Formal Models for Group-Centric Secure Information Sharing

March 23, 2009
Technical Report CS-TR-2009-002
Department of Computer Science
The University of Texas, San Antonio, TX 78249.

Ram Krishnan[1], **Ravi Sandhu**[2], **Jianwei Niu**[3], **William H. Winsborough**[4]
[1] George Mason University, [2,3,4] University of Texas at San Antonio

ABSTRACT

We develop the foundations for a theory of Group-Centric Secure Information Sharing (g-SIS), characterize a specific family of models in this arena and identify several directions in which this theory can be extended. The traditional approach to information sharing, characterized as Dissemination-Centric in this paper, focuses on attaching attributes and policies to an object (sometimes called "sticky policies") as it is disseminated from producers to consumers in a system. In contrast, Group-Centric sharing envisions bringing the subjects and objects together in a group to facilitate sharing. The metaphor is that of a secure meeting room where participants and information come together to enable parties to "share" information for some common purpose.

We formalize the concept of an Information-Sharing Group using Linear Temporal Logic (LTL), by specifying g-SIS properties. We begin with a core set of properties (Simultaneity, Provenance, Persistence, Availability, etc.) that any g-SIS model must satisfy. Next we identify additional properties regarding specific variations of group operations (Strict, Liberal, Lossy, Lossless, etc.). Finally, we specify the correct authorization behavior for a sub-family of g-SIS specifications using LTL and formally prove that the specifications satisfy the properties using the model checker NuSMV.

1. INTRODUCTION AND MOTIVATION

This paper introduces and formalizes the concept of Group-Centric Secure Information Sharing (g-SIS). The need to share information is driven by multiple forces. Post 9/11, the need-to-share principle has supplanted the traditional need-to-know which caused failure to "connect the dots." In our information age, businesses collaborate not only with allies, but also with competitors to advance their own interests. For the individual citizen, modern healthcare requires timely sharing of medical information amongst care providers while maintaining privacy. The explosive phenomena of social networking enables individuals to interact without geographic barriers, but with an expectation of security and privacy.

In all of these cases, we see the need to "share but protect." This paper will develop formal models and analysis for a style of information sharing that we call Group-Centric. Intuitively, users and information come together in a group to facilitate sharing. We identify two metaphors: a secure meeting room and a subscription service. A meeting room brings people together to "share" information for some common purpose. The purpose can range from collaboration on a specific goal-oriented task (such as designing a new product) to participation in a shared activity (such as a semester long class) to a dynamic coalition (such as a mission-oriented group that is driven towards completion of a particular task). Subscription metaphor speaks to a potentially larger scale sharing with a publisher disseminating information to subscribers who in turn participate in blogs and forums. We show that these simple and familiar metaphors enable a rich space of policies that will be investigated in a systematic incremental fashion. In particular, we show that the temporal interactions of users joining and leaving the group and information being added and removed is critical to determination of who can see what in the group.

Secure meeting room Visualize a conversation room (albeit virtual) where users may join, leave and re-join, but only hear the conversation occurring during their participation period. In general, a meeting room has the notion of simultaneous presence of participants engaged in the meeting. For simplicity, we will also use this metaphor to include the idea of a "secure document room", such as often used during merger and acquisition. Users bring documents to this room wherein they are asynchronously accessible by participants from different stakeholders and third-party agents. Users' participation may be intermittent as influenced by their availability, need to know, etc. Let us consider some example scenarios:

Program committee meeting: In a program committee meeting, members are privy only to conversations occurring in their presence. Alice, a committee member, may be excused from the room when her paper is being discussed and may re-join the room after that discussion has concluded. The conversation that occurred during her absence is not accessible to her. In a different setting, all conversations are recorded on a smartboard in the room and as Alice re-joins she is able to see what happened during her absence.

Collaborative product development: Consider collaborative product design between ABC Corp. and XYZ Corp. Say ABC establishes a group by pulling in engineers from across the company. Certain sensitive documents are provided to these ABC engineers, but these are not accessible to XYZ engineers. Documents created after the XYZ engineers join the group are shared by both ABC and XYZ engineers. Moreover, both XYZ and ABC engineers retain access to such new documents even after leaving the group. In a different consulting scenario, incoming XYZ engineers may access the sensitive ABC documents during their membership period, but lose access once the collaboration ends.

Employee stock options: Stock option benefits typically change over time. New employees only get to see benefits as of their joining and not previously existing ones. Similarly, organizations may share some information with existing employees, but withhold it from future employees. In these cases, certain objects are shared only with existing group members and not with future members. Furthermore, when employees leave the company, they may be allowed to retain certain information (such as their profile, recommendations, etc.), but denied access to sensitive proprietary information (such as design documents, code, etc.).

Subscription service Another metaphor is that of the subscription service where access to content may depend upon when the subscription began and the terms of subscription. The publisher may contribute information that the participants read and possibly respond to via content in associated blogs and forums.

Magazine Subscription: Consider an online news magazine ABS that offers four levels of membership. Level 1 ($10/year) subscribers can access news articles that are published after they started paying the subscription fee. If they cancel their subscription, they completely lose access. Level 2 ($12/year) is similar to Level 1 except subscribers can retain access to news articles that they paid for even after canceling their subscription. In addition to Level 2 services, Level 3 ($15/year) subscribers can access rich archives filled with post-news analysis, predictions, annotations and opinions from experts. But if they cancel their subscription, they lose all access. Level 4 ($17/year) is similar to Level 3, except even after canceling membership, subscribers can login and view all articles that they had access before leaving.

Secure multicast: In secure multicast [31], new members joining the group cannot access content transmitted prior to their join time. This is referred to as "backward secrecy". Similarly, members leaving the group can no longer access any new content. This is referred to as "forward secrecy". Thus access is dependant on when the subjects join and leave the group.

Clearly, the "secure meeting room" metaphor suggests a smaller-scale information sharing scenario whereas the "subscription service" metaphor indicates a potentially larger-scale. These examples illustrate two important principles in the group-centric approach. The first principle is "share but differentiate". Sharing is enabled by joining and adding information to group. Yet, users' access is differentiated by the time at which they join and the time at which the requested information is added to the group, as well as possibly by other attributes. The second principle is the notion of "groups within groups". That is, in a given g-SIS system, there may be multiple related groups. The relationship between these groups can be of any number of varieties familiar to computer scientists. One well-known structure is that of a hierarchy, where subjects at a higher level dominate those at the lower levels in terms of read access. Another common relationship is that of mutual exclusion where the same user is prohibited from joining conflicting groups.

Methodology and Focus.

Methodologically, we believe that secure systems design is best accomplished in two layers enabling consideration of system design issues a few at a time. It is useful to consider the interaction of operations that affect authorizations independent of the way in which those operations affect some particular notion of authorization state. Thus, instead of specifying the effect of various operations on an authorization state, we specify the circumstances in which an action is authorized directly in terms of the operations that have come before. To this end, we propose to use temporal logic as a precise, well understood means to specify and analyze those circumstances. Thereby alternative designs can be specified with precise semantics, compared, and studied with respect to their suitability for a given application. The designer's intuitions about dependencies among design characteristics can be captured and assessed. Informed by such analysis, the designer can select among the various operations and associated semantic specifications to define what we think of as the *application policy layer* of the system design [35]. Of course, real systems have specific representations of authorization state, which are modified by performing operations and examined to determine whether an action is permitted. This is part of the design of what we characterize as the *enforcement layer* of the system design. In general, but particularly in distributed systems, design decisions must be made at this layer that have a significant impact on security and functionality. In this paper, we focus solely on the *application policy layer*. A complete analysis of enforcement layer design issues is out of scope for this paper.

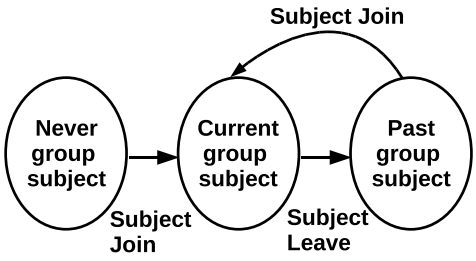

Figure 1: Subject Membership States.

