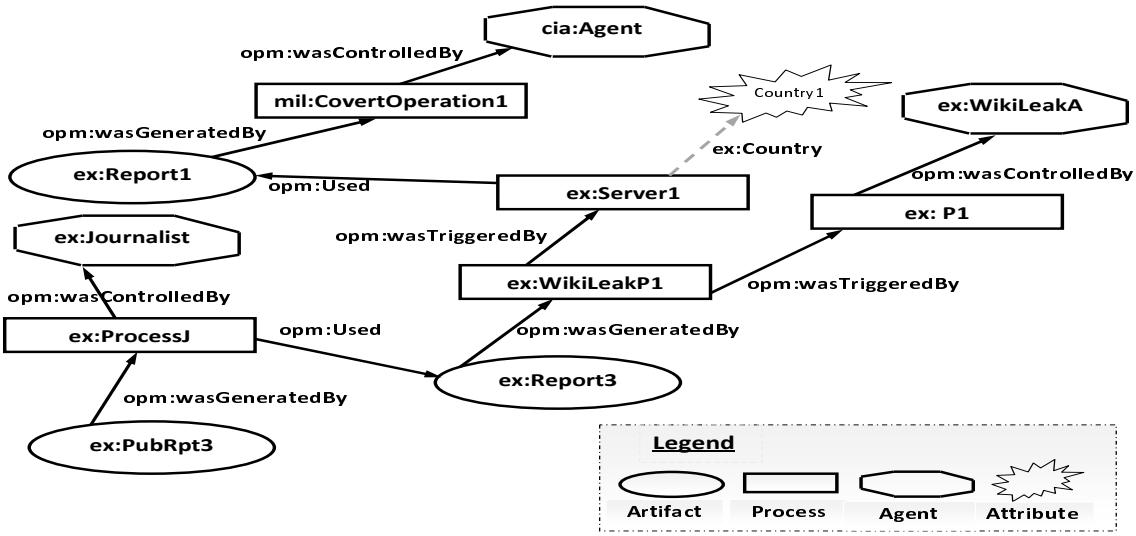


Figure 3: Provenance Graph

2.2.2 Use Case: Intelligence Example

Figure 3 shows an intelligence example as a provenance graph using a RDF representation that outlines a flow of a document through a server located in some country. This document was given to a journalist. The contents of this provenance graph could serve to evaluate the trustworthiness of the server from which the document originated. This provenance graph also shows the base skeleton of the actual provenance, which is usually annotated with RDF triples indicating contextual information, e.g. time and location. Note that the predicates are labeled with the OPM abstract predicate labels and that the final report can be traced back to a CIA agent.

We now use one of the new features that extends SPARQL with regular expressions [10] and an optimization technique from [3] to define a resource (or subgraph) of the provenance graph in Figure 3 as follows:

Example 1 ***(Integrity Query)***

```
Select ?x
{ ex:Report3 arq:OnPath("([opm:WasGeneratedBy]/
       [opm:WasTriggeredBy]/[ex:Country])" ?x). }
```

This query would return the country as a binding to the variable x and could be used to verify if ex:Report3 is in fact a high integrity report. A similar query could be used to identify a resource in the provenance graph that is protected by an access control policy.

We now show how to carry out redaction on a provenance graph as follows: Assume an agent provides the provenance of ex:Report3 to a journalist, but the information related to the CIA agent (cia:agent) must be redacted before the provenance is released. We illustrate this redaction in Figure 4, which also illustrates a rule manipulation over a provenance graph. The cloud in Figure 4 signals that some part of the provenance graph from Figure 3 is omitted.

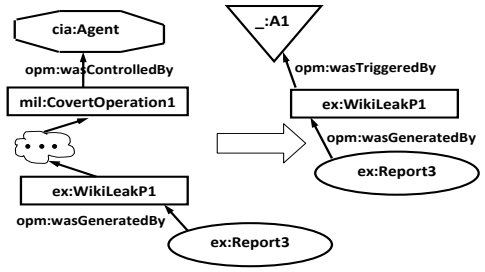
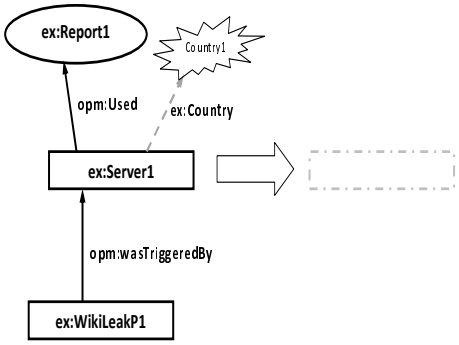


