# A Lattice Interpretation of Group-Centric Collaboration with Expedient Insiders

Khalid Zaman Bijon*, Tahmina Ahmed*, Ravi Sandhu* and Ram Krishnan†
*Institute for Cyber Security & Department of Computer Science
†Institute for Cyber Security & Department of Electrical and Computer Engineering
University of Texas at San Antonio

***Abstract*—For various reasons organizations need to collaborate with external consultants, e.g. domain specialists, on specific projects. Many security-oriented organizations deploy multi-level systems which enforce one directional information flow in a lattice of security labels. However, traditional lattice constructions are not suitable for accommodating external consultants, since such consultants are not "true insiders" but rather "expedient insiders" who should receive much more limited privileges than employees. An authorization model for group-centric collaboration with expedient insiders (GEI) has been recently proposed, wherein organizations create groups and replicate the organizational lattice with selected content for such collaborations [4]. Motivated by GEI, in this paper, we formulate a novel lattice construction wherein a new collaboration category is introduced for each new collaboration group, in a manner significantly different from the usual process of defining new security categories in a lattice. In particular, a collaboration category brings together only the required objects and users. We develop a formal model for lattices with collaborative compartments (LCC) comprising administrative and operational parts covering the life-cycle of such collaborations. We formally prove the equivalence of LCC and GEI, thereby precisely characterizing the information flow and security properties of GEI which heretofore had only been informally considered. This equivalence shows that GEI can be realized via LBAC with minimal operational disruptions.**

**Keywords- Group Centric Collaboration; Information Sharing; Lattice Based Access Control;**

## I. INTRODUCTION

Security-oriented organizations, encompassing government, military and commercial sectors, deploy lattice-based access control or LBAC [10] (also commonly known as multilevel security or mandatory access control) to enforce one directional information flow within the organization. In general, LBAC supports a variety of security policies including confidentiality, integrity and separation. A lattice is a set of security labels wherein a person is cleared and an object is classified at a particular label in the lattice. A security label is typically constructed as a combination of a security level and a subset of the categories. Security levels are totally ordered (e.g. top secret (TS) $>$ secret (S) $>$ classified (C) $>$ unclassified (U)), while the categories are unordered (e.g. ProjA, ProjB and ProjC). Information flow in a lattice is aligned with the partial ordering of the security labels. For example, a subject cleared to the security label $\langle$S, {ProjA}$\rangle$, may only get access to an object that is classified at same or lower label, i.e. $\langle$S, {ProjA}$\rangle$, $\langle$C, {ProjA}$\rangle$, $\langle$U, {ProjA}$\rangle$, $\langle$S, $\emptyset\rangle$, $\langle$C, $\emptyset\rangle$, or $\langle$U, $\emptyset\rangle$. The resulting information sharing within the organization is based on the long-term relationship between the organization and its employees, as well as employees' responsibility, competence, trustworthiness and accountability.

Organizations increasingly need to share information with outside consultants. These consultants are not regular employees and do not have same accountability or trustworthiness as employees. They are not "true insiders" but rather "expedient insiders" as their presence in the organization is transient and opportunistic rather than persistent. Therefore, information sharing with such expedient insiders in similar fashion as true insiders, i.e. assign a security label from existing lattice to each consultant, will lead to excessive exposure of sensitive information beyond that needed for the collaboration.

Recently a model has been proposed [4] for such collaborations between a single organization and outside consultants. This model motivates the Group-Centric concept [7] as an appropriate means for this purpose. A new group is established for each collaboration. Each group replicates the organizational lattice structure in an identical but separate copy. Initially, the group does not contain any users or objects. Gradually the organization populates the group with selected objects, true insiders and expedient insiders. For ease of reference we call the model of [4] as the GEI model, for group-centric collaboration with expedient insiders. In GEI, separate piece of lattices are maintained for the organization and and for each collaboration groups. Thus, information flow and security properties for the overall system are only informally addressed.

In this paper, we develop an alternate model in which the organizational lattice is augmented for each collaboration group. Specifically, our goal is to develop a single lattice structure that functionally meets all the requirements proposed in GEI with security equivalence. In our model, an organization creates a new collaboration category for each newly started collaboration group and removes it upon end of the collaboration. Similar to a traditional category, a collaboration category is a collection of objects, users and subjects brought together for a particular purpose. We will show later that collaboration categories behave very differently from regular categories, in their relationship to the rest of the lattice. We develop a formal model for operational and administrative management for the complete life-cycle of this lattice construction. We call the resulting model LCC, for LBAC with collaboration compartments. Besides lattice constructions, LCC provides

necessary functionalities to capture all the requirements for a collaboration with expedient insiders proposed in GEI. Finally, we prove the equivalence of LCC and GEI using state-matching reductions [11] which preserve security properties. Hence LCC and GEI have equivalent security behavior. The security properties of LCC follow from the well-known formal security properties of LBAC. Our equivalence result extrapolates these properties to GEI, whose security properties have only been informally considered and motivated heretofore.

In the rest of the paper, section II discusses related work. Section III gives an overview of GEI. Lattice based access control models with traditional compartments (LTC) and collaboration compartments (LCC) are discussed in sections IV and V respectively. Section VI formally specifies the authorization model for LCC. The equivalence of GEI and LCC is shown in section VII and section VIII gives our conclusions.

## II. Related Work

The conceptual foundation of lattice-based information was defined by Denning [5]. Bell-Lapadula [2] showed how lattices can be applied to enforce confidentiality policies. Lattice-based integrity policies were addressed by Biba [3]. A lattice interpretation of the Chinese wall policy of separation was defined by Sandhu [10]. In addition to these traditional policies, more recently, a number of authors have addressed lattices in a variety of modern domains. Ray et al [9] introduce a location-based mandatory access control model for a system that uses wireless networks and mobile devices. They extend the Bell-Lapadula model by attaching a security label to every location that represents the type of information that can be accessed from that location. Jafarian et al [6] propose context awareness in mandatory access control in order to provide more dynamism in policy specification. For instance, in their model, a user's access to an object should satisfy some timing constraints besides satisfying their security labels.This paper similarly extends lattices beyond their traditional domains to incorporate collaboration.

## III. Group-Centric Collaboration with Expedient Insiders (GEI)

In this section we review the group-centric model for collaboration with expedient insiders [4], which we call GEI. In GEI, an organization may establish any number of distinct collaboration groups, all of which use the identical but separate lattice structure as the organization. In a group, true insiders retain the same security clearances as they have in the organization. An administrative user of the organization is selected as admin of the group and has complete control on which users (true or expedient insiders) and objects can get membership in the group. A group admin also assigns a suitable security clearance to every newly joining expedient insider.

GEI does not attempt to modify organizational lattice structure for the purpose of collaboration with expedient insiders. Rather, it maintains separate copies of identical lattice structure as the organization with selected true insiders, expedient insiders and objects. Figure 1 shows administrative and user operations for the life-cycle of such a collaboration group.