Our focus in this paper is on semantics of the basic group operations and their temporal interactions. We propose an abstract set of group operations: Join and Leave for subjects, Add and Remove for objects. Subjects may Join, Leave and re-Join the group. This is illustrated in figure 1. Similarly, objects may be Added, Removed and re-Added to the group. Further each of these operations could be of various types such as Lossy/Lossless, Restorative/Non-Restorative, etc. For example, in Lossless Join, a joining subject never loses access to objects authorized prior to joining the group. Similarly, in Restorative Join, the joining subject may regain access to objects authorized during past membership period. There may also be additional operations such as post-dated Join, back-dated Join, substitution of one subject by another, etc. One of the goals of our future work is to formally analyze the expressive power of these operations, and identify the essential ones versus those that add convenience but no new fundamental expressive power (see for e.g., [11, 37]). In general, there may be any number of such variations beyond those explicitly identified in this paper. Temporal aspects of access control have been previously studied (e.g., [10]), where temporal aspects are introduced as extensions to prior models of role based access control. In g-SIS the temporal interactions are so fundamental that they must be investigated and understood before even the first model can be formulated.

We recognize the importance of authorization for these operations. It is clearly not sufficient for a security policy to specify the semantics of Join, Leave, Add and Remove. A complete policy must also specify the authorization for these operations. In simple cases, a distinguished group owner may be responsible for all of these operations. More realistically the authorization will be decentralized and distributed. The problem of decentralized authorization and its administration has been investigated in the access control literature for over three decades. The formal treatment of this area began with the landmark paper of HRU [19] that showed that so-called safety analysis in even this simple model was undecidable. Numerous papers on models such as take-grant [26], SPM [32], TAM etc. [33, 20] followed. With the growing popularity of role-based access control in the 1990's, decentralized administrative models were proposed in that context [34]. Clearly, the area of decentralized authorization and its administration has been extensively studied. We hypothesize that suitable administration models for g-SIS can be developed utilizing the insights of this accumulated previous work. These will necessarily be application dependant. Consider two different g-SIS applications, one where users need to pay to join a group and another where users are admitted based on organizational needs. The administrative model for these two applications is likely to be different.

We believe that authorizations concerning the operational aspects that bear on group membership is a more immediate and

novel problem, and this will be the focus of this research.[1] Without a basic understanding of the semantics of group operations, we believe that it would be premature to consider administrative models. Henceforth, we leave the development of an administrative g-SIS model for future work.

In this paper we develop the foundations for a theory of Group-Centric Information Sharing, characterize a specific sub-family of models in this arena and identify several directions in which this theory can be extended. The principal contributions of this paper are as follows. We formalize the concept of Group-Centric Information Sharing using Linear Temporal Logic (LTL) [27], by specifying three levels of properties. We first specify the Core Properties (Simultaneity, Provenance, Persistence, etc.) that must be satisfied by any g-SIS specification. Next we specify, the Membership and Membership Renewal Properties which are based on specific variations of group operations. These properties need not be satisfied by all g-SIS specifications but suggest some useful operational semantics for many applications. Finally, we specify the correct authorization behavior for a sub-family of g-SIS specifications[2] and show using model checking that they satisfy the properties.

The remainder of this paper is organized as follows. In section 2, we discuss related work. In section 3, we discuss the formal g-SIS language and various g-SIS properties. In section 4, we discuss a sub-family of g-SIS specifications and show by using model checking that the specifications satisfy the Core Properties. In section 5, we re-visit our metaphors and discuss how the group operations can express many scenarios. In section 6, we identify several directions in which this work can be extended and conclude.

2. RELATED WORK

The traditional approach to information sharing, which we characterize as Dissemination-Centric in this paper, focuses on attaching attributes and policies to an object as it is disseminated from producers to consumers in a system. These policies are sometimes described as being "sticky". As an object is disseminated further down a supply chain the policies may get modified, such modification itself being controlled by existing policies. This mode of information sharing goes back to early discussions on originator-control systems [18, 28, 4, 29] in the 1980's and Digital Rights Management in the 1990's and 2000's. XrML [1], ODRL [3] and XACML [2] are recent examples of policy languages developed for this purpose. Dissemination-Centric Sharing describes in advance the characteristics of subjects who may access the object by attaching "sticky policies" to be enforced when a subject accesses the object. The vision of Group-Centric Sharing differs in that it advocates bringing the subjects and objects together to facilitate sharing.

We envision that Dissemination-Centric and Group-Centric Sharing will co-exist in a mutually supportive manner. For example, objects could be added with "sticky" policies in a Group-Centric model. In this case, the objects may have controls imposed by both the Group-Centric model and the "sticky policies". Also, the "sticky policies" on the object could determine whether or not an object can be added to the group in the first place. It may turn out that at a theoretical level whatever Dissemination-Centric can achieve Group-Centric can also achieve and vice versa. But at a pragmatic level, we believe these are significantly different approaches to information sharing.

Older approaches to Secure Information Sharing (SIS) can be classified into at least three categories. First is Discretionary Access Control (DAC) [17, 25, 16] which proposes to enforce controls on sharing information at the discretion of the "owner" of the object. Although, this is similar in objective to SIS, DAC does not correlate the controls on copies with that of the original.

The second is Mandatory Access Control (MAC) [8, 15, 16] which allows information to flow in one direction in a lattice of security labels. Copies of information made from one or more objects inherit the least upper bound of the labels from the individual objects. Thereby the copies are controlled at least as strictly as the original. Historically, one directional information flow has not been the most common requirement of SIS. In particular, MAC does not allow the owner of the object to share information but also to protect it from other users in the same or higher security levels. MAC also suffers from covert channel issues whereby information flow contrary to the labels can occur via malware.

The third is Originator Control or ORCON [18, 28, 4, 29] in which the owner of the object decides which user(s) may have access to it. The owner is the principal source of the policy to be enforced. As information flows from one container to another, the policy is also propagated. In other words, it is a "sticky policy".

Recently, information sharing challenges have been considered in the context of Dynamic Coalition Problem or DCP (see [30, 12, 21, 22, 7, 38] for example). The DCP is concerned with the challenges involved when a coalition is dynamically formed, for example, in response to a crisis. Government, civilian and other commercial organizations may need to form a coalition (who may otherwise distrust each other) and share information quickly to solve the problem at hand. In [9], the authors provide a formal temporal authorization model that focuses on database management systems. Specifically, the model extends traditional authorizations with the notion of temporal intervals of their validity. In [5], the authors use a role-based delegation framework for specifying policies for resource and information sharing within and across organizations. Our work largely differs from all these approaches in that we solely focus on the policy models for the group-centric SIS problem. Further, formal specification of g-SIS properties using LTL enables us to automate verification using model checking techniques.

To the best of our knowledge, this is the first effort towards developing a formal model for the group-centric approach to SIS. At a policy level, the closest work to group-centric sharing that can be found in the literature is in the area of Secure Multicast [31]. It will be evident later that the g-SIS specifications subsume policies considered by Secure Multicast. A conceptual framework of the Group-Centric approach was presented in [24] and [23].

3. FORMAL REQUIREMENTS OF G-SIS SYSTEMS

In this section, we present a collection of core properties that must be satisfied by any g-SIS system. These properties characterize a g-SIS system. After that, we discuss several orthogonal aspects of candidate g-SIS operation semantics and provide specifications for a specific sub-family of these operation semantics. We begin by defining the g-SIS language that we use to state g-SIS properties and specifications.

[1] We propose to use temporal logic for specifying the semantics of these operations, primarily because of the importance of temporal interactions between these. An alternative might be to use more native access control formalisms such as HRU [19]. It would be an interesting exercise to recast the results we obtain in this research within the HRU formalism perhaps as a prelude to developing formal administrative models. Nevertheless we feel that starting our investigation with a HRU-style formalism would not be productive.

[2] We confine our attention to correct authorization behavior with respect to *read* access in a *single group* using LTL. Our future work involves extensions to other forms of accesses such as *write* or *update* and *multiple groups*.

Table 1: Intuitive summary of temporal operators used in this paper

Future/Past	Operator	Read as	Explanation
Future	$\bigcirc$	Next	($\bigcirc$ p) means that the formula p holds in the next state.
	$\Box$	Henceforth	($\Box$ p) means that the formula p will continuously hold in all future states starting from the current state.
	$\mathcal{U}$	Until	(p $\mathcal{U}$ q) means that q will occur sometime in the future and p will hold at least until the first occurrence of q.
	$\mathcal{W}$	Unless	(p $\mathcal{W}$ q) is a weaker form of (p $\mathcal{U}$ q). It says that p holds either until the next occurrence of q or if q never occurs, it holds throughout.
Past	$\ominus$	Previous	($\ominus$ p) means that formula p held in the previous state.
	$\blacklozenge$	Once	($\blacklozenge$ p) means that formula p held at least once in the past.
	$\mathcal{S}$	Since	(p $\mathcal{S}$ q) means that q happened in the past and p held continuously from the position following the last occurrence of q to the present.