Figure 4: Redaction Policy

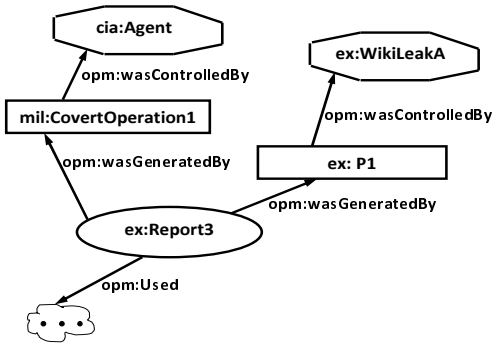
2.2.3 Embeddings and Valid Provenance Graphs

A graph rewriting system should be capable of specifying under what conditions a graph manipulation operation is valid. The embedding instructions normally contain a fair amount of information and are usually very flexible. Therefore, allowing the policy designer to specify the embeddings may become error-proned. The OPM nomenclature places a restriction on the set of admissible RDF graphs, which we call valid OPM graphs. These restrictions serve to control a graph transformation process (also a graph rewriting process) by ruling out transformations leading to non-admissible graphs.

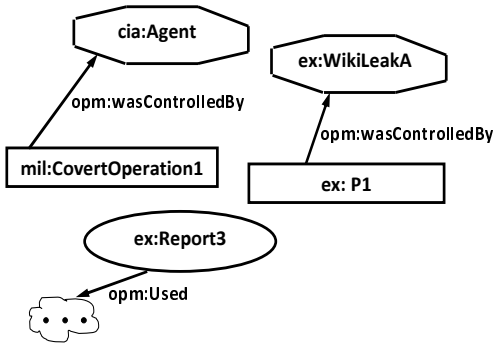
Let there be a rule in Figure 5(a) that replaces a one subgraph with a null (or empty) graph. Figures 5(b)-(d) show the effects of carrying out a graph transformation step using an embedding instruction. Figures 5(b) is the result of performing a transformation using the rule in



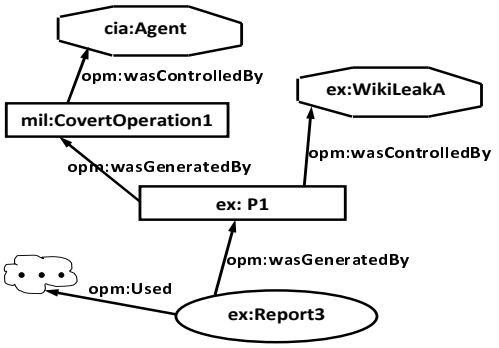
(a) Redaction Policy



(b) Redacted Graph 1



(c) Redacted Graph 2



(d) Redacted Graph 3

Figure 5: Graph Transformations

Figure 5(a) and the following embedding instruction:

```
<ex:Report3> <opm:WasGeneratedBy> <mil:CovertOperation1>
<ex:Report3> <opm:WasGeneratedBy> <ex:P1>
```

Figures 5(c) is the result of performing a transformation using the rule in Figure 5(a) but with an empty embedding instruction.

Figure 5(d) is the result of performing a transformation using the rule in Figure 5(a) and the following embedding instruction:

```
<ex:P1> <opm:WasGeneratedBy> <mil:CovertOperation1>
```

The only provenance graph of interest to us is the one in Figure 5(b). This is a valid OPM graph under the transformation of the rule in Figure 5(a). Figure 5(b) conforms to the OPM nomenclature convention, and each causal dependency in Figure 5(b) existed in Figure 3. Figure 5(c) is a valid OPM graph, but the causal relationships are not preserved, for example there is a causal relationship between ex:Report3 and cia:Agent in Figure 3, which is absent in Figure 5(c). Figure 5(d) is not a valid OPM graph since the RDF triple

```
<ex:P1> <opm:WasGeneratedBy> <mil:CovertOperation1>
```

does not conform to the OPM nomenclature convention. In addition there is no causal relationship between ex:P1 and mil:CovertOperation1 in Figure 3.

2.2.4 Discussion

Our solution is not limited to any particular representation of provenance, since we do not restrict the input provenance to be in any specific format. Instead the input provenance could be in any other format, for example XML, Relational or RDF. The causal relationships among the provenance entities and the graph operations over a provenance database are easily visualized using a data model, which supports the directed graph structure of provenance. In addition, existing tools can be used to convert other data format to RDF [2], thus making our unified framework flexible enough to support other data models for provenance. The architecture of our framework can be extended to take any high level description of provenance, while internally working with the RDF graph representation of provenance.

