Group-Centric
Secure Information Sharing Models

Ram Krishnan
PhD Candidate
Dissertation Directors:
Dr. Ravi Sandhu and Dr. Daniel Menascé

Dissertation Defense
Nov 25th 2009
Presentation Outline

- Introduction
- Policy Models for Group-Centric Secure Information Sharing (g-SIS)
- Enforcement Models for g-SIS
- Implementation Model for g-SIS
Introduction and Motivation

- Secure Information Sharing
  - Share *but* protect
  - A fundamental problem in cyber security

- Dissemination-Centric Sharing
  - Dissemination chain with “sticky” policies on objects
  - E.g. ORCON, DRM, ERM, XrML, ODRL, etc.

- Query-Centric Sharing
  - Queries wrt a particular dataset
  - More generally, addresses de-aggregation/inference problem
Introduction and Motivation (contd)

- **Group-Centric Sharing**
  - Sharing for a specific purpose or mission
    - E.g. Collaboration in joint product design, merger and acquisition, etc.
  - Emerging needs in Government and Commercial Organizations
    - E.g. Mission critical operations post 9/11, Inter-organizational collaboration, etc.
  - Brings users & objects together in a group
    - Secure Meeting Room
    - Subscription Model
Problem Statement

- One of the central problems in information sharing is the ability to securely share information for a specific purpose or mission by bringing together users and information.
- There is no existing model that addresses this problem.

Contribution

Thesis Statement

• It is possible to systematically develop Policy, Enforcement and Implementation models for Group-Centric Sharing

Security and system goals (objectives/policy)

• Necessarily informal

Policy models

• Specified using users, subjects, objects, admins, labels, roles, groups, etc. in an ideal setting.
• Security analysis (objectives, properties, etc.).

Enforcement models

• Approximated policy realized using system architecture with trusted servers, protocols, etc.
• Enforcement level security analysis (e.g. stale information due to network latency, protocol proofs, etc.).

Implementation models

• Technologies such as Cloud Computing, Trusted Computing, etc.
• Implementation level security analysis (e.g. vulnerability analysis, penetration testing, etc.)

Concrete System

• Software and Hardware
Group-Centric Sharing (g-SIS)

**Operational aspects**

- **Group Characteristics**
  - Core properties
  - Membership semantics
  - Membership renewal semantics
  - g-SIS specification

- **Object Model**
  - Read-only
  - Read-write (versioning?)

- **User-Subject Model**
  - Read-only subjects may read multiple groups
  - Read-write subjects restricted to single group

**Administrative aspects**

- Authorizations for Join, Add, etc.

**Inter-group relations**

- Subordination
- Conditional Membership
- Mutual Exclusion
g-SIS Operations

GROUP Authz (u,o,r)?

- Users
  - Strict Join
  - Liberal Join
  - Strict Add
  - Liberal Add

- Objects
  - Strict Add
  - Liberal Add

- Strict Leave
- Liberal Leave
- Strict Remove
- Liberal Remove
Core g-SIS Properties

- **Authorization Persistence**
  - *Authorization cannot change if no group event occurs*

  \[ \varphi_0 = \Box (\text{Authz} \rightarrow (\text{Authz} \lor (\text{Join} \lor \text{Leave} \lor \text{Add} \lor \text{Remove}))) \]
  \[ \varphi_1 = \Box (\neg \text{Authz} \rightarrow (\neg \text{Authz} \lor (\text{Join} \lor \text{Leave} \lor \text{Add} \lor \text{Remove}))) \]

- **Authorization Provenance**
  - *Authorization can begin to hold only after a simultaneous period of user and object membership*

  \[ \varphi_2 = (\neg \text{Authz} \lor (\text{Authz} \land (\neg \text{Leave} \land \text{Join}) \land (\neg \text{Remove} \land \text{Add}))) \]
Core g-SIS Properties (contd)

- **Bounded Authorization**
  - *Authorization cannot grow during non-membership periods*
    
    \[ \varphi_3 = \Box((\text{Leave} \land \neg \text{Authz}) \rightarrow (\neg \text{Authz} \lor \text{Join})) \]
    
    \[ \varphi_4 = \Box((\text{Remove} \land \neg \text{Authz}) \rightarrow (\neg \text{Authz} \lor \text{Add})) \]

- **Availability**
  - *On add, authorization should hold for all existing users at add time*
    
    \[ \varphi_5 = \Box(\text{Join} \rightarrow (\text{Add} \rightarrow ((\text{Authz} \lor \text{Leave}) \lor \text{Leave}))) \]

\[ \neg \text{Leave} \]

\[ \text{Join} \quad \text{Add} \land \text{Authz} \]
Satisfaction and Independence