3.1 g-SIS Language

We use Linear Temporal Logic to characterize g-SIS properties and specifications. A brief overview of temporal operators used in this paper is given in table 1. To formalize the LTL language we use and its semantics, suppose S is a finite set of subjects, O is a finite set of objects, and R is a finite set of actions, such as read and write. Let P be a set of predicates over sorts S, O and/or R, and let $\{A, B\}$ be a partition of P. Predicates in A are called actions and intuitively encode actions or events that occurred in the transition to the current state. Predicates in B are used to encode aspects of a given state, such as operations that are authorized or not authorized. We let $\mathcal{F}$ be the set of atomic formulas obtained by applying a predicate $p \in P$ to a list of arguments of the appropriate number and sorts. LTL formulas are constructed from $\mathcal{F}$ by applying logical connectives and temporal operators in the usual way.

For the purpose of this paper, a *g-SIS language* is required to satisfy the following[3]. It must include a collection of join-group events, leave-group events, add-object events, and remove-object events: $A = \{\text{join}_i | 1 \leq i \leq m\} \cup \{\text{leave}_i | 1 \leq i \leq n\} \cup \{\text{add}_i | 1 \leq i \leq p\} \cup \{\text{remove}_i | 1 \leq i \leq q\}$, $B = \{\text{Authz}\}$, and $R = \{r\}$, where *r* refers the right to exercise "read" operations. Also, an atomic formula in a g-SIS language should be formed in the natural way: for all $s \in S$, $o \in O$, $r \in R$, $\text{join}_i(s), \text{add}_i(o), \ldots, \text{Authz}(s, o, r) \in \mathcal{F}$.

Formally, a state is a function from variable-free formulas $\mathcal{F}$ into the set $\{\text{True}, \text{False}\}$. We use Σ to denote the set of all states. A *trace* σ is an infinite sequence of states, that is, it is an ω-sequence in Σ^ω. In the following, we often wish to write sub-formulas that state, for example, some type of join event occurs. It is therefore convenient to introduce the following shorthands:

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

The properties we consider treat the authorization a subject has to access an object independently of actions involving other subjects and objects. Thus, it is often convenient to omit the parameters in all of the predicates. For instance, when we write $\text{Authz} \rightarrow (\text{Join} \wedge (\neg(\text{Leave} \vee \text{Remove}) \ \mathcal{S} \ \text{Add}))$ we mean $\forall s \in S. \forall o \in O. \text{Authz}(s, o, r) \rightarrow (\text{Join}(s) \wedge (\neg(\text{Leave}(s) \vee \text{Remove}(o)) \ \mathcal{S} \ \text{Add}(o)))$. Note that Join, Leave, Add, Remove and Authz, all refer to the same pair of *s* and/or *o*. In addition to using these shorthands in formulas, we continue to use these words to informally refer to intuitive notions of corresponding operations.

[3]We suggest that the language represented here should be a sub-language of any g-SIS language designed in the future.

Well-Formed Traces.

We now introduce four formulas that define what we call *well formed* traces.

A. An object cannot be Added and Removed and a subject cannot Join and Leave at the same time[4].

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

B. For any given subject or object, two types of operation cannot occur at the same time.

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

C. If a subject s joins a group, s cannot join again unless s first leaves the group. A similar rule applies for other operations.

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

D. A Leave event cannot occur before Join. Similarly for objects.

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

3.2 Core g-SIS Properties

We begin with the Core properties, all of which must be satisfied by any g-SIS specification[5]. Next we specify a few useful additional properties. We specify these properties with the assumption that Join, Leave, Add and Remove are the only events that influence a g-SIS specification. If need be, these properties can be

[4]Note that here and below we introduce names of the form τ_j for each of the formulas for later reference. The equality introduces shorthands for the respective formulas

[5]While we do not claim that these properties are complete, the notions of simultaneity, provenance, persistence, bounded authorization and availability are nevertheless core to g-SIS. It is beyond our current scope to discuss completeness of core g-SIS properties.
Note that each of these formulas define a *safety* property [6, 36] in the sense that any trace that does not satisfy the property can be recognized as such by examining a finite prefix of the trace.

extended to g-SIS specifications involving additional aspects (such as a "role") in the context of a given application.

1. Overlapping Membership (Simultaneity) Property: Intuitively, a group subject can access a group object only if they were both members of the group at least once: $\Box(\text{Authz} \rightarrow (\blacklozenge\text{Add} \wedge \blacklozenge\text{Join}))$. However, we further wish to require that the subject and object were *simultaneously* members of the group at some point in the past. This is formalized in φ_0 below.

$$\varphi_0 = \Box(\text{Authz} \rightarrow \blacklozenge(\text{Add} \wedge (\neg\text{Leave } \mathcal{S}\ \text{Join})) \vee \blacklozenge(\text{Join} \wedge (\neg\text{Remove } \mathcal{S}\ \text{Add})))$$

This formula states that if a subject is able to access an object then there exists a point in the past ($\blacklozenge$) where (a) the object was added at a time when the subject was a current member ($\neg\text{Leave } \mathcal{S}\ \text{Join}$) *or* (b) the subject joined the group at a time when the object was a current member ($\neg\text{Remove } \mathcal{S}\ \text{Add}$).

2. Authorization Provenance: Authorization Provenance requires that authorization cannot begin for the first time during a non-membership period. The Overlapping Membership property does not capture this notion.

$$\varphi_1 = \Box(\text{Authz} \rightarrow \blacklozenge(\text{Join} \wedge (\neg\text{Leave } \mathcal{U}\ \text{Authz})))$$

This formula states that if Authz holds in any state, then it should have held during the subject's membership (sometime after subject Join) at least once in the past.

3. Persistence Properties: These properties talk about the conditions under which authorization periods may begin and end.

Authorization Persistence: When a subject s is authorized to access an object o, it remains so at least until a group event involving s or o occurs[6].

$$\varphi_2 = \Box(\text{Authz} \rightarrow (\text{Authz } \mathcal{W}\ (\text{Join} \vee \text{Leave} \vee \text{Add} \vee \text{Remove})))$$

Revocation Persistence: When a subject s is not authorized to access an object o, it remains so at least until a group event involving s or o occurs.

$$\varphi_3 = \Box(\neg\text{Authz} \rightarrow (\neg\text{Authz } \mathcal{W}\ (\text{Join} \vee \text{Leave} \vee \text{Add} \vee \text{Remove})))$$

4. Bounded Authorization: These properties require that authorizations not begin during non-membership periods of subjects and objects. Note that periods of authorization and non-authorization begin in the state in which the Join, Leave, Add or Remove occurs.

Bounded Subject Authorization: The set of all objects that a subject can access during non-membership periods is bounded at Leave time. This set cannot grow until the subject rejoins.

$$\varphi_4 = \Box((\text{Leave} \wedge \neg\text{Authz}) \rightarrow (\neg\text{Authz } \mathcal{W}\ \text{Join}))$$

[6]Note that a subject may Join a group, Leave subsequently and retain access to some objects. It is possible this subject may lose access to these objects by Joining the group again. We call this a Lossy Join where joining a group may actually require a subject to relinquish prior authorizations. This is the reason we include enabling operations such as Join and Add in the $\mathcal{W}$ (unless) predicate in the above formula. In general, there are many flavors of group operations that will be discussed in detail in the following sections.

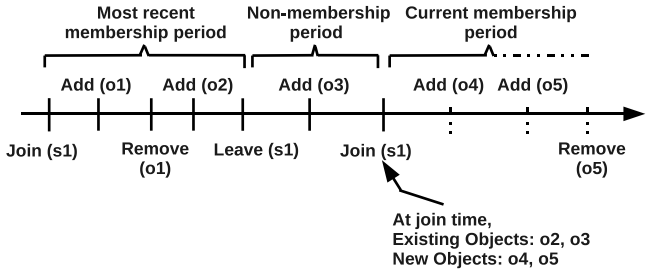

Figure 2: Subject Operations Illustration.

