On Data Provenance in Group-centric Secure Collaboration

Jaehong Park, Dang Nguyen and Ravi Sandhu
Institute for Cyber Security
University of Texas at San Antonio
jae.park@utsa.edu, dnguyen@cs.utsa.edu, ravi.sandhu@utsa.edu

***Abstract*—In this paper, we explore data provenance in a group-centric secure collaboration environment. In collaborations, participating organizations are likely to want certain trustworthiness on the data that are shared from other organizations and some assurance on how the shared data are used by users regardless of their organizations. By utilizing data provenance in group collaboration environment, we can provide the participating organizations with various provenance information that can establish trustworthiness and assurance on the shared data.**

To achieve this, we first identify what kind of operation information can be and should be captured as provenance data and how this information can be expressed in a formal representation which can be queried via the provenance system for certain utilities. We show the identified provenance data for a group collaboration application can provide some unique provenance utilities such as ability to trace the origins or usages of a shared data object even if it was created in a different organization. We utilize Open Provenance Model (OPM) [13] to capture various group collaboration operations identified in [12] and introduce a provenance system for a group collaboration environment that utilizes Resource Description Framework (RDF) data representations [10] and GLEEN-enabled SPARQL query language [7].

***Index Terms*—Provenance; Security; Collaboration; Group Collaboration; Information Sharing; Access Control;**

I. Introduction

In this paper we focus on data provenance in inter-organizational collaboration. More specifically, we utilize a group-centric collaboration environment where information is shared and created in a collaboration group. Here, the collaboration groups are controlled by participating organizations. The participating organizations assign members and add data resources to a group. The concept of "group-centric" sharing has been discussed in [11]. The main focus of [11] was in group operations such as user join/leave and object add/remove. A more recent study in [12] further discussed administrative and usage operations found in group collaboration and mainly focused on authorization issues of information sharing in collaboration groups that are created and administered by multiple organizations.

In such an environment, each participating organization shares its own data with other organizations in a collaboration group. Also they may use data that are shared by other organizations. This can cause certain concerns regarding where the data came from and who influenced the data to be in the current state. Therefore, in a group collaboration environment, it is in each organization's interest to be able to retrieve information about how and by whom a shared data object is created, modified or used in a group collaboration environment.

To achieve this, we first need to identify what kind of operation information can be and has to be captured and expressed. In this paper, we utilize the Open Provenance Model (OPM) [13] and Resource Description Framework (RDF) to properly capture and express both administrative operations and users' (group members') usage operations in a group-centric secure collaboration specified in [12]. Once captured, the provenance data can be utilized to compute some data dependencies if applicable. Such information can be retrieved by utilizing a query language. We introduce a provenance system for a group collaboration environment that utilize the GLEEN-enabled SPARQL query language to query provenance data stored in RDF triples. We further show some utilities of provenance by means of a group collaboration example.

The remainder of this paper is organized as follows. In section II, we discuss some characteristics and limitations of data provenance and summarize some basic aspects of OPM that are necessary to understand this paper. Section III discusses a group-centric collaboration environment and a data object versioning model that this paper is based on. In section IV we present provenance data and data object dependencies of various operations found in a group collaboration system using OPM notations. Section V includes a discussion on how the provenance data is expressed and queried in a provenance system, and then discusses some utilities of provenance by using a running example. Section VI discusses related work and section VII concludes the paper.

II. Data Provenance

In this section, we discuss some characteristics of data provenance and summarize OPM which we used to capture necessary information of operations that can be found in a group collaboration environment.

A. Some Characteristics of Data Provenance

In recent years, researchers have studied data provenance issues extensively in various computing and application environments. Generally speaking, many of these studies emphasize that data provenance can provide pedigree, usage tracking,

COLLABORATECOM 2011, October 15-18, Orlando, United States

DOI 10.4108/icst.collaboratecom.2011.247192

versioning capability, etc. While this could be true in theory, in a real world system, some of these utilities will be more critical than others. Fundamentally, the utilities of provenance largely depend on the kinds of provenance data that are captured in a system. We believe that capturing complete provenance data for all the operations occurred in a system is neither feasible nor necessary.

In a provenance system, while many computing operations and data dependencies can be captured by the system, there are certain data object (or node) dependencies that can be captured properly only by users' manual declaration. For example, if a user creates a new document from two existing documents, only the user herself can tell whether the newly created document is derived from any or all of the existing versions or not. While this could be done automatically by a system to a certain degree, for example, by comparing contents of these documents, there is no guarantee for the accuracy of the result since ultimately it is the user's intention that defines the dependency.

In addition, even with user's manual input, capturing a complete list of provenance data or data dependency is not likely to be possible in a system. This is largely because human memory is not capable of identifying all the source information of their ideas or creations. Consider an example where a researcher writes a scientific article with a list of citations. While the author may try as hard as possible to identify all the sources from where the ideas are derived, some ideas could be simply based on years of study and experience. Hence it is not likely to be possible to generate a complete list of data dependencies.

At the same time, capturing some information of the activities that occurred in a system may not provide any additional utility of provenance. For example, attribute update operations could be critical for authorization process, but capturing these operations in provenance data may not provide any additional utility. Also, we do not need to capture provenance data of all activities if they do not contribute in achieving particular goals of a provenance system. Depending on the goals of a provenance system, some activities are not necessary to be considered in the provenance system.

Having these constraints, it is our interest to identify the kinds of operations that can be and need to be captured as provenance data, how the captured provenance data can be used in a provenance system and what utilities of provenance can be achieved with the given provenance data. To properly discuss these issues, we need a specific computing application environment where a set of operations can be specified and expressed and some reasonably significant utilities of the provenance data can be identified.

B. Open Provenance Model

Recently, the OPM core specification v1.1 has been proposed by a group of researchers based on various requirements associated with the usage and employment of provenance in various application domains that are identified in a series of information provenance challenges [13]. The OPM provides a technology-agnostic definition of provenance.

