Towards a Framework for Group-Centric Secure Collaboration

Ram Krishnan*, Ravi Sandhu†, Jianwei Niu† and William Winsborough†
*George Mason University, Email: rkrishna@gmu.edu
†Institute for Cyber Security, University of Texas at San Antonio,
Email: ravi.sandhu@utsa.edu, niu@cs.utsa.edu, wwinsborough@acm.org

***Abstract*—The concept of groups is a natural aspect of most collaboration scenarios. Group-Centric Secure Information Sharing models (g-SIS) have been recently proposed in which users and objects are brought together to promote sharing and collaboration. Users may join, leave and re-join and objects may be added, removed and re-added. Furthermore, objects embodying new intellectual property may be created in the group during collaboration, some of which may flow back to the participating entities. Authorizations may depend on various aspects including time of join and add, the user's role, etc. In this paper, we outline three example scenarios of inter-organizational collaboration. We develop a complete authorization model for one of these scenarios comprising administrative and operational components. We conclude the paper by proposing an initial framework for developing more sophisticated models for inter-organizational collaboration.**

I. Introduction

Collaboration, in general, can be of many different types and at different scales. In most inter-organizational, a group is established between participating organizations to promote sharing and collaboration. Each organization, a stake holder, may share its resources (e.g. users and objects) with others by adding them to the common group. As collaboration proceeds, objects may be created (e.g. new intellectual property) which may be transferred back to the stake holders. Such collaboration scenarios have many characteristics: small-scale vs large-scale, short-term vs long-term, dynamic vs static membership, equal vs unequal stakeholders, unilateral vs bi/multi-lateral administration, uniform vs differentiated permissions, etc. Joint product design between two or more organizations, merger and acquisition scenarios, program committee meetings, micro-collaboration such as in a research paper where multiple authors collaborate, unilateral collaboration between a product team in an organization and temporary consultants/contractors, interest groups such as in IETF where members participate in writing RFCs, etc. are few examples of collaboration scenarios with such characteristics. On the other hand, we have massively large-scale, highly distributed collaboration scenarios where users may not represent a specific organization and the notion of group tends to be fuzzy. Examples of such scenarios include Wikipedia, Crowd-Sourcing, etc.

In this paper, we focus on inter-organizational and similar collaboration scenarios where a clear notion of a collaboration group exists. The concept of "Group-Centric" sharing has already been introduced in [6] where the primary focus was on the semantics of group operations (user join and leave and object add and remove). In a group, users may join, leave and re-join and objects may be added, removed and re-added. Authorization may depend on the temporal relationship between the user and the object in the group with respect to say the time the user joined and the time the object was added. For instance, a user may be allowed to access all objects in the group regardless of when he/she joined (or when the object was added) or only those that were added after he/she joined the group. Several alternative operation semantics that are useful in group-centric information sharing were proposed in [6]. However, the administrative aspects such as who authorizes such group operations were not addressed for this specific application. More generally, we expect that the administrative model will always be strongly influenced by the application context for group-centric sharing. Since our focus is on inter-organizational collaboration, we strongly believe that it is important to specify a concrete administrative model.

Our contribution in this paper is two-fold. First, we formally specify a complete and novel model for group-centric collaboration in inter-organizational scenarios. The specification comprises of an administrative model and an operational model. In the administrative model, we specify the authorizations for an essentially complete set of administrative operations such as establishing and disbanding a collaboration group, joining (and leaving) users to the group, adding (and removing) objects and managing the results of the collaboration (such as exporting/importing newly created objects in the group), etc. In the operational model, we first propose a user-subject model. The user represents a human in the system who may create subjects—processes/programs, to exercise his/her privileges in the system. A user in an organization who belongs to a collaboration group may create subjects to read and write objects in both the organization and the group. Clearly, it is important to control information flow so that objects or their content cannot be arbitrarily transferred to unauthorized entities by malicious subjects. In the operational model, we specify authorizations for normal user operations such as creating and killing a subject, reading and updating an object, etc. We believe that these relatively simple models can serve the needs of many real-world collaborations. However, clearly, there will also be a need for more sophisticated models in other situations. To this this end, our second contribution is to introduce a framework for developing more sophisticated models

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8305
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8305

of group-centric collaboration. For example, in such models, attributes such as the user's role may influence authorizations in a group. The framework addresses issues involved in three phases of collaboration: Begin Collaboration phase (where the group is set up), Collaboration phase (where users and objects are shared and new objects may be created) and End Collaboration phase (where the results are shared between the stake holders). In this paper, we solely focus on policy models for inter-organization collaboration, thereby clearly separating enforcement (e.g. security architecture and protocols) and implementation (e.g. underlying technology, operating system and software components) issues as illustrated in [10].

The remainder of this paper is organized as follows. In section II we discuss three distinct inter-organizational collaboration scenarios. We formally specify a complete model for one of the scenarios in section III. In section IV, we present a framework for developing group-centric collaboration models. Finally, we discuss related work and conclude in section V.

II. Collaboration Scenarios

We discuss three distinct collaboration scenarios below. Later we specify a complete formal policy model for one of them. We then generalize these in to a unified framework for developing policy models for group-centric collaboration.

a) Scenario #1 (Bilateral Organizational Collaboration): Consider two organizations A and B that want to collaborate for some common purpose such as developing Intellectual Property (IP). We refer to A and B as Source Organizations (SO). A Collaboration Group (CG) is established and each organization shares its resources (users, objects, etc.) by adding them to the group. Thus the objective of the collaboration is to share assets, develop IP and take-away identical set of results. All users in CG are treated equally in terms of the privileges that they can exercise in the group. While user and object membership change is expected, it is not expected to be frequent. User authentication to CG is performed through an inter-organizational federated identity system. That is, there is no local id for CG and users use their SO id to participate in CG. Further, we also assume federated CG administration. That is, the respective SO (A and B) is responsible for adding and removing its users and objects to CG. For example, if A decides to remove its user from CG, B is not required to participate in that administrative process. However, a collective CG admin is established for controlling objects such as IP that may be created in CG. CG admins are representatives from A and B. CG admins decide what information from CG flows back to participating SOs. In this case, during and at the end of collaboration, CG admins publish intermediate results and the final IP back to both A and B.

b) Scenario #2 (Unilateral Organizational Collaboration): Consider organizations A and B where A is a defense services organization and B is a consulting company. This is similar to scenario #1, except A is the ultimate stakeholder and sets of consultants from B are admitted to CG periodically. Org A users in CG are long-term members, whereas different sets of B consultants are brought into CG at different periods of