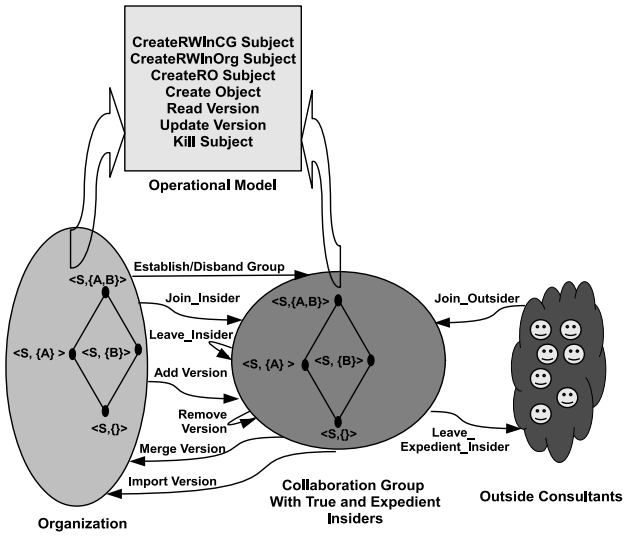


Fig. 1. **Operations in GEI.** *The group and the organization have separate but identical security lattices. Named arrows show the administrative operations where each entry gives the operation name followed by its principal target. User operations are shown in the rectangle at the top. Note that in the group, object and true insiders come from the "organization". while expedient insiders come from "outside consultants."*

A user can create subjects and exercise privileges in a group if he has the group membership. A subject is an instance of a user in the system with specific set of privileges. Subjects can be read-write or read-only. A read-write subject can read, update and create objects, however, its access is confined within the group it was created. A read-only subject can read an object from any group to which the user belongs and/or the Org (for true insiders), however, it cannot write or create objects. A subject can get the same or lower clearance as the creating user. The read operation is restricted by the usual simple-security property in which subjects can only read an object with same or lower security class. The strict form of star-property is applied for update (write) operations in which subjects can only write or create objects at same level.

In GEI, an object can have multiple versions. An update operation creates a new object version. The admin can add selected versions of an object from the organization to a group. During creation, an object inherits the security clearance of the creating subject as its classification. This classification is carried over to all future versions of the object. Non-admin users (true or expedient insiders) are not allowed to add any object to the group. They can only contribute information via newly created objects or modifying exising objects in the group. An admin also has the privilege to import or merge specific versions of objects from the group to the organization.

## IV. LBAC with Traditional Compartments (LTC)

In this section, we review components of the traditional lattice structure and corresponding information flow policy.

TABLE I
FORMAL DEFINITION OF LATTICES FROM COMPONENTS

| A: Lattice with Traditional Compartments (LTC) | B: Lattice with Collaboration Compartments (LCC) |
|---|---|
| L: is a finite set of linearly ordered security levels<br>C: is a finite set of unordered categories<br>SL: is a finite set of security labels where<br>SL = (L × $2^C$)<br>$\succeq$: is a finite dominance relation defined so that $\succeq \subseteq$ SL × SL, where<br>$\succeq$ = {((l1,c1),(l2,c2)) \| ∧ (l1,c1) ∈ SL ∧ (l2,c2) ∈ SL ∧<br>l1 $\succeq$ l2 ∧ c1 ⊇ c2}<br>⊕: SL × SL → SL is a join operator defined as<br>∀l1,l2 ∈ L and ∀c1,c2 ∈ C<br>(l1,c1) ⊕ (l2,c2) = (max(l1,l2),c1∪c2) | L: is a finite set of linearly ordered security levels<br>C: is a finite set of unordered categories<br>CC: is a finite set of unordered collaboration categories<br>Org, is the entity Organization, a constant<br>SysHigh: system high (constant label)<br>SysLow: system low (constant label)<br>SL: is a finite set of security labels where<br>SL = {(L × $2^C$)×(CC ∪ {Org})} ∪ {SysHigh, SysLow}<br>$\succeq$: is a finite dominance relation defined so that $\succeq \subseteq$ SL × SL, where<br>$\succeq$ = { ((l1,c1,cc1), (l2,c2,cc2)) \| (l1,c1,cc1) ∈ SL ∧ (l2,c2,cc2) ∈ SL<br>∧ l1 $\succeq$ l2 ∧ c1 ⊇ c2 ∧ cc1=cc2}<br>∪ {(SysHigh,x),(x,SysLow) \| x∈ SL }<br>⊕: SL × SL → SL is a join operator defined as<br>∀l1,l2 ∈ L and ∀c1,c2 ∈ C and ∀cc1,cc2 ∈ CC ∪ {Org}<br>(l1,c1,cc1) ⊕ (l2,c2,cc2) = (max(l1,l2),c1∪c2,cc1), if cc1=cc2<br>(l1,c1,cc1) ⊕ (l2,c2,cc2)=SysHigh, if cc1≠cc2<br>∀l ∈ L and ∀c ∈ C and ∀cc ∈ CC ∪ {Org}<br>(l,c,cc) ⊕ SysHigh = SysHigh, SysHigh ⊕ (l,c,cc) = SysHigh<br>(l,c,cc) ⊕ SysLow = (l,c,cc), SysLow ⊕ (l,c,cc) = (l,c,cc)<br>SysHigh ⊕ SysHigh = SysHigh, SysHigh ⊕ SysLow = SysHigh<br>SysLow ⊕ SysHigh = SysHigh, SysLow ⊕ SysLow = SysLow |

Traditional lattices are constructed by combining two components, security levels and security categories. Security levels are linearly ordered, e.g. Top Secret > Secret > Classified > Unclassified. On the other hand, security categories are unordered denoting a set of incomparable information types, e.g. salary information and medical information. Each subset of the categories is called a compartment. A security label is formed by combining a security level with a subset of categories (i.e., a compartment). Table I-A gives the formal definition of security label formation.

In LTC, a dominance relation ($\succeq$) specifies if there is an information flow from label A to B. The notation "A$\succeq$B" denotes that there is information flow from B to A or "A dominates B". Note that, A$\succeq$B if security level of A dominates security level of B and A's security category is a superset of B's, that is category (A) $\supseteq$ category (B). $\succeq$ is a partial order on SL. The binary join operator ($\oplus$) specifies the resulting label when information from two labels is combined. For example, A $\oplus$ B = C means that objects containing information from labels A and B should be labeled as C.

According to Denning [5], under certain assumptions the above mentioned components, SL, $\succeq$ and $\oplus$, form a finite lattice structure. To this end, Denning provides the following axioms for a lattice.

1) SL is a finite set of security labels
2) $\succeq$ is a partial order on SL
3) There is a lower bound that is dominated by all labels in SL
4) $\oplus$ is a least upper bound operator on SL

Note that the join operator $\oplus$ needs to be totally defined so it is possible to combine information from any pair of labels in SL and assign the result a label. This requirement guarantees there is a unique highest security label dominating all labels in SL. Formal definitions of $\succeq$ and $\oplus$ is given in Table I-A. In LTC, security levels generally remain unchanged, however, a new security category could be added if the system includes a new information category. Every time a new category is added, new labels are formed, and the dominance relations ($\succeq$) are appropriately adjusted.