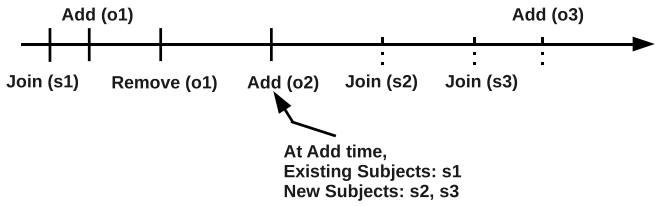

Figure 3: Object Operations Illustration.

The above property states that new authorizations cannot be granted to a subject during non-membership period. Any object that is accessible after Leave should have been authorized at the time of Leave.

Bounded Object Authorization: The set of all subjects who can access a removed object is bounded at Remove time, which cannot grow until re-Add.

$$\varphi_5 = \Box((\text{Remove} \wedge \neg\text{Authz}) \rightarrow (\neg\text{Authz } \mathcal{W}\ \text{Add}))$$

5. Availability: Availability specifies the conditions under which authorization *must* succeed.

$$\varphi_6 = \Box(\text{Join} \rightarrow ((\text{Add} \rightarrow \text{Authz})\ \mathcal{W}\ \text{Leave}))$$

This property states that after a subject joins a group, any object that is added subsequently should be authorized. Obviously, the subject should be a current member when the object in question is added. Note that this authorization will persist as guided by the Authorization Persistence Property.

We next discuss additional properties that are based on specific variations of group operations. Unlike the core properties, not all g-SIS specifications are required to satisfy these properties. Instead, these properties define certain group operation semantics that are useful for many applications. We classify these properties into two categories: Membership and Membership Renewal Properties.

3.3 Membership Properties

Membership properties characterize the semantics of authorizations enabled when a subject joins or an object is added and those which are disabled when a subject leaves or an object is removed from the group. In the following subsection (Membership Renewal Properties), we consider properties when a subject or an object is re-admitted.

Strict Join (SJ) Vs Liberal Join (LJ): In SJ, the joining subject can only access those objects added after Join time. LJ additionally allows the subject to access objects that were added prior to join time. Suppose that in figure 2 the second Join ($s1$) is an SJ. Then $s1$ can access $o4$ and $o5$ and cannot access $o3$. If the Join was an

LJ instead of SJ, $s1$ can also access $o2$ and $o3$. This can be formalized by requiring that join_i, a type of Join, would be admitted as SJ only if it satisfies α_0 stated below:

$$\alpha_0 = \Box(\text{Authz} \rightarrow \blacklozenge(\text{Add} \wedge (\neg\text{Leave } \mathcal{S}\ \text{join}_i)))$$

In a g-SIS specification with LJ, there exists at least one well-formed trace for which Authz does not satisfy α_0.

Strict Leave (SL) Vs Liberal Leave (LL): In SL, the leaving subject loses access to all objects. In LL, the leaving subject can retain access to objects authorized prior to the time of Leave. In figure 2, on SL, $s1$ loses access to all group objects ($o1$ and $o2$) authorized during the membership period. An LL will allow $s1$ to retain access to $o2$ (and possibly $o1$). leave_i, a type of Leave, would be admitted as SL only if it satisfies α_1 below:

$$\alpha_1 = \Box(\text{Authz} \rightarrow (\neg\text{leave}_i\ \mathcal{S}\ \text{Join}))$$

In a g-SIS specification with LL, there exists at least one well-formed trace that does not satisfy α_1.

Strict Add (SA) Vs Liberal Add (LA): In SA, the added object can be accessed only by subjects already in the group. In LA, there are no such restrictions. The added object may be accessed by subjects that join (e.g., LJ) later. If Add ($o2$) in figure 3 is an SA, only $s1$ can access the object. Subjects $s2$ and $s3$, joining later, cannot access this object. But on LA current subject $s1$ and future subjects $s2$ and $s3$ may access $o2$. add_i, a type of Add, would be admitted as SA only if it satisfies α_2 below:

$$\alpha_2 = \Box(\text{add}_i \rightarrow (\neg\blacklozenge\text{Join} \rightarrow (\neg\text{Authz } \mathcal{W}\ \text{Add})))$$

In a g-SIS specification with LA, there exists at least one well-formed trace that does not satisfy α_2.

Strict Remove (SR) Vs Liberal Remove (LR): In SR, the removed object cannot be accessed by any subject. In LR, subjects who had access to the object at the time of remove may continue to access (of course subjects joining later are not allowed to access the removed object—this respects the Overlapping Membership core property). In figure 3, if Remove ($o1$) is an SR, every group subject (including $s1$) loses access to $o1$. If Remove ($o1$) is an LR, $s1$ can continue to access $o1$. However $s2$ and $s3$ will not have access to $o1$. remove_i, a type of Remove, would be admitted as SR only if it satisfies α_3 below:

$$\alpha_3 = \Box(\text{remove}_i \rightarrow (\neg\text{Authz } \mathcal{W}\ \text{Add}))$$

In a g-SIS specification with LR, there exists at least one well-formed trace that does not satisfy α_3.

3.4 Membership Renewal Properties

Membership Renewal Properties characterize what, if any, authorizations from previous membership period(s) are enabled or disabled when members re-join and subsequently leave the group. In the meeting room metaphor, Alice may leave the room and re-enter later. These properties are concerned with her authorizations from her previous session(s) in the room and its continuity when she leaves the room again. As the name implies, these properties are applicable only to returning members. We discuss these below.

Lossless Vs Lossy Join: In Lossless Join, a re-joining subject does not lose authorization(s) held immediately prior to re-joining. A Join operation that causes a subject to lose some or all prior authorizations is called Lossy. Suppose in figure 2 $s1$ retains access to $o2$ at the time of Leave (due to LL). When $s1$ re-joins subsequently, in a Lossless Join (regardless of whether it is a SJ or LJ), access to $o2$ will not be revoked. If access to $o2$ is revoked by re-joining the group, the Join is Lossy. The following formula characterizes Lossless Join:

$$\beta_0 = \Box((\text{Join} \wedge \neg\text{Remove} \wedge \ominus\ \text{Authz}) \rightarrow \text{Authz})$$

In a g-SIS specification with Lossy Join, there exists at least one well-formed trace that does not satisfy the above property.

A Lossy Join is useful in scenarios when authorizations from past membership and those from the new membership are in conflict of interest. For example, if a student registers for a course, drops after the mid-term and re-registers the following semester, he/she may be required to relinquish access to exercise solutions and other materials from past enrollment. The student would be given a Lossy Join in this scenario.

Non-Restorative Vs Restorative Join: In a Non-Restorative Join, authorizations from past membership periods may not be explicitly restored at the time of re-join. On the other hand, a Restorative Join explicitly restores authorizations from past membership periods. Suppose in figure 2 when $s1$ leaves, SL is applied and SJ is applied on re-join. A Restorative SJ in this scenario will allow $s1$ to re-gain access to $o2$ from past membership period. Note that the notion of Restorative LJ is subtle but important. Suppose $o1$ was removed with LR and an SL is applied at the time of Leave. In this case, s1 will continue to access $o1$ until the time of Leave. If LL is applied on re-join, a Restorative LJ will allow $s1$ to re-gain access to $o1$, but a Non-Restorative LJ will not.

Formalizing Non-Restorative Join is complicated because we want our characterization to be independent of the exact semantics of the Join operation in question. Intuitively, we want to require that the Non-restorative Join does not add any authorizations that it would not have added if the subject had a different history. However LTL does not enable one to compare different traces. The solution we take is to consider two different subjects within a single trace. Because the two subjects can have different histories with the same trace, this strategy enables us to formalize the property:

$$\rho_1 = \text{join}_i(s1) \wedge \text{join}_i(s2)$$
$$\rho_2 = (\text{Authz}(s1, o, r) \wedge \neg\text{Authz}(s2, o, r)) \rightarrow \ominus\ (\text{Authz}(s1, o, r) \wedge \neg\text{Authz}(s2, o, r))$$
$$\beta_1 = \forall i \Box(\rho_1 \wedge \rho_2)$$

In formula ρ_1, subjects $s1$ and $s2$ both join the group at the same time by means of the same type of Join. ρ_2 says that if $s1$ is authorized to access an object in the current state and $s2$ is not, this should also be the case in the previous state (and vice-versa). The Non-Restorative Join property is characterized by formula ρ. It states that if two subjects Join the group at the same time with the same type of Join, then any difference in access at Join time is the result of some operation prior to the current Join operation. Let us use formula ρ_2 to understand the intuition. Because both $s1$ and $s2$ Join at the same time with same type, any access that is necessarily enabled by this Join for $s1$, would also be enabled for $s2$. Any additional access that $s1$ may have that $s2$ does not have could arise only because $s1$ had access to that object before joining the group. This captures the fact that access is not restored from past but is a consequence of the type of Leave operation applied to the subject when he/she left the group.