- The Core Properties are Satisfiable
  
  There exists a trace in which  \( \bigwedge_{i \in [0..5]} \varphi_i \) is true

- The Core Properties are Independent
  
  Neither prove nor refute one of the properties from others

  \[ \bigwedge_{i \neq j, j \in [0..5]} \varphi_j \rightarrow \varphi_i \text{ is not valid } \forall i \in [0..5] \]

  \[ \bigwedge_{i \neq j, j \in [0..5]} \varphi_j \rightarrow \neg \varphi_i \text{ is not valid } \forall i \in [0..5] \]
Membership Semantics

- Strict Vs Liberal Operations
  - User operations: <SJ, LJ> and <SL, LL>
  - Object operations: <SA, LA> and <SR, LR>

- SJ (u): User not authorized to access objects added prior to join time
- SA (o): Users joining after add time not authorized to access o
- LL (u): User retains access to objects authorized at leave time
- LR (o): Users authorized to access o at remove time retain access
Membership Renewal Semantics

• Lossless Vs Lossy Join
  • Lossless: Authorizations from past membership period not lost
  • Lossy: Some authorizations lost at rejoin time

• Restorative Vs Non-Restorative Join
  • Restorative: Authorizations from past membership restored
  • Non-Restorative: Past authorizations not restored at rejoin time

• Gainless Vs Gainful Leave

• Restorative Vs Non-Restorative Leave
The $\pi$-System Specification

- Allows all membership ops (Strict and Liberal user/object ops)
- Allows selected membership renewal ops
  - Lossless and Non-Restorative Join
  - Gainless and Non-Restorative Leave

\[
\forall i. \text{Type}(\text{join}_i) \in \{\text{SJ, LJ}\} \times \{\text{Lossless}\} \times \{\text{Non-Restorative}\}
\]
\[
\forall i. \text{Type}(\text{leave}_i) \in \{\text{SL, LL}\} \times \{\text{Gainless}\} \times \{\text{Non-Restorative}\}
\]
\[
\forall i. \text{Type}(\text{add}_i) \in \{\text{SA, LA}\}
\]
\[
\forall i. \text{Type}(\text{remove}_i) \in \{\text{SR, LR}\}
\]
The $\pi$-System Specification (contd)

**$\pi$-system g-SIS Specification:**

$$\pi = \Box (\text{Authz} \leftrightarrow \lambda_1 \lor \lambda_2) \land \bigwedge_{0 \leq j \leq 3} \tau_j$$

$$\lambda_1 = (\neg \text{SL} \land \neg \text{SR}) \ S (\text{SA} \lor \text{LA}) \land (\neg \text{LL} \land \neg \text{SL}) \ S (\text{SJ} \lor \text{LJ}))$$

$$\lambda_2 = (\neg \text{SL} \land \neg \text{SR}) \ S (\text{LJ} \land (\neg \text{SR} \land \neg \text{LR}) \ S \text{LA}))$$

The $\Pi$-system satisfies the core g-SIS properties

$$\pi \models \bigwedge_{0 \leq i \leq 5} \varphi_i$$

That is, the $\Pi$-system is a $g$-SIS specification
Fixed Operation Models

- 16 possible initial models with fixed operations
  - E.g. (SJ, SL, SA, SR) or (LJ, LL, LA, LR) for all users and objects

- Can be reduced to 8 fixed operation models
  - E.g. With SJ, object add semantics has no significance on user’s authorization
Enforcement Models for g-SIS

- Security and system goals (objectives/policy)
- Policy models
- Enforcement models
- Implementation models
- Concrete System
g-SIS Enforcement Model

- Enforcement Components
  - Control Center (CC)
  - Group Administrator (GA)
  - Users
- Allows Offline Access
- Assumes a Trusted Reference Monitor (TRM)
  - Resides on group user’s access machines
  - Enforces group policy
  - Synchronizes attributes periodically with server
- Objects Available Via Super-distribution
Interaction b/w Various Components

Att (u) = \{JoinTS, LeaveTS, gKey, N, ORL\}
Att (o) = \{AddTS\}

1.1 Request Join
1.2 Authorize Join
1.3 Fwd authorization to Join with integrity evidence
1.4 Provision Group Attributes

2.1 Add Object o
2.2 Approve and Release Object o

3. Access objects

4.1 Request Refresh
4.2 Update Attributes
5.1 Leave User u
5.2 Remove Object o
5.2 Update:
   a. RemoveTS (o) = Current Time
   b. ORL = ORL U \{AddTS (o), RemoveTS(o)\}