Usually, every user is assigned a single clearance to a particular security label. Users can create subjects and assign their same clearance or a clearance of a lower label dominated by user's assigned clearance. Objects are also classified to a specific security label. Object versioning is not typically considered in traditional LBAC so each object has only one version in the system. In a lattice, there are several access control models for enforcing one directional information flow. For example, in the Bell-Lapadula model, subjects can only read objects with the same or lower classification and write objects with the same or higher classification. Sometimes write is limited to be at the same level.

## V. LBAC WITH COLLABORATION COMPARTMENTS (LCC)

As we discussed, collaboration with expedient insiders within traditional lattice structure is inappropriate for the following reasons.

- Assigning expedient insiders a clearance to a label might unnecessarily expose critical information beyond that necessary for the collaboration purpose.
- Pursuing collaboration with expedient insiders within the main lattice might unnecessarily involve many true insiders who are actually not participating in the collaboration.
- In the presence of two type of users, true insiders and expedient insiders, privilege management within main lattice might become very complex.

We develop a new lattice construction process with collaboration compartments in which an organization may efficiently

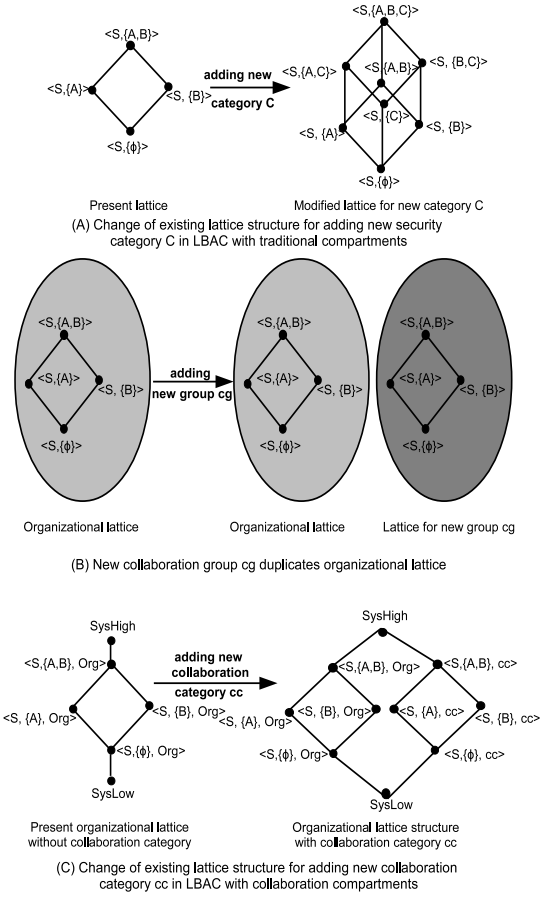


Fig. 2. *Lattice representation of three different systems: (A) LTC, (B) GEI and (C) LCC.*

perform group-centric collaboration with expedient insiders. In this process, each collaboration group introduces a new collaboration category in the system. A collaboration category is different than traditional security categories. In particular, a collaboration category does not create new combinations with existing traditional security categories or with other existing collaboration categories. Rather it creates new security labels in combination with the entire set of security labels of the organization (denoted as Org). Table I-B gives the formal definition of this security label formation from three components: security levels, traditional security categories and the newly defined collaboration categories.

Figure 2 shows three different types of lattices for LTC, GEI and LCC systems respectively. For each type, initially, current lattice of the organization contains one security level Secret (S) and two security categories (A and B). Figure 2-A shows the change involved in the lattice structure in LTC for adding a new security category C. Addition of C creates new labels and a new upper bound is also established. Figure 2-B shows that the existing lattice is duplicated for the new collaboration group cg in GEI system. Here, the cg and the organization maintains separate and disjoint copies of the lattice. However, in LCC only one lattice is maintained for the organization and collaboration groups. To this end, we introduce two fixed security labels system high (SysHigh) and system low (SysLow) that define the upper bound and lower bound of the lattice respectively. In Figure 2-C, a new collaboration group creates a new collaboration category (cc) that basically duplicates the existing organizational lattice similar to GEI but SysHigh and SysLow tie these two pieces into one lattice.

In LCC, a security label A dominates label B if security level of A dominates security level of B, security category of A is a superset of security category of B (category(A)$\supseteq$category(B)) and both A and B belong to Org or the same cc. Formal definition of the dominance relation ($\succeq$) is given in Table I-B. Join operator ($\oplus$), in this system, specifies that information from two labels can be combined and gives a formula for least upper bound if collaboration categories of these two labels are the same. Otherwise, combination of two labels results in SysHigh. $\oplus$ is formally defined in Table I-B. Finally, the lattice is formed by five elements: SL, $\succeq$, $\oplus$, SysHigh and SysLow. Note that, SysHigh and SysLow are two constant labels unpopulated with users, subjects and objects. However, they are necessary to maintain one single lattice structure in LCC. Actually SysLow could be populated with information available on the public Internet.

The main motivation behind this lattice construction is to facilitate the organization to share information in a "need to consult" basis. Thus, regular user operations should be properly defined so that they cannot flow information between collaboration groups and organization. Only administrative users of the organization should be able to select and share specific information with a collaboration group from the organization. Again, expedient insiders should not be assigned organizational labels. To this end, in the following section, we formally specify an authorization model consisting of separate administrative and operational components.

## VI. Formal Specification of the Authorization Model for LCC

In LCC, relevant attributes of the entities are utilized to authorize each operation. For example, before an operation (such as writing an object) is permitted all the involved entities' (e.g. user, object, etc) attributes (user id, object type, etc) are evaluated in order to approve it. For this purpose we follow the well-known UCON model [8] and specifically its pre-authorization component. UCON provides mutable attributes which may be updated dynamically as each operation is authorized. For example, if the operation is to merge two versions of an object, the object version attributes are updated accordingly. Details of the attributes definition are provided in section VI-B. For convenience, we use the term cc for collaboration category, Org for the organization, and cc admin for administrative members of cc.

### A. Overview

An organization may establish a number of distinct collaboration categories (cc) for the purpose of collaboration with expedient insiders and create new security labels identical to

TABLE II
SETS AND ATTRIBUTE SPECIFICATION

**Global Sets and Symbols:**

*Notation:* We use upper-case roman letter(s) subscript with $\gamma$ to represent a finite set of an entity at current state $\gamma$ and corresponding calligraphic letter(s) to represent a countably infinite set of the same entity. This helps us represent currently existing names of an entity and overall namespace of the same. For example, $U_\gamma$ represents the set of existing users at a state state $\gamma$ in the system (finite) and $\mathcal{U}$ represents the set of all possible users (countably infinite).