In Restorative Join, there exists at least one well-formed trace that does not satisfy the Non-Restorative Join property. If a subject joins a group using Restorative Join, some or all of the accesses to objects authorized during past membership period may be restored (unless it has been removed). Note that this is in addition to the authorizations that current Join may enable. The formula $\Box(\text{Join} \wedge ((\neg\text{Leave} \wedge \neg\text{Remove})\ \mathcal{S}\ (\text{Leave} \wedge \ominus\ \text{Authz})) \rightarrow \text{Authz})$, for

example, characterizes a type of Restorative Join where *all* past authorizations are restored.

A Restorative Join is applicable in scenarios where an incentive is provided for a subject to re-join the group. On the other hand, in subscription service scenarios, a Restorative and Non-Restorative Join may be priced differently, which may decide what prior authorizations to their past subscription materials will be restored.

Gainless Vs Gainful Leave: After re-joining the group, a subsequent Leave operation could either be Gainless or Gainful. In Gainless Leave, authorizations that never held during current membership period cannot be obtained by leaving the group. On the other hand, a Gainful Leave allows new authorizations to be granted at the time of Leave. Suppose that in figure 2 a Lossless SJ is applied when $s1$ re-joins the group. Because of SJ, only $o4$ and possibly $o5$ can be accessed. If $s1$ leaves the group in the future with LL, a Gainless LL will not grant any new authorizations other than that to $o4$ and $o5$. A Gainful LL, for example, may additionally grant access to $o3$. β_2 characterizes Gainless Leave:

$$\beta_2 = \Box((\text{Leave} \wedge (\neg\text{Join}\ \mathcal{U}\ (\text{Authz} \wedge \neg\text{Join}))) \rightarrow \ominus((\neg\text{Authz} \wedge \neg\text{Join})\ \mathcal{S}\ (\text{Authz} \wedge (\neg\text{Join}\ \mathcal{S}\ \text{Join}))))$$

This formula says that if the subject is authorized to access an object during non-membership period then it should have been authorized during the most recent membership period. In a g-SIS specification with Gainful Leave, there exists at least one well-formed trace that does not satisfy the Gainful Leave property.

A Gainful Leave is useful in scenarios where an incentive is provided for a subject to leave the group. This is commonplace in voluntary retirement or a severance package for employees.

Non-Restorative Vs Restorative Leave: In Non-Restorative Leave, authorizations that the subject had prior to joining the group are not explicitly restored at Leave time. In Restorative Leave, some or all of such authorizations are restored at Leave time. Suppose in figure 2 $s1$ left the group with LL and re-joins with Lossy SJ. In this case, $s1$ possibly loses access to both $o1$ and $o2$ at re-join time. Later on, if $s1$ leaves with Gainful LL, a Restorative Leave will allow $s1$ to re-gain access to $o1$ and $o2$ at the time of leave, but a Non-Restorative leave will not.

In the meeting room metaphor, suppose Alice is re-invited as a consultant on demand and is required to relinquish her past authorizations due to a conflict of interest with new authorizations that will be enabled by current membership. After Alice performs her duties and leaves the group, it is natural that she will need access to objects for which she lost authorization when joining the group. The following formula characterizes Non-Restorative Leave:

$$\beta_3 = \Box(\text{Leave} \wedge \text{Authz} \rightarrow \ominus\ \text{Authz})$$

In Restorative Leave, there exists at least one well-formed trace that does not satisfy the Non-Restorative Leave Property. For example, the formula $\Box((\text{Leave} \wedge \neg\text{Remove} \wedge \ominus((\neg\text{Leave} \wedge \neg\text{Remove})\ \mathcal{S}\ (\text{Join} \wedge \ominus\ \text{Authz}))) \rightarrow \text{Authz})$ characterizes a specific type of Restorative Leave where access to *all* objects authorized prior to Join is restored.

4. G-SIS SPECIFICATION

A g-SIS specification formally defines the conditions under which authorization is granted in terms of past joins, leaves, adds, and removes. In this section, we present the syntactic form of such specifications written in LTL and a collection of semantic constraints, which we call *core properties*, that they are required to satisfy. As a proof of concept of our approach, we develop a sub-family of g-SIS specifications and successfully verify that these specifications entail the core properties.

4.1 Syntax and Semantics Definitions

The syntax and semantics of a g-SIS specification are defined as follows:

DEFINITION 4.1 (SYNTACTIC CORRECTNESS). *A g-SIS specification is* syntactically correct *if it is of the form:*

$$\gamma = \forall s \in S.\forall o \in O.\Box(\text{Authz}(s,o,r) \leftrightarrow \psi(s,o)) \wedge \bigwedge_{0\leq j\leq 3} \tau_j$$

in which ψ is an LTL formula constructed by using temporal operators and predicates in A, and the conjunction τ_j specifies the well-formedness requirements of a g-SIS trace.

DEFINITION 4.2 (SEMANTIC CORRECTNESS). *A g-SIS specification γ is* semantically correct *if:*

$$\gamma \vDash \bigwedge_{0\leq i\leq 6} \varphi_i$$

Thus a g-SIS specification must obey the well-formedness requirements and the core properties. Note that the specification is not required to satisfy Membership and Membership Renewal properties (α's and β's) since they are not core to g-SIS.

4.2 Design of Mixed g-SIS Specification

In this section, we discuss the design of a g-SIS specification that is based on a selection of Membership and Membership Renewal group operations discussed earlier. We formally show that such a specification satisfies the g-SIS properties.

4.2.1 Design Scope

In section 3, we discussed a few useful Membership and Membership Renewal properties. While a g-SIS specification should satisfy all core properties, it may opt to satisfy a subset of Membership and Membership renewal properties depending on the requirements posed by an application. In other words, Membership and Membership Renewal properties are orthogonal and the operations can be mixed. We characterize the notion of an operation type below:

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

Mixed g-SIS specifications allow any variation of Membership operations (Strict/Liberal) for subjects and objects. In such a specification, different subjects and objects may be given different types of respective operations. For example, SJ for $s1$, LJ for $s2$, SA for $o1$, LA for $o2$, etc.

REMARK 4.3. *In this paper, we confine our attention to Mixed g-SIS specifications in which the operations are of the types indicated below:*

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

In the following formulas, for convenience, we assume that both SJ and LJ are operations of type Lossless and Non-Restorative. Similarly, both SL and LL are Gainless and Non-Restorative operations.

4.2.2 Mixed Specification Design

There are two scenarios to consider when a subject requests access to an object: (a) the subject Join event occurred prior to object Add event and (b) the object Add event occurred prior to subject Join event. Intuitively, if the specification correctly addresses these two scenarios, it would be complete. We now separately consider these two scenarios. Formula λ_1 addresses the scenario where the object is added after the subject joined the group (figure 4).

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

Since Join occurred prior to Add, regardless of whether the object was LA'ed[7] or SA'ed or whether the subject was SJ'ed or LJ'ed, the subject should be authorized to access the object in both cases as per our core Availability Property (φ_6). Formula λ_1 says that the subject has not been SL'ed and the object has not been SR'ed since it was added with SA or LA. Further, when the Add occurred, the subject was a current member (that is, no SL or LL since SJ or LJ).

In figure 4, an SL or SR since object add time denies access to the requested object. However, it is alright for an LL or LR to occur during that period. Recall that an LR authorizes current subjects at remove time to retain access and LL authorizes a leaving subject to retain access to objects authorized during membership period. Similarly, if the subject was not a current member when the object was added (for e.g., joined and left the group before the object was added), authorization cannot hold as per Overlapping Membership Property.

Scenarios (b) where an Add occurs prior to Join is more interesting. As shown in figure 5, there are four possible cases. Let us first consider cases (a) and (b) where the object is SA'ed to the group. Recall that an SA'ed object can be accessed only by *existing* subjects (that is, the subjects who joined the group prior to object Add). Clearly, regardless of the type of Join, the subject is not authorized to access objects that were SA'ed prior to the subject Join time. Thus Authz cannot hold in cases (a) and (b).

