Group-Centric Secure Information Sharing Models

Ram Krishnan

PhD Candidate

Dissertation Directors:

Dr. Ravi Sandhu and Dr. Daniel Menascé

Dissertation Defense

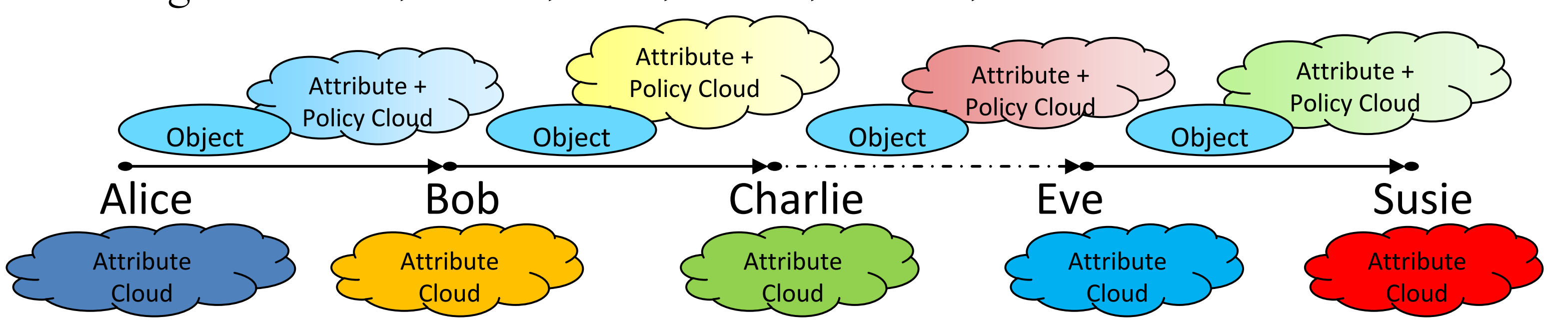
Nov 25th 2009

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

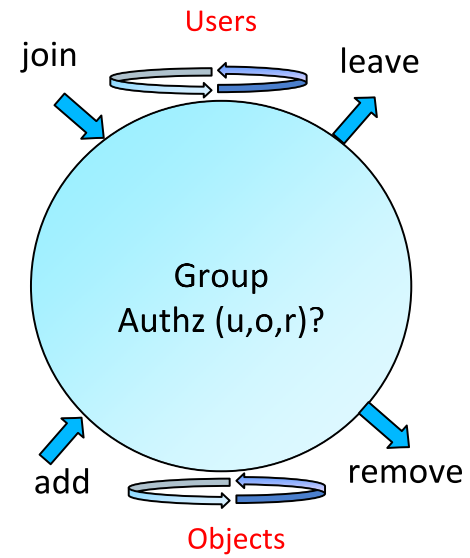
- Secure Information Sharing
  - Share *but* protect
  - A fundamental problem in cyber security
- Dissemination-Centric Sharing
  - Dissemination chain with "sticky" policies on objects
  - E.g. ORCON, DRM ,ERM, XrML, ODRL, etc.



- 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

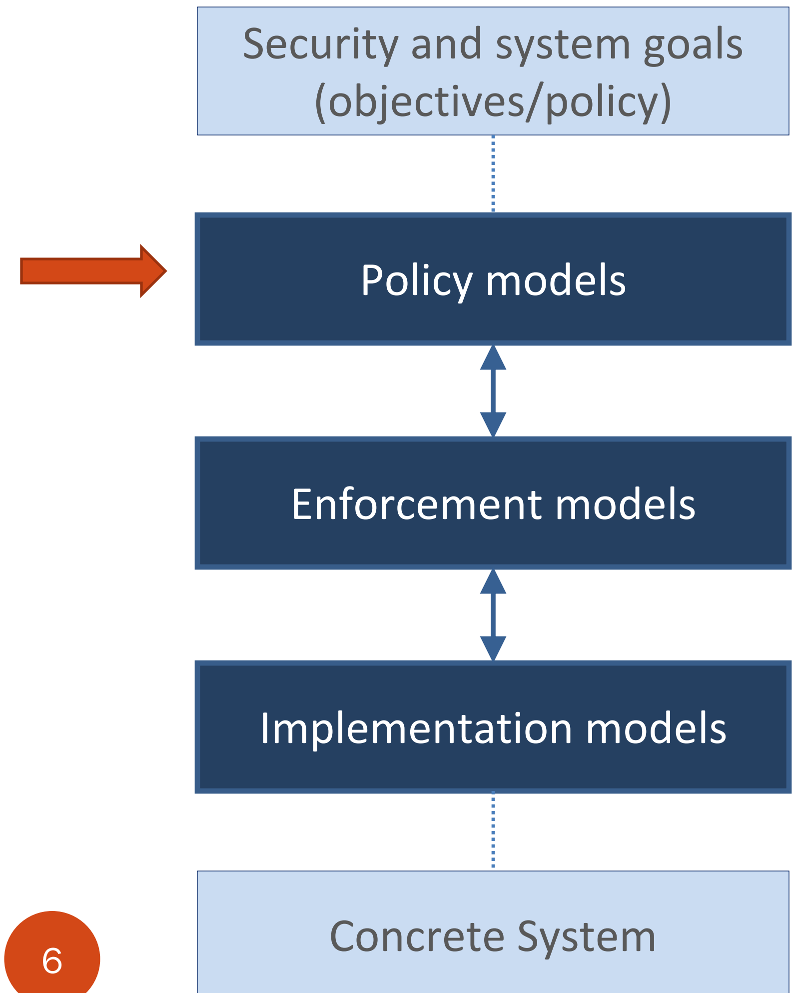
- One of the central problems in information sharing is the ability to securely share information for a specific purpose or mission by bringing together users and information
- There is no existing model that addresses this problem

Contribution

- A first step towards a formal and systematic study of Group-Centric Secure Information Sharing Models

Thesis Statement

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



- 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 Cloud Computing, Trusted Computing, etc.
- Implementation level security analysis (e.g. vulnerability analysis, penetration testing, etc.)

- 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

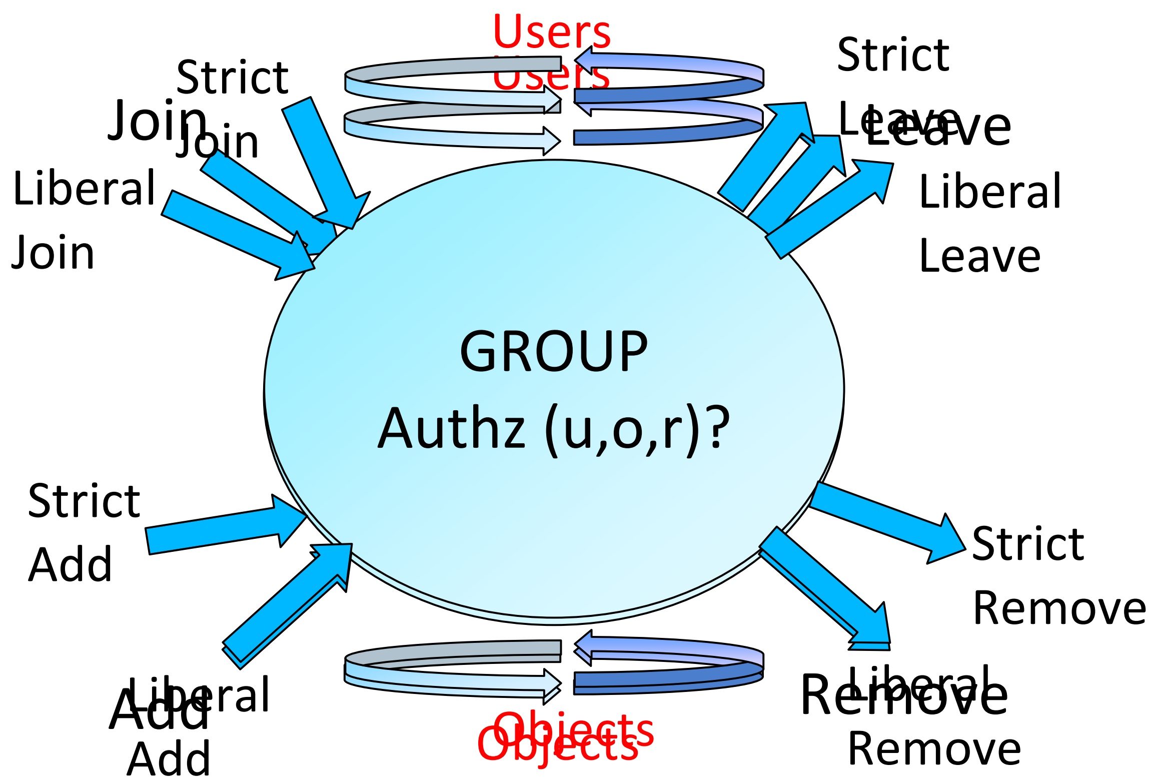
Administrative aspects

- Authorizations for Join, Add, etc.

Inter-group relations

- Subordination
- Conditional Membership
- Mutual Exclusion

g-SIS Operations



Core g-SIS Properties

- <u>Authorization Persistence</u>
  - *Authorization cannot change if no group event occurs*

$$\varphi_0 = \Box(\text{Authz} \rightarrow (\text{Authz}\ \mathcal{W}\ (\text{Join} \vee \text{Leave} \vee \text{Add} \vee \text{Remove})))$$

$$\varphi_1 = \Box(\neg\text{Authz} \rightarrow (\neg\text{Authz}\ \mathcal{W}\ (\text{Join} \vee \text{Leave} \vee \text{Add} \vee \text{Remove})))$$

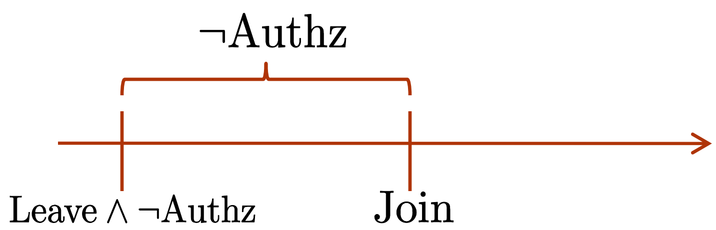
- <u>Authorization Provenance</u>
  - *Authorization can begin to hold only after a simultaneous period of user and object membership*