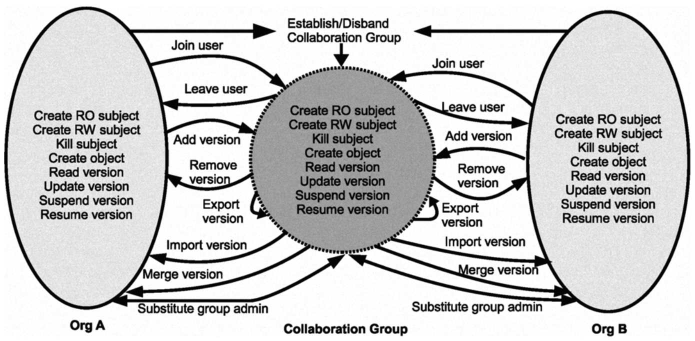


Fig. 1. Each entry gives the operation name followed by its principal target (specified in further detail in the subsequent tables). Operations labeled over arcs belong to the administrative model and those within the group or organizations (all identical) belong to the operational model.

time for a specific purpose. Thus, B users are authorized to access objects added only during their period of membership but A users may access all objects in CG. Users in CG may be substituted with another user temporarily from corresponding SO. Substituted users inherit all permissions of the outgoing user and no further substitution is allowed. Authentication to CG is performed through a federated id between A and B. However, A controls membership to CG. That is A may decide to add and remove users and objects to CG at its discretion. At the end of collaboration, A takes away the results. B is compensated for its service.

c) Scenario #3 (Qualified Open Collaboration): Consider an interest group such as in IETF where many users jointly work on an RFC. Here users share, collaborate and finally publish or release the results of collaboration, say, to the public. All users are treated equally in CG. Users are authenticated to CG based on a local identity infrastructure. Membership change is expected to be highly frequent with different users coming in and going out and possibly re-joining while contributing information and collaborating in CG. We assume that CG is self-administered. That is membership to CG is controlled by CG admin(s) who also control final release of the results of the collaboration.

These scenarios albeit relatively simple, illustrate the relevance of groups in a wide variety of collaboration scenarios.

III. Formal Specification of Bilateral Inter-Organizational Collaboration

We formally specify a complete policy model for the bilateral inter-organizational collaboration discussed in scenario #1 earlier. We closely follow the attribute-based UCON model for usage control [7]. Authorization to perform an operation (such as reading an object) is expressed over the attributes (e.g. user id, group membership, etc.) of the entities involved in the operation (such as the user and the object). A unique capability of UCON is that if the operation succeeds, the attributes may be updated. For example, if the operation is to remove a user from a group, the user's membership attribute is updated accordingly. In the following, for convenience, we interchangeably use the terms "organization" and "org" and "collaboration group" and "group" respectively.

A. Overview

We assume a federated administrative model for managing user and object membership in the group. User membership in the group is controlled by respective org's admins. Each collaborating org has total control on users and objects that they want to contribute or remove from the group.

In a distributed sharing setting such as inter-organizational collaboration, we expect that objects will be versioned. We assume a general version model where each write operation on an object creates a new version of that object. Orgs can share a specific version of an object with other orgs by adding it to the collaboration group. The same version may be shared by an org with one or more collaboration groups. Newer versions of the object created in the org are not automatically available to users in the group unless explicitly shared. Similarly, newer versions created in the group from the version shared by an org are not automatically available to users in the org unless explicitly shared. Such new versions in the group can be merged back to that org as allowed by the group admins.

New objects (and new versions of those objects) may be created in the group during collaboration. The group's admins (representatives from each collaborating org) may export specific versions of such objects to all collaborating organizations (recall that in scenario #1 each org is an equal stakeholder). We use a pull model here where specific versions in the group are marked as importable and it is the responsibility of collaborating orgs' admins to import such versions back to their respective orgs as and when they are made available.

Figure 1 illustrates a group that is established between two orgs A and B (more generally, any number of orgs may participate). Operations labeled over the arcs are administrative operations and those that are contained within the group and the orgs belong to the operational model where users create subjects (processes) and exercise their privileges.

B. Attributes

Table I specifies the sets and attributes in the system. UNIV_ORG, UNIV_CG, UNIV_U, UNIV_S, UNIV_O, UNIV_V represent the universal sets (name spaces) of orgs, groups established between such orgs, users, subjects, objects and object versions respectively. Similarly sets ORG, CG, U, S and O represent the existing sets of respective entities. Note that since each object in the set O require the name space of UNIV_V, we do not maintain a single set of existing versions. As will be discussed later, we address this by maintaining an attribute of the object, currV, which keeps track of the existing versions of that object.

User Attributes: "User" is the representation of a human in the system. We assume that a user can be a member of only one organization. Thus attribute uorg specifies the membership organization of a user. But there could be any number of groups established between the organizations and the user may be a member of more than one group. Thus ucg maps the user to a set of groups as indicated by 2^{CG} (the power set of CG). The orgadmin attribute specifies if the user is an administrator of his/her membership organization. The user could be assigned as an administrator of one or more groups as specified by the cgadmin attribute.

Object Attributes: Each object is a member of a single entity (an org or a group) as represented by the member attribute. The object implicitly becomes a member of an org or a group depending on where it is created. The currV attribute maintains the list of current versions of an object.

Object Version Attributes: Each object may have many versions and specific versions may be shared by organizations with one or more groups. Also, specific versions in a group may be made available to all participating organizations by the group's admins. Thus given an object and version, attribute vMember lists the orgs and/or groups to which the version belongs and vSuspended specifies if the version has been suspended. In our model, we support a Suspend operation instead of a Delete operation. In the case the version was created in a group, importable specifies if the version can be imported by all orgs associated with the group (we will discuss this process in detail in section III-C).

Group Attributes: The assoc attribute of a group lists the set of collaborating orgs that are associated with the group.

Subject Attributes: "Subjects" represent processes that execute on behalf of the user. The sOwner attribute specifies the user that owns the subject (the creator of the subject), type specifies whether the subject is read-only or read-write and belongsTo specifies the org or group in which the subject was created.

C. Administrative Model

In table II, we formally specify a set of administrative operations that are relevant to the collaboration group (as illustrated in figure 1). Various semantics for user Join and Leave and object Add and Remove groups operations have been proposed in [6]. For instance, users may join the group with a Strict Join (SJ) or Liberal Join (LJ). In SJ, the joining user may only access objects that are added after join time. In LJ, the users may also access objects added prior to join time. Similarly, users may leave with a Strict Leave (SL) or Liberal Leave (LL). In SL, the user loses complete access to all objects in CG. In LL, the user retains access to objects authorized during their membership period in CG. Further, objects could be added and removed in Strict or Liberal fashion. In Strict Add (SA), only existing users at add time in CG may access the object. Future CG members cannot access strictly added object in the past. Liberal Add (LA), on the hand, has no such restrictions. In Strict Remove (SR), no CG user will be able to access the removed object after remove time. Liberal Remove (LR) allows those CG members who had authorization at remove time to retain access. In the models