Consider cases (c) and (b) where the object is LA'ed to the group. In case (c), the object is LA'ed and the subject is SJ'ed. An SJ'ed subject is authorized to access only those objects that were added after join time. Thus (c) is also a failed case. Authorization is successful in case (d) where both Add and Join are Liberal operations. An LJ'ed subject can access all current LA'ed objects and any newly added object in the future (LA or SA). We can now formulate λ_2 as shown below.

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

Figure 6 illustrates λ_2. It says that the subject has not been SL'ed and the object has not been SR'ed since the subject LJ'ed the group. Further, at Join time, the object in question was still part of the group (that is, it has not been LR'ed or SR'ed since it was added). We can now formalize a Mixed g-SIS specification.

DEFINITION 4.4 (π-SYSTEM). *The π-system is given by:*

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

[7]We use terminology of the form "LA'ed" to refer to the fact that object o is Liberally Added in a state.

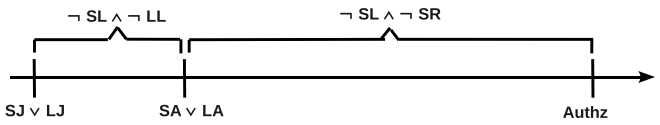

Figure 4: Mixed Operations - Formula λ_1.

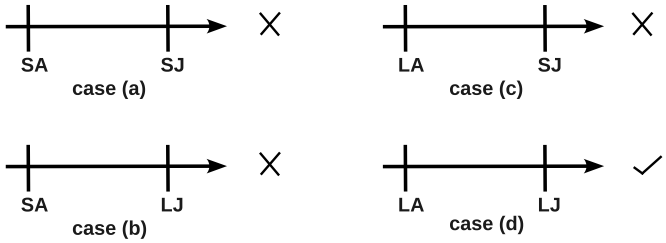

Figure 5: Cases when Add occurs prior to Join.

Note that π is a syntactically correct g-SIS specification. π says that a subject is authorized to access an object if and only if λ_1 or λ_2 holds and the trace is well-formed. Note that this definition is consistent with definition 4.1. We show that π is also semantically correct in the following subsection.

4.2.3 Formal Analysis

In this section, we show that a Mixed g-SIS Specification (π-system) entails the Core and Membership Renewal properties. Since Membership operations are mixed, a single Membership property cannot be verified for this specification. Later, we show the verification of Membership Properties in a specification where the operations are fixed for all subjects and objects in a group. We begin with the Entailment Theorem.

THEOREM 4.5 (ENTAILMENT THEOREM). *The π-system entails the Core Properties (φ_0 to φ_6) and Membership Renewal Properties (β_0 to β_3):*

$$\pi \vDash (\bigwedge_{0 \le q \le 6} \varphi_q \wedge \bigwedge_{0 \le r \le 3} \beta_r)$$

We utilize the model checker NuSMV [14] to prove this theorem. Model checking is an automated formal analysis approach that takes a finite model of a system and its properties written in temporal logic formulas as input to verify if the properties hold in the system. In the case that a property fails to hold, a model checker produces a counterexample consisting of a trace that shows how the failure can arise and can be used to correct the model or the property specification.

A NuSMV model describes how variables can be modified in each step of a system execution. The model of the π-system for the purpose of our proof is very simple. The NuSMV model (appendix A) is expressed in terms of events Join, Leave, Add and Remove (declared as boolean variables) that are allowed to occur concurrently in a non-deterministic manner. The theorem is expressed as an implication having the Mixed g-SIS specification in the antecedent and the Core and Membership Renewal properties as the consequent. NuSMV takes the model and the LTL formula to verify if the formula holds in all possible traces generated by the model. As one can see, this is a non-traditional use of model checker NuSMV since the model is not constrained. The output from NuSMV in appendix A shows that the LTL formula hold against the model. Thus we verify the Entailment Theorem, i.e.,

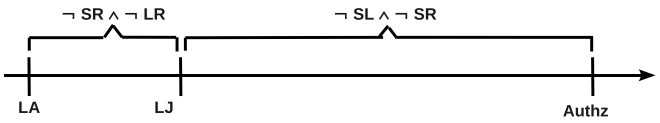

Figure 6: Mixed Operations - Formula λ_2.

the π-system satisfies the Core and Membership Renewal properties.

The significance of the Entailment Theorem is two-fold. First, it shows that any specification that one derives from the family of Mixed g-SIS specifications (π) is guaranteed to be a *g-SIS* specification (i.e., it would satisfy the core properties.) For example, one can derive the Most Restrictive Mixed g-SIS specification by substituting all operations with Strict versions in π. Such a specification is guaranteed to be admitted as g-SIS. Next, the Core g-SIS properties are consistent with respect to π. That is, the core properties *can* be satisfied by the π-system and do not conflict with each other. In other words, the Core properties are enforceable. Further, note that the theorem proves that formula π is semantically correct (the semantic definition of a g-SIS specification 4.2 is actually weaker).

COROLLARY 4.6 (MIXED G-SIS SEMANTIC CORRECTNESS). *The π-system is a semantically correct g-SIS specification:*

$$\pi \vDash \bigwedge_{0 \leq i \leq 6} \varphi_i$$

As mentioned earlier, the Most Restrictive Mixed g-SIS specification is one where only Strict operations are allowed. Since Liberal operations are not allowed in such a specification, we substitute Liberal operations with "False" in formulas λ_1 and λ_2 and thereby obtain the μ-system defined below.

DEFINITION 4.7 (μ-SYSTEM). *The μ-system is given by:*

$$\mu = \Box(\text{Authz} \leftrightarrow ((\neg\text{SL} \wedge \neg\text{SR}) \; \mathcal{S} \; (\text{SA} \wedge (\neg\text{SL} \; \mathcal{S} \; \text{SJ})))) \wedge \bigwedge_{0 \leq i \leq 3} \tau_i$$

Note that specification μ is a synctactically and semantically correct g-SIS specification since it is a specific form of π. Further, since only Strict operations are allowed, μ must satisfy all the Strict versions of Membership Properties (formulas α_0 to α_3).

THEOREM 4.8 (MOST RESTRICTIVE ENTAILMENT THEOREM). *The μ-system entails the Core Properties (φ_0 to φ_6), Membership Properties (α_0 to α_3) and Membership Renewal Properties (β_0 to β_3).*

$$\mu \vDash (\bigwedge_{0 \leq j \leq 6} \varphi_j \wedge \bigwedge_{0 \leq k \leq 3} \alpha_k \wedge \bigwedge_{0 \leq l \leq 3} \beta_l)$$

We take a similar approach to that of Theorem 4.5 to prove Theorem 4.8. The proof is given in appendix B. Note that with Strict and Liberal variations of Membership operations, there are 16 possible specifications where these operations are fixed (i.e., same type) for every subject and object in the group. One can derive more specifications from the π-system where some of the operations are mixed while others are fixed for all subjects and objects in the group.

5. DISCUSSION

We now discuss how these group operations apply to the scenarios discussed earlier in section 1.

Magazine Subscription: In general, subject operations define the semantics of most subscription models. ABS's subscription models would fall into one of the four categories: (SJ, SL), (SJ, LL), (LJ, SL) and (LJ, LL). (SJ, SL) would be the Level 1 membership that does not allow members to access archives. When the subscribers stop paying the fee, they completely lose access to all objects. (SJ, LL) would be Level 2 membership, which differs from Level 1 in that it lets leaving members retain access to what they paid for. (LJ, SL) would be Level 3 membership, which differs from Level 1 in that subscribers can also access the archives during their membership period. Finally, (LJ, LL) would be Level 4 membership which differs from Level 3 access in that leaving members retain access to what they paid for.

If ABS Corp. were to find some inappropriate content after publication, they can remove the article with SR. On the other hand, ABS Corp. may decide to remove content that they do not want their new subscribers to view. But the existing subscribers may continue to access the removed article since they have already 'paid for the content. This is achieved by removing that article with LR.

From time-to-time, ABS Corp. may offer promotions only to their current subscribers—such as discounted price for highly-rated reports and other multimedia content. Such offers are added to the group with SA. This way the offer is made available only to current subscribers. Also, a Restorative Join would allow re-joining subscribers to regain past accesses.