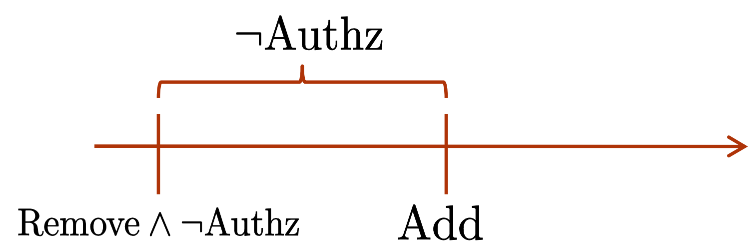
$$\varphi_2 = (\neg\text{Authz}\ \mathcal{W}\ (\text{Authz} \wedge (\neg\text{Leave}\ \mathcal{S}\ \text{Join}) \wedge (\neg\text{Remove}\ \mathcal{S}\ \text{Add})))$$

Core g-SIS Properties (contd)

- Bounded Authorization
  - *Authorization cannot grow during non-membership periods*

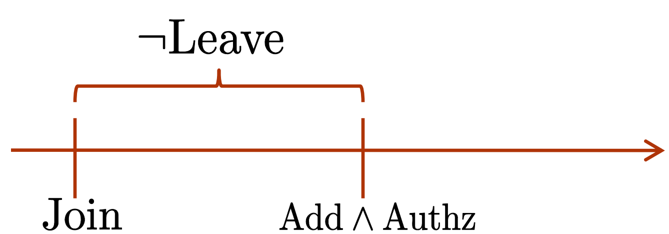
$$\varphi_3 = \Box((\mathrm{Leave} \wedge \neg\mathrm{Authz}) \rightarrow (\neg\mathrm{Authz}\ \mathcal{W}\ \mathrm{Join}))$$
$$\varphi_4 = \Box((\mathrm{Remove} \wedge \neg\mathrm{Authz}) \rightarrow (\neg\mathrm{Authz}\ \mathcal{W}\ \mathrm{Add}))$$





- Availability
  - *On add, authorization should hold for all existing users at add time*

$$\varphi_5 = \Box(\mathrm{Join} \rightarrow (\mathrm{Add} \rightarrow ((\mathrm{Authz}\ \mathcal{W}\ \mathrm{Leave})\ \mathcal{W}\ \mathrm{Leave})))$$



Satisfaction and Independence

- The Core Properties are Satisfiable

There exists a trace in which $\bigwedge_{i\in[0..5]} \varphi_i$ is true

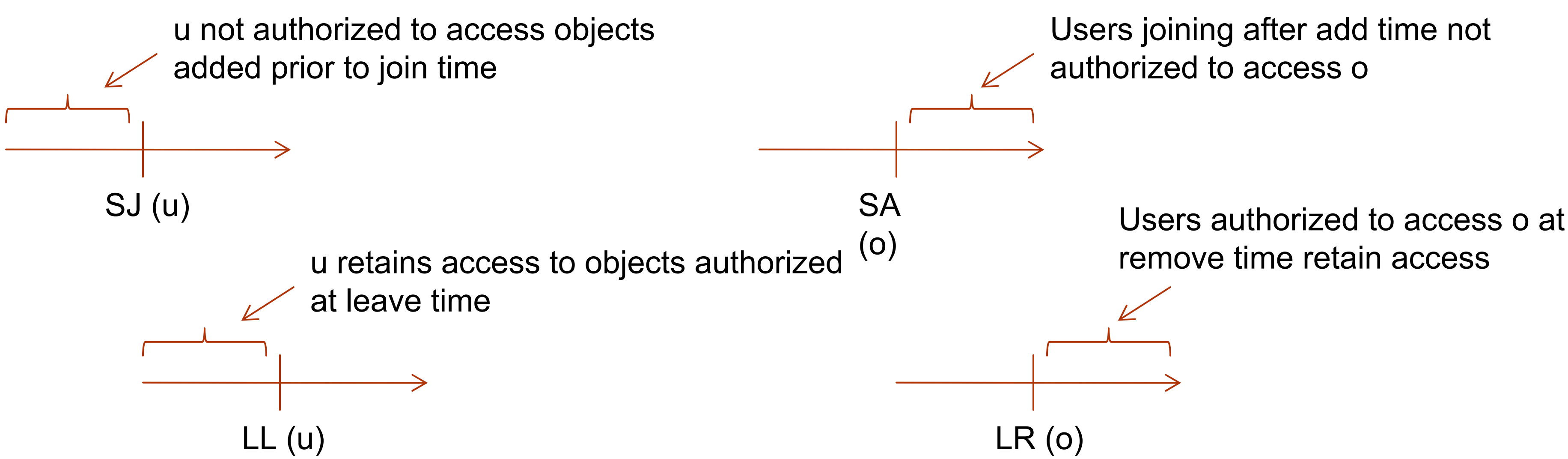
- The Core Properties are Independent
  - Neither prove nor refute one of the properties from others

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

$\bigwedge_{i\neq j, j\in[0..5]} \varphi_j \rightarrow \neg\varphi_i$ 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>



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 π-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

$$\begin{aligned}
\forall i.\mathrm{Type}(\mathrm{join}_i) &\in \{\mathrm{SJ}, \mathrm{LJ}\} \times \{\mathrm{Lossless}\} \times \{\text{Non-Restorative}\} \\
\forall i.\mathrm{Type}(\mathrm{leave}_i) &\in \{\mathrm{SL}, \mathrm{LL}\} \times \{\mathrm{Gainless}\} \times \{\text{Non-Restorative}\} \\
\forall i.\mathrm{Type}(\mathrm{add}_i) &\in \{\mathrm{SA}, \mathrm{LA}\} \\
\forall i.\mathrm{Type}(\mathrm{remove}_i) &\in \{\mathrm{SR}, \mathrm{LR}\}
\end{aligned}$$

The π-System Specification (contd)

Well-formed traces

Π-system g-SIS Specification:

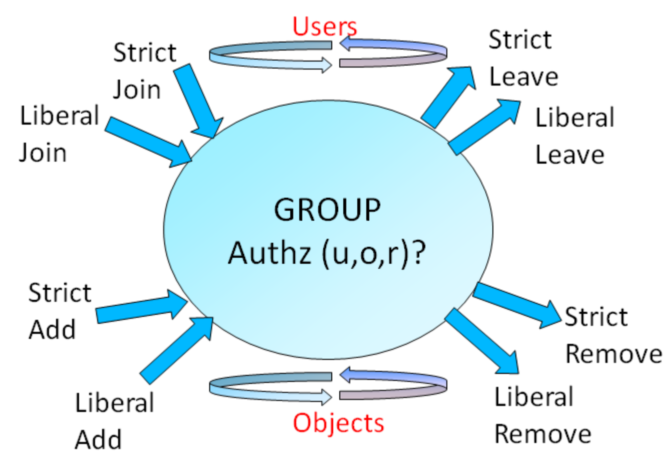
$$\pi = \Box(\text{Authz} \leftrightarrow \lambda_1 \vee \lambda_2) \wedge \bigwedge_{0 \leq j \leq 3} \tau_j$$

$$\lambda_1 = ((\neg\text{SL} \wedge \neg\text{SR})\ \mathcal{S}\ ((\text{SA} \vee \text{LA}) \wedge ((\neg\text{LL} \wedge \neg\text{SL})\ \mathcal{S}\ (\text{SJ} \vee \text{LJ}))))$$

Add after Join

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

Add before Join



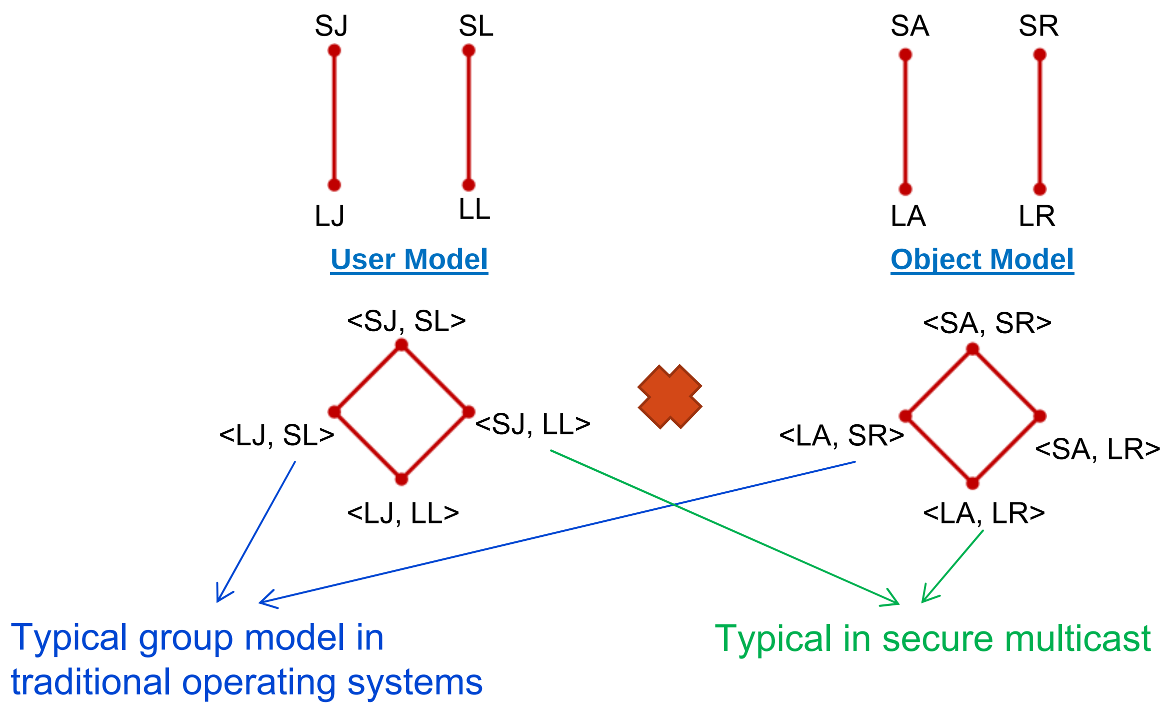
The Π-system satisfies the core g-SIS properties