Authz(u, o, r) \leftrightarrow Add\_TS(o) \geq Join\_TS(u) \land Leave\_TS(u) = NULL \land o \notin ORL

Fixed Operation Model: (<SJ, SL>, <LA, SR>)
Concept of Stale-Safety

AIP:
Authorization
Information Point

ADP:
Authorization
Decision Point

AEP:
Authorization
Enforcement Point

Update
Staleness in g-SIS

RT$_i$: Refresh Times

Join (u)  Add (o1)  Add (o2)  Leave (u)

RT$_3$  RT$_4$  RT$_5$  RT$_6$

Request (u, o2, r)  Request (u, o1, r)

RT$_7$

Authz held at RT$_6$

Authz never held

Authz($u$, $o$, $r$) $\leftrightarrow$ Add$_{TS}(o) \geq$ Join$_{TS}(u) \land$ Leave$_{TS}(u) =$ NULL $\land$ $o \notin$ ORL

Fixed Operation Model: ($<SJ$, $SL>$, $<LA$, $SR>$)
Stale-Safe Security Properties

• If a user is able to perform an action on an object, the authorization to perform the action is guaranteed to have held sometime prior to perform

• Weak Stale-Safety
  • Allows safe authorization decision without contacting the CC
  • Achieved by requiring that authorization held at the most recent refresh time

• Strong Stale-Safety
  • Need to obtain up to date authorization information from CC after a request is received
  • If CC is not available, decision cannot be made
Weak and Strong Stale-Safe Properties

Weak Stale-Safety:
\[ \square (\text{perform} \rightarrow (\varphi_1 \lor \varphi_2)) \]

Strong Stale-Safety:
\[ \square (\text{perform} \rightarrow \varphi_2) \]

\[ \varphi_1 \equiv \lozenge (\neg \text{perform} \land (\neg \text{RT} \lor (\text{RT} \land \text{Authz}))) \]

\[ \varphi_2 \equiv \lozenge (\neg \text{perform} \land (\neg \text{RT})) S (\text{RT} \land \text{Authz} \land ((\neg \text{perform} \land (\neg \text{RT} \lor (\text{RT} \land \text{Authz})))) S \text{request})) \]
Verification using Model Checking

\[ \Delta_2 \models \square (\text{perform} \rightarrow (\varphi_1 \lor \varphi_2)) \]

\[ \Delta_1 \not\models \square (\text{perform} \rightarrow (\varphi_1 \lor \varphi_2)) \]
Implementation Model and PoC

- Security and system goals (objectives/policy)
- Policy models
- Enforcement models
- Implementation models
- Concrete System
Implementation Model

- Specified TPM-based protocols for g-SIS Enforcement Model
- Proof-of-Concept
  - Assumed the presence of a Trusted Computing Base on client machines
  - Implemented secure provisioning of group credentials on the user machine
Contribution

• Policy Layer
  • Formal characterization of Group-Centric models
    • Identification of a core set of properties required of all g-SIS specifications
    • Proof of Independence and Satisfaction of core properties
    • A set of useful group operation semantics
  • A family of g-SIS specifications (\(\pi\)-system) supporting a variety of group operation semantics
    • A formal proof that the \(\pi\)-system satisfies the core properties

• Enforcement Layer
  • Identification and specification of stale-safe security properties
  • Verification of stale-safety of g-SIS enforcement model

• Implementation Layer
  • TPM-based protocols for g-SIS enforcement model
  • Provisioning protocol proof-of-concept
A few things that I did not talk about...

- **Policy Layer**
  - Detailed versioning model
  - Case-study of inter-organizational collaboration scenario
    - Administrative Component
    - Operational Component with a user-subject model
    - A framework for developing more sophisticated g-SIS models

- **Enforcement Layer**
  - Super-distribution, Micro-distribution and Hybrid enforcement models
  - Model checking g-SIS enforcement model using NuSMV

- **Implementation Layer**
  - Approach for access control of group credentials in user’s machine
  - TPM-based protocols for super-distribution and hybrid model
  - Proof of Concept design of provisioning protocol
Future Work

- Inter-group Relations
  - Subordination, conditional membership, mutual exclusion
  - Handling relationship changes
  - Handling information flow
- Administrative Models for g-SIS
- Need Other Access Control Components in Practical Scenarios
  - Meaningfully combine DAC, LBAC, RBAC and ABAC in g-SIS
- Generalization of Stale-safety to Multiple Authorization Information Points
  - Extension to ABAC
- Complete Implementation
Related Publications