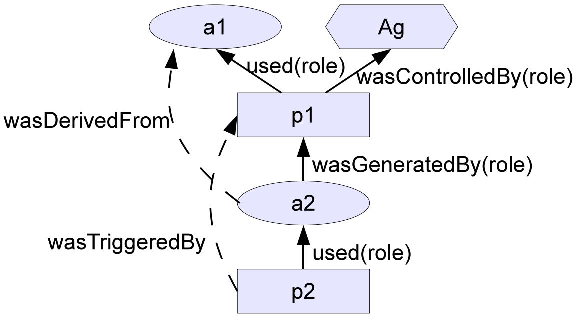


Fig. 1. A Sample Diagram of OPM Components

The main concern of OPM is to represent the execution process that led to a particular state of a data object. In essence, OPM aims to capture the causality dependencies of the computing operations, data objects, and execution context between any two of such states. In an associating OPM graphical representation, there are three main types of nodes: artifact which represents a state of a data object, process which represents an operation, and agent which represents an execution context. The direct causality dependency relationships between any pair of these nodes are captured by five different types of edges: $used(Role)$, $wasGeneratedBy(Role)$, $wasControlledBy(Role)$, $wasTriggeredBy$, and $wasDerivedFrom$, which altogether form a directed acyclic graph. Figure 1 demonstrates how the above nodes and edges interact in a generically captured use case. The three types of nodes are differentiated with different graphical representations: artifacts are represented by ellipses, processes by rectangles, and agents by octagons. The $used(Role)$, $wasGeneratedBy(Role)$ and $wasControlledBy(Role)$ edges are used to express the system-captured relationships between the nodes. They are represented by solid lines, differentiated by the annotations on the edges. Roles are used to give additional semantics to the associated edges. The $wasTriggeredBy$, and $wasDerivedFrom$ edges are represented by dashed lines. They are used to provide additional dataflow-oriented and process-oriented views of the provenance data. They may not be fully captured by the system and may require user's manual declaration in such cases. Figure 1 can be described as follows. The agent Ag controlled the process $p1$ which used the artifact $a1$ to generate the new artifact $a2$ which was then used by the process $p2$. Notice the direction of the arrows specifies a causality relationship instead of a data flow. The source of the arc represents the effect while the destination represents the cause. Also, the fact that $p1$ used $a1$ and generated $a2$ does not guarantee that $a2$ was derived from $a1$, hence that needs to be asserted with the $wasDerivedFrom$ edge from $a2$ to $a1$. The $wasTriggeredBy$ edge in Figure 1 shows the dependency of processes. We do not utilize this kind of edge in this paper.

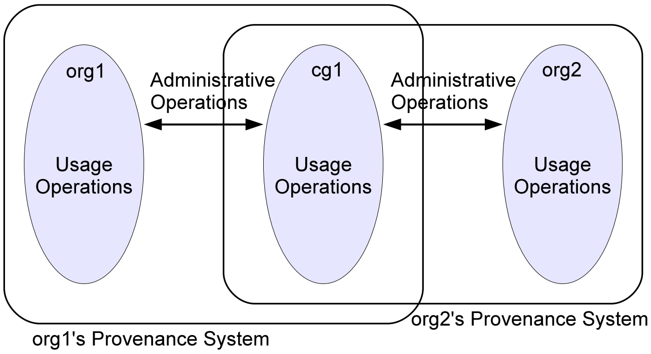


Fig. 2. A Conceptual View of Provenance Systems in A Group-centric Collaboration Environment

To distinguish nodes of the same type that are captured within the same graph, OPM assigns each of the nodes a unique identifier. For example, two instances of a process that perform the same operation are differentiated by their unique identities. The usage of assigned identities is also applied to other components of OPM where distinguishability is required under the same context.

OPM is also capable of describing multiple views of the same process at different levels of abstraction within the same graph. A specific abstraction view and its associated semantics are captured in an abstract form of a series of operations which are called “accounts” in OPM. The use of accounts to provide all ranges of description between abstract and detailed levels gives the users efficient utilizations of provenance data.

To capture the unique semantics of the operations within a particular application domain, OPM allows more detailed descriptions to be associated with the nodes and edges in the provenance graph. This is enabled through the annotation framework. The framework allows subtypes of edges to be defined and properties of nodes to be annotated. These subtypes of edges or node dependencies are defined in an OPM profile for a specific application domain.

III. A Group-centric Collaboration Environment and Object Versioning

A. Review of a Group-centric Secure Collaboration Framework

Collaboration comes in many different forms and sizes. To facilitate scenarios where a well-defined collaboration group exists, the concept of a Group-Centric sharing framework was recently introduced [12]. In this inter-organizational collaboration framework, the participating organizations collaborate through an agreed structure defined as a group. In a collaboration group, organizations share resources, which are termed objects. A version control system is applied on these shared objects. Users, who are granted access, can perform collaborative work on these objects. Organizations can create as many collaboration groups as necessary.

In [12], the administrative and operational aspects of the framework are addressed separately with two component submodels. The models are specified following the attribute-based UCON model for usage control [15]. The administrative sub-model is responsible for the management of groups as well as users and objects in the collaboration groups. The set of corresponding operations include: Establish/Disband for managing the group, Join/Disband/Substitute for managing users/admins, and Add/Remove/Export/Import/Merge for managing objects that are shared or natively created within the groups. The usage or operational sub-model, in contrast, is concerned with the management of users' activities within the collaboration groups as well as the respective organizations. The set of operations corresponding to these group-centric entities include: CreateRO/CreateRW/Kill for data flow control, Read/Update/Create for usage of objects/versions, and Suspend/Resume for controlling usage of objects/versions.

B. A Group Collaboration Environment for Data Provenance

In group-centric collaboration, in general there could be multiple organizations and these organizations could establish multiple collaboration groups for different purposes. For simplicity, here we assume that two participating organizations $org1$ and $org2$ have established one collaboration group $cg1$. As identified in [12], there are two types of operations. Administrative operations are performed to establish/disband groups together with group administrators, substitute group administrators, join/leave group members in a group, add/remove organization data to and from a group, etc. This means provenance data includes operations not only on shared data but also on groups and users. Usage operations are performed against data objects accessed via either an organization or a collaboration group. Also, there are two types of data objects in a collaboration group. Firstly there are pre-existing data objects shared by organizations in the collaboration group, and secondly there are data objects that are natively created in the collaboration group.