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

That is, the Π-system is a *g-SIS* specification

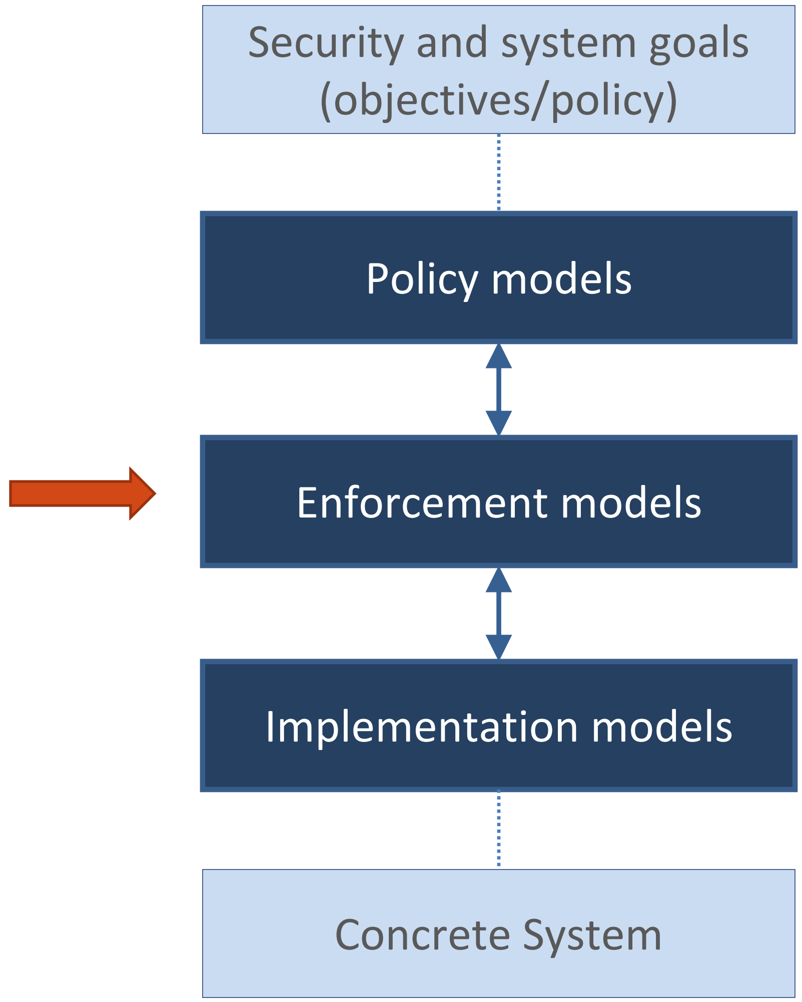
Fixed Operation Models

- 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

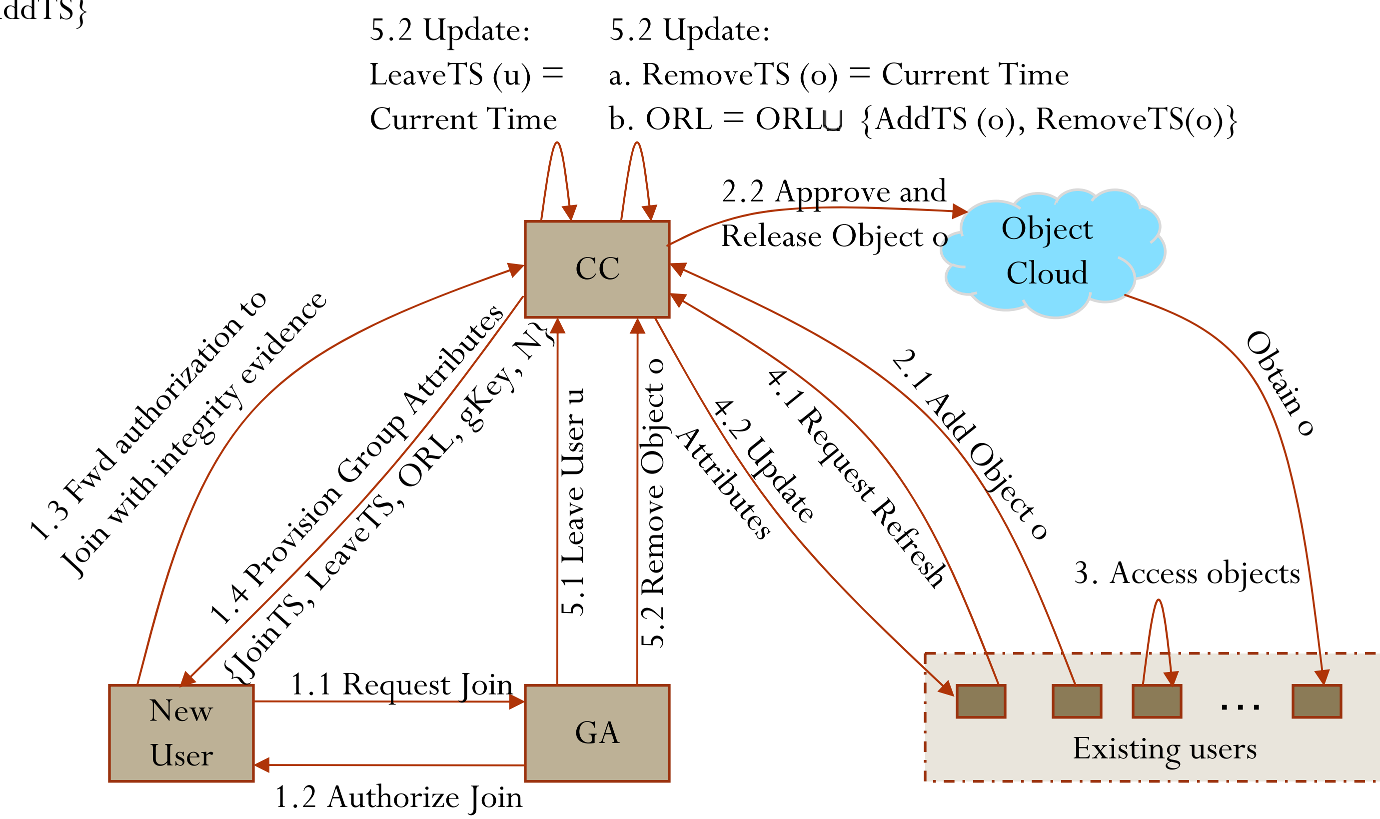


g-SIS Enforcement Model

- 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}



$$\mathrm{Authz}(u,o,r) \leftrightarrow \mathrm{Add\_TS}(o) \geq \mathrm{Join\_TS}(u) \wedge \mathrm{Leave\_TS}(u) = \mathrm{NULL} \wedge o \notin \mathrm{ORL}$$

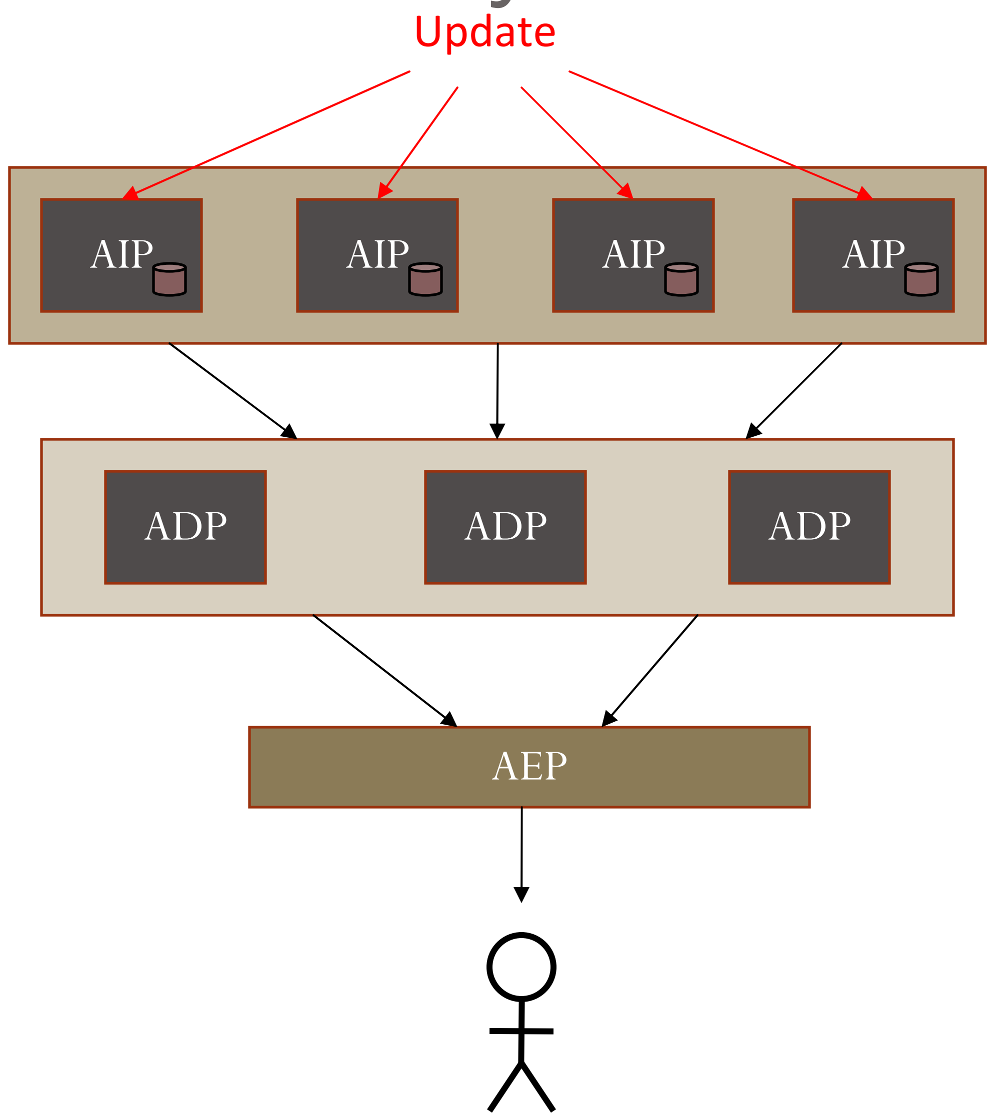
Fixed Operation Model: (<SJ, SL>, <LA, SR>)