Questions and Comments

Thank you!
Backup
Introduction & Motivation

- **Secure Information Sharing**
  - Share but protect
  - A fundamental problem in cyber security
- **Dissemination-Centric Sharing**
  - Dissemination chain with sticky policies on objects
  - E.g. ORCON, DRM, ERM, XrML, ODRL, etc.
- **Query-Centric Sharing**
  - Queries wrt a particular dataset
  - More generally addresses de-aggregation/inference problem
- **Group-Centric Sharing**
  - Sharing for a specific purpose or mission
    - E.g. Collaboration in joint product design, merger and acquisition, etc.
  - Emerging needs in Government and Commercial Organizations
    - E.g. Mission critical operations post 9/11, Inter-organizational collaboration, etc.
  - Brings users & objects together in a group
    - Secure Meeting Room
    - Subscription Model
Thesis Statement

• It is feasible to systematically develop Policy, Enforcement and Implementation models for Group-Centric Sharing
  • Consider temporal aspects in this initial work
Technical Approach

- Study Policy, Enforcement and Implementation aspects of Group-Centric Secure Information Sharing

- Security and system goals (objectives/policy)
  - Necessarily informal
  - Specified using users, subjects, objects, admins, labels, roles, groups, etc. in an ideal setting.
  - Security analysis (objectives, properties, etc.).
  - Approximated policy realized using system architecture with trusted servers, protocols, etc.
  - Enforcement level security analysis (e.g. stale information due to network latency, protocol proofs, etc.).
  - Technologies such as SOA, Cloud, SaaS, Trusted Computing, MILS, etc.
  - Implementation level security analysis (e.g. vulnerability analysis, penetration testing, etc.)
  - Software and Hardware
Related Publications


Information Protection Models

- Traditional models do capture important SIS aspects
  - But not satisfactory
- Discretionary Access Control (DAC)
  - Owner based discretion
  - Fails to distinguish copy from read
- Lattice Based Access Control (E.g. Bell-LaPadula)
  - One directional information flow in a lattice of security labels
  - Rigid and coarse-grained due to strict one-directional information flow within predefined security labels
- Role Based Access Control (E.g. RBAC96)
  - Effective administration
  - Too flexible; does not directly address information sharing
- Attribute Based Access Control (E.g. UCON)
  - Obligations, Conditions, etc.
  - Too flexible; does not directly address information sharing
Secure Information Sharing (SIS)

- **Share *but* protect**
  - A fundamental problem in cyber security
- **Traditional models do capture important SIS aspects**
  - But not satisfactory
  - Discretionary Access Control (owner control)
    - Too fine-grained, lacks copy control
  - Bell-LaPadula (information flow)
    - Too rigid and coarse-grained
  - Role-Based Access Control (effective administration)
    - Too general and does not directly address information sharing
  - UCON/ABAC also too general
- **Primary issues**
  - Copy control
  - Manageability
Dissemination-Centric Sharing

- Extensive research in the last two decades
  - ORCON, DRM, ERM, XrML, ODRL, etc.
- Copy/usage control has received major attention
- Manageability problem largely unaddressed
Roles Vs Groups in SIS

• Roles
  • Users get same set of privileges on role assignment
  • Does not consider timing of assignment/activation
  • Temporal RBAC considers specific timing aspects
    • E.g. authorizations for when a role can be activated

• Groups
  • Privileges may differ with time of join, leave, etc.
  • Sharing is promoted within and across groups
  • Inter-group relationship may differ from that of roles
Group-Centric Sharing (g-SIS)

- Brings users & objects together in a group
  - Two metaphors
    - Secure Meeting Room
    - Subscription Model

- Operational aspects
  - Group characteristics
    - E.g. What are the properties of a group?
  - Group operation semantics
    - E.g. What is authorized by join, add, etc.?