TABLE I
ATTRIBUTE DEFINITIONS

Universal sets of names: UNIV_ORG: The universe of organizations UNIV_CG: The universe of collaboration groups UNIV_U: The universe of users UNIV_S: The universe of subjects UNIV_O: The universe of objects UNIV_V: The universe of versions	**Existing sets of names:** ORG: Set of all existing collaborating organizations CG: Set of all existing groups established between orgs in ORG U: Set of all existing users S: Set of all existing subjects O: Set of all existing objects
User Attributes: $\text{Att (U)} = \{\text{uorg}, \text{ucg}, \text{orgadmin}, \text{cgadmin}\}$ $\text{uorg} : \text{U} \rightarrow \text{ORG}$ $\text{ucg} : \text{U} \rightarrow 2^{\text{CG}}$ $\text{orgadmin} : \text{U} \rightarrow \{\text{True}, \text{False}\}$ $\text{cgadmin} : \text{U} \rightarrow 2^{\text{CG}}$	**Objects Attributes:** $\text{Att (O)} = \{\text{member}, \text{currV}\}$ $\text{member} : \text{O} \rightarrow \text{ORG} \cup \text{CG}$ $\text{currV} : \text{O} \rightarrow 2^{\text{UNIV\_V}}$
Object Version Attributes: $\text{Att (O, UNIV\_V)} = \{\text{vMember}, \text{vSuspended}, \text{importable}\}$ $\text{vMember} : \text{O} \times \text{UNIV\_V} \hookrightarrow 2^{\text{ORG} \cup \text{CG}}$ $\text{vSuspended} : \text{O} \times \text{UNIV\_V} \hookrightarrow \{\text{True}, \text{False}\}$ $\text{importable} : \text{O} \times \text{UNIV\_V} \hookrightarrow \{\text{True}, \text{False}\}$ As shown, vMember, vSuspended and importable are partial functions that are undefined for object versions that do not currently exist.	
Group Attributes: $\text{Att (CG)} = \{\text{assoc}\}$ $\text{assoc} : \text{CG} \rightarrow 2^{\text{ORG}}$	**Subject Attributes:** $\text{Att (S)} = \{\text{sOwner}, \text{type}, \text{belongsTo}\}$ $\text{sOwner} : \text{S} \rightarrow \text{U}$ $\text{type} : \text{S} \rightarrow \{\text{ro}, \text{rw}\}$ $\text{belongsTo} : \text{S} \rightarrow \text{ORG} \cup \text{CG}$

we discuss, we confine the authorization semantics to Liberal Join, Strict Leave for users and Liberal Add, Strict Remove for objects. Note that any variation of these semantics will only affect the authorizations of Join, Leave, Add and Remove operations leaving the rest of the specification intact.

We do not consider organizational administrative operations such as the user joining and leaving the org due to space limitations and organizational variations. In table II, the first column specifies the operation that is to be performed, the second column specifies the conditions under which the operation is authorized and the final column specifies the attributes and sets that need to be updated if the operation succeeds. In the third column, an attribute/set name denoted with a superscripted prime refers to its new value after the update and that without the prime refers to its existing value before the update.

- **Establish** collaboration group: A set of administrative users come together to form a collaboration group between their representative organizations. In the first column in table II, $uSet$ is the set of users and cg is the group to be established. Note that cg cannot be an existing group name. Since UNIV_CG is the set of universe of group names and CG the existing group names, we require that cg be picked from the unused group names set UNIV_CG − CG. In all of the following operations, we take a similar approach whenever a new name is required. In column 2, for Establish to succeed, we require that there be exactly one administrative user in the set $uSet$ from each collaborating organization. That is any two users in $uSet$ cannot belong to the same organization. In other scenarios, this operation may simply be performed by one admin user from a single organization.
Subsequently, in column 3, the assoc attribute of cg is updated to the set of organizations of the users in $uSet$ and every user in $uSet$ is made an administrator of cg by setting the respective user's cgadmin attribute. Finally, since the group name cg is now used, we update the existing group names set CG accordingly.
- **Join** user to group: An org's admin user $u1$ is allowed to join the same org's user $u2$ to the group cg if an association exists between $u1$ and $u2$'s org and cg. Note that $u1$ should also be an admin of cg (that is, $u1$ participated in cg's Establish process and became the group admin at that time or was later substituted to be the group admin). $u2$'s ucg attribute is updated to reflect the fact that the user is now a member of cg.
- **Leave** user from group: Leave operation is similar to join, except all the subjects executing on the user's behalf that were created in the group should be killed. To this end, we update the existing subjects set S by removing all the subjects executing in the group that are owned by the leaving user. As we will see in the operational model, this ensures that such subjects cannot continue performing any actions.
- **Add** object version to group: A user u is allowed to add a version v of an object o to a group cg if u is an admin of the org to which the object version belongs and also an admin of cg. After add, attribute vMember is updated to reflect that v is a member of cg.