Concept of Stale-Safety

AIP:
Authorization
Information Point

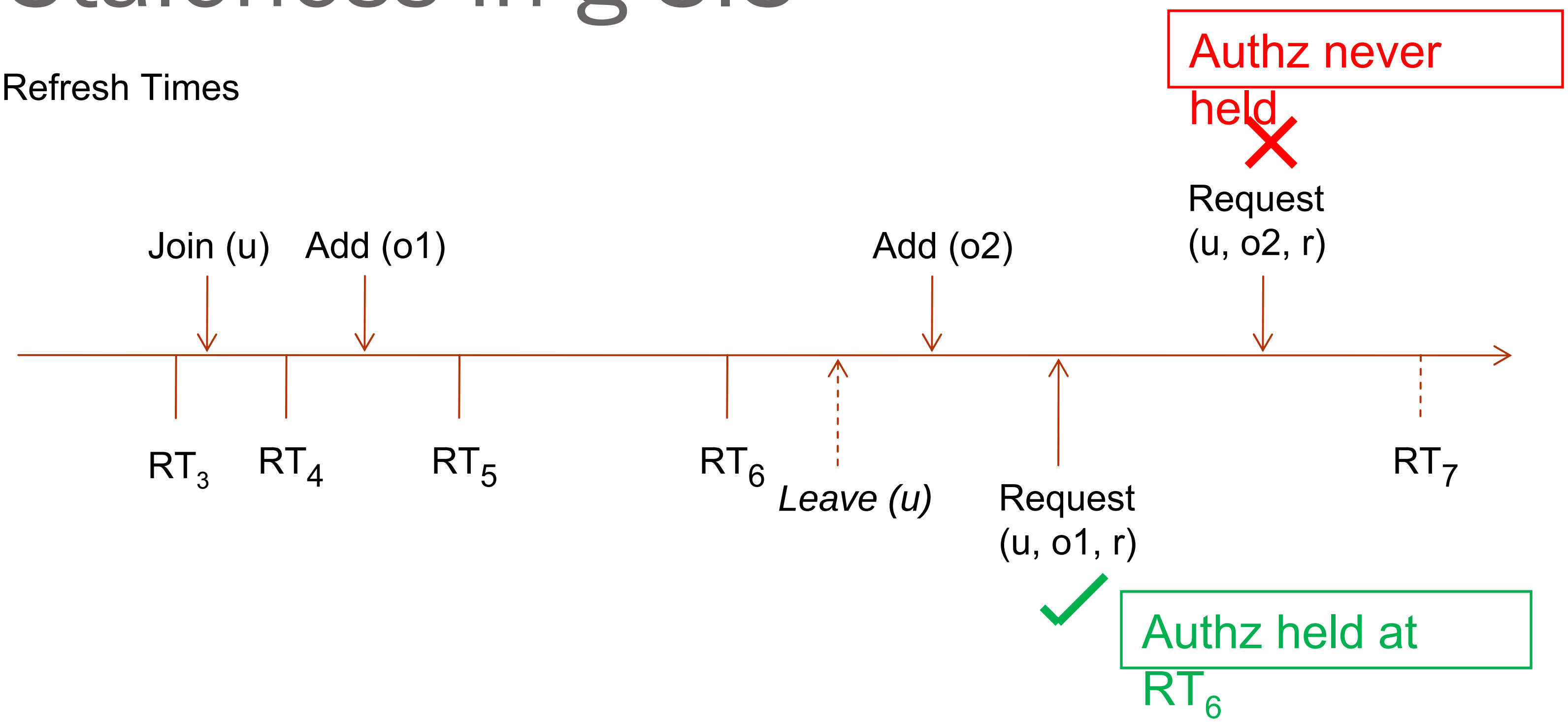
ADP:
Authorization
Decision Point

AEP:
Authorization
Enforcement Point



Staleness in g-SIS

RT_i: Refresh Times



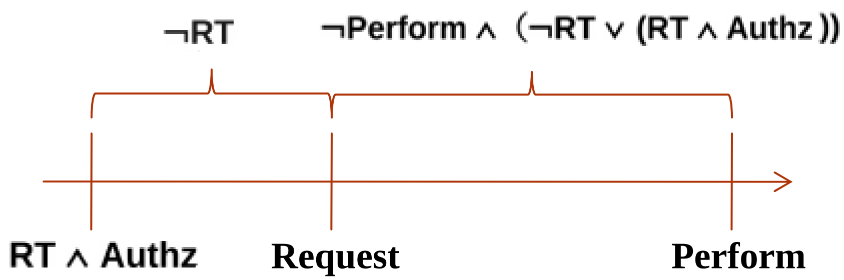
$$\text{Authz}(u, o, r) \leftrightarrow \text{Add\_TS}(o) \geq \text{Join\_TS}(u) \wedge \text{Leave\_TS}(u) = \text{NULL} \wedge o \notin \text{ORL}$$

Fixed Operation Model: (<SJ, SL>, <LA, SR>)

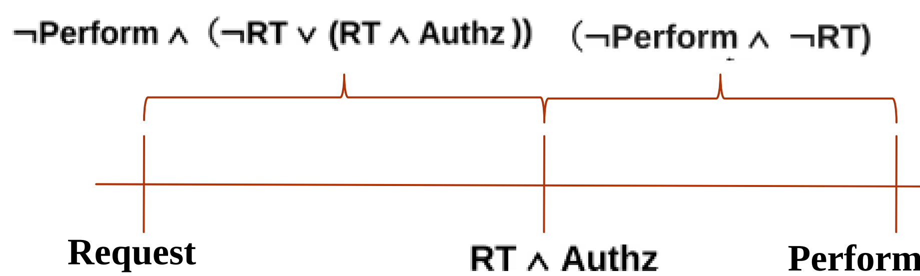
Stale-Safe Security Properties

- 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



Formula φ_1



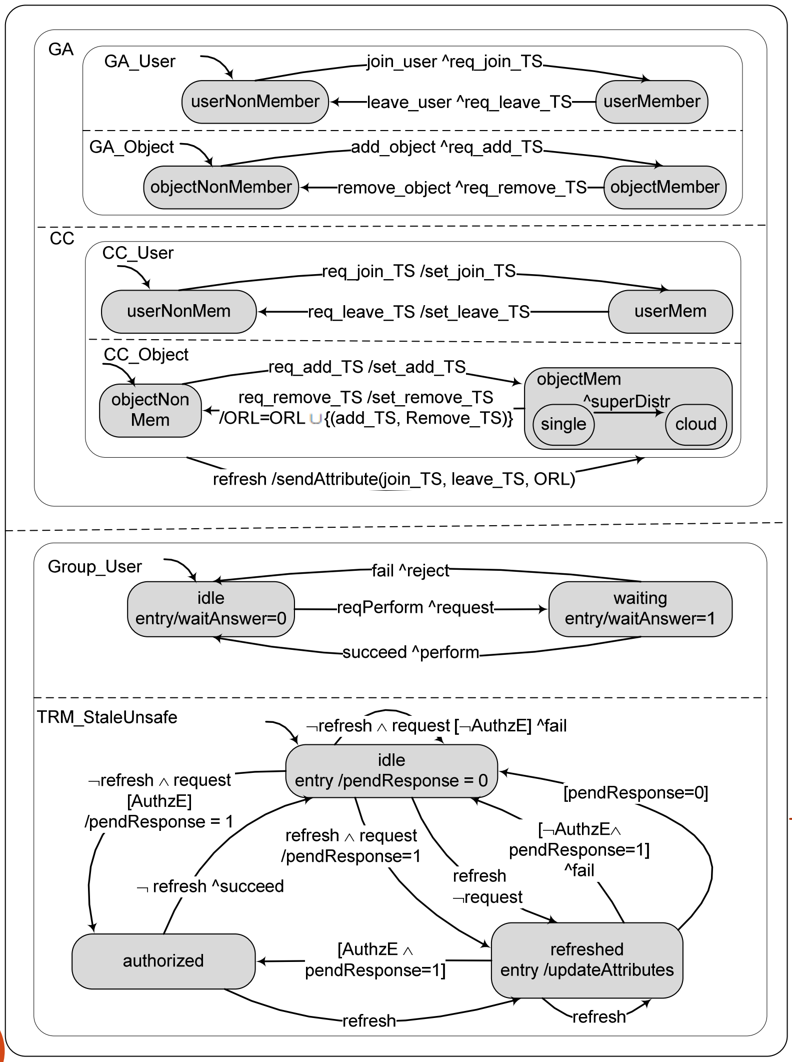
Formula φ_2

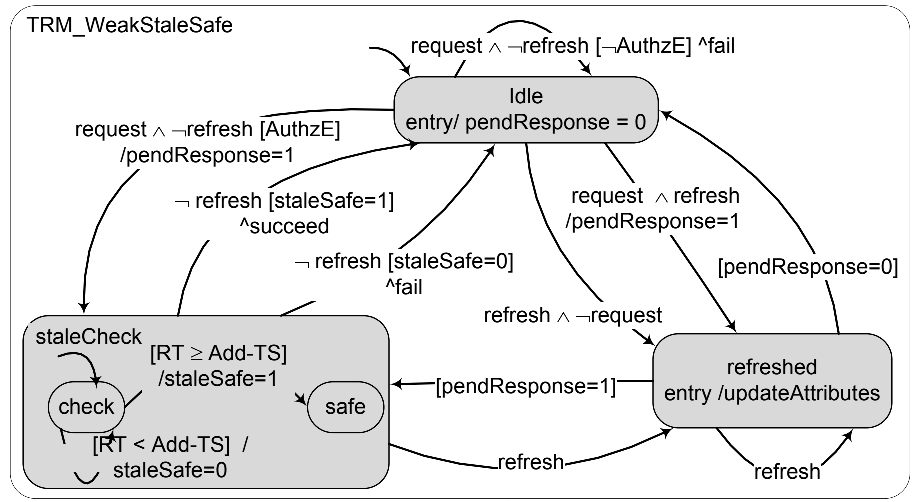
$$\varphi_1 \equiv \ominus\, (\neg \text{perform} \wedge (\neg \text{RT} \vee (\text{RT} \wedge \text{Authz})))\ \mathcal{S}\ (\text{request} \wedge (\neg \text{RT}\ \mathcal{S}\ (\text{RT} \wedge \text{Authz})))$$

$$\varphi_2 \equiv \ominus\, (\neg \text{perform} \wedge \neg \text{RT})\ \mathcal{S}\ (\text{RT} \wedge \text{Authz} \wedge ((\neg \text{perform} \wedge (\neg \text{RT} \vee (\text{RT} \wedge \text{Authz})))\ \mathcal{S}\ \text{request}))$$

Weak Stale-Safety:	$\square\, (\text{perform} \rightarrow (\varphi_1 \vee \varphi_2))$
Strong Stale-Safety:	$\square\, (\text{perform} \rightarrow \varphi_2)$

