A Lattice Interpretation of Group-Centric Collaboration with Expedient Insiders

Khalid Zaman Bijon, Tahmina Ahmed, Ravi Sandhu, Ram Krishnan
Institute for Cyber Security
University of Texas at San Antonio

Expedient Insiders

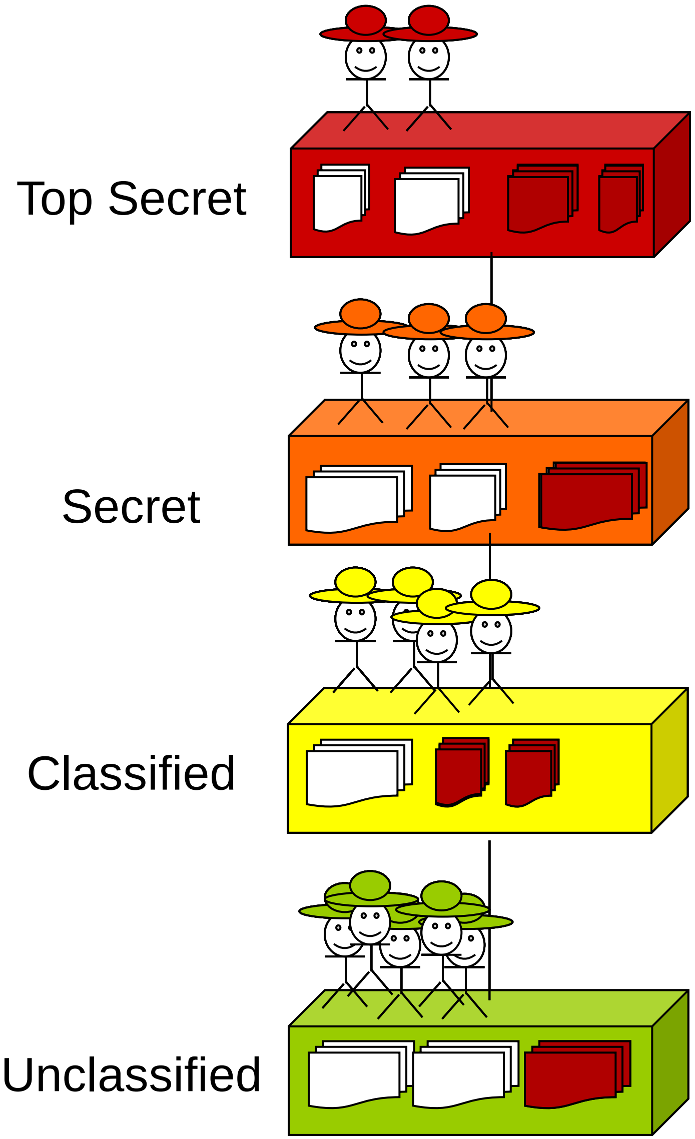
- Who are expedient insiders?
  - Any outside Collaborators, i.e. Domain specialists, cyber-security experts, etc.

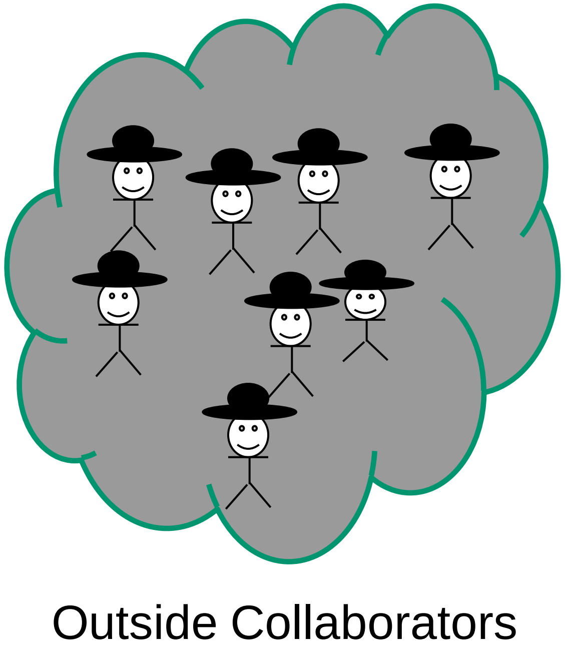
- Difference with respect to true insiders
  - Transient rather than persistent
  - Information sharing is based on need-to-consult basis
  - Less commitment than long time employees

What are the Challenges?