TABLE II
ADMINISTRATIVE MODEL

Operation	Authorization Requirement ($\rightarrow$)	Updates
$\forall uSet \subseteq \mathrm{U}.$ $\forall cg \in \mathrm{UNIV\_CG} - \mathrm{CG}.$ **Establish**$(uSet, cg)$	$\forall u1, u2 \in uSet.(\mathrm{orgadmin}(u1) \wedge$ $(u1 \neq u2 \rightarrow \mathrm{uorg}(u1) \neq \mathrm{uorg}(u2)))$	$\mathrm{assoc}'(cg) = \bigcup_{\forall u \in uSet} \mathrm{uorg}(u)$ $\forall u \in uSet.\mathrm{cgadmin}'(u) = \mathrm{cgadmin}(u) \cup \{cg\}$ $\mathrm{CG}' = \mathrm{CG} \cup \{cg\}$
$\forall u1, u2 \in \mathrm{U}.\forall cg \in \mathrm{CG}.$ **Join**$(u1, u2, cg)$	$\mathrm{orgadmin}(u1) \wedge$ $cg \in \mathrm{cgadmin}(u1) \wedge (\mathrm{uorg}(u1) = \mathrm{uorg}(u2)) \wedge$ $cg \notin \mathrm{ucg}(u2) \wedge \mathrm{uorg}(u1) \in \mathrm{assoc}(cg)$	$\mathrm{ucg}'(u2) = \mathrm{ucg}(u2) \cup \{cg\}$
$\forall u1, u2 \in \mathrm{U}.\forall cg \in \mathrm{CG}.$ **Leave**$(u1, u2, cg)$	$\mathrm{orgadmin}(u1) \wedge$ $cg \in \mathrm{cgadmin}(u1) \wedge (\mathrm{uorg}(u1) = \mathrm{uorg}(u2)) \wedge$ $cg \in \mathrm{ucg}(u2) \wedge \mathrm{uorg}(u1) \in \mathrm{assoc}(cg)$	$\mathrm{ucg}'(u2) = \mathrm{ucg}(u2) - \{cg\}$ $\forall s \in \mathrm{S}.\mathrm{sOwner}(s) = u2 \wedge \mathrm{belongsTo}(s) = cg.$ $\mathrm{S}' = \mathrm{S} - \{s\}$
$\forall u \in \mathrm{U}.\forall o \in \mathrm{O}.$ $\forall v \in \mathrm{currV}(o).\forall cg \in \mathrm{CG}.$ **Add**(u, o, v, cg)	$\mathrm{uorg}(u) = \mathrm{member}(o) \wedge \mathrm{uorg}(u) \in \mathrm{assoc}(cg)$ $\wedge\ cg \in \mathrm{cgadmin}(u) \wedge \mathrm{orgadmin}(u)$ $\wedge\ \mathrm{uorg}(u) \in \mathrm{vMember}(o, v) \wedge cg \notin \mathrm{vMember}(o, v)$	$\mathrm{vMember}'(o, v) = \mathrm{vMember}(o, v) \cup \{cg\}$
$\forall u \in \mathrm{U}.\forall o \in \mathrm{O}.$ $\forall v \in \mathrm{currV}(o).\forall cg \in \mathrm{CG}.$ **Remove**(u, o, v, cg)	$\mathrm{uorg}(u) = \mathrm{member}(o) \wedge \mathrm{uorg}(u) \in \mathrm{assoc}(cg)$ $\wedge\ cg \in \mathrm{cgadmin}(u) \wedge \mathrm{orgadmin}(u)$ $\wedge\ \mathrm{uorg}(u) \in \mathrm{vMember}(o, v) \wedge cg \in \mathrm{vMember}(o, v)$	$\mathrm{vMember}'(o, v) = \mathrm{vMember}(o, v) - \{cg\}$
$\forall u1, u2 \in \mathrm{U}.\forall cg \in \mathrm{CG}.$ **Substitute**$(u1, u2, cg)$	$\mathrm{orgadmin}(u2) \wedge cg \in \mathrm{cgadmin}(u1) \wedge$ $\mathrm{uorg}(u1) = \mathrm{uorg}(u2) \wedge \mathrm{uorg}(u1) \in \mathrm{assoc}(cg)$	$\mathrm{cgadmin}'(u1) = \mathrm{cgadmin}(u1) - \{cg\}$ $\mathrm{cgadmin}'(u2) = \mathrm{cgadmin}(u2) \cup \{cg\}$
$\forall uSet \subseteq \mathrm{U}.\forall cg \in \mathrm{CG}.$ $\forall o \in \mathrm{O}.\forall v \in \mathrm{currV}(o).$ **Export**$(uSet, cg, o, v)$	$\forall u1 \in uSet.cg \in \mathrm{cgadmin}(u1) \wedge$ $\mathrm{assoc}(cg) = \bigcup_{\forall u2 \in uSet} \mathrm{uorg}(u2) \wedge$ $\neg \mathrm{vSuspended}(o, v) \wedge cg \in \mathrm{vMember}(o, v) \wedge$ $\mathrm{member}(o) = cg \wedge \neg \mathrm{importable}(o, v)$	$\mathrm{importable}'(o, v) = \mathrm{True}$
$\forall u \in \mathrm{U}.\forall o1, o2 \in \mathrm{O}.$ $\forall v1 \in \mathrm{currV}(o1).\forall cg \in \mathrm{CG}.$ **Import**$(u, o1, v1, o2, cg)$	$\mathrm{member}(o1) = cg \wedge \mathrm{importable}(o1, v1) \wedge$ $cg \in \mathrm{cgadmin}(u) \wedge \mathrm{orgadmin}(u) \wedge$ $\mathrm{uorg}(u) \in \mathrm{assoc}(cg) \wedge \neg \mathrm{vSuspended}(o1, v1) \wedge$ $\mathrm{member}(o2) = \mathrm{uorg}(u)$	$\mathrm{currV}'(o2) = \mathrm{currV}(o2) \cup \{v2\}$ $/*\ where\ v2 \in \mathrm{UNIV\_V} - \mathrm{currV}(o2)$ $is\ a\ newly\ generated\ version\ id\ */$ $\mathrm{vMember}'(o2, v2) = \mathrm{vMember}(o2, v2) \cup \mathrm{uorg}(u)$
$\forall uSet \subseteq \mathrm{U}.\forall cg \in \mathrm{CG}.$ $\forall o \in \mathrm{O}.\forall v \in \mathrm{currV}(o).$ **Merge**$(uSet, cg, o, v)$	$\bigcup_{\forall u1 \in uSet} \mathrm{uorg}(u1) = \mathrm{assoc}(cg) \wedge$ $\forall u2 \in uSet.(cg \in \mathrm{cgadmin}(u2)) \wedge$ $\exists u3 \in uSet.\mathrm{uorg}(u3) = \mathrm{member}(o) \wedge$ $cg \in \mathrm{vMember}(o, v)$	$\mathrm{vMember}'(o, v) = \mathrm{vMember}(o, v) \cup \mathrm{member}(o)$
$\forall uSet \subseteq \mathrm{U}.\forall cg \in \mathrm{CG}.$ **Disband**$(uSet, cg, disband)$	$\forall u1 \in uSet.(\mathrm{orgadmin}(u1) \wedge cg \in \mathrm{cgadmin}(u1)) \wedge$ $\bigcup_{\forall u \in uSet} \mathrm{uorg}(u) = assoc(cg)$	$\forall u2 \in \mathrm{U}.\mathrm{uorg}(u2) \in \mathrm{assoc}(cg).$ $(\mathrm{ucg}'(u2) = \mathrm{ucg}(u2) - \{cg\}$ $\mathrm{cgadmin}'(u2) = \mathrm{cgadmin}(u2) - \{cg\})$ $\mathrm{assoc}'(cg) = NULL$ $\forall o \in \mathrm{O}.cg = \mathrm{member}(o).\mathrm{member}'(o) = NULL$ $\forall o \in \mathrm{O}.\forall v \in \mathrm{currV}(o).cg \in \mathrm{vMember}(o, v).$ $\mathrm{vMember}'(o, v) = \mathrm{vMember}(o, v) - \{cg\}$ $\mathrm{CG}' = \mathrm{CG} - \{cg\}$ $\mathrm{S}' = \mathrm{S} - \bigcup_{\forall s \in \mathrm{S}.\mathrm{belongsTo}(s) = cg} s$

- **Remove** object version from group: This is similar to Add operation. Again, u is both an admin of the group from which the version is being removed and the org that initially shared the version with the group.
- **Substitute** group admin: In the event a *cg* admin is unavailable or has to leave the group, we provide a Substitute operation to replace the leaving admin. Here, current *cg* and org admin $u1$ needs to be substituted with another org admin $u2$. Note that we make $u2$ a cgadmin in the update column. This operation is essentially delegation of permissions and more sophisticated approaches may be taken as discussed in delegation in RBAC [1]. For our purpose, this simple substitution operation serves to illustrate the requirement.
- **Export** a version in group to all source orgs: We use a pull model for exporting any object version created newly in the group. That is, the *cg* admins mark an object version in *cg* as importable and the collaborating orgs can subsequently copy that version to their respective orgs using the following Import operation. In table II, a set of *cg* admins come together to authorize Export. Export requires that every user in $uSet$ is a *cg* admin and users in $uSet$ represent every org associated with *cg*. The object version should belong to *cg* and not have been suspended. Note that Export can only be performed on versions of an object that was natively created in *cg* (as ensured by the requirement member (*o*)=*cg*). The importable attribute of the object version is set to True.