We also assume, as shown in Figure 2, that conceptually each organization facilitates its own provenance system which captures provenance data for usage operations against data objects managed within the organization and shared by the organization in a collaboration group as well as data objects that are natively created in the group. The provenance system also captures provenance data for administrative operations against the collaboration group, group members and data objects in the organization and in the collaboration group. [1] If mutually agreed, the participating organizations can query the other organization's provenance data using the overlapping provenance data as connectors.

[1] There can be several different ways to structure the overall provenance system in a group collaboration environment. For example, it is possible that $org2$ is only allowed to capture provenance data for its own user's operations or operations on their local data objects while $org1$ captures as discussed above. This could make sense, for example, in case a government organization collaborates with a contracting company where the contracting company's access to provenance data is restricted by the government. Another example could be that each participating organization and shared group maintains its own provenance system. In this case, the provenance data captured and maintained by a collaboration group may need to be accessible by the participating organizations even after the collaboration group is disbanded.

C. Data Object Versioning Model

In the paper, we use the terms objects, versions and copies. We assume that one object can have multiple versions, and each version can have multiple identical copies. The versions of an object form a rooted tree structure, relating a parent version to its immediate children versions. For provenance purposes each copy (identical in content) is considered as a separate "object." Each copy of a version of an object is shown as an artifact node in the OPM graph.

IV. Provenance Data for Group-centric Secure Collaboration

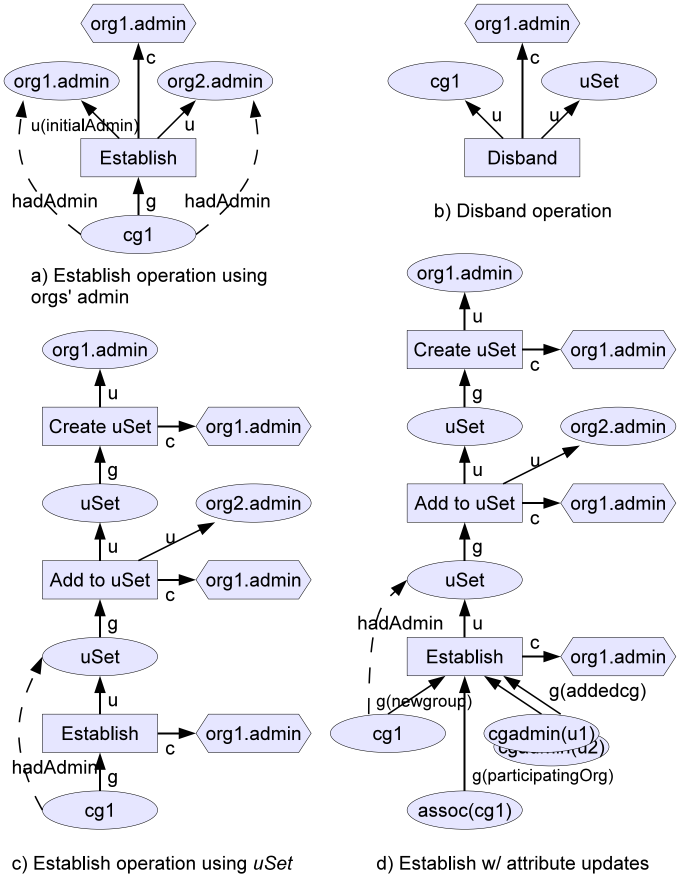
In order to discuss utility of provenance data, we first have to identify operations that can be performed on data objects and dependency of the data objects that are formed as a result of an operation or a set of operations. In this paper we utilize OPM notations to show these operations and data object dependencies.

As mentioned earlier, [12] identified various administrative operations on groups, administrators, regular users, and data objects as well as regular users' usage operations on data objects in a group. It is not necessary to capture all these operations in provenance data. Many of these operations are for authorization purpose and are not meaningful for provenance. Note that how much information of an operation can be or should be captured in provenance data depends on the participating organizations' agreement and provenance system design details. Hence, there could be many variations of the general theme of this section. Also, in the system of Figure 2 the provenance data captured in $org1$ could be different from those captured in $org2$ for the same operation. Further, there could be other operations (e.g., object duplication and deletion) or the existing operations could be refined to capture richer semantics (e.g., update operation can integrate some content of another data object into the updating object). Here we mainly focus on the operations identified in [12].

A. Provenance Data of Administrative Operations

In this subsection, we discuss how administrative operations identified in [12] can be expressed in OPM.

Establish$(uSet, cg)$: Establish collaboration group cg. In general, a collaboration group is established together with a set of administrative users who represent their own organizations. While there could be multiple ways to do this depending on how participating organizations agree, we assume that one of these administrative users establishes a collaboration group on behalf of other users.[2] Figure 3a) illustrates that the $establish$ process "$wasControlledBy$" (shown as an arrow labeled "c") $org1.admin$ and "$used$" (shown as arrows labeled "u") $org1.admin$ and $org2.admin$. The artifact $cg1$ "$wasGeneratedBy$" (shown as an arrow labeled "g") the $establish$ process. In other words, an administrator of an organization $org1.admin$ established a collaboration group $cg1$ together with two group administrators $org1.admin$ and $org2.admin$. Here, we have a subtype of $wasDerivedFrom$ (shown in dashed arrows) named as $hadAdmin$ to show more meaningful dependency of provenance data artifacts. The provenance data of the $establish$ operation can be also captured in a way discussed in [12]. This is shown in Figure 3c). Here, $org1.admin$ created a $uSet$, added a set of administrative users to the $uSet$ and then used it to establish a collaboration group $cg1$. In addition, [12] discussed that, as shown in Figure 3d), firstly $assoc$ attribute of cg1 was created/updated to include the participating organizations (all organizations found in $uSet$) and secondly participating users' $cgadmin$ attributes were updated to include $cg1$ as part of the groups they administer.