1. *Information selection for collaboration*
2. *Restrict unnecessary access*
3. *Import Results*

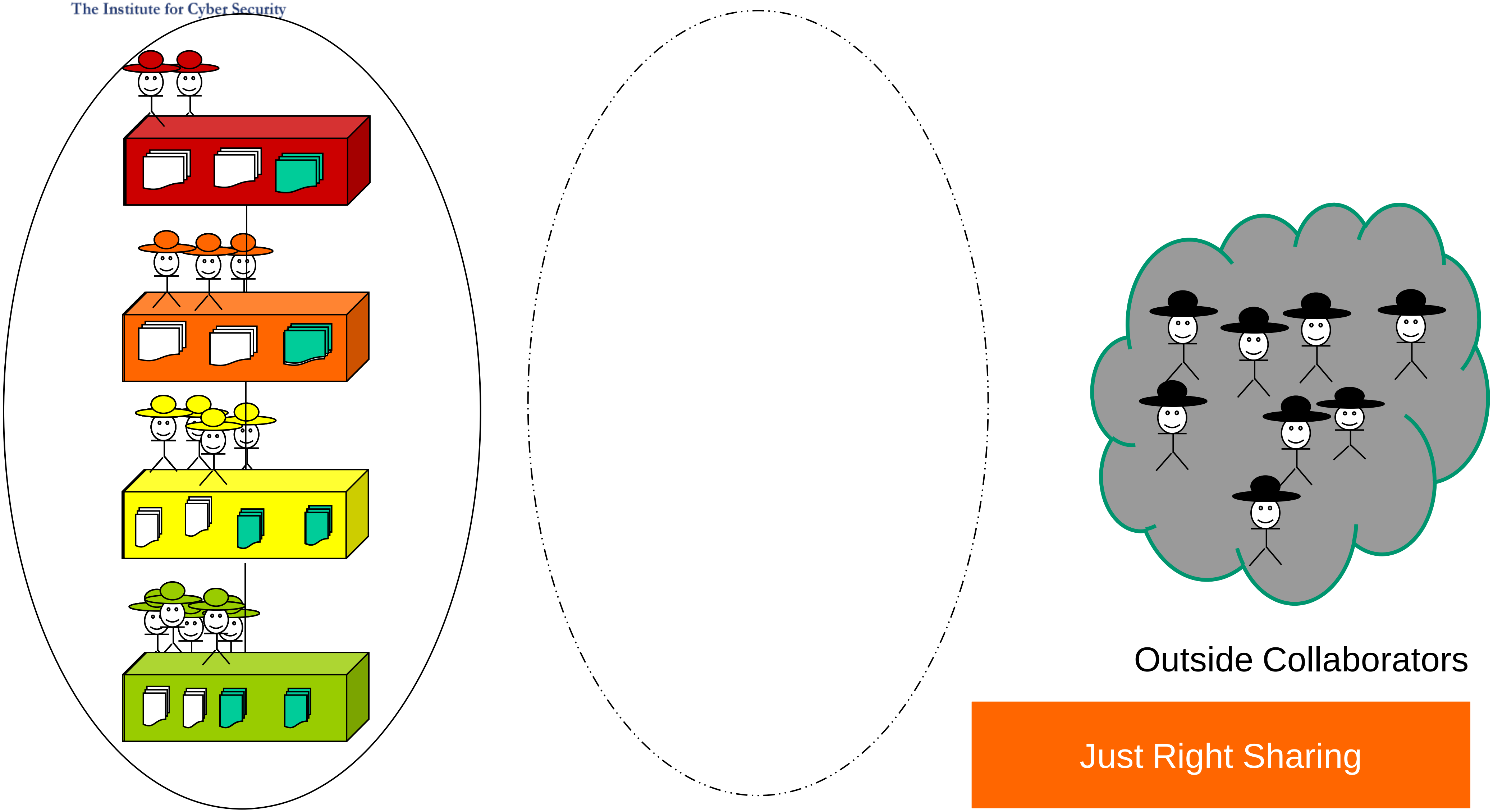
Collaboration with Expedient Insiders in Traditional LBAC