- **Import** a version from group to org: The semantics of import operation is to copy the importable object version in *cg* to the actor's organization. The *cg* admin u (who is also an admin of his/her organization) copies the importable object version $o1, v1$ to object version $o2, v2$ in u's organization (a new version $v2$ of the object $o2$ is created in org as part of Import). This is allowed if an association exists between u's organization and cg and importable attribute of $o1, v1$ is True. Similar to Export, Import can only be performed on versions of an object that was natively created in *cg*. The currV attribute of $o2$ is updated to reflect the fact that $v2$ has been used.
- **Merge** a version to org: The "merge" operation is performed on new object versions in *cg* that were created from a version that was previously shared by an organization with *cg*. Versions can only be merged back to an organization that shared an older version in the past. Hence, we verify that the object is actually a member of one of the collaborating organizations. Similar to Export, we need one admin from each collaborating organization in *cg* to authorize this operation. This is required to ensure that content of the version that is to be merged back to one of the collaborating organizations is appropriate.
- **Disband** group: The semantics of this operation is to delete the group. We assume that each of the collaborating organizations have copied all the group objects back to their respective organizations before Disband. Similar to group creation, we require that one administrator from each associated organization in the group be present to authorize this operation. Once disbanded, the corresponding attributes of every user, object and object version in the group are updated as shown. Also, all subjects executing in the group should be killed.

D. Operational Model

For the operational model, the user-subject relationship is critical. We assume "user" represents a human who can create "subjects" in the system that execute on his/her behalf. We assume that each user may have multiple subjects, however each subject is owned by one user. The user may be a member of one or more collaboration groups established between any number of orgs. A typical user will then create a subject to read and write objects in the user's org and those in one or more collaboration groups of which he/she is a member. While it may be reasonable to trust the user in this scenario, we cannot trust subjects (programs) running on behalf of the user due to trojan horse issues. Clearly, it is critical to control information flow here. For instance, we should not allow a subject to copy arbitrary information from the org to one of the user's membership groups or copy information from one group (between org A and B) to another group (between A and C) if the user is member of both groups.

To this end, in our user-subject model, we allow two types of subjects specified at creation time by the user: read-only and read-write. A user can create a read-only subject to freely read information from his/her org and all groups that he/she is a member of. However read-only subjects cannot write information anywhere. Such a subject is convenient for the purpose of aggregation. The user can use a read-only subject to aggregate information from various groups in one place in order to read them, scan for alerts, etc. If the user wants to write information, he/she needs to create a read-write subject. However, due to the possibility of malicious subjects (trojan horses that may copy arbitrary information from one group to other or from an org to group and vice versa), we restrict read-write subjects to one group specified at creation time. Suppose an org A user is a member of group 1 and group 2 that are established between orgs A and B and A and C respectively. A read-write subject created by the user in group 1 will not be allowed to read information from group 1 and write to group 2. However, it can read and write any object that is in group 1. If the user needs to write to objects in group 2, he/she needs to create a separate read-write subject in group 2.

We discuss a set of operations involved in the operational model as specified in table III. The operations contained within the group or the orgs in figure 1 belong to the operational model. These operations allow a user to create subjects and exercise privileges in a group or an org.

- **CreateRO and CreateRW** subject: A user can create a read-only or read-write subject in either an organization or a group if he/she is a member of the respective entity. Note that the subject's type attribute is appropriately set to read-only or read-write and the creating user is made the owner of the subject. The parameter *entity* specifies where the subject should be rooted (the user's organization or a specific group of which the user is a member) and belongsTo attribute is set accordingly.
- **Kill** subject: A subject can be killed by either the owner of the subject or by the administrator of the organization or the group where the subject is rooted. The existing subjects set S is updated once the subject is killed.
- **Read** an object version: A subject can only read unsuspended object versions. If the subject is of type read-only, the subject can read object versions from the owner's organization or any of the groups of which the user who owns the subject is a member. On the other hand, if it is a read-write subject, the subject can read object versions only from the organization or group where it is rooted (as specified by the belongsTo attribute). Note that we need an ongoing authorization check even after the initial authorization for Read. In the event s is no longer an element of S (which happens if the subject is killed or if the user leaves group) or the version being read gets suspended or ceases to exist in the group or org, the read operation should stop: $\text{stopped}(s, o, v, \mathit{Read}) \leftarrow s \notin S \vee \text{vSuspended}(o, v) \vee (\text{uorg}(\text{sOwner}(s)) \notin \text{vMember}(o, v) \wedge \text{vMember}(o, v) \cap \text{ucg}(\text{sOwner}(s)) = \phi)$. The notion of stopped is adopted from the UCON model [7]. An enforcement model will provide additional details of how the stopped operation may be enforced.