a) Establish operation using orgs' admin

b) Disband operation

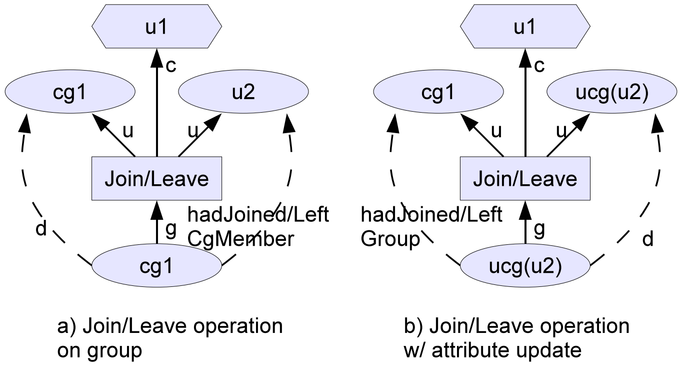
c) Establish operation using *uSet*

d) Establish w/ attribute updates

Fig. 3. OPM Diagrams for Establish/Disband Operations

While these additional updates on related attributes are discussed in [12], these activities may not need to be captured in provenance data. This is because capturing the "establish" operation as shown Figure 3a) might be enough to provide sufficient provenance utility. Capturing additional details of creation/update activities on attributes may not provide any additional significant provenance utility. Specifically, Figure 3a) will be enough to identify who created the group or who

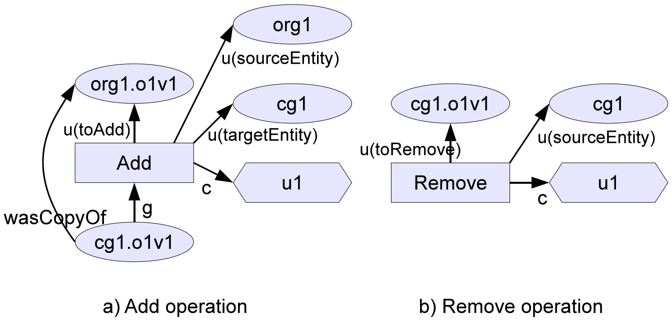
[2] We do not attempt to identify an exhaustive list of the possible scenarios for establishing a collaboration group. Rather we show a couple of possible ways how provenance data for a collaboration group establishment can be expressed using OPM and further discuss the captured provenance data. This also applies to the other operations discussed here.



a) Join/Leave operation on group

b) Join/Leave operation w/ attribute update

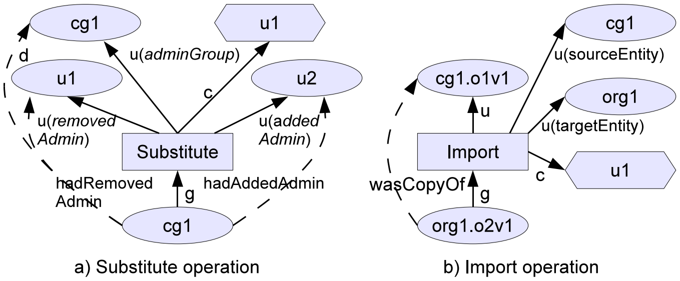
Fig. 4. OPM Diagrams for Join/Leave Operations



a) Add operation

b) Remove operation

Fig. 5. OPM Diagrams for Add/Remove Operations



a) Substitute operation

b) Import operation

Fig. 6. OPM Diagrams for Substitute/Import Operations

were the participating organizations or administrative users of the group. Capturing how $uSet$, $accoc$ or $cgadmin$ attributes were created/updated can be useful only if we need to verify some specific aspects such as who added a certain user in $uSet$, which administrative user is added first, etc.[3] At the same time, attribute updates shown in Figure 3c) and d) are just one way of conducting the details of the operation and can be subsumed in the approach shown in Figure 3a).

Disband$(uSet, cg)$: Disband group. The provenance data of this operation allows users to query who disbanded the collaboration group. While [12] requires agreement of all administrative users for this operation, provenance data only captures who actually conducted the operation and does not reflect the authorization processes. Figure 3b) shows an administrative user $org1.admin$ who disbanded collaboration group $cg1$ and a set of administrative users $uSet$. Capturing provenance data of the $establish$ and $disband$ operations allows users' to query pedigree and disposition of the collaboration group. This also means that the group is considered an OPM artifact.

Join/Leave$(u1, u2, cg)$: Join/Leave user to/from group.[4] Suppose an administrative user $u1$ from a participating organization included a user $u2$ as a member of collaboration group $cg1$. The provenance data of this operation can be expressed in OPM as shown in Figure 4a). Here, "$join/leave$" operation processes were controlled by $u1$ and used $u2$ and $cg1$, and a new $cg1$ was generated from the "$join/leave$" processes. In [12], a necessary update activity on the attribute ucg of $u2$ is captured to reflect the fact that the user is now a member of $cg1$ (see Figure 4b)). However, as similarly discussed in the $establish$ operation above, this update can be seen as one of multiple ways of performing the "$join/leave$" operations. For example, instead of using the ucg attribute of a user, we can utilize the ucg' attribute of $cg1$ to capture all the group members. Therefore, Figure 4b) can be subsumed in a more general operation description shown in Figure 4a). In the Figure 4a), two subtypes of "$wasDerivedFrom$" named "$hadJoinedCgMember$" and "$hadLeftCgMember$" were introduced to capture the dependencies of provenance data artifacts.

Add$(u, o, v, org, cg,)$: Add object version from org to group. The add operation creates a copy of an object version from an organization to a collaboration group. In Figure 5a), $u1$ added a copy of an object $org1.o1v1$ from an organization $org1$ to a group $cg1$. A subtype of "$wasDerivedFrom$" named "$wasCopyOf$" is identified to show a node dependency. Here, $org1$ is used as a source entity and $cg1$ is used as a target entity. While both source and target entities are captured here, if this provenance data is captured by $org1$, the source entity information may not need to be captured since it is always $org1$. However, if this provenance data is captured by organizations other than the source entity, say $org2$, the provenance data in $org2$ will need to include both the source and target entities information. While the source organization information could be found in source data object, we do not assume that this is always the case. Hence the source entity information is shown explicitly in the OPM diagram.

Remove(u, o, v, cg): Remove object version from group. The $remove$ operation deletes a copy of an object version from the entity where it is located. In Figure 5b), $u1$ removed $cg1.o1v1$ from $cg1$.