Verification using Model Checking

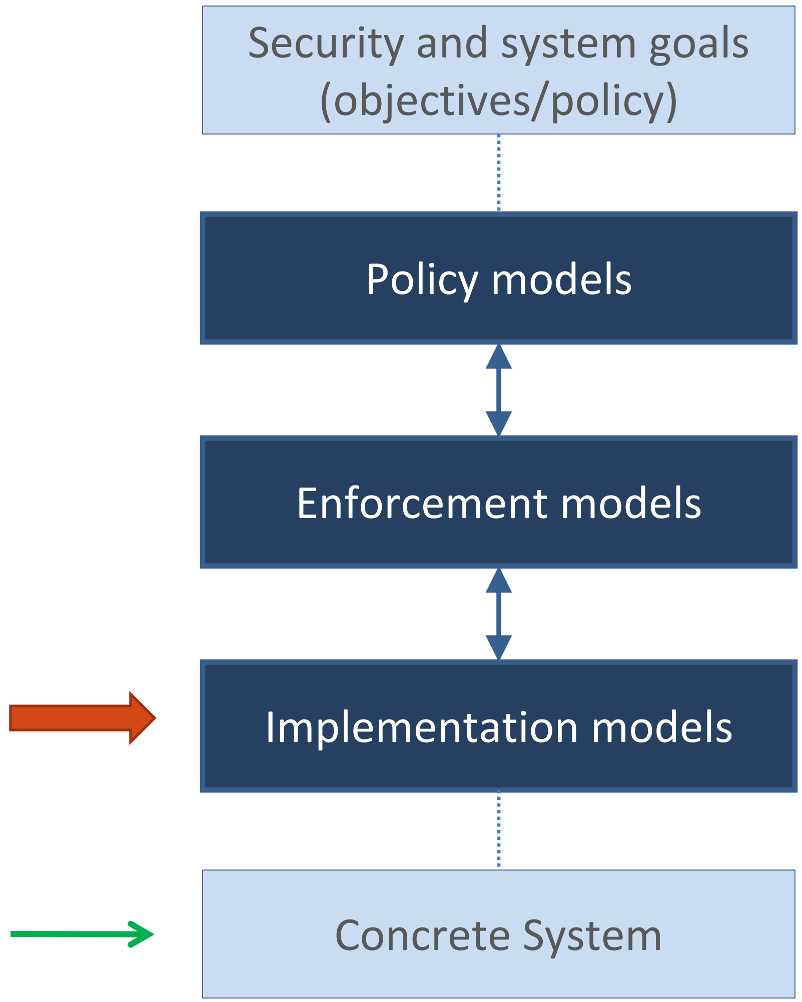




$$\Delta_2 \models \Box(\text{perform} \rightarrow (\varphi_1 \vee \varphi_2))$$

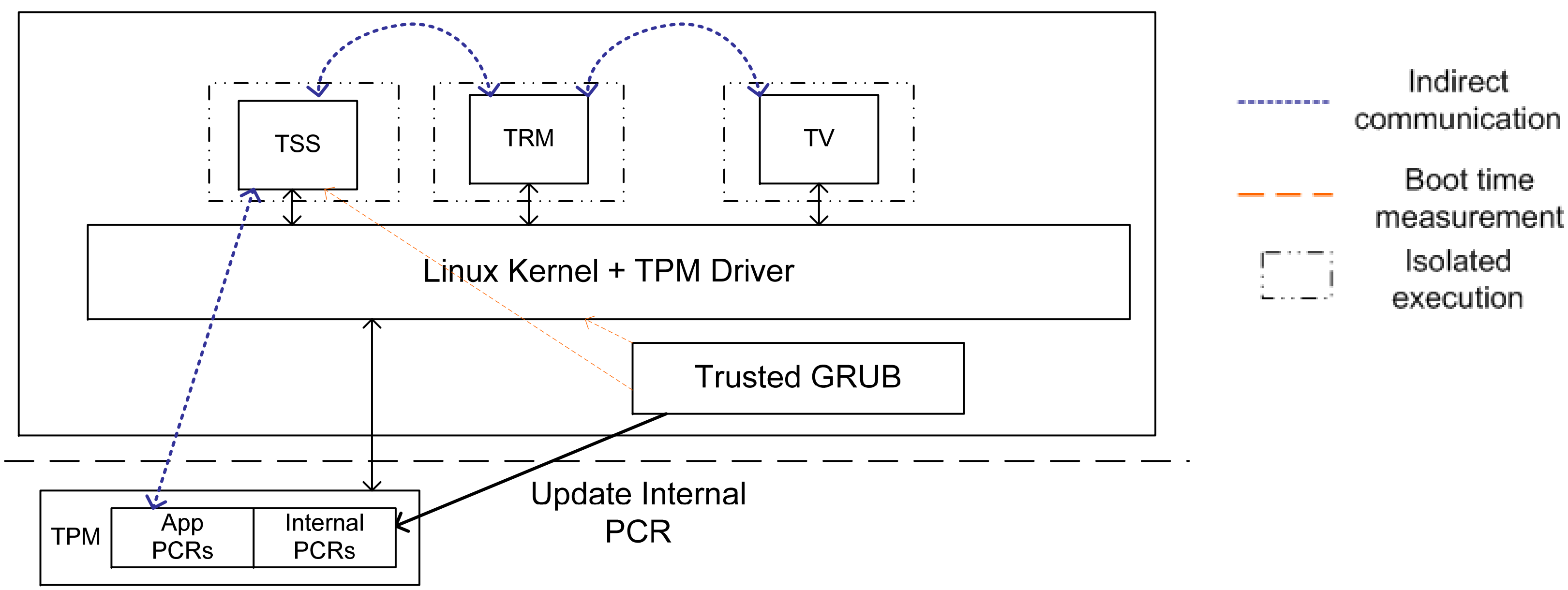
$$\Delta_1 \not\models \Box(\text{perform} \rightarrow (\varphi_1 \vee \varphi_2))$$

Implementation Model and PoC



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 (π-system) supporting a variety of group operation semantics
    - A formal proof that the π-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

- Ram Krishnan, Ravi Sandhu , Jianwei Niu and William Winsborough, Towards a Framework for Group-Centric Secure Collaboration, *Proceedings of IEEE International Conference on Collaborative Computing (CollaborateCom)*, Crystal City, Washington D.C., Nov 11-14, 2009. P
- Ram Krishnan and Ravi Sandhu, A Hybrid Enforcement Model for Group-Centric Secure Information Sharing, *Proceedings of IEEE International Symposium on Secure Computing (SecureCom 2009)*, August 29-31, Vancouver, Canada. E
- Ram Krishnan, Ravi Sandhu , Jianwei Niu and William Winsborough, Foundations for Group-Centric Secure Information Sharing Models, *Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT 2009),* June 3-5, Stresa, Italy. P
- Ram Krishnan and Ravi Sandhu, Enforcement Architecture and Implementation Model for Group-Centric Information Sharing, *International Workshop on Security and Communication Networks (IWSCN 2009),* May 20-22, 2009, Trondheim, Norway. EI
- Ram Krishnan, Ravi Sandhu , Jianwei Niu and William Winsborough, A Conceptual Framework for Group-Centric Secure Information Sharing Models, *Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS 2009)* , March 10 – 12 2009, Sydney, Australia. P
- Ram Krishnan, Jianwei Niu, Ravi Sandhu and William Winsborough, Stale-Safe Security Properties for Group-Based Secure Information Sharing, *Proceedings of ACM workshop on Formal Methods in Security Engineering (FMSE 2008)*. Oct 27- Oct 31 2008, Alexandria, Virginia, USA. E
- Manoj Sastry, Ram Krishnan and Ravi Sandhu, A New Modeling Paradigm for Dynamic Authorization in Multi-domain Systems, *Proceedings of International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS 2007)* . September 13-15, 2007, St. Petersburg, Russia. P
- Ram Krishnan, Ravi Sandhu and Kumar Ranganathan, PEI Models towards Scalable, Usable and High-Assurance Information Sharing, *Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT 2007)* . June 20-22, 2007, Nice-Sophia Antipolis, France. PEI

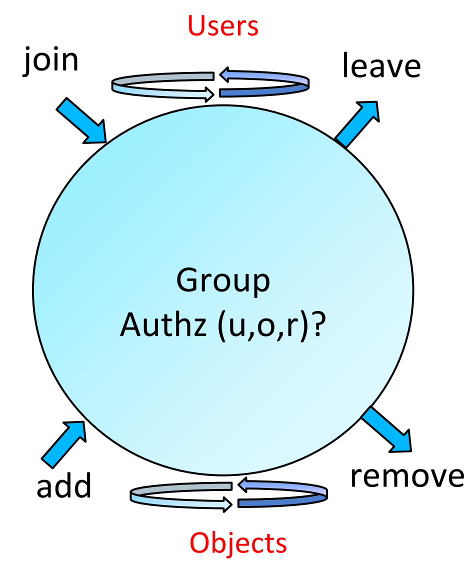
Questions and Comments

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

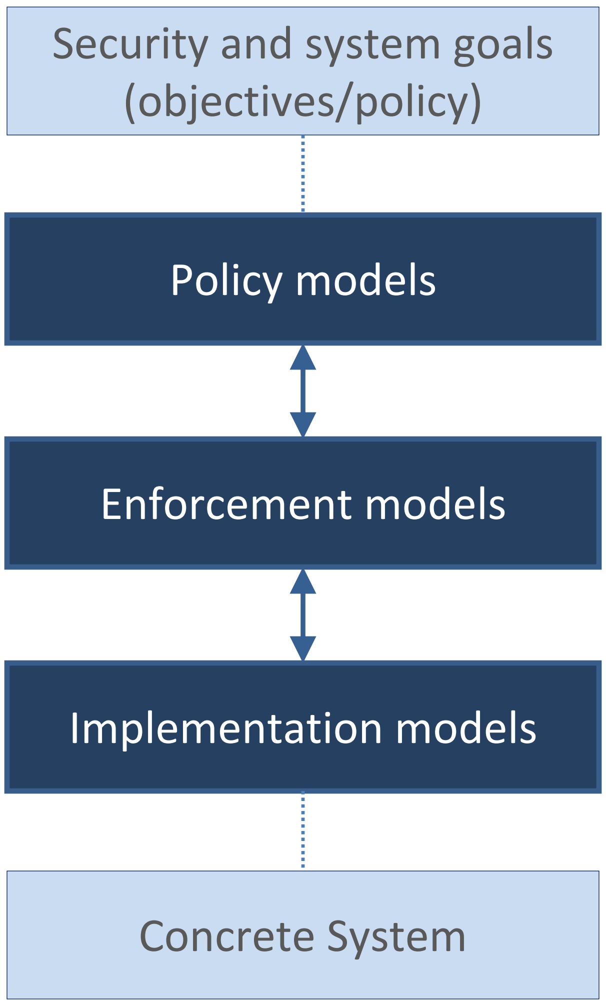


Thesis Statement

- 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



- 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

- Ram Krishnan, Ravi Sandhu , Jianwei Niu and William Winsborough, Towards a Framework for Group-Centric Secure Collaboration, *To appear in the 5th IEEE International Conference on Collaborative Computing (CollaborateCom)*, Crystal City, Washington D.C., Nov 11-14, 2009.
- Ram Krishnan and Ravi Sandhu, A Hybrid Enforcement Model for Group-Centric Secure Information Sharing, *Proceedings of IEEE International Symposium on Secure Computing (SecureCom 2009)*, August 29-31, Vancouver, Canada.
- Ram Krishnan, Ravi Sandhu , Jianwei Niu and William Winsborough, Foundations for Group-Centric Secure Information Sharing Models, *Proceedings of 14th ACM Symposium on Access Control Models and Technologies (SACMAT 2009),* June 3-5, Stresa, Italy.
- Ram Krishnan and Ravi Sandhu, Enforcement Architecture and Implementation Model for Group-Centric Information Sharing, *International Workshop on Security and Communication Networks (IWSCN 2009),* May 20-22, 2009, Trondheim, Norway.
- Ram Krishnan, Ravi Sandhu , Jianwei Niu and William Winsborough, A Conceptual Framework for Group-Centric Secure Information Sharing Models, *Proceedings of 4th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2009)* , March 10 – 12 2009, Sydney, Australia.
- Ram Krishnan, Jianwei Niu, Ravi Sandhu and William Winsborough, Stale-Safe Security Properties for Group-Based Secure Information Sharing, *Proceedings of 6th ACM workshop on Formal Methods in Security Engineering (FMSE 2008)*. Oct 27- Oct 31 2008, Alexandria, Virginia, USA.
- Manoj Sastry, Ram Krishnan and Ravi Sandhu, A New Modeling Paradigm for Dynamic Authorization in Multi-domain Systems, *Proceedings of 4th International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS 2007)* . September 13-15, 2007, St. Petersburg, Russia.
- Ram Krishnan, Ravi Sandhu and Kumar Ranganathan, PEI Models towards Scalable, Usable and High-Assurance Information Sharing, *Proceedings of 12th ACM Symposium on Access Control Models and Technologies (SACMAT 2007)* . June 20-22, 2007, Nice-Sophia Antipolis, France.