We are currently improving or unified framework with new functionalities to address some of the open issues with our graph rewriting system. One of the new functions takes as input a valid OPM graph, a production rule and a set of embedding instructions and return a valid OPM graph.

A new direction we are investigating is the optimization of our framework, which uses regular expressions for the queries that enforces our policies. Our goal is to

use the notion that if two automata accept the same language, then one of the languages may be redundant in our framework. This is based on an algorithmic translation of a finite regular expression over the RDF representation of a provenance graph to a finite state machine. Therefore, this direction will allow us to derive an optimized and compact set of policies. We can also compare policies for overlaps as well as identify conflicts and suitable resolutions.

3 Related Work

Graph transformation is already applied to access control [12, 7]; provenance and access control are also well studied [4, 14]. Our work combines these approaches. Our work is motivated by [4, 5, 13, 16] where the focus is on representing provenance as a directed graph structure. This contrasts some approaches, where the flow of information between the various sources and the causal relationships between entities are not immediately obvious. There are also previous works on the efficiency of a graph rewriting system [9, 3]. We utilize some of these techniques in our unified framework.

4 Conclusion

In this paper we propose a unified framework that allows a domain user a choice of policies for both protecting and sharing provenance information. Our work extends previous policy definitions to support provenance. We demonstrate the success of our framework by leveraging over a closely knit set of open technologies (RDF, SPARQL, OPM). We plan to pursue this avenue of research further with the emphasis on optimization and policy conflict resolution in the presence of large provenance graphs and large policy sets.

References

[1] Redact Privacy Information - Redact-It Software. Online at http://www.redact-it.com/.

[2] BIZER, C. D2R MAP-A database to RDF mapping language. *WWW (Posters)* (2003).

[3] BLOSTEIN, D., FAHMY, H., AND GRBAVEC, A. Issues in the practical use of graph rewriting. In *Graph Grammars and Their Application to Computer Science* (1996), Springer, pp. 38–55.

[4] BRAUN, U., SHINNAR, A., AND SELTZER, M. Securing provenance. In *Proceedings of the 3rd conference on Hot topics in security* (2008), USENIX Association, p. 4.

[5] CADENHEAD, T., KHADILKAR, V., KANTARCIOGLU, M., AND THURAISINGHAM, B. A language for provenance access control. In *Proceedings of the first ACM conference on Data and application security and privacy* (2011), ACM, pp. 133–144.

[6] CADENHEAD, T., KHADILKAR, V., KANTARCIOGLU, M., AND THURAISINGHAM, B. Transforming Provenance using Redaction. In *Proceedings of the Sixteenth ACM Symposium on Access Control Models and Technologies (SACMAT)* (2011), ACM.

[7] CORRADINI, A., HEINDEL, T., HERMANN, F., AND KÖNIG, B. Sesqui-pushout rewriting. *Graph Transformations* (2006), 30–45.

[8] COTTRILLE, S. Selective Document Redaction, Dec. 19 2007. US Patent App. 11/960,522.

[9] DÖRR, H. *Efficient graph rewriting and its implementation*. Springer, 1995.

[10] HARRIS, S., AND SEABORNE, A. SPARQL 1.1 Query Language. *W3C Working Draft* (2010).

[11] KLYNE, G., CARROLL, J., AND MCBRIDE, B. Resource description framework (RDF): Concepts and abstract syntax. *Changes* (2004).

[12] KOCH, M., MANCINI, L., AND PARISI-PRESICCE, F. Graph-based specification of access control policies. *Journal of Computer and System Sciences 71*, 1 (2005), 1–33.

[13] MOREAU, L., CLIFFORD, B., FREIRE, J., GIL, Y., GROTH, P., FUTRELLE, J., KWASNIKOWSKA, N., MILES, S., MISSIER, P., MYERS, J., ET AL. The Open Provenance Model—Core Specification (v1. 1). *Future Generation Computer Systems* (2009).

[14] NI, Q., XU, S., BERTINO, E., SANDHU, R., AND HAN, W. An access control language for a general provenance model. *Secure Data Management* (2009), 68–88.

[15] PRUD'HOMMEAUX, E., SEABORNE, A., ET AL. SPARQL query language for RDF. *W3C working draft 20* (2006).

[16] ZHAO, J. Open Provenance Model Vocabulary Specification. *Latest version: http://purl.org/net/opmv/ns-20100827* (2010).