Collaborative Product Development: Re-visiting our earlier collaborative product design example between ABC and XYZ Corp., ABC can create a group and admit their engineers with LJ. Proprietary documents are made available to ABC engineers by adding them with SA. Suppose incoming XYZ engineers are given an LJ, SA objects added earlier cannot be accessed by them. After the collaboration period, the respective engineers may leave the group with LL so that access to newly developed design documents can be retained. Note that our current Join semantics does not accommodate a scenario where an ABC engineer is admitted much later and still allow access to SA objects added earlier. Specifically, this requires a notion of a back-dated Join for the ABC engineer. Investigating such useful additional semantics, such as back-dated Join or suspending membership for brief period, is part of our future work. In another scenario, if XYZ engineers were invited as consultants on demand, they are given an SL so they cannot access the design documents after leaving the group. If they were to be re-invited later, a Restorative Join will allow them to access past design documents in order to continue their collaboration.

6. FUTURE WORK AND CONCLUSION

Our work on developing single group read-only g-SIS models will serve as a foundation for systematically extending this theory in several directions. We are investigating further extensions along three dimensions: read-write g-SIS model, multiple groups and application of attributes such as roles in g-SIS.

We strongly believe that information sharing, being distributed in nature, should support versioning so different subjects may edit and update an object at the same time without having to obtain a lock or "check out" the object. Thus writing an object creates a new version of that object. Versioning brings many interesting questions such as if the core properties identified here are adequate and how the inter-dependency between various versions can be handled.

Next, we need to investigate g-SIS models in the context of multiple groups. Completely unrelated groups would not be interesting so some relationship is necessary. One natural way to create a structure is to impose a hierarchy similar to information flow mod-

els such as Bell-LaPadula (BLP) [8], Biba [13], etc. and study how temporal interactions in g-SIS relates to these models.

Finally, we need to investigate how specifications that are purely based on temporal ordering of subject and object events can be complemented with attribute-based access control. We believe attributes such as "roles" play a vital role in information sharing.

In this paper, we proposed a group-centric family of models for Secure Information Sharing. We specified a core set of properties that should be satisfied by the g-SIS specifications. We also identified an additional set of properties in light of many variations of group operations. We formally specified the properties using LTL making them suitable to be verified by using model checking which is an automated verification technique. We discussed the specification of a family of g-SIS models that are based on mixed Membership operations. Finally, we showed using NuSMV that the specification semantically entails the g-SIS properties.

7. REFERENCES

[1] eXtensible rights Markup Language. *www.xrml.org*.

[2] OASIS eXtensible Access Control Markup Language . *www.oasis-open.org/committees/xacml/*.

[3] The Open Digital Rights Language Initiative. *www.odrl.net*.

[4] M. Abrams, J. Heaney, O. King, L. LaPadula, M. Lazear, and I. Olson. Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy. *Proc. of the 14th National Computer Security Conf.*, pages 257–266, 1991.

[5] G.-J. Ahn, B. Mohan, and S.-P. Hong. Towards secure information sharing using role-based delegation. *J. Network and Computer Applications*, 30(1):42–59, 2007.

[6] B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2:117–126, 1987.

[7] V. Atluri and J. Warner. Automatic Enforcement of Access Control Policies Among Dynamic Coalitions. *International Conference on Distributed Computing & Internet Technology, Bhubaneswar, India, Dec*, 2004.

[8] D. Bell and L. LaPadula. Computer Security Model: Unified Exposition and Multics Interpretation. *MITRE Corp., Bedford, MA, Tech. Rep. ESD-TR-75-306, June*, 1975.

[9] E. Bertino, C. Bettini, and P. Samarati. A temporal authorization model. In *Proc. of 2nd ACM Conference on Computer and communications security*, 1994.

[10] E. Bertino, P. Bonatti, and E. Ferrari. TRBAC: A temporal role-based access control model. *ACM Trans. on Information and System Security (TISSEC)*, 4(3):191–233, 2001.

[11] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. *ACM Trans. Inf. Syst. Secur*, 6:71–127, 2003.

[12] V. Bharadwaj and J. Baras. A framework for automated negotiation of access control policies. *DARPA Information Survivability Conference and Exposition*, 2, 2003.

[13] K. Biba. Integrity considerations for secure computer systems. *Technical Report TR-3153*, April 1977.

[14] A. Cimatti, E.M.Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model checker. *Journal on Software Tools for Tech. Transfer*, pages 410–425, 2000.

[15] D. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM*, 19(5):236–243, 1976.

[16] DoD National Computer Security Center (DoD 5200.28-STD). *Trusted Computer System Evaluation Criteria*, December 1985.

[17] G. Graham and P. Denning. Protection-principles and practice. *Proceedings of the AFIPS Spring Joint Computer Conference*, 40:417–429, 1972.

[18] R. Graubart. On the Need for a Third Form of Access Control. *Proceedings of the 12th National Computer Security Conference*, pages 296–304, 1989.

[19] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Comm. of the ACM*, August 1976.

[20] T. Jaeger and J. E. Tidswell. Practical safety in flexible access control models. *ACM Trans. Inf. Syst. Secur.*, 4(2):158–190, 2001.

[21] H. Khurana and V. Gligor. A Model for Access Negotiations in Dynamic Coalitions. *Proc. of the 13th IEEE Int. Wksp. on Enabling Tech.*, pages 205–210.

[22] H. Khurana, V. Gligor, and J. Linn. Reasoning about joint administration of access policies for coalition resources. In *Proc. of the 22nd Int. Conf. on Distributed Computing Systems*, page 429, 2002.

[23] R. Krishnan, R. Sandhu, J. Niu, and W. Winsborough. A conceptual framework for group-centric secure information sharing. *To appear in ACM Symposium on Information, Computer and Communications Security*, March 2009.

[24] R. Krishnan, R. Sandhu, and K. Ranganathan. PEI models towards scalable, usable and high-assurance information sharing. In *SACMAT '07*, pages 145–150. ACM.

[25] B. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974.

[26] R. Lipton and L. Snyder. A Linear Time Algorithm for Deciding Subject Security. *Journal of the ACM*, 24(3):455–464, 1977.

[27] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[28] C. McCollum, J. Messing, and L. Notargiacomo. Beyond the pale of MAC and DAC - defining new forms of access control. *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pages 190–200, 1990.

[29] J. Park and R. Sandhu. Originator control in usage control. *Policies for Distributed Systems and Networks*, 2002.

[30] C. Phillips Jr, T. Ting, and S. Demurjian. Information sharing and security in dynamic coalitions. *SACMAT*, 2002.

[31] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Surveys*, pages 309–329, September 2003.

[32] R. Sandhu. The schematic protection model: its definition and analysis for acyclic attenuating schemes. *Journal of the ACM (JACM)*, 35(2):404–432, 1988.

[33] R. Sandhu. The typed access matrix model. In *Proc. of the IEEE Symposium on Security and Privacy*, page 122, 1992.

[34] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Trans. on Inf. and Syst. Security*, 2(1):105–135, 1999.

[35] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and PEI models. In *Proc. of ACM Symposium on Information, Computer and Communications Security*, pages 2–12, 2006.

[36] A. P. Sistla. Safety, liveness and fairness in temporal logic. In *Formal Aspect of Computing*, pages 495–511, 1994.

[37] M. V. Tripunitara and N. Li. A theory for comparing the expressive power of access control models. *Journal of Computer Security*, 15(2):231–272, 2007.

[38] J. Warner, V. Atluri, R. Mukkamala, and J. Vaidya. Using semantics for automatic enforcement of access control

policies among dynamic coalitions. In *SACMAT '07*, pages 235–244, 2007.

APPENDIX

A. PROOF OF ENTAILMENT THEOREM

The following NuSMV model in the code listing simply allows group events such as SJ, LJ, SR, etc. to occur concurrently in a non-deterministic manner. This model produces all possible traces in terms of these group events. We concern ourselves only with the legal mixed g-SIS traces by formulating the g-SIS specification as the antecedent and the core and membership renewal properties as the consequent of an implication. The output shows that the g-SIS properties are satisfied by all well-formed g-SIS traces.

In the NuSMV model, the group events and authorization are declared as boolean variables in the VAR section. The DEFINE section consists of macro definitions to improve the readability of the code. The LTL formula to be checked are listed in the LTLSPEC section. Comments follow the symbols – –. The logic operators $\vee$, $\wedge$, and $\neg$ are represented as |, &, and !, respectively. The temporal logic operators $\bigcirc$, $\Box$, $\mathcal{U}$, $\mathcal{W}$, $\ominus$, $\blacklozenge$, and $\mathcal{S}$ are represented as X, G, U, W, Y, O and S respectively in the LTL formulas.