Information Protection Models

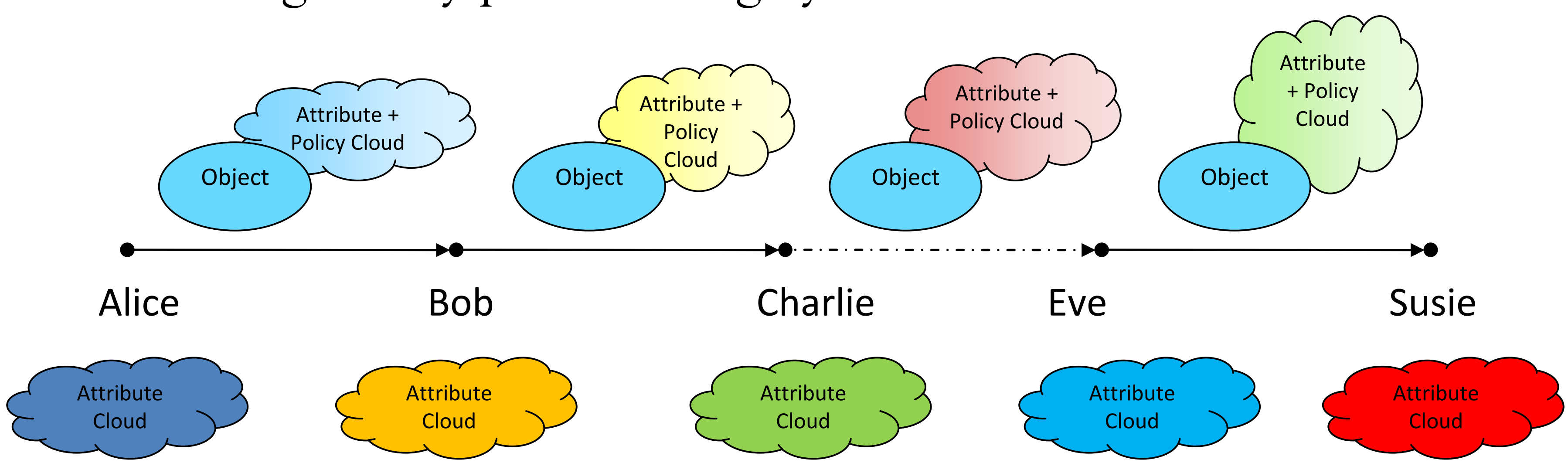
- Traditional models do capture important SIS aspects
  - But not satisfactory
- Discretionary Access Control (DAC)
  - <u>Owner based discretion</u>
  - Fails to distinguish copy from read
- Lattice Based Access Control (E.g. Bell-LaPadula)
  - One directional <u>information flow</u> in a lattice of 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 <u>administration</u>
  - Too flexible; does not directly address information sharing
- Attribute Based Access Control (E.g. UCON)
  - <u>Obligations, Conditions</u>, 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



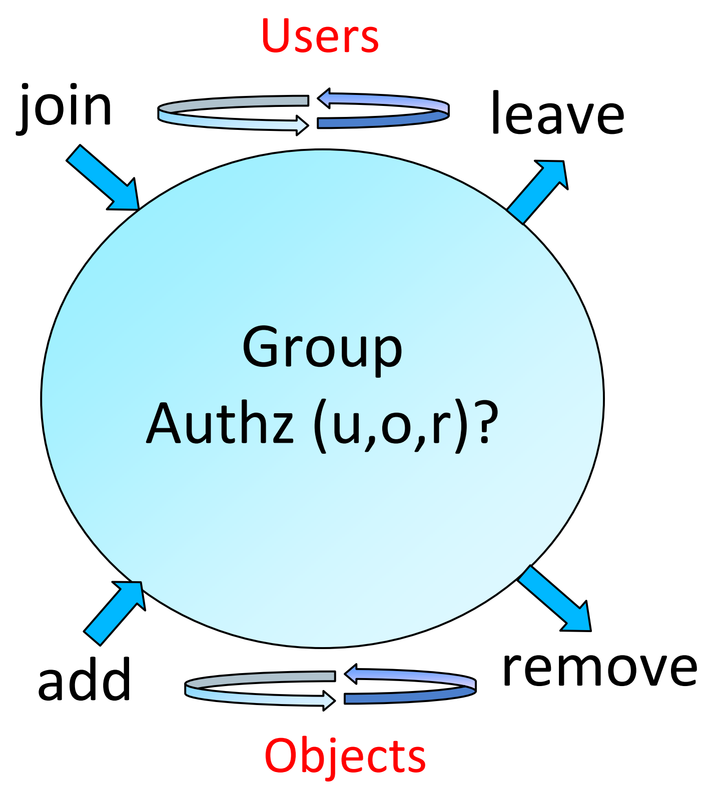
Dissemination Chain with Sticky Policies on Objects

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 ($\ominus p$)
  - Formula p held in the previous state
- Once p ($\blacklozenge p$)
  - 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 <u>J</u>oin, <u>L</u>eave, <u>A</u>dd and <u>R</u>emove to refer to some respective event type occurring

$$\begin{aligned}
\text{Join}(u) &= (\text{join}_1(u) \vee \text{join}_2(u) \vee \ldots \vee \text{join}_m(u)) \\
\text{Leave}(u) &= (\text{leave}_1(u) \vee \text{leave}_2(u) \vee \ldots \vee \text{leave}_n(u)) \\
\text{Add}(o) &= (\text{add}_1(o) \vee \text{add}_2(o) \vee \ldots \vee \text{add}_p(o)) \\
\text{Remove}(o) &= (\text{remove}_1(o) \vee \ldots \vee \text{remove}_q(o))
\end{aligned}$$

- Drop the parameters for convenience

$$\text{Authz} \rightarrow (\text{Join} \wedge (\neg(\text{Leave} \vee \text{Remove})))$$

$$\equiv$$

$$\forall u \in U. \forall o \in O. \text{Authz}(u, o, r) \rightarrow (\text{Join}(u) \wedge (\neg(\text{Leave}(u) \vee \text{Remove}(o))))$$

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

$$\underbrace{\left\{ \begin{array}{c} \text{join}_3(u_1) \\ \text{leave}_4(u_1) \end{array} \right\}}_{s_0} \underbrace{\left\{ \begin{array}{c} \text{add}_1(o_1) \\ \text{remove}_6(o_1) \end{array} \right\}}_{s_1} \underbrace{\left\{ \begin{array}{c} \text{join}_1(u_2) \\ \text{join}_6(u_2) \end{array} \right\}}_{s_2} \underbrace{\left\{ \begin{array}{c} \text{remove}_3(o_2) \\ \text{remove}_4(o_2) \end{array} \right\}}_{s_3}$$

E.g. 1 E.g. 2

✖ Malformed trace

- 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

$$\underbrace{\underset{s_0}{\{\text{leave}_1(u_1)\}} \; \underset{s_1}{\{\text{join}_1(u_1)\}}}_{\text{E.g. 1}} \; \underbrace{\underset{s_2}{\{\text{join}_1(u_2)\}} \; \underset{s_3}{\{\text{join}_2(u_2)\}}}_{\text{E.g. 2}}$$

✖ Malformed trace

LTL Specification of Well-Formed Traces

$$\tau_0 = \Box(\neg(\text{Add} \wedge \text{Remove}) \wedge \neg(\text{Join} \wedge \text{Leave}))$$

$$\begin{aligned}\tau_1 = &\forall i,j\ \Box((i \neq j) \rightarrow \neg(\text{join}_i \wedge \text{join}_j)) \wedge \\ &\forall i,j\ \Box((i \neq j) \rightarrow \neg(\text{leave}_i \wedge \text{leave}_j)) \wedge \\ &\forall i,j\ \Box((i \neq j) \rightarrow \neg(\text{add}_i \wedge \text{add}_j)) \wedge \\ &\forall i,j\ \Box((i \neq j) \rightarrow \neg(\text{remove}_i \wedge \text{remove}_j))\end{aligned}$$

$$\begin{aligned}\tau_2 = &\Box(\text{Join} \rightarrow \bigcirc\ (\neg\text{Join}\ \mathcal{W}\ \text{Leave})) \wedge \\ &\Box(\text{Leave} \rightarrow \bigcirc\ (\neg\text{Leave}\ \mathcal{W}\ \text{Join})) \wedge \\ &\Box(\text{Add} \rightarrow \bigcirc\ (\neg\text{Add}\ \mathcal{W}\ \text{Remove})) \wedge \\ &\Box(\text{Remove} \rightarrow \bigcirc\ (\neg\text{Remove}\ \mathcal{W}\ \text{Add}))\end{aligned}$$

$$\tau_3 = \Box(\text{Leave} \rightarrow \blacklozenge\text{Join}) \wedge \Box(\text{Remove} \rightarrow \blacklozenge\text{Add})$$

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

$$\gamma = \forall u \in U.\forall o \in O.\Box(\text{Authz}(u,o,r) \leftrightarrow \psi(u,o)) \wedge \bigwedge_{0 \le i \le 3} \tau_i$$

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

$$\gamma \vDash \bigwedge_{0 \leq i \leq 5} \varphi_i$$

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

LTL spec for Membership and Membership Renewal Properties (contd)