$L_\gamma$ = L, is a finite set of existing hierarchical ordered security levels /*For simplicity we assume fixed security levels. More generally new levels can be added*/
$C_\gamma$ = C, is a finite set of existing unordered categories /*For simplicity we assume fixed security categories. More generally new categories can be added*/
$CC_\gamma \subset \mathcal{CC}$, is a finite subset of countably infinite set of unordered collaboration categories $\mathcal{CC}$
Org, denotes the Organization, a constant
SysHigh, the system high (constant label) that dominates every security label in SL
SysLow, the system low (constant label) that is dominated by every security label in SL
$SL_\gamma = \{(L \times 2^C) \times (CC_\gamma \cup \{Org\})\} \cup \{SysHigh, SysLow\}$, is a finite set of existing security labels
$\succeq_\gamma$, is a finite dominance relation defined by the formula for $\succeq$ in Table I-B
$\oplus_\gamma$, is a join operator defined by the formula for $\oplus$ in Table I-B
$U_\gamma \subset \mathcal{U}$, is a finite subset of countably infinite set $\mathcal{U}$, i.e. existing users in $\gamma$
$O_\gamma \subset \mathcal{O}$, is a finite subset of countably infinite set $\mathcal{O}$, i.e. existing objects in $\gamma$
$S_\gamma \subset \mathcal{S}$, is a finite subset of countably infinite set $\mathcal{S}$, i.e. existing subjects in $\gamma$
$UTYPE_\gamma$ = UTYPE = {insider, expedient_insider, outsider} is the finite set of user's types
$STYPE_\gamma$ = STYPE = {RO, RW} is the finite set of subject's types

**User Related State Elements:**

hierclearanceOfUser: $U_\gamma \rightarrow L$, this function maps each user to a security level
compcategoryOfUser: $U_\gamma \rightarrow 2^C$, this function maps each user to a set of security categories
uCC: $U_\gamma \rightarrow 2^{CC_\gamma}$, this function maps each user to zero or more collaboration categories
orgAdmin: $U_\gamma \rightarrow$ {true,false}, this function maps each user to true if she is an admin of Org
ccAdmin: $U_\gamma \rightarrow 2^{CC_\gamma}$, this function maps each user to zero or more groups if he is an administrative user of a collaboration group
uType: $U_\gamma \rightarrow UTYPE_\gamma$, this function maps each user to a user type

**Objects Related State Elements:**

hierclassificationOfObject: $O_\gamma \rightarrow L$, this function maps each object to a security level
compcategoryOfObject: $O_\gamma \rightarrow 2^C$, this function maps each object to a set security categories
origin: $O_\gamma \rightarrow CC_\gamma \cup \{Org\}$, this function maps each object to the entity (collaboration category or Org) where it was created
$V_\gamma \subset \mathcal{V}$, is a finite subset of countably infinite set $\mathcal{V}$, i.e. existing versions in $\gamma$
versions: $O_\gamma \rightarrow 2^{V_\gamma} - \phi$, this function maps each object to all its existing versions in $\gamma$

**Subject Related State Elements:**

hierclearanceOfSubject: $S_\gamma \rightarrow L$, this function maps each subject to a security level
compcategoryOfSubject: $S_\gamma \rightarrow 2^C$, this function maps each subject to a set of security categories
owner: $S_\gamma \rightarrow U_\gamma$, this function maps each subject to the user who created it
belongsTo: $S_\gamma \hookrightarrow CC_\gamma$, this function maps each RW subject (not RO subject) to the collaboration category where it was created. Hence, it is a partial function
type: $S_\gamma \rightarrow STYPE_\gamma$, this function maps each subject to a subject type

**Object Version Related State Elements:**

For each $o \in O_\gamma$, $vMember_o$: versions(o)$\rightarrow 2^{CC_\gamma \cup \{Org\}} - \phi$, this functions maps each version of every object to one or more entity (collab category or Org) where this version is available to access
For each $o \in O_\gamma$, $hierclassificationOfVersion_o$: versions(o)$\rightarrow$ L, this function maps each version to a security level
For each $o \in O_\gamma$, $compcategoryOfVersion_o$: versions(o) $\rightarrow 2^C$ this function maps each version to a set of security categories

existing organizational labels. Establishing those labels causes modification of the dominance relation ($\succeq$) accordingly. After the lattice is re-constructed for a cc, the administrative user of cc might select and bring necessary objects, true insiders and expedient insiders into the new lattice. Joining a true insider to a collaboration category cc requires the true insider to be assigned to another clearance for a collaboration label belongs to cc. Thus, unlike LTC, a user needs multiple clearances in LCC. In this system, a user clearance to a particular label is formed by combining three components: security level, security categories and collaboration category. Note that, the third component represents the groups she is a member of. When a true insider joins a collaboration group cc, she gets the clearance to a label in cc that is equivalent to the organizational label to which she is cleared. In this process, the first two components of the clearance always are the same for each collaboration category she joins. On the other hand, admin of the cc can select an expedient insider and assign a suitable clearance on her first ever join to any collaboration group. An expedient insider can also have multiple clearances in order to participate in different collaboration groups, however, she cannot join to the organization. Joining to another cc adds another clearance to an expedient insider similarly to the case of true insiders.

In this system, each object is assigned a security classification as a label that consists of three components: security level, security category and origin. The last component specifies the cc or Org in which the object is created. Organizations need to share objects with different groups for the purpose of collaboration. For this purpose, we assume that objects are versioned. In a general versioning model, each write operation on an object creates a new version of the object. Each version of an object also has individual security classification. A cc administrator can bring specific versions of selected objects from the organization into the cc. Thus, a version can have multiple classifications in order to share with different groups. Similar to objects, the first two components of each classification of an object version are always same while the

last component represents the respective entities (cc and/or Org) in which it is available to access.

### B. Attributes

We now provide a formal specification of the model. Global sets and attributes used in LCC are given in Table II. U, O and S respectively represent the set of current existing users, objects and subjects in the system.

**User Attributes:** A user is a human being in the system. Users are of two types: True Insider or Expedient Insider as specified by the user attribute *utype*. A user's clearance is represented by three attributes: *hierclearanceOfUser*, *compcategroyOfUser* and *uCC*. First two attributes represent a user's security level and security categories respectively and both remain same for every entity (Org or cc) that the user joins. Attribute *uCC* lists the collaboration groups of which the user is a member. Attribute *ccAdmin* maintains the set of groups where the user is an administrator. Attribute *orgAdmin* specifies whether or not the user is an administrator of the organization.

**Object Attributes:** Similar to users, an object classification is also represented by three attributes: *hierclearanceOfObject*, *campcategroyOfObject* and *origin*. They represent an object's security level, security categories and the group or Org where the object was created respectively. The *versions* attribute maintains the set of existing versions of an object.

**Subject Attributes:** Attribute *owner* represents the user who created the subject. A subject is defined as either read-only (RO) or read-write (RW) by the *type* attribute. A subject's clearance is also represented by three components: *hierclearanceOfSubject*, *campcategroyOfSubject* and *belongsTo*. The first two represent security level and security category. The last one is necessary only for a RW subject and it represents Org or the cc in which the subject was created.