Code Listing

```
MODULE main
VAR
SL: boolean;
LL: boolean;
SA: boolean;
LA: boolean;
SJ: boolean;
LJ: boolean;
SR: boolean;
LR: boolean;

authz: boolean;

DEFINE
Join := SJ |LJ;
Leave := SL|LL;
Add := SA | LA;
Remove := SR | LR;

---------------------------------------
--Verification
LTLSPEC
G
(
--well-formed trace constraints
(!(Add & Remove) & !(Join & Leave) &
!(SJ & LJ) & !(SL & LL) &
!(SA & LA) & !(SR & LR) &
  (Join  -> X ((!Join U Leave) | G !Join)) &
(Leave -> X((!Leave U Join) | G!Leave)) &
(Remove -> X((!Remove U Add) | G!Remove)) &
(Add -> X((!Add U Remove) | G!Add)) &
(Leave -> O Join) & (Remove -> O Add) ) &

--Mixed g-SIS specification
(authz <-> ((((!SL & !SR) S ((SA | LA) &
((!LL & !SL) S (SJ | LJ)))) |
((!SL & !SR) S (LJ & ((!SR &
!LR) S LA)))))
)
->

--Core Properties
G (

--Availability
(Join ->
(((Add -> authz) U Leave) |
G (Add -> authz))) &

--Bounded Object Authorization
(Remove & !authz) -> ((!authz U Add) |
G !authz) &

--Bounded Subject Authorization
(Leave & !authz) -> ((!authz U Join) |
G !authz) &

--Authorization Persistence
(!authz -> ((!authz U (Join | Leave |
Add | Remove)) | G !authz)) &

--Revocation Persistence
(authz -> ((authz U (Join | Leave |
Add | Remove)) | G authz)) &

--Authorization Provenance
(authz -> O (Join & (!Leave U
authz))) &

--Overlapping Membership
(authz -> O (Add & (!Leave S Join)) |
O (Join & (!Remove S Add))) &

--Membership Renewal Properties

--Lossless Join
((Join & !Remove & Y authz) -> authz) &

--Gainless Leave
((Leave & (!Join U (authz & !Join))) ->
Y ((!authz & !Join) S
(authz & (!Join S Join)))) &

--Non-Restorative Leave
((Leave & authz) -> Y authz)
)
```

Running NuSMV shows that the LTLSPEC specification is true proving the Entailment Theorem 4.5.

```
--Run NuSMV

[ram@localhost smv]$ NuSMV gsis_v3.smv
*** This is NuSMV 2.4.3 (compiled on Mon May
5 02:33:40 UTC 2008)
*** For more information on NuSMV
see <http://nusmv.irst.itc.it>
*** or email to <nusmv-users@irst.itc.it>.
*** Please report bugs to <nusmv@irst.itc.it>.
```

```
-- specification ( G ((((((((((((!
(Add & Remove) & !(Join & Leave))
& !(SJ & LJ)) & !(SL & LL)) &
!(SA & LA)) & !(SR & LR)) & (Join
->  X ((!Join U Leave) |  G !Join)))
& (Leave ->  X ((!Leave U Join)
|  G !Leave))) & (Remove ->  X ((!Remove
U Add) | G !Remove))) & (Add ->
X ((!Add U Remove) |  G !Add))) & (Leave
->  O Join)) & (Remove ->  O Add)) &
(authz <-> (((!SL & !SR) S ((SA | LA) &
((!LL & !SL) S (SJ | LJ)))) |
((!SL & !SR) S (LJ & ((!SR & !LR)
S LA)))))) ->  G (((Join -> (((Add
-> authz) U Leave) | G (Add -> authz)))
& (Remove & !authz)) -> (((((!authz U Add)
|  G !authz) & (Leave & !authz))
-> ((((((((((!authz U Join) |  G !authz)
& (!authz -> ((!authz U (((Join | Leave)
| Add) | Remove)) | G !authz))) & (authz
-> ((authz U (((Join | Leave) | Add) |
Remove)) |  G authz))) & (authz ->
O (Join & (!Leave U authz)))) & (authz
-> ( O (Add & (!Leave S Join)) | O
(Join & (!Remove S Add))))) & (((Join &
!Remove) &  Y authz) -> authz)) & ((Leave
& (!Join U (authz & !Join))) ->
Y ((!authz & !Join) S (authz & (!Join
S Join))))) & ((Leave & authz)
->  Y authz)))))
```

is true

```
[ram@localhost smv]$
```

B. PROOF OF MOST RESTRICTIVE ENTAILMENT THEOREM

In this case, we incorporate the Membership properties (α_0 to α_3) into the consequent of the implication and run NumSV against such a specification.

Code Listing

```
MODULE main

VAR

SL: boolean;
    SA: boolean;
SJ: boolean;
SR: boolean;

authz: boolean;

DEFINE
Join := SJ;
    Leave := SL;
    Add := SA;
Remove := SR;

---------------------------------------
--Most Restrictive Entailment Theorem
LTLSPEC
G
(
--well-formed trace constraints
(!(Add & Remove) & !(Join & Leave) &
  (Join  -> X ((!Join U Leave) | G !Join)) &
(Leave -> X((!Leave U Join) | G!Leave)) &
(Remove -> X((!Remove U Add) | G!Remove)) &
(Add -> X((!Add U Remove) | G!Add)) &
(Leave -> O Join) & (Remove -> O Add) ) &

--Most Restrictive Mixed g-SIS Specification
(authz <-> ((!SL & !SR) S (SA & (!SL S SJ))))
)
->

--Core Properties
G (
(Join -> (((Add -> authz) U Leave) |
G (Add -> authz))) &
(Remove & !authz) -> ((!authz U Add) |
G !authz) &
(Leave & !authz) -> ((!authz U Join) |
G !authz) &
(!authz -> ((!authz U (Join | Leave |
Add | Remove)) | G !authz)) &
(authz -> ((authz U (Join | Leave |
Add | Remove)) | G authz)) &
(authz -> O (Join & (!Leave U authz))) &
(authz -> O (Add & (!Leave S Join)) |
O (Join & (!Remove S Add))) &

--Membership Renewal Properties
--Lossless Join
((Join & !Remove & Y authz) -> authz) &

--Gainless Leave
((Leave & (!Join U (authz & !Join))) ->
Y ((!authz & !Join) S (authz &
(!Join S Join)))) &

--Non-Restorative Leave
((Leave & authz) -> Y authz)

--Membership Properties
--alpha0
(authz -> O (Add & (!Leave S Join))) &

--alpha1
(authz -> (!SL S Join)) &

--alpha2
(SA -> (!O Join -> ((!authz U Add)
| G !authz))) &

--alpha3
(SR -> ((!authz U Add) | G !authz))
)
```

Running NuSMV shows that the LTLSPEC specification is true proving the Most Restrictive Entailment Theorem 4.8.

```
[ram@localhost smv]$ NuSMV gsis_v4_strict.smv

-- specification ( G (((((((!(Add & Remove)
```

```
& !(Join & Leave)) & (Join ->  X ((!Join U
Leave) | G !Join))) & (Leave ->  X ((!Leave
U Join) | G !Leave))) & (Remove ->  X ((
!Remove U Add) | G !Remove))) & (Add ->
X ((!Add U Remove) | G !Add))) & (Leave ->
O Join)) & (Remove -> O Add)) & (authz <->
((!SL & !SR) S (SA & (!SL S SJ))))) ->  G
(((Join -> (((Add -> authz) U Leave) |  G
(Add -> authz))) & (Remove & !authz)) ->
((((!authz U Add) |  G !authz) & (Leave &
!authz)) -> ((((((((((((!authz U Join) |
G !authz) & (!authz -> ((!authz U (((Join |
 Leave) | Add) | Remove)) |  G !authz))) &
(authz -> ((authz U (((Join | Leave) | Add)
| Remove)) |  G authz))) & (authz ->
O (Join & (!Leave U authz)))) & (authz ->
( O (Add & (!Leave S Join)) |  O (Join &
(!Remove S Add))))) & (((Join & !Remove) &
Y authz) -> authz)) & ((Leave & (!Join U
(authz & !Join))) ->  Y ((!authz & !Join)
S (authz & (!Join S Join))))) & ((Leave &
authz) ->  Y authz)) & (authz ->  O (Add &
(!Leave S Join)))) & (authz -> (!SL S
Join))) & (SA -> (!( O Join) -> ((!authz U
Add) |  G !authz)))) & (SR -> ((!authz U Add)
|  G !authz))))))
```

is true.

```
[ram@localhost smv]$
```