Sharing more information than necessary
Open to more true-insiders than necessary

Group Centric Collaboration with Expedient Insiders (GEI)[1]



Outside Collaborators

Just Right Sharing

Organization

Collaboration Group with Expedient Insider

[1] K. Bijon, R. Sandhu, and R. Krishnan. A group-centric model for collaboration with expedient insiders in multilevel systems. In *Secots*, 2012.

Group Centric Collaboration with Expedient Insiders (GEI)1

- Organizations and groups maintain separate piece of lattice

- Information flow and security properties for the overall system are informally addressed

- No comparison with traditional LBAC

Motivation & Goal:

- Construct a single lattice for group-centric organizational collaboration
- Achieve all requirements of GEI as well as well-known formal security properties of a LBAC system
- Proof of equivalence with GEI

1. K. Bijon, R. Sandhu, and R. Krishnan. A group-centric model for collaboration with expedient insiders in multilevel systems. In *Secots*, 2012.

Traditional Lattice Based Access Control (Traditional-LBAC)

- Traditional-LBAC
  - Information objects are attached with security labels.
  - Information flows on partial ordered of those security labels
  - A security label is formed by combining a security level with a subset of security categories
  - Security levels are ordered (e.g. TS>S>U>C)
  - Security categories are unordered (e.g. ProjA, ProjB)
  - A user is cleared to a
  - Users can access obje
    dominated by their sec

These security labels are not suitable for expedient insiders (i.e. too many sharing)

Need to find a way to construct security labels (solely for a collaboration purpose)

Lattice with Collaborative Compartments (LCC)

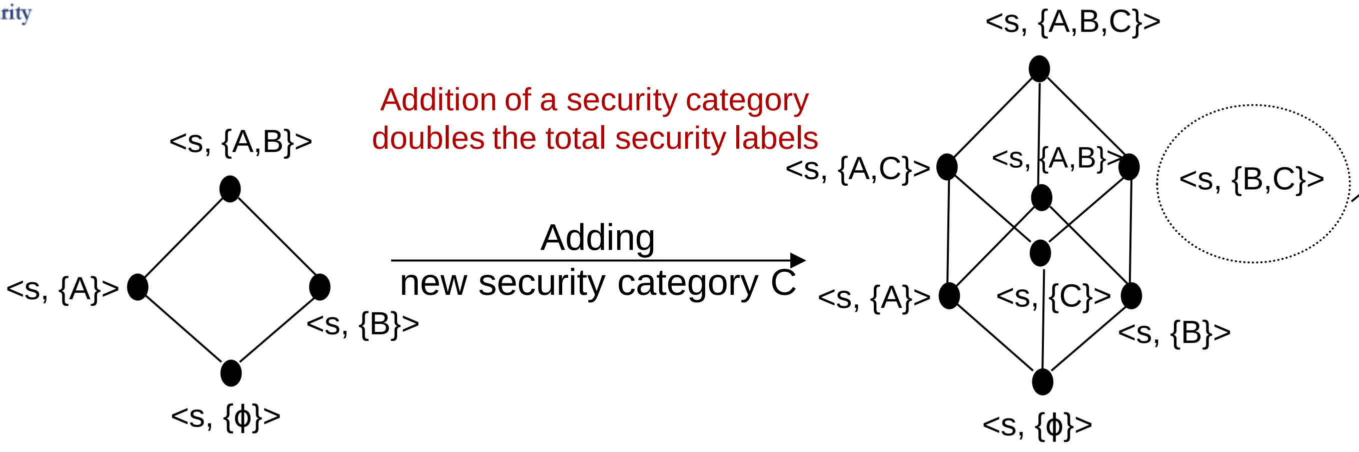
- Each collaboration group introduces a new collaboration category (cc).