Substitute$(u1, u2, cg)$: Substitute group admin. The $substitute$ operation removes an existing administrative user and add another administrative user in a collaboration group. In Figure 6a), $u1$ substituted herself with $u2$ as an

[3] Further, note that provenance data can also capture various contextual information (e.g., timestamp, location, computing platform, etc.) to provide additional utility. In this paper, we do not consider such contextual information since it can be simply added to provenance data without worrying about data flow or node dependency.

[4] Although $join$ and $leave$ operations are shown in a single process in Figure 4 for convenience, they are two separate operations and occurrence of each operation should be captured by a separate process.

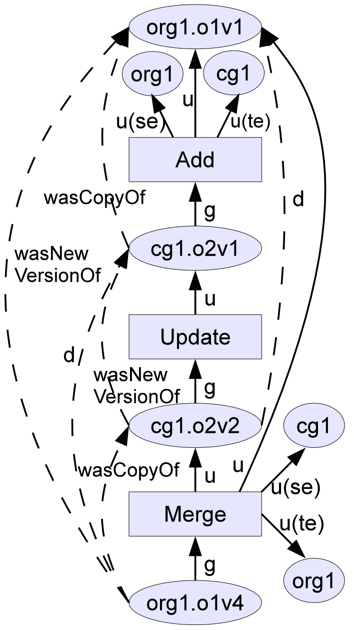


Fig. 7. OPM Diagrams for Merge Operation

administrative user in $cg1$. The roles of these $used$ edges are captured in $u(role)$ format. In Figure 6a), $cg1, u1$ and $u2$ are used with roles $adminGroup$, $removedAdmin$ and $addedAdmin$, respectively. Two subtypes of "$wasDerivedFrom$" named "$hadRemovedAdmin$" and "$hadAddedAdmin$" are identified to show the node dependencies. The OPM diagram also shows a generic "$wasDerivedFrom$" arrow to capture dependency between the previous and current state of $cg1$.

Import$(u, o1, v1, o2, cg, org)$: Import a version from group to organization. The $import$ operation copies a version of an object that was natively created in a collaboration group into an organization. In Figure 6b), an object version $cg1.o1v1$ was copied from $cg1$ to an organization $org1$ and named as $org1.o2v1$. While $org1.o2v1$ is an exact copy of $cg1.o1v1$, $org1.o2v1$ is treated as a new object. The "$wasCopyOf$" shown in Figure 6b) shows the dependency of the two data objects. While these two copies are considered different objects and cannot be connected in a version control system, using the dependency arrow $wasCopyOf$, users in a collaboration group or in an organization can trace the usage information of a particular object version that are imported to another organization even if the user does not belong to the organization. While [12] discusses "$export$" operation to capture the fact that all the administrative users should agree to make an object version exportable, this operation is identified for authorization purpose, hence not included in this paper.

Similar to the add operation, provenance data for the $import$ operation may or may not include the source entity information depending on whether the organization of the provenance system is the one who performed the operation or not.

Merge$(uSet, cg, o, v, org)$: Merge a version from group to organization. The $merge$ operation creates a newer object version of an existing object in an organization. This new version created in the organization is a copy of an object version that is created in a collaboration group as a result of the $update$ operation on a version of the object that is previously added from the organization to the group. (Additional details on the $update$ operation will be discussed in the next subsection.)

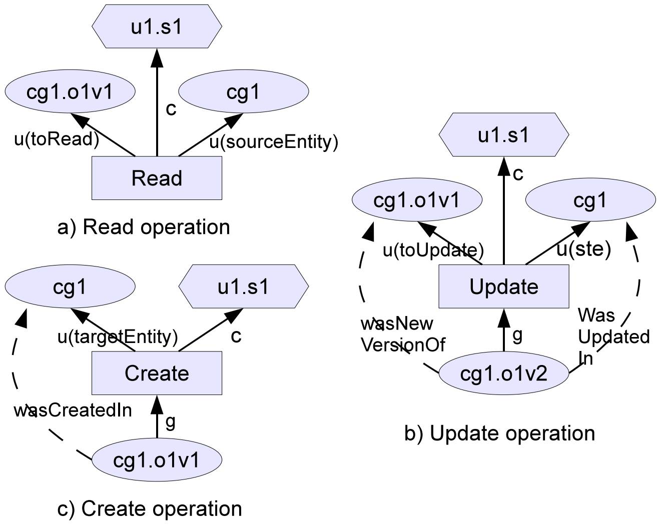
The $merge$ operation needs some precedent operations that should have occurred in advance. At least one add operation and then one $update$ operation on the added version are necessary to perform a $merge$ operation on the updated version. In Figure 7, $org1.o1v1$ was added to $cg1$ then the added version $cg1.o2v1$ was updated in $cg1.o2v2$ which then is merged back into the original organization $org1$ as a new version of $org1.o1v1$ shown as $org1.o1v4$. Here, the new version $org1.o1v4$ is an exact same copy of $cg1.o2v2$.[5] Figure 7 shows that in the $merge$ operation, $cg1$ was used as a source entity (shown as $u(se)$) and $org1$ was used as a target entity (shown as $u(te)$). Two subtypes of "$wasDerivedFrom$" named "$wasCopyOf$" and "$wasNewVersionOf$" are used to show the dependencies of object artifacts. Having the dependency of data objects allows users to trace information flow and usage history on the various versions of a particular object as well as copies of the versions. For simplicity, agent nodes for add, $update$ and $merge$ operation processes are omitted in Figure 7 though every process needs an agent.

B. Provenance Data of User's Usage Operations

In this section, we discuss provenance data of user's usage operations identified in [12]. [12] assumes that a user represents a human who creates a subject in a system and a subject exercises operations on behalf of the user. While user-subject distinction is critical for information flow control in group collaboration setting, provenance data only capture operation events that already occurred in a system and does not worry about how the operations are authorized. Therefore, though we use subjects as agents who controlled operations (as shown in Figure 8), this is not necessarily critical in this paper.