Operation	Explanation	Property
Strict Join (SJ)	Only objects added after join time can be accessed	$\alpha_0 = \Box(\text{Authz} \rightarrow \blacklozenge(\text{Add} \wedge (\neg\text{Leave}\ \mathcal{S}\ \text{join}_i)))$
Liberal Join (LJ)	Can access objects added before and after join time	There exists a well-formed trace that does not satisfy α_0
Strict Leave (SL)	Lose access to all objects on leave	$\alpha_1 = \Box(\text{Authz} \rightarrow (\neg\text{leave}_i\ \mathcal{S}\ \text{Join}))$
Liberal Leave (LL)	Retain access to objects authorized before leave time	There exists a well-formed trace that does not satisfy α_1
Strict Add (SA)	Only users who joined prior to add time can access	$\alpha_2 = \Box(\text{add}_i \rightarrow (\neg\blacklozenge\text{Join} \rightarrow (\neg\text{Authz}\ \mathcal{W}\ \text{Add})))$
Liberal Add (LA)	Users who joined before or after add time may access	There exists a well-formed trace that does not satisfy α_2
Strict Remove (SR)	All users lose access on remove	$\alpha_3 = \Box(\text{remove}_i \rightarrow (\neg\text{Authz}\ \mathcal{W}\ \text{Add}))$
Liberal Remove (LR)	Users who had access at remove time retain access	There exists a well-formed trace that does not satisfy α_3

Operation	Explanation	Property
Lossless Join	Authorizations prior to join time is not lost	$\beta_0 = \Box((\text{Join} \wedge \neg\text{Remove} \wedge \circleddash\ \text{Authz}) \rightarrow \text{Authz})$
Lossy Join	Authorizations from prior to join may be lost	There exists a well-formed trace that does not satisfy β_0
Non-Restorative Join	Authorizations from past membership periods not explicitly restored	$\rho_1 = (\text{join}_i(u1) \wedge \text{join}_i(u2) \wedge \text{Authz}(u1,o,r) \wedge \neg\text{Authz}(u2,o,r))$ $\rho_2 = \circleddash\ (\text{Authz}(u1,o,r) \wedge \neg\text{Authz}(u2,o,r))$ $\beta_1 = \forall i \Box(\rho_1 \rightarrow \rho_2)$
Restorative Join	Authorizations from past membership may be restored	There exists a well-formed trace that does not satisfy β_1
Gainless Leave	Authorizations that never held during most recent membership period cannot be obtained	$\beta_2 = \Box((\text{Leave} \wedge (\neg\text{Join}\ \mathcal{U}\ (\text{Authz} \wedge \neg\text{Join}))) \rightarrow \circleddash\ ((\neg\text{Authz} \wedge \neg\text{Join})\ \mathcal{S}\ (\text{Authz} \wedge (\neg\text{Join}\ \mathcal{S}\ \text{Join}))))$
Gainful Leave	New authorizations may be granted at Leave time	There exists a well-formed trace that does not satisfy β_2
Non-Restorative Leave	Authorizations that the user had prior to joining the group are not explicitly restored	$\beta_3 = \Box(\text{Leave} \wedge \text{Authz} \rightarrow \circleddash\ \text{Authz})$
Restorative Leave	Authorizations from prior to join time may be restored	There exists a well-formed trace that does not satisfy β_3

Verification Using Model Checker

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

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

- 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

No Versioning	Versioning
1. Multiple users may update, latest write is committed (destructive write).	1. Multiple users may update, each update creates a new version.
2. No write after leave.	2. No write after leave.
3. Coarse-grained authorization (specified on the whole object).	3. Fine-grained. Authorization can differ for different versions of the same object.
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.

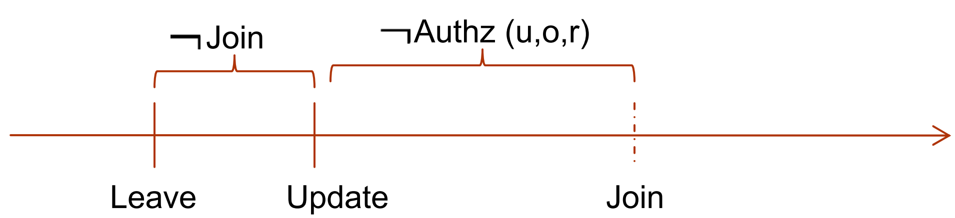
Core Properties (no versioning)

- New Operation: Update(o)
- Provenance Both user and object should be current members

$$\Box(\mathrm{Authz}(u,o,w) \rightarrow (\mathrm{Authz}(u,o,r) \wedge (\neg \mathrm{Leave}\ \mathcal{S}\ \mathrm{Join}) \wedge (\neg \mathrm{Remove}\ \mathcal{S}\ \mathrm{Add})))$$

- Bounded Authorization
  - Past users cannot read after update

$$\Box(\mathrm{Update} \wedge (\neg \mathrm{Join}\ \mathcal{S}\ \mathrm{Leave}) \rightarrow (\neg \mathrm{Authz}(u,o,r)\ \mathcal{W}\ \mathrm{Join}))$$



  - Past users cannot write. No write on past objects.

$$\Box(\mathrm{Leave} \rightarrow \forall o.\neg \mathrm{Authz}(u,o,w)\ \mathcal{W}\ \mathrm{Join})$$

$$\Box(\mathrm{Remove} \rightarrow \forall u.\neg \mathrm{Authz}(u,o,w)\ \mathcal{W}\ \mathrm{Add})$$

Core Properties (versioning)

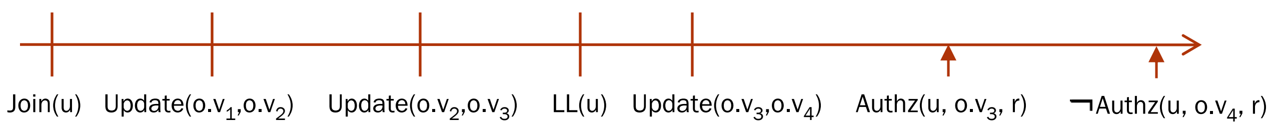
- New operation: Update($o.v_i$, $o.v_j$)

Version to update

New updated Version

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

$$\Box(\mathrm{Authz}(u, o.v_i, w) \rightarrow (\mathrm{Authz}(u, o.v_i, r) \wedge (\neg \mathrm{Leave}\ \mathcal{S}\ \mathrm{Add}) \wedge (\neg \mathrm{Remove}\ \mathcal{S}\ (\mathrm{Add} \vee \exists v_j.\mathrm{Update}(o.v_j, o.v_i)))))$$



- Bounded Authorization

$$\Box(\mathrm{Leave}(u) \rightarrow (\forall o.\forall v_i.\neg \mathrm{Authz}(u, o.v_i, w)\ \mathcal{W}\ \mathrm{Join}))$$

$$\Box(\mathrm{Remove}(o.v_i) \rightarrow (\forall u.\neg \mathrm{Authz}(u, o.v_i, w)\ \mathcal{W}\ \mathrm{Join}))$$

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($o.v_i$, $o.v_j$)

Version to update

New updated Version

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: Update($o.v_i$, $o.v_j$)

Version to update

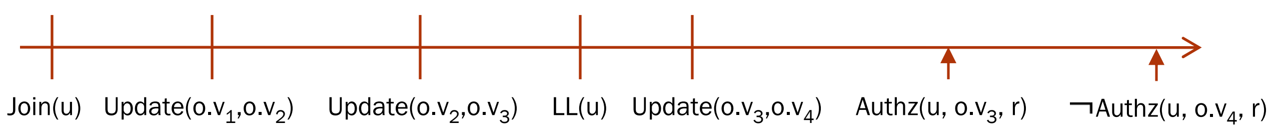
New updated Version

- Authorization Persistence

$$\Box(\text{Authz} \rightarrow (\text{Authz}\ \mathcal{W}\ (\text{Join} \vee \text{Leave} \vee \text{Add} \vee \text{Remove} \vee \text{Update})))$$
$$\Box(\neg\text{Authz} \rightarrow (\neg\text{Authz}\ \mathcal{W}\ (\text{Join} \vee \text{Leave} \vee \text{Add} \vee \text{Remove} \vee \text{Update})))$$

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

$$\Box(\text{Authz}(u, o.v_i, w) \rightarrow (\text{Authz}(u, o.v_i, r) \wedge (\neg\text{Leave}\ \mathcal{S}\ \text{Add}) \wedge (\neg\text{Remove}\ \mathcal{S}\ (\text{Add} \vee \exists v_j.\text{Update}(o.v_j, o.v_i)))))$$



Core Properties (continued)

- Bounded Authorization

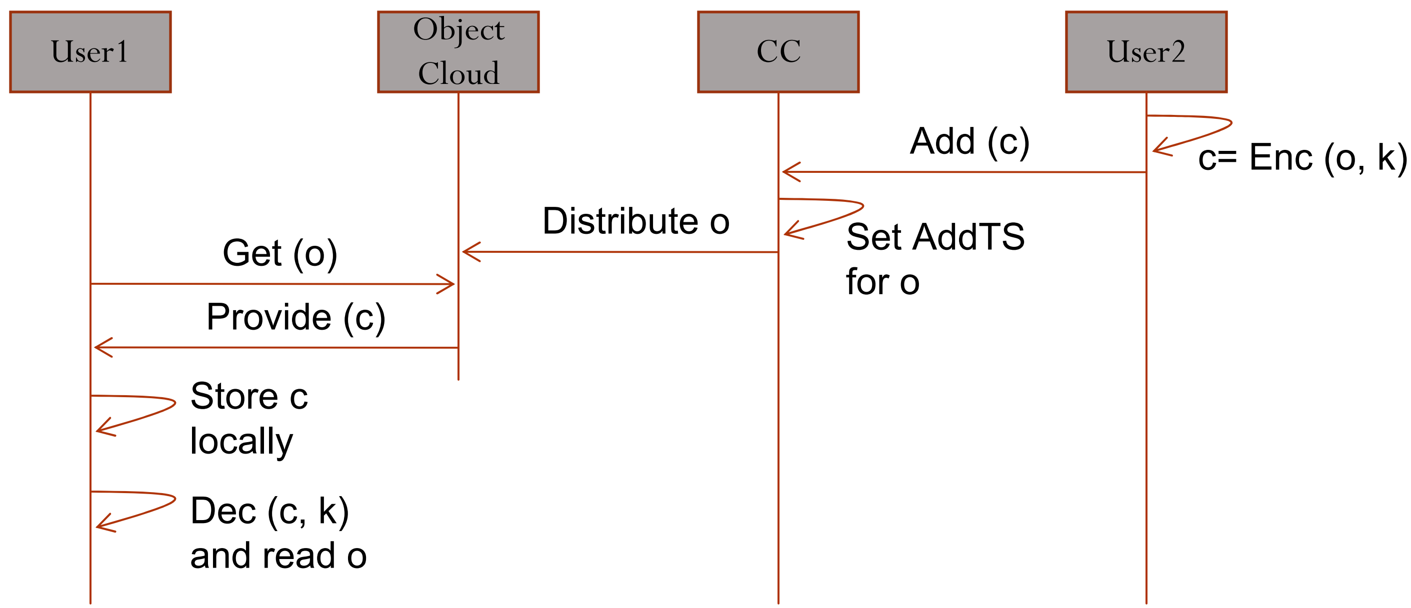
$$\Box(\text{Leave}(u) \rightarrow (\forall o.\forall v_i.\neg\text{Authz}(u, o.v_i, w)\ \mathcal{W}\ \text{Join}))$$

$$\Box(\text{Remove}(o.v_i) \rightarrow (\forall u.\neg\text{Authz}(u, o.v_i, w)\ \mathcal{W}\ \text{Join}))$$