- New security labels are formed for each cc in combination with the entire set of security labels of the organization (different than new traditional security categories)

- Existing lattice structure is modified accordingly (different than new traditional security categories)

- One single lattice structure is maintained for all collaboration groups and organization.

Lattice with Collaborative Compartments (LCC)



Addition of a security category doubles the total security labels

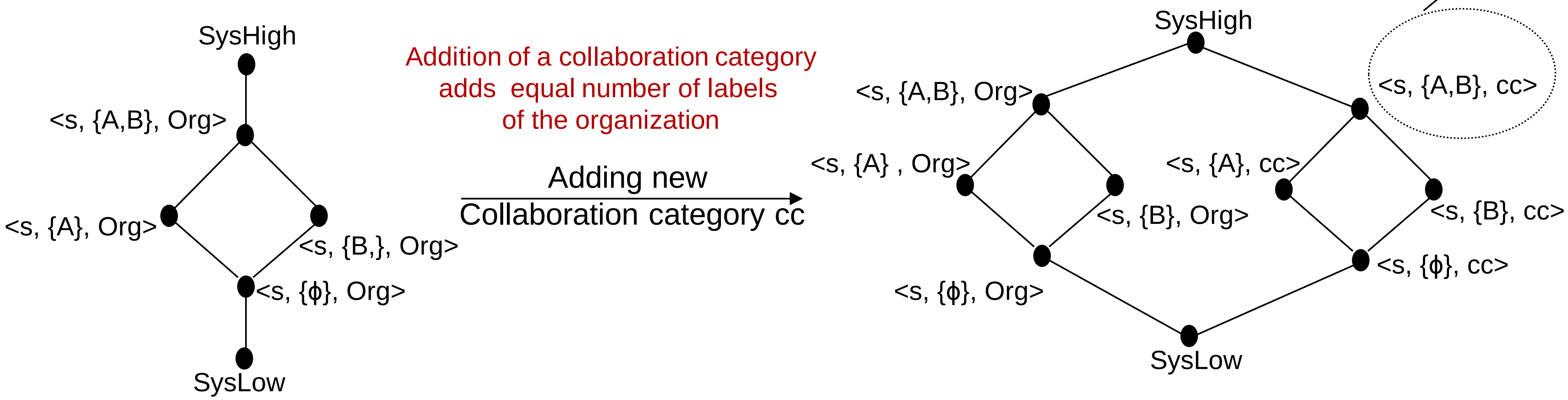
Security label Consists of security Level and categories

Present Lattice

Modified Lattice after new security category c

Security label Consists of security Level and categories and entities (org or Collaboration category)

Change of Lattice structure for adding new security category in Traditional LBAC



Addition of a collaboration category adds equal number of labels of the organization

Present Organizational Lattice without collaboration category

Modified Lattice after adding collaboration category cc

Change of Lattice structure for adding new collaboration category in LCC

Formal Definition of Lattices from components

A: Lattice with Traditional Compartments (LTC)

L: is a finite set of linearly ordered security levels

C: is a finite set of unordered categories

SL: is a finite set of security labels where

$SL = (L \times 2^{C})$

$\succeq$: is a finite dominance relation defined so that $\succeq \subseteq SL \times SL$, where

$\succeq = \{((l1,c1),(l2,c2)) \mid \wedge (l1,c1) \in SL \wedge (l2,c2) \in SL \wedge$

$l1 \succeq l2 \wedge c1 \supseteq c2\}$

$\oplus$: $SL \times SL \rightarrow SL$ is a join operator defined as

$\forall l1,l2 \in L$ and $\forall c1,c2 \in C$

$(l1,c1) \oplus (l2,c2) = (\max(l1,l2), c1 \cup c2)$

B: Lattice with Collaboration Compartments (LCC)

L: is a finite set of linearly ordered security levels

C: is a finite set of unordered categories

CC: is a finite set of unordered collaboration categories

Org, is the entity Organization, a constant

SysHigh: system high (constant label)