Read$(s, o, v, entity)$: Read an object version. The $read$ operation occurred on an object version by a subject in an entity. Entity information is captured if an object version is read in a collaboration group since there could be multiple groups in a single provenance system. Provenance data of $read$ operation against an imported or merged object in an organization is not likely to be captured by a provenance system of another organization. However it may need to be traced by another organization since the data object may have been

[5] Note that the $merge$ operation creates an exact copy of an object version in the collaboration group into an existing version tree in the organization where the original version in the collaboration group is $added$ from. This is different from merging two versions found in a same version tree of an object within an organization. While the latter could be useful, we do not consider this kind of "content merge" operation. For example, if $org1.o1v1$ was updated within $org1$ after it was added to $cg1$, $org1.o1v4$ is still a new version of $org1.o1v1$ but not a new version of the updated version of $org1.o1v1$.



a) Read operation

b) Update operation

c) Create operation

Fig. 8. OPM Diagrams for Read/Update/Create Operations

used or updated earlier by the tracing organization. For this, provenance data needs to include source entity information. This applies to both $update$ and $create$ operations discussed below.

Update$(s, o, v, entity)$: Update an object version. Similar to the $read$ operation, the $update$ operation occurred on an object version by a subject in an entity but creates a new version. In Figure 8b), $cg1.o1v1$ was updated and a new version $cg1.o1v2$ was created. In the diagram, two subtypes of "$wasDerivedFrom$" named $wasNewVersionOf$ and $wasUpdatedIn$ were identified to show the node dependencies. Note that, in the $merge$ operation diagram (Figure 7), the $entity$ node and $wasUpdatedIn$ arrow are not shown for simplicity.

Create$(s, o, entity)$: Create an object. The $create$ operation creates a data object in an entity. This is an initial version of the object. In Figure 8c), $cg1.o1v1$ was created in $cg1$ hence $cg1.o1v1$ has a $wasCreatedIn$ edge to $cg1$.

In addition to these three operations, [12] identified $createRO$ and $createRW$ operations as well as $kill$, $suspend$ and $resume$ operations. These operations are not discussed in this paper since they are identified mainly for authorization and information flow control purposes.

V. A Provenance System for Group-centric Secure Collaboration

In this section we describe how to express provenance data using Resource Description Framework (RDF) data representation [10] and show how the node dependencies and roles identified in the previous section can be stored in an OPM profile for group collaboration. We further discuss the query expression of the provenance system and then show three query examples to show some utilities of provenance data in group collaboration environment.

A. Provenance Data Expressions

We utilize the RDF data representation to express the provenance data for group collaboration since RDF supports a directed graph structure. The list that we identify in this section is not exhaustive. Specifically, we present the subtypes of $wasDerivedFrom$ and the roles of $used$ and $wasGeneratedBy$ in triples.

As discussed earlier, OPM identified five basic causal edges between three node types of artifact, process and agent. They are expressed in RDF representation as follows.

```
<opm:process><opm:used><opm:artifact>
<opm:artifact><opm:wasGeneratedBy><opm:process>
<opm:process><opm:wasControlledBy><opm:agent>
<opm:process><opm:wasTriggeredBy><opm:process>
<opm:artifact><opm:wasDerivedFrom><opm:artifact>
```

In addition to this, OPM introduced a notion of an OPM profile to specify a specialized OPM for different application domains. In the previous section, we identified several subtypes of "$wasDerivedFrom$" edges to add more semantics on the node dependencies so that more meaningful queries can be available to users. These subtypes for group collaboration provenance (gcp) are listed below. Using these triples, one can express the node dependencies identified in the previous section.

```
<gcp:artifact><gcp:wasCopyOf><gcp:artifact>
<gcp:artifact><gcp:wasNewVersionOf><gcp:artifact>
<gcp:artifact><gcp:HadAdmin><gcp:artifact>
<gcp:artifact><gcp:HadJoinedCgMember><gcp:artifact>
<gcp:artifact><gcp:HadLeftCgMember><gcp:artifact>
<gcp:artifact><gcp:HadRemovedAdmin><gcp:artifact>
<gcp:artifact><gcp:HadAddedAdmin><gcp:artifact>
<gcp:artifact><gcp:wasCreatedIn><gcp:artifact>
<gcp:artifact><gcp:wasUpdatedIn><gcp:artifact>
```

Provenance data forms a directed acyclic graph (DAG). Once stored as provenance data, these triples can be queried to construct a subset of the provenance graph that meets the querying criteria. For example one can find all the previous versions that show where an object version is coming from. As identified in the OPM standard [13], a role designates an artifact's or agent's function in a process so it can be differentiated among several use, generation, or controlling relations. The following is the list of triples that specifies various roles of $used$ (or u in short) and $wasGeneratedBy$ (or g in short) edges that can be found in a group collaboration environment.

```
<gcp:process><gcp:u(sourceEntity)><gcp:artifact>
<gcp:process><gcp:u(targetEntity)><gcp:artifact>
<gcp:process><gcp:u(adminGroup)><gcp:artifact>
<gcp:process><gcp:u(removedAdmin)><gcp:artifact>
<gcp:process><gcp:u(addedAdmin)><gcp:artifact>
<gcp:artifact><gcp:u(initialAdmin)><gcp:process>
<gcp:artifact><gcp:u(toJoin)><gcp:process>
<gcp:artifact><gcp:u(toLeave)><gcp:process>
<gcp:artifact><gcp:u(toAdd)><gcp:process>
<gcp:artifact><gcp:u(toRemove)><gcp:process>
<gcp:artifact><gcp:u(toImport)><gcp:process>
<gcp:artifact><gcp:u(toMergeTo)><gcp:process>
<gcp:artifact><gcp:u(toMergeFrom)><gcp:process>
<gcp:artifact><gcp:u(toRead)><gcp:process>
<gcp:artifact><gcp:u(toUpdate)><gcp:process>

<gcp:artifact><gcp:g(toEstablish)><gcp:process>
<gcp:artifact><gcp:g(toJoin)><gcp:process>
<gcp:artifact><gcp:g(toLeave)><gcp:process>
<gcp:artifact><gcp:g(toAdd)><gcp:process>
<gcp:artifact><gcp:g(toSubstitute)><gcp:process>
<gcp:artifact><gcp:g(toImport)><gcp:process>
<gcp:artifact><gcp:g(toMerge)><gcp:process>
<gcp:artifact><gcp:g(toCreate)><gcp:process>
<gcp:artifact><gcp:g(toUpdate)><gcp:process>
```

While the provenance data for the operations in a group-centric collaboration can be expressed in the forms of the above listed triples, not all operation information is expressed in the above list. It is not our goal to identify a complete list of these triples. Rather we show what kind of roles can be identified for those operations discussed in the previous section so they can be used to specify query statements. Also note that some of these roles are not specified in the OPM diagram for simplicity.