- Administrative aspects
  - E.g. Who authorizes join, add, etc.?
  - May be application dependant

- Inter-group relations
Group-Centric Sharing (g-SIS)

- Operational aspects
  - Object Model
    - Read-only
    - Read-Write (With and without versioning)
  - User-Subject Model
    - Read-only subjects can read from multiple groups
    - Read-write subjects can read and write only in one group
- Group characteristics
  - Core properties
    - Independence and Satisfiability
  - Operation semantics
    - Membership semantics
    - Membership renewal semantics
- Administrative aspects
  - E.g. Who authorizes join, add, etc.?
- Inter-group relations
  - Subordination, Conditional Membership, Mutual Exclusion
Linear Temporal Logic (summary)

- Next $p$ ($\bigcirc p$)
  - Formula $p$ holds in the next state
- Henceforth $p$ ($\Box p$)
  - Starting from current state, $p$ will continuously hold in all the future states
- $p$ until $q$ ($p \mathcal{U} q$)
  - $q$ will occur sometime in the future and $p$ will hold at least until the first occurrence of $q$
- $p$ unless $q$ ($p \mathcal{W} q$)
  - $p$ holds either until the next occurrence of $q$ or if $q$ never occurs, it holds throughout

- Previous $p$ ($\bigcirc p$)
  - Formula $p$ held in the previous state
- Once $p$ ($\Diamond p$)
  - Formula $p$ held at least once in the past
- $p$ since $q$ ($p \mathcal{S} q$)
  - $q$ happened in the past and $p$ held continuously from the position following the last occurrence of $q$ to the present
Notations

• Use _Join, _Leave, _Add and _Remove to refer to some respective event type occurring

\[
\begin{align*}
\text{Join}(u) &= (\text{join}_1(u) \lor \text{join}_2(u) \lor \ldots \lor \text{join}_m(u)) \\
\text{Leave}(u) &= (\text{leave}_1(u) \lor \text{leave}_2(u) \lor \ldots \lor \text{leave}_n(u)) \\
\text{Add}(o) &= (\text{add}_1(o) \lor \text{add}_2(o) \lor \ldots \lor \text{add}_p(o)) \\
\text{Remove}(o) &= (\text{remove}_1(o) \lor \ldots \lor \text{remove}_q(o))
\end{align*}
\]

• Drop the parameters for convenience

\[
\text{Authz} \rightarrow (\text{Join} \land (\neg(\text{Leave} \lor \text{Remove})))
\]

\[
\equiv
\]

\[
\forall u \in U. \forall o \in O. \text{Authz}(u, o, r) \rightarrow (\text{Join}(u) \land (\neg(\text{Leave}(u) \lor \text{Remove}(o))))
\]
Well-Formed Traces

- Multiple events cannot occur in a state for the same user (or object)
  - **E.g. 1** User cannot join and leave in the same state
  - **E.g. 2** Two types of join cannot occur in the same state

\[ \{ \text{join}_3(u_1) \} \quad \{ \text{add}_1(o_1) \} \quad \{ \text{join}_1(u_2) \} \quad \{ \text{remove}_3(o_2) \} \]

- User events should occur alternatively beginning with a join event
  - **E.g. 1** leave cannot occur before join
  - **E.g. 2** join should be followed by a leave before another join

\[ \{ \text{leave}_1(u_1) \} \quad \{ \text{join}_1(u_1) \} \quad \{ \text{join}_1(u_2) \} \quad \{ \text{join}_2(u_2) \} \]
LTL Specification of Well-Formed Traces

\[ \tau_0 = \Box \neg (\text{Add} \land \text{Remove}) \land \neg (\text{Join} \land \text{Leave}) \]

\[ \tau_1 = \forall i, j \Box ((i \neq j) \rightarrow \neg (\text{join}_i \land \text{join}_j)) \land \\
\forall i, j \Box ((i \neq j) \rightarrow \neg (\text{leave}_i \land \text{leave}_j)) \land \\
\forall i, j \Box ((i \neq j) \rightarrow \neg (\text{add}_i \land \text{add}_j)) \land \\
\forall i, j \Box ((i \neq j) \rightarrow \neg (\text{remove}_i \land \text{remove}_j)) \]

\[ \tau_2 = \Box (\text{Join} \rightarrow \Diamond (\neg \text{Join} \lor \text{Leave})) \land \\
\Box (\text{Leave} \rightarrow \Diamond (\neg \text{Leave} \lor \text{Join})) \land \\
\Box (\text{Add} \rightarrow \Diamond (\neg \text{Add} \lor \text{Remove})) \land \\
\Box (\text{Remove} \rightarrow \Diamond (\neg \text{Remove} \lor \text{Add})) \]

\[ \tau_3 = \Box (\text{Leave} \rightarrow \Diamond \text{Join}) \land \Box (\text{Remove} \rightarrow \Diamond \text{Add}) \]
g-SIS Specification (Syntactic Correctness)