**Object Version Attributes:** Each object can have a number of versions and each version may be shared by one or more groups. The attributes $hierclearanceOfVersion_o$ and $campcategroyOfVersion_o$ represent security level and security category of a version of object o. Note that, these two attributes are same for all versions of an object. (More generally, these could be allowed to vary by version so long as derivative version labels go up in one or both components.) $vMember_o$ lists entities (i.e., groups and/or Org) that share this version.

### C. Administrative Model:

In LCC, a collaboration category (cc), representing a collaboration group, might utilize several features of group-centric secure information sharing concepts. Different semantics of four core group operations (user Join and Leave and object Add and Remove) have been proposed in [7]. For simplicity, in this model, we confine the authorization semantics to Liberal Join and Strict Leave for users and Liberal Add and Strict Remove for objects. This is the typical semantics used for groups in traditional access control systems. Note that any variation of these semantics will only affect the authorizations of Read and Update operations leaving the rest of the specification essentially intact. Table III formally specifies a set of administrative operations. The first column specifies the operation that is to be performed. The second column specifies the authorization queries that need to be satisfied to authorize the operation. Attributes and sets that will be updated after an authorized operation are listed in the third column, with the "$\prime$" symbol indicating the value after the update. These administrative operations are discussed below.

- **Establish** a collaboration category: Administrative user u establishes a collaboration category (cc) and becomes its admin. The cc is added to the set of existing collaboration categories (CC), new labels are created for cc and the dominance relation ($\succeq$) and join operator ($\oplus$) are recalculated using formulas in Table I-B.
- **Add_Clearance** to a true insider: The cc admin u1 gives a clearance cc to a true insider u2 by including cc in *uCC* of u2.
- **Remove_Clearance** from a true insider: The cc admin u1 revokes the clearance from a true insider u2 by removing cc from *uCC* of u2.
- **Join_Outsider** to a group: Group administrator u1 can enroll an outsider u2 as an expedient insider who does not have current group membership. However, he might hold membership to other collaboration groups of the organization. In that case, u2 retains the same security level and security category for this group. Otherwise, group administrator u1 assigns an appropriate security clearance.
- **Leave_Expedient_Insider** from a group: This operation revokes cc clearance from an expedient insider. Further, it kills subjects that were created by that insider in the group cc. Note that if this results in *uCC*(u2) becoming empty then, on future join, the security clearance of this user will be set to a new value.
- **Add** an object version to a group: Admin of cc adds a version of object o from Org to cc by updating *vMember* attribute of the version accordingly.
- **Remove** an object version from a group: Admin of cc removes a version of object o from cc by removing cc from *vMember* of the version.
- **Import** a version from a cc to Org: A version v1 of an object o1 can be copied to Org from a group cc by cc admin. This operation can only be performed on an object o1 that is natively created in cc, whereby *origin*(o1)=cc. The cc administrator copies the object version v1 of o1 to a new version v2 of o2 where *origin*(o2)=Org. Note that, security level and security category of o1 and o2 should be equal for a successful import.
- **Merge** a version from a cc to Org: This operation merges back a new version of an object from a cc to Org where the object was created in Org. If the operation is successful, Org is included to *vMember* of the version of object o.
- **Disband** a collaboration category: A cc admin disbands the collaboration category. Prior to Disband, the group

TABLE III
ADMINISTRATIVE MODEL

| Operation | Authorization Query | Updates |
|---|---|---|
| **Establish**(u, cc)<br>/*Admin user u establishes new collaboration category cc*/ | $u \in U \wedge cc \notin CC \wedge$ orgAdmin(u)=True | $ccAdmin'(u) = ccAdmin(u) \cup \{cc\}$<br>$CC' = CC \cup \{cc\}$<br>$SL' = \{(L \times 2^C) \times (CC' \cup \{Org\})\} \cup \{SysHigh , SysLow\}$<br>$\succeq'$ and $\oplus'$ are recalculated using formulas in Table I-B |
| **Add_Clearance**(u1,u2,cc)<br>/*Admin u1 grants cc clearance to a true insider u2*/ | $u1 \in U \wedge u2 \in U \wedge cc \in CC \wedge$ $cc \in ccAdmin(u1) \wedge$ $uType(u2) = Insider \wedge cc \notin uCC(u2)$ | $uCC'(u2) = uCC(u2) \cup \{cc\}$ |
| **Remove_Clearance**(u1,u2,cc)<br>/*Admin u1 revokes cc clearance from a true insider u2*/ | $u1 \in U \wedge u2 \in U \wedge cc \in CC \wedge$ $cc \in ccAdmin(u1) \wedge cc \in uCC(u2)$ $\wedge uType(u2) = Insider$ | $uCC'(u2) = uCC(u2) - \{cc\}$<br>$S' = S$<br>***forall*** $s \in S$<br>***if*** $owner(s) = u2 \wedge belongsTo(s) = cc$<br>***then*** $S' = S' - \{s\}$<br>/*Kill subjects in cc those are owned by the respective insider*/ |
| **Join_Outsider**(u1,u2,cc,sl,cp)<br>/*Admin u1 grants cc clearance to an outsider u2*/ | $u1 \in U \wedge u2 \in U \wedge cc \in CC \wedge$ $cc \in ccAdmin(u1) \wedge cc \notin uCC(u2)$ $\wedge uType(u2)=$ Outsider $\wedge$ $s1 \in L \wedge cp \subseteq C$ | $uType'(u2) =$ Expedient_Insider<br>***if*** $uCC(u2) = \emptyset$ ***then*** $hierclearanceOfUser'(u2) = sl$<br>$compcategoryOfUser'(u2) = cp$<br>$uCC'(u2) = uCC(u2) \cup \{cc\}$ |
| **Leave_Expedient_Insider**(u1,u2,cc)<br>/*Admin u1 revokes cc clearance from an expedient insider u2*/ | $u1 \in U \wedge u2 \in U \wedge cc \in CC \wedge$ $cc \in ccAdmin(u1) \wedge cc \in uCC(u2)$ $\wedge uType(u2) =$ Expedient_Insider | $uCC'(u2) = uCC(u2) - \{cc\}$<br>$S' = S$<br>***forall*** $s \in S$<br>***if*** $owner(s) = u2 \wedge belongsTo(s) = cc$<br>***then*** $S' = S' - \{s\}$<br>/*Kill subjects in cc those are owned by the respective insider*/<br>***if*** $uCC'(u2) = \emptyset$ ***then*** $hierclearanceOfUser' =$<br>$hierclearanceOfUser-\{u2 \rightarrow hierclearanceOfUser(u2)\}$<br>$compcategoryOfUser' =$<br>$compcategoryOfUser-\{u2 \rightarrow compcategoryOfUser(u2)\}$<br>$uType'(u2) =$ Outsider |
| **Add**(u,o,v,cc)<br>/*Admin u adds version v of object o from Org to cc*/ | $u \in U \wedge cc \in CC \wedge o \in O \wedge$ $v \in versions(o) \wedge cc \in ccAdmin(u)$ $\wedge cc \notin vMember_o(v)$ | $vMember'_o(v) = vMember_o(v) \cup \{cc\}$ |
| **Remove**(u,o,v,cc)<br>/*Admin u removes version v of object o from cc*/ | $u \in U \wedge cc \in CC \wedge o \in O \wedge$ $v \in versions(o) \wedge cc \in ccAdmin(u)$ $\wedge cc \in vMember_o(v)$ | $vMember'_o(v) = vMember_o(v) - \{cc\}$ |
| **Import**(u,o1,v1,o2,cc)<br>/*Admin u imports version v1 of object o1 to new version v2 of object o2 in Org*/ | $u \in U \wedge cc \in CC \wedge v1 \in versions(o)$ $\wedge o1,o2 \in O \wedge origin(o2) = Org \wedge$ $cc \in ccAdmin(u) \wedge origin(o1) = cc$ $\wedge hierclassificationOfObject(o1) =$ $hierclassificationOfObject(o2) \wedge$ $compcategoryOfObject(o2) =$ $compcategoryOfObject(o1)$ | $versions'(o2) = versions(o2) \cup \{v2\}$<br>/*v2 is newly created version id*/<br>$vMember'_{o2}(v2) = \{Org\}$<br>$hierclassificationOfVersion'_{o2}(v2) =$<br>$hierclassificationOfObject(o2)$<br>$compcategoryOfVersion'_{o2}(v2) = compcategoryOfObject(o2)$ |
| **Merge**(u,o,v,cc)<br>/*Admin u merges version v of object o from cc to Org*/ | $u \in U \wedge cc \in CC \wedge o \in O \wedge$ $v \in versions(o) \wedge cc \in ccAdmin(u)$ $\wedge cc \in vMember_o(v) \wedge$ $origin(o) = Org \wedge v \in versions(o)$ | $vMember'_o(v) = vMember_o(v) \cup \{Org\}$ |
| **Disband**(u, cc)<br>/*Admin u disbands a collaboration category cc*/ | $u \in U \wedge cc \in CC \wedge$ $cc \in ccAdmin(u)$ | $uCC' = uCC$, $ccAdmin' = ccAdmin$, $O' = O$<br>$S' = S$, $vMember' = vMember$<br>***forall*** $u1 \in U$<br>***if*** $cc \in uCC(u1)$ ***then*** $uCC'(u1) = uCC'(u1) - \{cc\}$<br>***if*** $cc \in ccAdmin(u1)$ ***then*** $ccAdmin'(u1) = ccAdmin'(u1) - \{cc\}$<br>***forall*** $o \in O$<br>***if*** $origin(o) = cc$ ***then*** $O' = O' - \{o\}$<br>***forall*** $o \in O$ ***and forall*** $v \in versions(o)$<br>***if*** $cc \in vMember_o(v)$ ***then*** $vMember'_o(v) = vMember'_o(v) - \{cc\}$<br>***forall*** $s \in S$<br>***if*** $belongsTo(s)=cc$ ***then*** $S' = S' - \{s\}$<br>$CC' = CC - \{cc\}$<br>$SL' = \{(L \times 2^C) \times (CC' \cup \{Org\})\} \cup \{SysHigh , SysLow\}$<br>$\succeq'$ and $\oplus'$ are recalculated using formulas in Table I-B |