TABLE III
OPERATIONAL MODEL

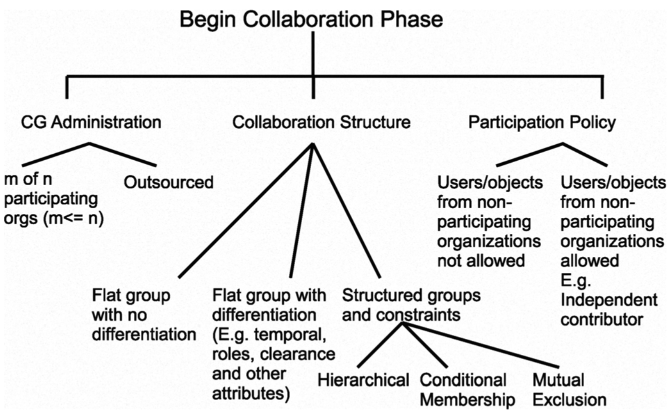
Operation	Authorization Requirement ($\longrightarrow$)	Updates
$\forall u \in \mathrm{U}.\forall s \in \mathrm{UNIV\_S} - \mathrm{S}.$ $\forall entity \in ORG \cup \mathrm{CG}.$ $\mathbf{CreateRO}(u, s, entity)$	$\mathrm{uorg}(u) = entity \vee entity \in \mathrm{ucg}(u)$	$\mathrm{sOwner}'(s) = u$ $\mathrm{type}'(s) = ro$ $\mathrm{belongsTo}'(s) = entity$ $\mathrm{S}' = \mathrm{S} \cup \{s\}$
$\forall u \in \mathrm{U}.\forall s \in \mathrm{UNIV\_S} - \mathrm{S}.$ $\forall entity \in ORG \cup \mathrm{CG}.$ $\mathbf{CreateRW}(u, s, entity)$	$\mathrm{uorg}(u) = entity \vee entity \in \mathrm{ucg}(u)$	$\mathrm{sOwner}'(s) = u$ $\mathrm{type}'(s) = rw$ $\mathrm{belongsTo}'(s) = entity$ $\mathrm{S}' = \mathrm{S} \cup \{s\}$
$\forall u \in \mathrm{U}.\forall s \in \mathrm{S}.$ $\mathbf{Kill}(u, s)$	$\mathrm{sOwner}(s) = u \vee$ $(\mathrm{orgadmin}(u) \wedge \mathrm{belongsTo}(s) = \mathrm{uorg}(u)) \vee$ $\mathrm{belongsTo}(s) \in \mathrm{cgadmin}(u)$	$\mathrm{sOwner}'(s) = NULL$ $\mathrm{type}'(s) = NULL$ $\mathrm{belongsTo}'(s) = NULL$ $\mathrm{S}' = \mathrm{S} - \{s\}$
$\forall s \in \mathrm{S}.\forall o \in \mathrm{O}.$ $\forall v \in \mathrm{currV}(o).$ $\mathbf{Read}(s, o, v)$	$\neg\mathrm{vSuspended}(o, v) \wedge$ $((\mathrm{type}(s) = ro \wedge (\mathrm{uorg}(\mathrm{sOwner}(s)) \in \mathrm{vMember}(o, v) \vee$ $\mathrm{vMember}(o, v) \cap \mathrm{ucg}(\mathrm{sOwner}(s)) \neq \phi)) \vee$ $(\mathrm{type}(s) = rw \wedge \mathrm{belongsTo}(s) \in \mathrm{vMember}(o, v)))$	None
$\forall s \in \mathrm{S}.\forall o \in \mathrm{O}.$ $\forall v1 \in \mathrm{currV}(o).$ $\mathbf{Update}(s, o, v1)$	$\mathrm{type}(s) = rw \wedge \mathrm{belongsTo}(s) \in \mathrm{vMember}(o, v1) \wedge$ $\neg\mathrm{vSuspended}(o, v1)$	$\mathrm{currV}'(o) = \mathrm{currV}(o) \cup \{v2\}$ $/{*}where\ v2 \in \mathrm{UNIV\_V} - \mathrm{currV}(o)$ $is\ a\ newly\ generated\ version\ id{*}/$ $\mathrm{vMember}'(o, v2) = \mathrm{belongsTo}(s)$ $\mathrm{vSuspended}'(o, v2) = \mathrm{False}$ $\mathrm{importable}'(o, v2) = \mathrm{False}$
$\forall s \in \mathrm{S}.\forall o \in \mathrm{UNIV\_O} - \mathrm{O}.$ $\mathbf{Create}(s, o)$	$\mathrm{type}(s) = rw$	$\mathrm{currV}'(o) = \mathrm{currV}(o) \cup \{v\}$ $/ * where\ v \in \mathrm{UNIV\_V}$ $is\ a\ newly\ generated\ version\ id{*}/$ $\mathrm{vMember}'(o, v) = \mathrm{belongsTo}(s)$ $\mathrm{member}'(o) = \mathrm{belongsTo}(s)$ $\mathrm{vSuspended}'(o, v) = \mathrm{False}$ $\mathrm{importable}'(o, v) = \mathrm{False}$ $\mathrm{O}' = \mathrm{O} \cup \{o\}$
$\forall s \in \mathrm{S}.\forall o \in \mathrm{O}.\forall v \in \mathrm{currV}(o).$ $\mathbf{Suspend}(s, o, v)$	$\mathrm{type}(s) = rw \wedge \mathrm{belongsTo}(s) \in \mathrm{vMember}(o, v) \wedge$ $\neg\mathrm{vSuspended}(o, v)$	$\mathrm{vSuspended}'(o, v) = \mathrm{True}$
$\forall s \in \mathrm{S}.\forall o \in \mathrm{O}.\forall v \in \mathrm{currV}(o).$ $\mathbf{Resume}(s, o, v)$	$\mathrm{type}(s) = rw \wedge \mathrm{belongsTo}(s) \in \mathrm{vMember}(o, v) \wedge$ $\mathrm{vSuspended}(o, v)$	$\mathrm{vSuspended}'(o, v) = \mathrm{False}$

- **Update** an object version: Subjects of type read-write may update un-suspended object versions. Updating an object version creates a new version of the object in the same organization or group where the subject is rooted. As shown, the new version $v2$ is made a member of the organization or group where the subject is rooted. Again, $\mathrm{stopped}(s, o, v1, Update) \leftarrow s \notin S \vee \mathrm{belongsTo}(s) \notin \mathrm{vMember}(o, v1)$.
- **Create** an object: A read-write subject can create a new object in the organization or group where it is rooted. As shown, the object is created with a new version v which is made a member of the subject's rooted org or group. Also, $\mathrm{stopped}(s, o, Create) \leftarrow s \notin S$.
- **Suspend/Resume** a version: A read-write subject can suspend or resume an object version that is member of the organization or group where the subject is rooted. vSuspended is set to True or False accordingly. Also, $\mathrm{stopped}(s, o, v, Suspend) \leftarrow s \notin S \vee \mathrm{belongsTo}(s) \notin \mathrm{vMember}(o, v)$. The condition under which the Resume operation should stop is identical to Suspend.

IV. GROUP-CENTRIC COLLABORATION FRAMEWORK

In this section, we present a framework for developing group-centric collaboration models. The framework consolidates many issues that are typical in most inter-organizational collaboration scenarios as discussed earlier.

Formal specification of one of the scenarios allowed us to realize a host of issues that needed to be resolved even in the most simple inter-organizational collaboration scenario where two or more orgs establish a group to share their resources. The resources we considered included only users and objects. There were no sub-groups or hierarchical groups within the collaboration group and all users had the same permissions in the group (that is permissions did not depend on other user attributes such as role). Based on this detailed case study, we consolidate various alternatives and issues into a framework for developing group-centric collaboration models.



Begin Collaboration Phase
CG Administration
Collaboration Structure
Participation Policy
m of n participating orgs (m<= n)
Outsourced
Users/objects from non-participating organizations not allowed
Users/objects from non-participating organizations allowed E.g. Independent contributor
Flat group with no differentiation
Flat group with differentiation (E.g. temporal, roles, clearance and other attributes)
Structured groups and constraints
Hierarchical
Conditional Membership
Mutual Exclusion

Fig. 2. Begin Collaboration Phase.