- Defines precisely when authorization holds
- A g-SIS specification is *syntactically correct* if
  - Stated in terms of past user and object operations
  - Satisfies well-formedness constraints

\[
\gamma = \forall u \in U. \forall o \in O. (\Box(\text{Authz}(u, o, r) \leftrightarrow \psi(u, o))) \land \bigwedge_{0 \leq i \leq 3} \tau_i
\]

specified using join, leave, add and remove

Well-formedness constraints

- A g-SIS specification is *semantically correct* if it satisfies following core properties
g-SIS Specification (Semantic Correctness)

- Semantically correct if it satisfies the core g-SIS properties

\[
\gamma \models \bigwedge_{0 \leq i \leq 5} \varphi_i
\]
Group Operation Semantics

- **Membership semantics**
  - Authorizations enabled by current membership (join & add)
  - And authorizations disabled at the time of leave and remove

- **Membership Renewal Semantics**
  - Authorizations enabled from prior membership period
  - And those disabled at subsequent leave time
<table>
<thead>
<tr>
<th>Operation</th>
<th>Explanation</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict Join (SJ)</td>
<td>Only objects added after join time can be accessed</td>
<td>$\alpha_0 = \Box (\text{Authz} \rightarrow \Diamond (\text{Add} \land \neg \text{Leave} \lor \text{Join}))$</td>
</tr>
<tr>
<td>Liberal Join (LJ)</td>
<td>Can access objects added before and after join time</td>
<td>There exists a well-formed trace that does not satisfy $\alpha_0$</td>
</tr>
<tr>
<td>Strict Leave (SL)</td>
<td>Lose access to all objects on leave</td>
<td>$\alpha_1 = \Box (\text{Authz} \rightarrow \neg \text{leave} \lor \text{Join})$</td>
</tr>
<tr>
<td>Liberal Leave (LL)</td>
<td>Retain access to objects authorized before leave time</td>
<td>There exists a well-formed trace that does not satisfy $\alpha_1$</td>
</tr>
<tr>
<td>Strict Add (SA)</td>
<td>Only users who joined prior to add time can access</td>
<td>$\alpha_2 = \Box (\text{add} \rightarrow \neg \Diamond \text{Join} \lor \neg \text{Authz} \lor \text{Add}))$</td>
</tr>
<tr>
<td>Liberal Add (LA)</td>
<td>Users who joined before or after add time may access</td>
<td>There exists a well-formed trace that does not satisfy $\alpha_2$</td>
</tr>
<tr>
<td>Strict Remove (SR)</td>
<td>All users lose access on remove</td>
<td>$\alpha_3 = \Box (\text{remove} \rightarrow \neg \text{Authz} \lor \text{Add})$</td>
</tr>
<tr>
<td>Liberal Remove (LR)</td>
<td>Users who had access at remove time retain access</td>
<td>There exists a well-formed trace that does not satisfy $\alpha_3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Explanation</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lossless Join</td>
<td>Authorizations prior to join time is not lost</td>
<td>$\beta_0 = \Box ((\text{Join} \land \neg \text{Remove} \lor \Diamond \text{Authz}) \rightarrow \text{Authz})$</td>
</tr>
<tr>
<td>Lossy Join</td>
<td>Authorizations from prior to join may be lost</td>
<td>There exists a well-formed trace that does not satisfy $\beta_0$</td>
</tr>
<tr>
<td>Non-Restorative</td>
<td>Authorizations from past membership periods not</td>
<td>$\rho_1 = (\text{join}_1(u1) \land \text{join}_1(u2)) \land$</td>
</tr>
<tr>
<td>Join</td>
<td>explicitly restored</td>
<td>$\rho_2 = \Diamond (\text{Authz}(u1, o, r) \land \neg \text{Authz}(u2, o, r))$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\beta_1 = \forall i \Box (\rho_1 \rightarrow \rho_2)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There exists a well-formed trace that does not satisfy $\beta_1$</td>
</tr>
<tr>
<td>Restorative Join</td>
<td>Authorizations from past membership may be restored</td>
<td>$\beta_2 = \Box ((\text{Leave} \land \neg \Diamond \text{Join} \lor \Diamond \neg \text{Join}) \rightarrow$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Diamond ((\neg \text{Authz} \land \neg \text{Join}) \lor \text{Authz} \land (\neg \text{Join} \lor \text{Join}))$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There exists a well-formed trace that does not satisfy $\beta_2$</td>
</tr>
<tr>
<td>Gainsless Leave</td>
<td>Authorizations that never held during most recent</td>
<td>$\beta_3 = \Box (\text{Leave} \land \neg \text{Authz} \rightarrow \Diamond \text{Authz})$</td>
</tr>
<tr>
<td>Gainful Leave</td>
<td>membership period cannot be obtained</td>
<td>There exists a well-formed trace that does not satisfy $\beta_3$</td>
</tr>
<tr>
<td>Non-Restorative</td>
<td>New authorizations may be granted at Leave time</td>
<td></td>
</tr>
<tr>
<td>Leave</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restorative Leave</td>
<td>Authorizations that the user had prior to joining the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>group are not explicitly restored</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Authorizations from prior to join time may be restored</td>
<td></td>
</tr>
</tbody>
</table>