administrator need to Merge and Import necessary objects from the group to the organization. After Disband, the corresponding attributes of every true insider, expedient insider and object version are updated accordingly. Every subject executing in the group is killed and and every object with *origin* cc is deleted. Finally, cc is removed from CC, relevant labels are also removed from SL and $\succeq$ and $\oplus$ recalculated using formulas in Table I-B.

### D. Operational Model:

We specify necessary operations for user activities in this lattice setup. A user can create subjects and exercise privileges in a cc or Org. Only true insiders can create a subject in Org. A subject inherits the same or lower security clearance from the user. A user may create multiple subjects, however, a subject is owned by only one user. In LCC, an insider (both true and expedient) can have multiple clearances. For the purpose of aggregation, a user can create a read-only subject that reads an object version from any group and/or the Org (for true insiders) to which the user has a clearance. A read-only subject is unable to write. In order to write a user must create a read-write subject which is confined to write either only in a single group or only in the Org, depending on where it was created. Additionally the scope of a subject's read operation is restricted by simple security property and write operation is restricted by strict star-property (i.e., write can only be at the same label as the subject's label).

- **CreateRWInCC** or **CreateRWInOrg** subject: A user can create read-write subject in Org or in a cc where she is a member. In order to create a read-write subjects in Org the user needs to be a true insider. The user also determines the security clearance of the subject which is equal or lower than the user's own clearance. The *type* attribute of the subject is RW.
- **CreateRO** subject: A user can create a read-only subject. The user determines the security clearance of the subject which is equal or lower than the user's own clearance. The subject's *type* attribute is RO.
- **Read** an object version: Read is one of the most critical operations of the system. In order to aggregate information, a RO subject can read any object version from every cc and/or Org in which the subject's owner is a member, while a RW subject is restricted to read only in the group or Org in which it is created. By the simple security property, in order to authorize a read the clearance of a subject must be higher or equal to the classification of the target object version.
- **Update** an object version: A read-write subject can update a version of an object using this function. Note that each update creates a new version of the object and the version inherits the security classification from the previous version. Our model enforces strict star-property in which objects classification should be equal to the subjects clearance for a successful Update.
- **Create** an object version: Using this function a subject can create a new object and the newly created object gets the same security clearance as the subject.
- **Kill** a subject: A user can kill her subject by this command. It could be performed either by the owner of the subject or the cc admin.

## VII. Equivalence of GEI and LCC

In this section, we aim to show that our developed LBAC with collaborative compartment (LCC) model is equivalent to a previously developed model for a group-centric collaboration with expedient insiders (GEI) [4]. We utilize the framework of Tripunitara and Li [11] for comparing expressive power of access control models to show this equivalence. The proof is non-trivial and, due to space limitations, full details are provided elsewhere [1]. Here, we provide an outline of this proof to show equivalence.

Tripunitara and Li have convincingly argued that equivalency of access control models can be proved by comparing their expressive power based on state matching reductions (a simulation process) that preserve security properties. Thus, we show the formal proof of equivalence of LCC and GEI by proving that there exists a state matching reduction from GEI to LCC and vice versa.

Tripunitara and Li define an access control model as an access control scheme represented as a 4-tuple $\langle \Gamma, Q, \vdash, \Psi \rangle$. $\Gamma$ is a set of states where each element contains the necessary information to decide access control on that particular state. Q is a set of queries and $\vdash$: $\Gamma \times Q \rightarrow \{true, false\}$ is the entailment relation that specifies whether a query $q \in Q$ is true or false in a particular state $\gamma \in \Gamma$. Finally, $\Psi$ is a set of state transition rules where a particular $\psi \in \Psi$ determines how the state changes.