B. Query Expressions

Having the above subtype and role triples defined in an OPM profile and having provenance data which are stored in triples as described in the OPM profile, one can utilize SPARQL [16] (a standard query language for RDF) to query provenance information by stating a consecutive path of specific triple types of subject, predicate, and object in WHERE clause. For example, consider the following SPARQL query statement.

```
SELECT ?ver
WHERE{
    gcp:cg1_o2v2 gcp:wasCopyOf ?obj.
    ?obj gcp:wasNewVersionOf ?ver.}
```

Here, using the $gcp : wasCopyOf$ predicate, the query finds the original object of $cg1\_o2v2$ in a source entity. Then it finds the previous version of the original object in the source entity. The "?" symbol in front of a character sting denotes a variable.

However, this approach is not ideal to express recursive path patterns. For example, in the above query statement, if we want to specify the query so it can capture all the previous versions, it is not practical to list all the possible combinations in the query statement. To address issue, [7] developed the GLEEN path expression library as a plugin for the ARQ query engine. ARQ is a query engine for Jena, a semantic web framework for Java which supports the SPARQL RDF query language [1]. The ARQ engine provides a property function in which a custom triple matching function can be used in the predicate position instead of using a uniquely identified ontology property that is found in standard SPARQL. In this paper, we utilize the SPARQL query language together with the GLEEN OnPath property function to express regular expression-based path patterns in a query.[6] This kind of path patterns are necessary to build a query template for certain types of user inquiries so the templates can be used to generate actual query statements on the fly whenever a user's inquiry is placed. The syntax of GLEEN OnPath property function is structured as follows:

```
subject gleen:OnPath (pathExpression object)
```

Here, the subject and object may be either the URI (i.e gcp:o1v1) or a variable (i.e. ?agent). The path expression is a collection of representations of "opm" and "gcp" edges. GLEEN supports the following regular expression: operators '?' (zero or one), '*' (zero or more), '+' (one or more), '|' (alternation), and '/' (concatenation).

Using regular expression based path patterns, this GLEEN query expression can be used to create all the direct and indirect, further abstracted node dependencies at the time operation information is captured as provenance data. For example, when a merge operation occurred, a triple of direct node dependency that includes $wasCopyOf$ can be identified using a standard SPARQL query. At the same time, all the indirect (or further abstracted) node dependencies (also called accounts in OPM) from the target object node can be identified by utilizing the GLEEN OnPath function with the regular expression based path patterns that are mapped to the subtypes of $wasDerivedFrom$. Having these abstracted OPM accounts identified, user inquiries on provenance information can rely on the standard SPARQL query language without using any regular expression based path pattern in the query statement.

Another way to support provenance information retrieval is that node dependencies of a target object can be computed on the fly at the time of a user inquiry on provenance information. This approach does not require building all the identifiable direct and indirect node dependencies. However the node dependencies of a target object need to be computed which can be done using the GLEEN OnPath function together with

[6] A similar approach is also used in [5].

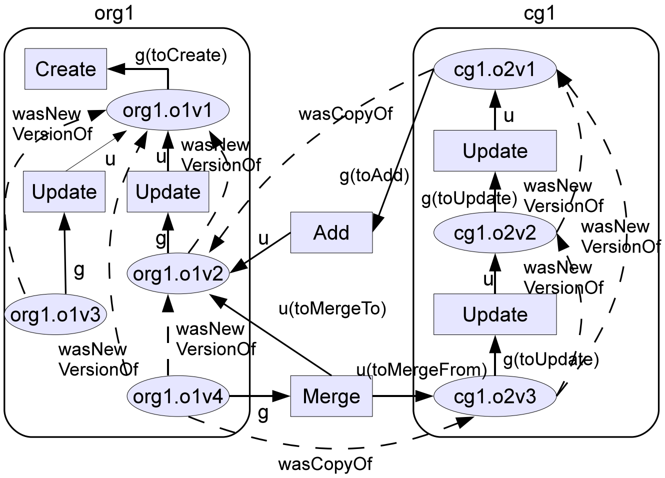


Fig. 9. An OPM Diagram for Provenance Data example

regular expression based path patterns in the query statement.

Here, if inquiries on provenance information occurred more frequently than the conducting operations that need to be captured as provenance data, the first approach could be more efficient than the second approach.

C. Query Examples

In this subsection, we utilize a provenance graph shown in Figure 9 as a running example. The example shows operations occurred in both $org1$ and $cg1$. The OPM diagram for the running example does not show the agent nodes and edges from them for simplicity. We assume that a query can access the provenance data of both $org1$ and $cg1$.

Using the running example, we create three provenance data retrieval cases to show how queries can be constructed and what kind of node dependencies and roles in the OPM profile are used for the queries. These provenance data retrievals will show some possible utilities of provenance data in group collaboration system environment. In particular, the provenance data of operations occurred in multiple entities can be accessed to support an organization's traceability of shared object that are located in different entities. For example, while versioning systems cannot relate objects in different entities, by using provenance data this can be easily achieved. Note that it is not our goal to identify all the possible utilities of provenance system in a group collaboration environment.

```
<Example 1>

SELECT ?obj ?proc
WHERE{
   gcp:cg1.o2v3 gleen:OnPath(
        ”[gcp:wasNewVersionOf]∗” ?obj).
   ?obj [gcp:g(toCreate)]|
             [gcp:g(toAdd)] ?proc.}
```

Example 1. This query is to identify the very initial version of an object version and whether it is created in the current group or added from an organization. This is done by first finding itself and all the previous versions ($cg1.o2v3$, $cg1.o2v2$, $cg1.o2v1$) and then identifying one that is either created or added. This will return $cg1.o2v1$ since it was derived from an add process.

```
<Example 2>

SELECT ?agent
WHERE{
   gcp:org1.o1v2 gleen:OnPath(
        ”[gcp:wasNewVersionOf]∗” ?obj).
   ?obj [gcp:g(toUpdate)]|
        [gcp:g(toCreate)] ?proc.
   ?proc [gcp:wasControlledBy] ?agent.}
```