Verification Using Model Checker

- Model allows join, leave, add and remove to occur concurrently, non-deterministically and in any order

\[ \pi \rightarrow \bigwedge_{0 \leq i \leq 5} \varphi_i \]

- The above implication is used as the LTLSPEC
- The model checker generates a counter-example if the specification is false
- Used the open-source NuSMV model checker
# Read-Write Object Model

<table>
<thead>
<tr>
<th>No Versioning</th>
<th>Versioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Multiple users may update, latest write is committed (destructive write).</td>
<td>1. Multiple users may update, each update creates a new version.</td>
</tr>
<tr>
<td>2. No write after leave.</td>
<td>2. No write after leave.</td>
</tr>
</tbody>
</table>
| 4. Tricky issues with Liberal operations. E.g. On LL, past users may read new writes by group users.  
   4.1 Fix: Past LL users cannot read after write. | 4. No such issues. Past LL users may continue to read versions authorized at leave time. Provenance property rules out access to new versions after leave. |
Core Properties (no versioning)

- New Operation: Update(o)
- **Provenance** Both user and object should be current members
  \[ \Box (\text{Authz}(u, o, w) \rightarrow (\text{Authz}(u, o, r) \land (\neg \text{Leave } S \land \text{Join}) \land (\neg \text{Remove } S \land \text{Add})) \) \]
- **Bounded Authorization**
  - Past users cannot read after update
    \[ \Box (\text{Update} \land (\neg \text{Join } S \land \text{Leave}) \rightarrow (\neg \text{Authz}(u, o, r) \land \text{W Join})) \]
  - Past users cannot write. No write on past objects.
    \[ \Box (\text{Leave} \rightarrow \forall o. \neg \text{Authz}(u, o, w) \land \text{W Join}) \]
    \[ \Box (\text{Remove} \rightarrow \forall u. \neg \text{Authz}(u, o, w) \land \text{W Add}) \]
Core Properties (versioning)

- **New operation:** \( \text{Update}(o.v_i, o.v_j) \)

- **Authorization Provenance**
  - User needs to be a current member to write
  - Access can be frozen at leave time even with Liberal Leave or Remove

\[
\Box (\text{Authz}(u, o.v_i, w) \rightarrow (\text{Authz}(u, o.v_i, r) \land (\neg \text{Leave } S \land \text{Add}) \land
\neg \text{Remove } S (\text{Add} \lor \exists v_j. \text{Update}(o.v_j, o.v_i))))
\]

- **Bounded Authorization**

\[
\Box (\text{Leave}(u) \rightarrow (\forall o. \forall v_i. \neg \text{Authz}(u, o.v_i, w) \land \text{Leave}))
\]

\[
\Box (\text{Remove}(o.v_i) \rightarrow (\forall u. \neg \text{Authz}(u, o.v_i, w) \land \text{Leave}))
\]
Read-Write (versioning)

- An object is composed of multiple versions
- An update operation creates a new version
- A specific version of an object may be updated
  - Basically, versions are immutable
- New operation:
  - Update(o.v\textsubscript{i}, o.v\textsubscript{j})
Core Properties (Continued)

- **Version dependency properties**
  - If current user can read base version of o, all other versions of o can also be read
  - If some version of o can be read, all prior versions of o can also be read
  - If user can write some version of o, then he/she can write all versions of o
    - Note only current members can write
Core Properties (versioning)

- **New operation:** \( \text{Update}(o.v_i, o.v_j) \)

- **Authorization Persistence**

  - User needs to be a current member to write
  - Access can be frozen at leave time even with Liberal Leave or Remove

- **Authorization Provenance**

  - \( \Box (\text{Authz} \rightarrow (\text{Authz} \cup (\text{Join} \lor \text{Leave} \lor \text{Add} \lor \text{Remove} \lor \text{Update}))) ) \)
  - \( \Box (\neg \text{Authz} \rightarrow (\neg \text{Authz} \cup (\text{Join} \lor \text{Leave} \lor \text{Add} \lor \text{Remove} \lor \text{Update}))) ) \)