According to [11], given two access control schemes A=$\langle \Gamma^A, Q^A, \vdash^A, \Psi^A \rangle$ and B=$\langle \Gamma^B, Q^B, \vdash^B, \Psi^B \rangle$ a mapping from A to B is defined as a function $\sigma$ that maps each pair $\langle \gamma^A, \psi^A \rangle$ to a pair $\langle \gamma^B, \psi^B \rangle$, and each query $q^A$ to $q^B$. Formally a mapping is represented as $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \rightarrow (\Gamma^B \times \Psi^B) \cup Q^B$. States $\gamma^A$ and $\gamma^B$ are said to be equivalent under the mapping $\sigma$ when for every $q^A \in Q^A$, $\gamma^A \vdash^A q^A$ if and only if $\gamma^B \vdash^B \sigma(q^A)$. This leads up to the definition for a state matching reduction as follows.

**Definition 1.** *(State Matching Reduction) Given two schemes A and B, a mapping $\sigma$ from A to B is a state matching reduction if for every $\gamma^A \in \Gamma^A$ and every $\psi^A \in \Psi^A$, we have the following two properties where $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$.*

1) *For every state $\gamma_1^A$ in scheme A such that $\gamma^A \xrightarrow{*}_{\psi^A} \gamma_1^A$, there exists $\gamma_1^B$ in scheme B such that $\gamma^B \xrightarrow{*}_{\psi^B} \gamma_1^B$ and $\gamma_1^A$ and $\gamma_1^B$ are equivalent.*
2) *For every state $\gamma_1^B$ in scheme B such that $\gamma^B \xrightarrow{*}_{\psi^B} \gamma_1^B$ there exists $\gamma_1^A$ in scheme A such that $\gamma^A \xrightarrow{*}_{\psi^A} \gamma_1^A$, and $\gamma_1^A$ and $\gamma_1^B$ are equivalent.*

The significance of state-matching reductions is expressed in Theorem 1 of [11] which asserts that: Given two schemes A and B, a mapping $\sigma$ from A to B is strongly security-preserving (in a precise formal sense) if and only if $\sigma$ is a state-matching reduction. Two schemes are said to be equivalent if

TABLE IV
OPERATIONAL MODEL

| Operation | Authorization Query | Updates |
|---|---|---|
| **CreateRWInCG**(u,s,cc,sl,cp) /*User u creates read-write subject s in a group cc*/ | u $\in$ U $\wedge$ s $\notin$ S $\wedge$ cc $\in$ uCC(u) $\wedge$ sl $\preceq$ hierclearanceOfUser(u) $\wedge$ cp $\subseteq$ compcategoryOfUser(u) | owner$'$(s) = u<br>hierclearanceOfSubject$'$(s) = sl<br>belongsTo$'$(s) = cc<br>compcategoryOfSubject$'$(u) = cp<br>type$'$(s) = RW<br>S$'$ = S $\cup$ {s} |
| **CreateRWInOrg**(u,s,sl,cp) /*Only true insider creates read-write subject in Org*/ | u $\in$ U $\wedge$ s $\notin$ S $\wedge$ utype(u) = Insider $\wedge$ sl $\preceq$ hierclearanceOfUser(u) $\wedge$ cp $\subseteq$ compcategoryOfUser(u) | owner$'$(s) = u<br>hierclearanceOfSubject$'$(s) = sl<br>belongsTo$'$(s) = Org<br>compcategoryOfSubject$'$(u) = cp<br>type$'$ (s) = RW<br>S$'$ = S $\cup$ {s} |
| **CreateRO**(u,s,sl,cp) /*User u creates read-only subject s*/ | u $\in$ U $\wedge$ s $\notin$ S $\wedge$ sl $\preceq$ hierclearanceOfUser(u) $\wedge$ cp $\subseteq$ compcategoryOfUser(u) | owner$'$(s) = u<br>hierclearanceOfSubject$'$(s) = sl<br>type$'$(s) = RO<br>compcategoryOfSubject$'$(u) = cp<br>S$'$ = S $\cup$ {s} |
| **Read**(s,o,v) /*Subject s reads the version v of object o*/ | s $\in$ S $\wedge$ o $\in$ O $\wedge$ v $\in$ versions(o) $\wedge$ hierclearanceOfSubject(s) $\succeq$ hierclassificationOfVersion$_o$(v) $\wedge$ compcategoryOfSubject(s) $\supseteq$ compcategoryOfVersion$_o$(v)$\wedge$ (type(s) = RO $\wedge$ ((uCC(owner(s)) $\cap$ vMember$_o$(v)$\neq \phi$) $\vee$ (utype(owner(s)) = Insider $\wedge$ {Org} $\in$ vMember$_o$(v)))) $\vee$ (type(s) = RW $\wedge$ (belongsTo(s) $\in$ vMember$_o$(v)))) | None |
| **Update**(s,o,v) /*Subject s updates the version v of object o. This function returns updated version v1*/ | s $\in$ S $\wedge$ o $\in$ O $\wedge$ v $\in$ versions(o) $\wedge$ hierclearanceOfSubject(s) = hierclassificationOfVersion$_o$(v) $\wedge$ compcategoryOfSubject(s) = compcategoryOfVersion$_o$(v)$\wedge$ (type(s) = RW $\wedge$ belongsTo(s) $\in$ vMember$_o$(v)) | versions$'$(o) = versions(o) $\cup$ {v1}<br>/*v1 is newly created version id*/<br>vMember$'_o$(v1) = vMember$_o$(v1) $\cup$ {cc}<br>hierclassificationOfVersion$'_o$(v1) = hierclassificationOfVersion$_o$(v)<br>compcategoryOfVersion$'_o$(v1) = compcategoryOfVersion$_o$(v) |
| **Create**(s,o) /*Subject s creates version v of object o. This function returns newly created version v*/ | s $\in$ S $\wedge$ o $\notin$ O $\wedge$ type(s)=RW | O$'$ = O $\cup$ {o}<br>versions$'$(o) = {v}<br>/*v is newly created version id*/<br>vMember$'_o$(v) = {belongsTo(s)}<br>origin$'$(o) = belongsTo(s)<br>hierclassificatonOfObject$'$(o) = hierclearanceOfSubject(s)<br>compcategoryOfObject$'$(o) = compcatgoryOfSubject(s)<br>hierclassificatonOfVersion$'_o$(v) = hierclearanceOfSubject(s)<br>compcategoryOfVersion$'_o$(v) = compcatgoryOfSubject(s) |
| **Kill**(u,s) /*User u kills subject s*/ | u $\in$ U $\wedge$ s $\in$ S $\wedge$ (owner(s) = u $\vee$ belongsTo(s) $\in$ ccAdmin(u)) | owner$'$ = owner - {s $\rightarrow$ owner(s)}<br>type$'$ = type - {s $\rightarrow$ type(s)}<br>hierclearanceOfSubject$'$ = hierclearanceOfSubject - {s $\rightarrow$ hierclearanceOfSubject(s)}<br>compcategoryOfSubject$'$ = compcategoryOfSubject - {s $\rightarrow$ compcategoryOfSubject(s)}<br>belongsTo$'$ = belongsTo - {s $\rightarrow$ belongsTo(s)}<br>S$'$ = S - {s} |

there is a state-matching reduction from one to the other, and vice versa.