SysLow: system low (constant label)

SL: is a finite set of security labels where

$SL = \{(L \times 2^{C}) \times (CC \cup \{Org\})\} \cup \{SysHigh, SysLow\}$

$\succeq$: is a finite dominance relation defined so that $\succeq \subseteq SL \times SL$, where

$\succeq = \{ ((l1,c1,cc1), (l2,c2,cc2)) \mid (l1,c1,cc1) \in SL \wedge (l2,c2,cc2) \in SL$

$\wedge\ l1 \succeq l2 \wedge c1 \supseteq c2 \wedge cc1{=}cc2\}$

$\cup \{(SysHigh,x),(x,SysLow) \mid x \in SL \}$

$\oplus$: $SL \times SL \rightarrow SL$ is a join operator defined as

$\forall l1,l2 \in L$ and $\forall c1,c2 \in C$ and $\forall cc1,cc2 \in CC \cup \{Org\}$

$(l1,c1,cc1) \oplus (l2,c2,cc2) = (\max(l1,l2), c1 \cup c2, cc1)$, if $cc1{=}cc2$

$(l1,c1,cc1) \oplus (l2,c2,cc2) {=} SysHigh$, if $cc1 \neq cc2$

$\forall l \in L$ and $\forall c \in C$ and $\forall cc \in CC \cup \{Org\}$

$(l,c,cc) \oplus SysHigh = SysHigh$, $SysHigh \oplus (l,c,cc) = SysHigh$

$(l,c,cc) \oplus SysLow = (l,c,cc)$, $SysLow \oplus (l,c,cc) = (l,c,cc)$

$SysHigh \oplus SysHigh = SysHigh$, $SysHigh \oplus SysLow = SysHigh$

$SysLow \oplus SysHigh = SysHigh$, $SysLow \oplus SysLow = SysLow$

True Insiders Vs Expedient Insiders In LCC

True Insiders	Expedient Insiders
1. Unlike traditional LBAC, users might have multiple clearances in this system. However, hierarchical clearance is always same for each user.	
2. True insiders might get the clearance to both organization or collaboration categories	2. Expedient insiders cannot get clearance to organization.
3. Can access all objects that - Satisfy dominance relation - in organization or joined collaboration categories	3. Can access all objects that - Satisfy dominance relation - in joined collaboration categories only

Object Version Model in LCC

- Each object can have multiple version. (necessary for sharing information among different collaboration groups and org)

- Security classification of an object and its versions could be different based on which groups or org it is belongs to. (However, hierarchical classification of them are always same).