- \( \Box (\text{Authz}(u, o.v_i, w) \rightarrow (\text{Authz}(u, o.v_i, r) \land (\neg \text{Leave} \lor \text{Add}) \land (\neg \text{Remove} \lor \exists v_j. \text{Update}(o.v_j, o.v_i))) ) \)
Core Properties (continued)

- **Bounded Authorization**
  
  \[
  \square (\text{Leave}(u) \rightarrow (\forall o. \forall v_i. \neg \text{Authz}(u, o.v_i, w) \lor \text{Join}))
  \]
  
  \[
  \square (\text{Remove}(o.v_i) \rightarrow (\forall u. \neg \text{Authz}(u, o.v_i, w) \lor \text{Join}))
  \]

- **Availability**
  
  \[
  \square (\text{Join} \rightarrow (\text{Add}\lor\text{Update}(o.v_i, o.v_j) \rightarrow \text{Authz}(u, o.v_j, w) \lor \text{Leave}) \lor \text{Leave})
  \]
Super-distribution

User1 → Get (o) → Provide (c) → CC → Store c locally → Dec (c, k) and read o → User2

Object Cloud → Distribute o → CC

c = Enc (o, k) → Add (c) → Set AddTS for o
Micro-Distribution and Hybrid Approach

**Micro-distribution:**
Obtain custom encrypted object from CC the first time
Subsequent accesses can be offline

**Hybrid approach:**
Split-key RSA, One split per user
CC participates in initial decryption
Subsequent accesses can be offline
Properties

\[ \neg \text{Leave} \quad \neg \text{Remove} \land \neg \text{Leave} \]

\[ \text{Join} \quad \text{Add} \quad \text{Authz} \]

\[ \text{Authz} \equiv ( (\neg \text{Remove} \land \neg \text{Leave}) \ S \ (\text{Add} \land \neg \text{Leave} \ S \ \text{Join}) ) \]

\[ \neg \text{RT} \quad \neg \text{Perform} \land ( \neg \text{RT} \lor (\text{RT} \land \text{Authz}) ) \]

\[ \text{RT} \land \text{Authz} \quad \text{Request} \quad \text{Perform} \]

\[ \neg \text{Perform} \land ( \neg \text{RT} \lor (\text{RT} \land \text{Authz}) ) \]

\[ \neg \text{Perform} \land ( \neg \text{RT} \lor (\text{RT} \land \text{Authz}) ) \quad (\neg \text{Perform} \land \neg \text{RT}) \]

\[ \text{Request} \quad \text{RT} \land \text{Authz} \quad \text{Perform} \]

\[ \text{Formula } \varphi_1 \]

\[ \varphi_1 \equiv \Box ( \neg \text{perform} \land (\neg \text{RT} \lor (\text{RT} \land \text{Authz}) )) \]

\[ S ( \text{request} \land (\neg \text{RT} \ S (\text{RT} \land \text{Authz} )) ) \]

\[ \Box ( \neg \text{perform} \land \neg \text{RT}) \ S (\text{RT} \land \text{Authz} \land ( (\neg \text{perform} \land (\neg \text{RT} \lor (\text{RT} \land \text{Authz} ))) \ S \text{request} )) \]

\[ \text{Formula } \varphi_2 \]

\[ \Box ( \text{perform} \rightarrow (\varphi_1 \lor \varphi_2 )) \]

\[ \Box ( \text{perform} \rightarrow \varphi_2 ) \]

\[ \text{Weak Stale-Safety:} \]

\[ \Box ( \text{perform} \rightarrow (\varphi_1 \lor \varphi_2 )) \]

\[ \text{Strong Stale-Safety:} \]

\[ \Box ( \text{perform} \rightarrow \varphi_2 ) \]
Verification (continued)

- Let
  - $\mathcal{C}_0$: composition of stale-unsafe FSMs
  - $\mathcal{C}_1$: composition of weak stale-safe FSMs
  - $\mathcal{C}_2$: composition of strong stale-safe FSMs

- Verified using model checking that:
  - Authz is enforced by $\mathcal{C}_0$, $\mathcal{C}_1$ and $\mathcal{C}_2$
  - $\mathcal{C}_0$ fails Weak and Strong Stale-Safe security properties
  - $\mathcal{C}_1$ satisfies Weak Stale-Safe security property
  - $\mathcal{C}_1$ fails Strong Stale-Safe security property
  - $\mathcal{C}_2$ satisfies Strong Stale-Safe security property
Strong Stale-Safe Machine