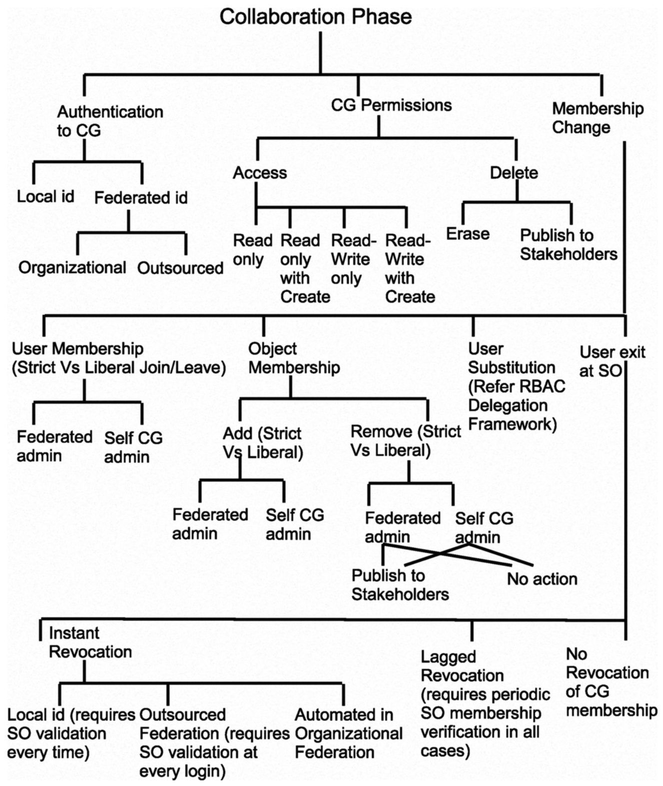


Fig. 3. Collaboration Phase.

At a high level, organizational collaboration scenarios can be classified into three distinct phases. In the Begin Collaboration phase, participating organizations make various fundamental decisions about the collaboration. For example, which entities may participate in the collaboration, the entity that controls the collaboration group, etc. Next, in the Collaboration Phase, users participate in the collaboration. During this phase, new ideas are born and in general, things change as the group evolves. The collaborating organizations need to decide on how to handle issues such as who controls membership to the collaboration group, what is the policy if a member exits the source organization, etc. Finally, in the End Collaboration phase, some or all of participating organizations take away the results of the collaboration. In the following discussion, we refer to the organizations that participate in the collaboration by contributing users and objects as Source Organizations (SO). The group that is established for collaboration is referred to as Collaboration Group (CG). Any other organization that facilitates the collaboration is referred to as third-party.

1) Begin Collaboration Phase: In this phase (figure 2), there are at least three important high-level decisions to be made. First a decision on who is responsible for administration of the CG needs to be made. For example, when 3 SOs A, B and C collaborate, CG may be controlled and hosted at one of the SOs or control may be shared between two of the SOs or all of them. Alternatively, the control could be outsourced to a third-party. Next, participating SOs need to decide if CG users will be differentiated in terms of privileges that they can exercise in the group where a specific group structure may be established for this purpose. In the simplest case, it could be a flat group where all users are treated equally or users may be differentiated as per their attributes such as their roles, time of joining the group, etc. In more complex settings, various structures and constraints could be established. For example, the groups could be set up in a hierarchical fashion where membership at higher group may automatically grant permissions to lower groups. Further constraints such as mutual exclusion (where membership in one group may be mutually exclusive to membership in one or more other groups) and/or conditional membership (where membership in a source group may be required for continued membership in another group) may be imposed. A final issue is whether users and objects from a non-participating organization are allowed to be present in CG. In some scenarios this may be desirable where although only two organizations formally initiate and establish a collaboration, a limited number of users may be admitted from a different organization. The users may also be individual contributors who may not belong to any organization. In other cases, this may not be allowed due to the sensitivity of collaboration.

2) Collaboration Phase: During this phase (figure 3), collaboration occurs and users and objects may be added and removed and the CG evolves. First, users need to be authenticated to CG. Authentication can be carried out locally by CG or could be federated to respective SOs. While maintaining a local identity infrastructure is not desirable in inter-organization collaboration, it is practical in scenarios where users don't belong to any organization, yet collaborate. In federated id systems, respective organizations authenticate their users and send a verifiable assertion that the user has been authenticated. CG can verify the assertion and decide to let the user in or not. Clearly, it is the responsibility of SOs to correctly authenticate their users.

The next issue is handling membership change. As mentioned earlier in section III-C, users may join and leave the group with SJ or LJ and SL or LL respectively. Administration of user join and leave may be federated or controlled locally by the CG. In federated admin, the SOs admit and remove their users to/from CG at their discretion. In self CG admin, the CG admins are responsible for managing users. Admission policy could vary depending on the scenario. For example, only faculty members from accredited universities may be allowed to join CG.

Similar to user membership, object membership may be either federated or self administered by CG. In federated admin, participating SOs decide what objects they want to contribute to CG. Note that federated object remove is tricky in some situations. Suppose that organization A adds an object to CG and organization B members in CG update those objects. In federated remove, if organization A decides to remove the object from CG, organization B CG members may lose access to that object. Clearly, the object update model is relevant in this scenario. If updating an object, creates a new version, it is easier to support any type of policy for object remove. Thus if organization A removes the original object, in some scenarios, newer versions of that object may be allowed to remain in CG. On the other hand, object remove may be controlled by CG admins locally. At remove time, they may choose to either publish the removed object to all the stakeholders in the collaboration or take no action, in which case the object is simply removed from the group with no information flowing back to other stakeholders. Further objects could be added and removed in Strict or Liberal fashion as mentioned in section III-C.

The next issue is whether user substitution is allowed which involves questions related to delegation of substituting user's permissions in CG. For example, it could be temporary or permanent substitution, single-step (where only one level of substitution allowed) vs multi-step (where subsequent substitutions may be allowed), etc. There are a number of such alternatives and many issues in the delegation framework [1] proposed for role-based access control apply here.

A final issue is how to handle CG users leaving their respective SOs. In certain scenarios, an instant or lagged revocation to CG may be required. In other scenarios, user's exit from SO may not affect their membership in CG. Instant revocation to CG can be enforced if the user's validation with the SO is carried out with every login. This is simple in organizational federated id scenarios. However, in both local id and outsourced federation, user's membership in SO needs to be verified every time the user attempts to login to CG. Similar issues exist in lagged revocation, except that membership validation with SO is performed periodically instead of doing it at every login. Clearly, instant revocation may incur significant performance penalty while lagged revocation may incur assurance penalty while achieving better performance.

3) End Collaboration Phase: When the collaboration ends (figure 4), CG may be torn down or suspended if future collaboration is anticipated. In either case, the results of the collaboration may need to published to the stakeholders of the collaboration. In some cases, participating organizations may all be equal stakeholders, while in others, only some of the participating organizations may be stakeholders and the remaining SOs may not own the final outcome. In certain unique scenarios (such as is confidential military operations), the results of collaboration may need to be destroyed after the mission is accomplished in which case there may not be any tangible information flowing back to stakeholders.

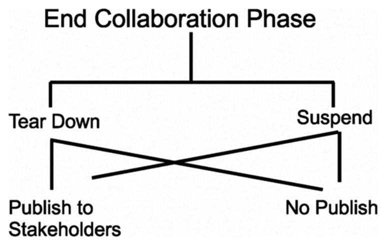


Fig. 4. End Collaboration Phase.