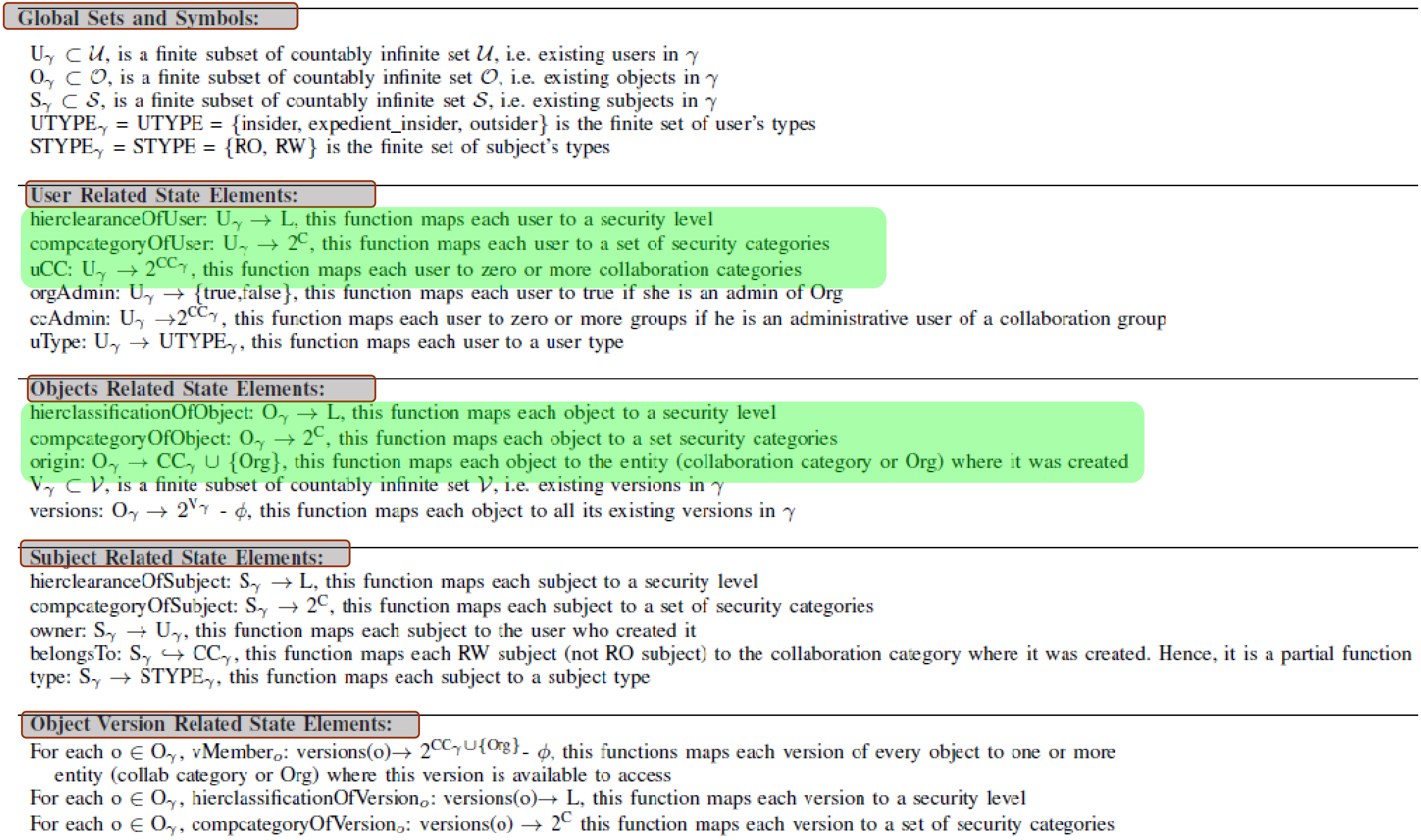
- Any update to an object version creates a new version of that object.

- Sharing an object to a group also creates a new object version

Read-Only Vs Read-Write Subject

<table>
<tr><th>Read Only</th><th>Read Write</th></tr>
<tr><td>1. Can not write, read is restricted by BLP simple security property</td><td>1. Can read and write, however, write is restricted by BLP strict * property</td></tr>
<tr><td colspan="2">2. User determines the security clearance (<= user’s clearance)</td></tr>
<tr><td colspan="2">3. Unlike users, a subject can have only one clearance.</td></tr>
<tr><td>4. Can read objects from any compartments where the user has clearance</td><td>4. restricted within the same collaboration category it was created</td></tr>
<tr><td>5. Read operation does not create new object versions</td><td>5. Only a write operation always create a new version of the respective object, however, does not change the classification of the version</td></tr>
</table>

Attribute Specification



Global Sets and Symbols:

$U_\gamma \subset \mathcal{U}$, is a finite subset of countably infinite set $\mathcal{U}$, i.e. existing users in γ
$O_\gamma \subset \mathcal{O}$, is a finite subset of countably infinite set $\mathcal{O}$, i.e. existing objects in γ
$S_\gamma \subset \mathcal{S}$, is a finite subset of countably infinite set $\mathcal{S}$, i.e. existing subjects in γ
$UTYPE_\gamma$ = 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 γ
versions: $O_\gamma \rightarrow 2^{V_\gamma} - \phi$, this function maps each object to all its existing versions in γ

Subject Related State Elements:

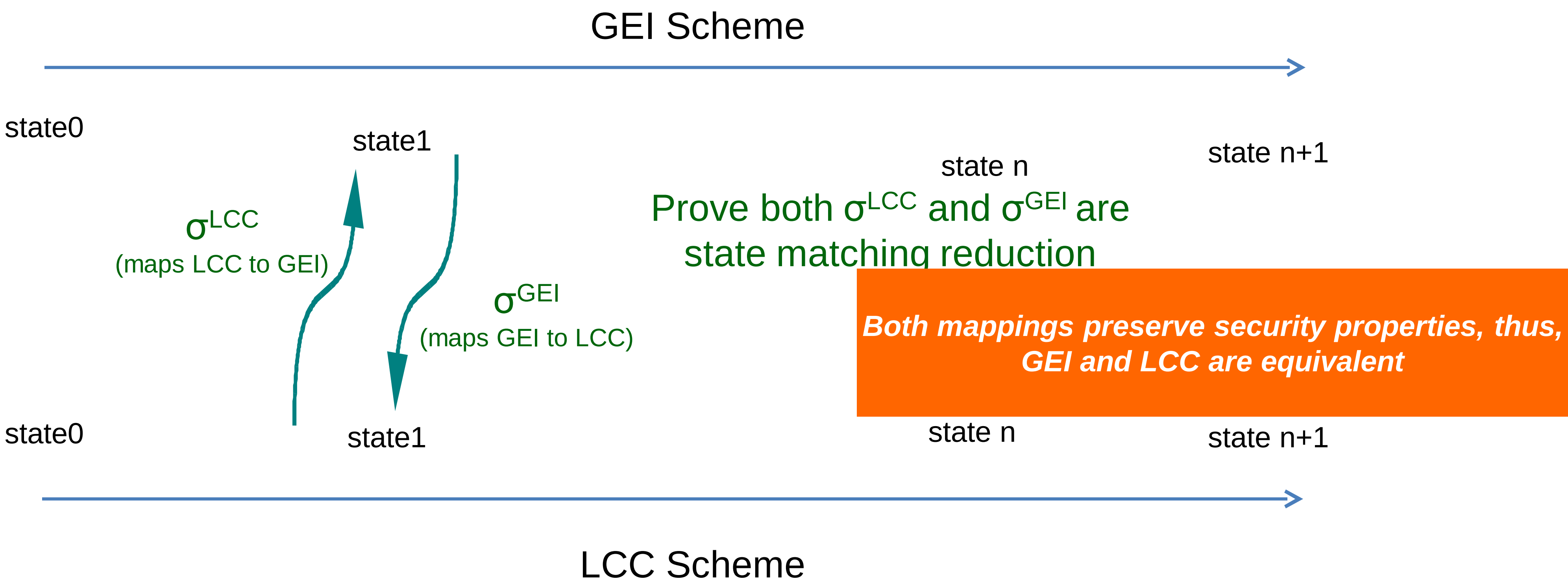
hierclearanceOfSubject: $S_\gamma \rightarrow L$, this function maps each subject to a security level
compcategoryOfSubject: $S_\gamma \rightarrow 2^C$, this function maps each subject to a set of security categories
owner: $S_\gamma \rightarrow U_\gamma$, this function maps each subject to the user 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

Proof of Equivalence of GEI[1] and LCC

- Developed operations for administrative and operational management for LCC
  - Operation name, authorization queries and updates of attributes
- Show proof of equivalence of GEI and LCC using method in Tripunitara and Li[2]



GEI Scheme

state0

state1

state n

state n+1

σ^{LCC}

(maps LCC to GEI)

σ^{GEI}

(maps GEI to LCC)

Prove both σ^{LCC} and σ^{GEI} are state matching reduction

Both mappings preserve security properties, thus, GEI and LCC are equivalent

state0

state1

state n

state n+1

LCC Scheme

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

Conclusion

- A new lattice construction process for group centric organizational collaboration with expedient insiders
  - Introduces collaboration category
  - separate compartments for organization and each collaboration groups.
  - Easy to identify the position of an expedient insider within the lattice

- Proof of Equivalence formally shows GEI also preserves the well-known security properties of a LBAC system.

Thank You ☺