Example 2. This query is to find out all the users who influenced a current object version within an entity. This may not identify all the versions of the same object. This is because if a different version ($org1.o1v3$) is created from a previous version ($org1.o1v1$) of the target object version ($org1.o1v2$), this query statement will only return $org1.o1v2$ and $org1.o1v1$ but not $org1.o1v3$. We think this is acceptable since $org1.o1v3$ did not influence $org1.o1v2$.

```
<Example 3>

SELECT ?agent
WHERE{
   gcp:org1_o1v4 gleen:OnPath(
        ”([gcp:wasNewVersionOf]|
             [gcp:wasCopyOf])∗” ?obj).
   ?obj [gcp:g(toUpdate)]|
        [gcp:g(toCreate)] ?proc.
   ?proc [gcp:wasControlledBy] ?agent.}
```

Example 3. This query is to verify users who may have influenced (including update and create) an object content regardless of the fact that whether the influence is done on a version of the same object or a version of a copied object of the object. This is particularly useful since the query finds all the users even if they influenced the object in a different entity.

VI. Related Work

Data provenance can be utilized in many different ways depending on the characteristic of a particular application domain within which the provenance is captured. For instance, identifying the source of a piece of data and its connection to other pieces in curated database [3] and the usage of provenance for reproducibility of workflow in Semantic Web [8] is a significant provenance utility in such domains. The group-centric secure collaboration we study in this paper is different from previously studied domains. Within this collaboration domain, we describe a provenance system that is able to capture and express data provenance that can be beneficially utilized.

The design for capturing provenance in the form of causal dependencies allows OPM to be employed in various different systems. Previous proposals for provenance models in [2], [3], [6], [9] are constrained to specific domains with different forms and purposes. [14] proposes a XACML-based access control policy language for data provenance requirements. [5] extend the language in [14] but instead choose OPM for the capability to use RDF and SPARQL with GLEEN-API for the data provenance of medical records. In this paper, we adapt OPM and RDF representation and SPARQL with GLEEN-API queries to demonstrate certain provenance utilities in group collaboration environment.

VII. Conclusion

In this paper, we discussed what kind of operations can be and need to be captured as provenance data in group collaboration environment. We then showed how we can express such provenance data in RDF triples so it can be retrieved by utilizing a regular expression based path patterns in the GLEEN-enabled SPARQL query language. We further showed some utilities of data provenance in a group collaboration environment using a sample example. While this is our initial step toward data provenance security and utility in group-centric collaboration, we believe this paper captured the necessary provenance data and querying mechanisms to support some utilities for the secure group collaboration. We anticipate further enhancements on this line of work for more secure and trustworthy group collaboration.

Acknowledgment

This work is partially supported by NSF grant CNS-1111925, AFOSR MURI and a research superiority grant from the State of Texas.

References

[1] ARQ - A SPARQL Processor for Jena, Available at: http://jena.sourceforge.net/ARQ/

[2] Benjelloun, O., Sarma, A.D., Halevy, A.Y., Theobald, M., Widom, J.: Databases with uncertainty and lineage. VLDB J. 17(2), 243-264 (2008).

[3] Buneman, P., Chapman, A., Cheney, J.: Provenance management in curated databases. In: SIGMOD 2006, pp. 539-550 (2006).

[4] J.J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, Jena: implementing the semantic web recommendations Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, pp. 74-83, ACM, 2004.

[5] T Cadenhead, V Khadilkar, Murat Kantarcioglu, Bhavani Thuraisingham, ”A language for Provenance Access Control” Proceedings of the CODASPY11, February 21-23, 2011.

[6] Chapman, A., Jagadish, H.V., Ramanan, P.: Efficient provenance storage. Proceedings of the ACM SIGMOD International Conference on Management of Data. In: Wang, J.T.L. (ed.) SIGMOD 2008, SIGMOD Conference, Vancouver, BC, Canada, June 10-12, ACM, New York (2008).

[7] L.t. Detwiler, D. Suciu,and J.F. Brinkley, Regular paths in SparQL: querying the NCI thesaurus AMIA Annual Symposium Proceedings, Vol. 2008, pp. 161, American Medical Informatics Association, 2008.

[8] Jennifer Golbeck. 2007. A Semantic Web and Trust Approach to the Provenance Challenge. Concurrency and Computation: Practice and Experience

[9] Heinis, T., Alonso, G.: Efficient lineage tracking for scientific workflows. Proceedings of the ACM SIGMOD International Conference on Management of Data. In: Wang, J.T.L. (ed.) SIGMOD 2008, SIGMOD Conference, Vancouver, BC, Canada, June 10-12, ACM, New York (2008)

[10] Resource description framework (RDF): Concepts and abstract syntax, Available at: http://www.w3.org/TR/rdf-concepts/, 2004.

[11] R. Krishnan, R. S. Sandhu, J. Niu, and W. H. Winsborough. Foundations for group-centric secure information sharing models. In ACM SACMAT, pages 115-124, 2009.

[12] R. Krishnan, R. S. Sandhu, J. Niu, and W. H. Winsborough. Towards a framework for group-centric secure collaboration. In IEEE Collaborate-Com, pages 1-10, 2009.

[13] L. Moreau, B. Clifford, J. Freire Y. Gil, P. Groth, J. Futrelle, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, and others, The Open Provenance Model Core Specification (v1.1) Future Generation Computer Systems, Elsevier, 2009.

[14] Qun Ni, Shouhuai Xu, Elisa Bertino, Ravi Sandhu, and Weili Han. An Access Control Language for a General Provenance Model. In the Proceedings of the 6th VLDB Workshop on Secure Data Management (SDM’09), August 28, 2009, Lyon, France.

[15] J. Park and R. Sandhu. The UCON_{ABC} Usage Control Model. ACM TISSEC, Volume 7, Number 1, February 2004, pages 128-174.

[16] SPARQL Query Language for RDF. Available at: http://www.w3.org/TR/rdf-sparql-query/, 2008.