Based on this framework, table IV outlines an informal policy model for the three scenarios discussed in section II (recall that the complete model presented was for scenario #1).

V. Related Work and Conclusion

The concept of virtual organizations/enterprises have been extensively studied in the past (see for example [2], [4] and [3]). Numerous authorization systems have been proposed for collaborative applications. A survey of many such existing systems can be found in [15]. Most existing work in the literature focuses on enforcement architecture for collaborative systems. For instance, in [11] and [8], the authors discuss enforcement approaches for supporting collaboration in distributed systems. A few access control models have also been proposed in the past. In [12], the authors focus on an access matrix based authorization model for groupware [5]. In [13], the author proposes an authorization model for cooperative systems using groups. The model supports various operations to create group structures such as hierarchy, roles, delegation, etc. Collaboration models using groups have also been proposed. For instance, a model for multiple collaboration protocols in which collaboration may occur within and across many groups is discussed in [9]. In [14], the authors introduce Task-Based Access Control (TBAC). Unlike traditional subject-object based models, decisions in TBAC are influenced by task-based contextual information. TBAC is an active model in the sense that permissions change dynamically depending on the current stage of the task.

Our goal in this paper was to identify some of the most fundamental issues in group-centric collaboration scenarios. Clearly, the issues discussed in this paper need to be addressed for building collaborative systems at any scale. Our work primarily differs from existing approaches in many aspects. First, we focus on inter-organizational group-centric collaboration where organizations collaborate by contributing their resources in terms of users and objects. Second, our focus is purely on the policy model, thereby it clearly separates enforcement and implementation issues as illustrated in [10]. Third, we clearly separate administrative issues from operational issues of the

TABLE IV
INFORMAL POLICY MODEL FOR THE 3 USAGE SCENARIOS BASED ON THE FRAMEWORK

Issues	Scenario #1	Scenario #2	Scenario #3
1. Who creates and owns CG?	Collective admin	Org A admins	CG admins
2. Differentiate Users?	No	Yes. Timeliness of membership.	No
3. User/object membership	Federated	Federated	Self-admin
4. Users/objects from 3rd party?	No	No	N/A
5. Authentication to CG	Federated id	Federated id	Local id
6. Membership policy	(LJ, SL), (LA, SR)	(LJ, SL) for A, (SJ, SL) for B, Mixed operations for objects (SA/LA, SR/LR)	(LJ, SL), (LA, SR)
9. Handling user exit from SO	Handled by federated id	Handled by federated id	N/A
10. Handling object deletes in SO	SR from group	SR/LR at CG admin discretion	SR/LR at CG admin discretion
11. User substitution	Yes. Single-step, temporary, complete inheritance.	Yes. Single-step, temporary, complete inheritance.	No
12. Disband CG	Orgs A and B take away results	Only org A takes away results	CG admins publish

model. To this end, we formally proposed both administrative and operational model for a specific usage scenario. Further, the operational model specifies the important user-subject model. This distinction of users from subjects is critical and seems to be missing in existing collaboration work. Finally, we also proposed a framework for developing richer models for group-centric collaboration.

In this paper, we only looked at establishing a single group for collaboration where all users are treated equally in terms of their permissions in the group. We showed that even in such a seemingly simply scenario, there are many fundamental decisions to be made and fairly sophisticated models can be developed. We are currently investigating verifying certain information flow properties against our model possibly using automated techniques such as model checking. Further, as with any information sharing systems, thorny covert channel issues may exist which may only be addressed using proper enforcement and implementation techniques. In the future, we are interested in looking at supporting structures for collaboration (e.g. hierarchical groups), applying attributes such as roles to users which influence their access in the group, etc. We are also interested in investigating other scenarios such as massively large scale collaboration like that of Wikipedia.

ACKNOWLEDGMENTS

The authors are partially supported by NSF grants IIS-0814027 and CNS-0716750, AFOSR MURI grant FA9550-08-1-0265, THECB ARP grant 010115-0037-2007 and grants from the State of Texas Emerging Tech. Fund and Intel Corp.

REFERENCES

[1] E. Barka and R. Sandhu. Framework for role-based delegation models. In *Proc. of the 16th Annual Computer Security Applications Conference*, page 168, Washington, DC, USA, 2000. IEEE Computer Society.
[2] L. Camarinha-Matos. *Virtual Organizations: systems and practices*. Springer-Verlag TELOS Santa Clara, CA, USA, 2004.
[3] L. Camarinha-Matos, H. Afsarmanesh, C. Garita, and C. Lima. Towards an architecture for virtual enterprises. *Journal of Intelligent Manufacturing*, 9(2):189–199, 1998.
[4] L. Camarinha-Matos, I. Silveri, H. Afsarmanesh, and A. Oliveira. Towards a framework for creation of Dynamic Virtual Organizations. In *Collaborative n/w and their Breeding Environments*. Springer, 2005.
[5] C. A. Ellis, S. J. Gibbs, and G. Rein. Groupware: Some issues and experiences. In *Comm. of the ACM*, pages 38–58, 1991.
[6] R. Krishnan, R. Sandhu, J. Niu, and W. H. Winsborough. Foundations for group-centric secure information sharing models. In *Proc. of the ACM Symp. on Access Control Models and Tech.*, pages 115–124, 2009.
[7] J. Park and R. Sandhu. The UCON$_{ABC}$ usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.
[8] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Proc. of Inter. Work. on Policies for Dist. Systs. and Netw.)*, page 50, 2002.
[9] W. Picard. Modeling structured non-monolithic collaboration processes. In *Collaborative Networks and their Breeding Environments, Proc. of the 6th IFIP Working Conf. on Virtual Enterprises (PRO-VE 2005), Valencia, Spain*. Springer, 2005.
[10] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and PEI models. In *Proc. of ACM Symp. on Inf., Comp. and Comm. Sec.*, pages 2–12, 2006.
[11] D. Shands, J. Jacobs, R. Yee, and E. J. Sebes. Secure virtual enclaves: Supporting coalition use of distributed application technologies. *ACM Trans. Inf. Syst. Secur.*, 4(2):103–133, 2001.
[12] H. Shen and P. Dewan. Access control for collaborative environments. In *Proc. of ACM Conf. on Comp. Supp. Coop. work*, pages 51–58, 1992.
[13] K. Sikkel. A group-based authorization model for cooperative systems. In *Proc. of the fifth conference on European Conference on Computer-Supported Cooperative Work*, pages 345–360, 1997.
[14] R. K. Thomas and R. S. Sandhu. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented autorization management. In *Proc. of IFIP TC11 WG11.3*, 1998.
[15] W. Tolone, G.-J. Ahn, T. Pai, and S.-P. Hong. Access control in collaborative systems. *ACM Comput. Surv.*, 37(1):29–41, 2005.