Thus, in order to show the proof of equivalence of LCC and GEI we have to show the following:

1) Represent LCC and GEI models as LCC and GEI schemes
2) Construct a mapping $\sigma^{LCC}$ that maps LCC to GEI
3) Prove that $\sigma^{LCC}$ mapping from LCC to GEI is a state matching reduction
4) Construct a mapping $\sigma^{GEI}$ that maps GEI to LCC
5) Prove that $\sigma^{GEI}$ mapping from GEI to LCC is a state matching reduction

**The LCC scheme**

$\Gamma$ In LCC scheme, each state is characterized by the sets and attributes given in Table II.

$\Psi$ A state transition rule, $\psi$, in the LCC scheme is an operation schema of administrative or operational model given in column 1 of Table III and IV respectively. Each operation takes a sequence of parameter and on successful execution it updates relevant state elements and causes a transition to

another state (shown in column 3 of Table III and IV) .

$Q$ Authorization queries given in column 2 of Table III and IV are the elements of Q in LCC scheme.

$\vdash$ Let q be an authorization query where q $\in$ Q. Then, given a state $\gamma$, $\gamma \vdash q$ if and only if q is true in state $\gamma$ and $\gamma \nvdash q$ otherwise.

**The GEI scheme**

$\Gamma$ Similar to the LCC scheme, elements of a state $\gamma \in \Gamma$ could be defined for GEI scheme. The complete set of elements of each state $\gamma \in \Gamma$ of GEI scheme is given in Table I of [1].

$\Psi$ State transition rules of GEI scheme are the operations of administrative and operational models given in [4]. Column 1 of Table II and III given in [1] specifies those operations.

$Q$ Authorization queries given in column 2 of Table II and III in [1] are the elements of Q in GEI scheme.

$\vdash$ Let q be an authorization query where q $\in$ Q. Then, given a state $\gamma$, $\gamma \vdash q$ if and only if q is true in state $\gamma$ and $\gamma \nvdash q$ otherwise.

**Construction of $\sigma^{LCC}$**

In [1], we construct a mapping $\sigma^{LCC}$ from LCC to GEI such that $\sigma^{LCC} : (\Gamma^{LCC} \times \Psi^{LCC}) \cup Q^{LCC} \rightarrow (\Gamma^{GEI} \times \Psi^{GEI}) \cup Q^{GEI}$. The mapping consists of three parts:

- $\sigma$ mapping of each $\gamma^{LCC} \in \Gamma^{LCC}$ to $\gamma^{GEI} \in \gamma^{GEI}$ where $\Gamma^{LCC}$ and $\Gamma^{GEI}$ are state elements given in Table I and Table IV respectively in [1].
- $\sigma$ mapping of each $\psi^{LCC} \in \Psi^{LCC}$ to $\psi^{GEI} \in \Psi^{GEI}$ where $\Psi^{LCC}$ and $\Psi^{GEI}$ are set of state transition rules given in column 1 of Tables II and III and Tables V and VI respectively in [1].
- $\sigma$ mapping of each $q^{LCC} \in Q^{LCC}$ to $q^{GEI} \in Q^{GEI}$ where $Q^{LCC}$ and $Q^{GEI}$ are authorization queries given in column 2 of Tables II and III and Tables V and VI in [1].

**Theorem 1.** *There exists a state matching reduction from LCC to GEI.*

***Proof Sketch:*** Section V of [1] shows proof that the mapping $\sigma^{LCC}$ from LCC to GEI satisfies properties 1 and 2 in Definition 1. Therefore, the mapping $\sigma^{LCC}$ from LCC to GEI is a state matching reduction.

**Construction of $\sigma^{GEI}$**

Similar to the mapping $\sigma^{LCC}$, section VI of [1] constructs a mapping $\sigma^{GEI}$ from GEI to LCC such that $\sigma^{GEI} : (\Gamma^{GEI} \times \Psi^{GEI}) \cup Q^{GEI} \rightarrow (\Gamma^{LCC} \times \Psi^{LCC}) \cup Q^{LCC}$.

**Theorem 2.** *There exists a state matching reduction from GEI to LCC.*

***Proof Sketch:*** Section VII of [1] shows proof that the mapping $\sigma^{GEI}$ from GEI to LCC satisfies properties 1 and 2 in Definition 1. Therefore, the mapping $\sigma^{GEI}$ is a state matching reduction.

**Theorem 3.** *The LCC and GEI scheme are equivalent.*

***Proof Sketch:*** This theorem follows from the proof of theorems 1 and 2.

The proof of equivalence establishes that either strategy (GEI or LCC) for group-centric collaboration with expedient insiders in a multilevel system can be used, since the security properties of information flow are preserved regardless. In LCC modifying the whole lattice structure for each collaboration group might be disruptive and cumbersome for the organization, thus, implementing the GEI duplicated lattice might be the best approach. On the other hand LCC may be easier to adapt from existing LBAC implementations.

## VIII. Conclusion

Our goal is to define a new lattice construction process for group-centric organizational collaboration with expedient insiders. To this end, we show that a new collaboration group introduces a collaboration category that eventually creates security labels solely for that collaboration group. Thus, position of an expedient insider could be easily determined based on her involvement in a particular collaboration group and organizations could be more selective in sharing information with them. We also propose authorization model consisting of separate administrative and operational components and show the equivalence of this model with a prior organizational collaboration model with expedient insiders proposed in [4].

**Acknowledgement.** This work is partially supported by an AFOSR MURI grant.

## References

[1] T. Ahmed, R. Sandhu, K. Bijon, and R. Krishnan. Equivalence of group-centric collaboration with expedient insiders (GEI) and LBAC with collaborative compartments (LCC). *CS-TR-2012-010, UTSA*, 2012.

[2] D. E. Bell and L. J. Lapadula. Secure computer systems: Mathematical foundations. *MITRE Corporation*, 1976.

[3] K. Biba. Integrity considerations for secure computer systems. *MITRE Corporation*, 1977.

[4] K. Bijon, R. Sandhu, and R. Krishnan. A group-centric model for collaboration with expedient insiders in multilevel systems. In *International Symp. on Security in Collaboration Technologies and Systems*, 2012.

[5] D. E. Denning. A lattice model of secure information flow. *Communication Of The ACM*, 19(5):236–243, May 1976.

[6] J. H. Jafarian and M. Amini. CAMAC: A Context-Aware Mandatory Access Control Model. *ISeCure*, Jan. 2009.

[7] R. Krishnan, J. Niu, R. Sandhu, and W. H. Winsborough. Group-centric secure information-sharing models for isolated groups. *TISSEC*, 2011.

[8] J. Park and R. Sandhu. The UCON$_{ABC}$ usage control model. *ACM Trans. on Information and Systems Security*, 7:128–174, 2004.

[9] I. Ray and M. Kumar. Towards a location-based mandatory access control model. *Computers & Security*, 25, 2006.

[10] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26:9–19, 1993.

[11] M. V. Tripunitara and N. Li. Comparing the expressive power of access control models. In *ACM CCS*. ACM, 2004.