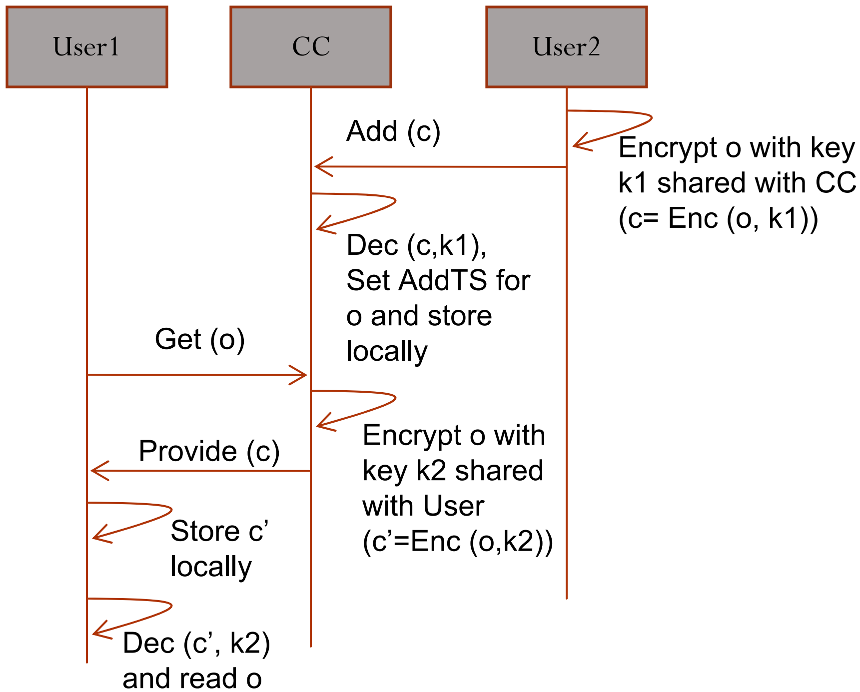
- Availability

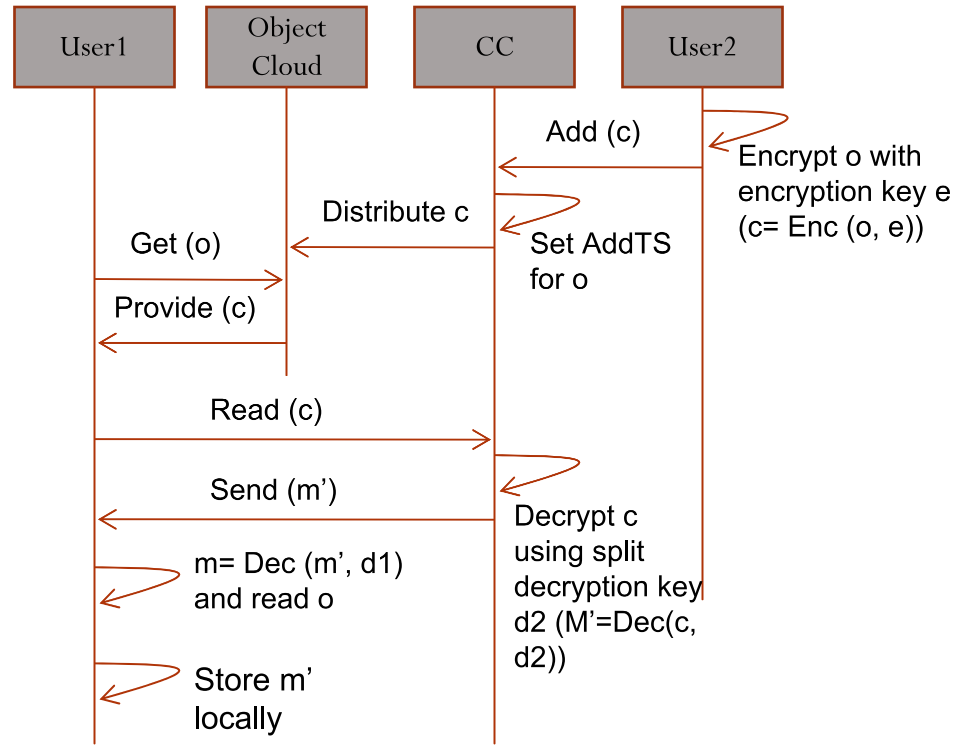
$$\Box(\text{Join} \rightarrow (\text{Add}\vee\text{Update}(o.v_i, o.v_j) \rightarrow \text{Authz}(u, o.v_j, w)\ \mathcal{W}\ \text{Leave})\ \mathcal{W}\ \text{Leave})$$

Super-distribution



Micro-Distribution and Hybrid Approach

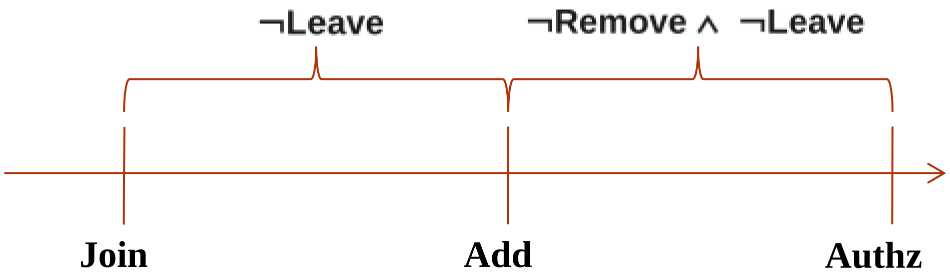




Micro-distribution:
Obtain custom encrypted object from CC the first time
Subsequent accesses can be offline

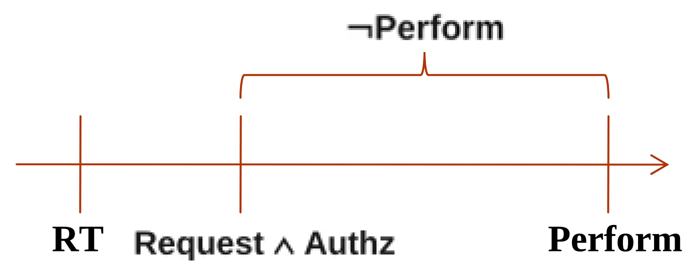
Hybrid approach:
Split-key RSA, One split per user
CC participates in initial decryption
Subsequent accesses can be offline

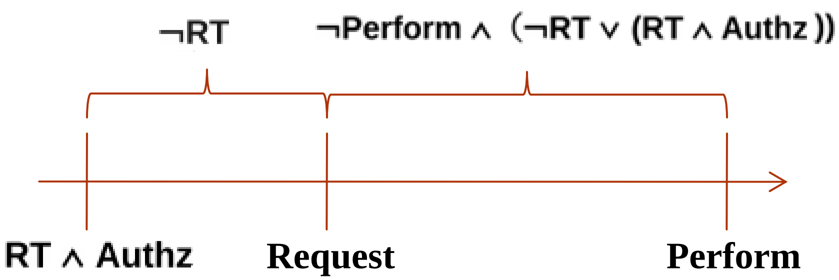
Properties



$$\text{Authz} \equiv ((\neg\text{Remove} \wedge \neg\text{Leave})\ \mathcal{S}\ (\text{Add} \wedge (\neg\text{Leave}\ \mathcal{S}\ \text{Join})))$$

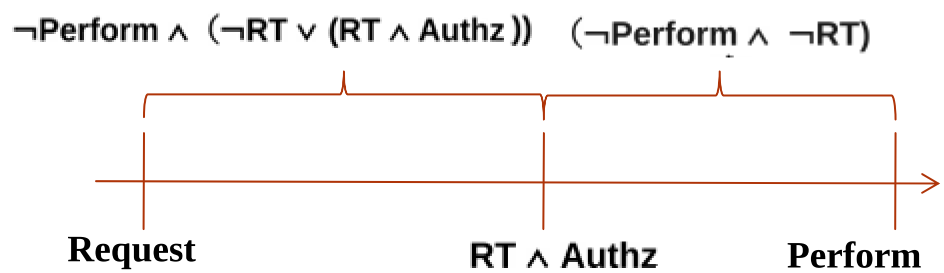
Stale-unsafe Decision





Formula φ_1

$$\varphi_1 \equiv \odot(\neg\text{perform} \wedge (\neg\text{RT} \vee (\text{RT} \wedge \text{Authz})))\ \mathcal{S}\ (\text{request} \wedge (\neg\text{RT}\ \mathcal{S}\ (\text{RT} \wedge \text{Authz})))$$



Formula φ_2

$$\varphi_2 \equiv \odot(\neg\text{perform} \wedge \neg\text{RT})\ \mathcal{S}\ (\text{RT} \wedge \text{Authz} \wedge ((\neg\text{perform} \wedge (\neg\text{RT} \vee (\text{RT} \wedge \text{Authz})))\ \mathcal{S}\ \text{request}))$$

Weak Stale-Safety: $\Box(\text{perform} \rightarrow (\varphi_1 \vee \varphi_2))$

Strong Stale-Safety: $\Box(\text{perform} \rightarrow \varphi_2)$

Verification (continued)

- Let
  - $\cent_0$: composition of stale-unsafe FSMs
  - $\cent_1$: composition of weak stale-safe FSMs
  - $\cent_2$: composition of strong stale-safe FSMs
- Verified using model checking that:
  - Authz is enforced by $\cent_0$, $\cent_1$ and $\cent_2$
  - $\cent_0$ fails Weak and Strong Stale-Safe security properties
  - $\cent_1$ satisfies Weak Stale-Safe security property
  - $\cent_1$ fails Strong Stale-Safe security property
  - $\cent_2$ satisfies Strong Stale-Safe security property

Strong Stale-Safe Machine