Foundations For Group-Centric Secure Information Sharing Models

Ram Krishnan
George Mason University
Fairfax, VA, USA
rkrishna@gmu.edu

Ravi Sandhu
Univ of Texas at San Antonio
San Antonio, TX, USA
ravi.sandhu@utsa.edu

Jianwei Niu
Univ of Texas at San Antonio
San Antonio, TX, USA
niu@cs.utsa.edu

William H. Winsborough
Univ of Texas at San Antonio
San Antonio, TX, USA
wwinsborough@acm.org

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. Traditional approach to information sharing, characterized as Dissemination-Centric, focuses on attaching attributes and policies to an object as it is disseminated from producers to consumers in a system. In contrast, Group-Centric sharing envisions bringing the users and objects together in a group to facilitate sharing. The metaphors "secure meeting room" and "subscription service" characterize the Group-Centric approach where participants and information come together to share for some common purpose. Our focus in this paper is on semantics of group operations: Join and Leave for users and Add and Remove for objects, each of which can have several variations called types. We use Linear Temporal Logic to first characterize the core properties of a group in terms of these operations. We then characterize additional properties for specific types of these operations. Finally, we specify the authorization behavior for read access in a single group for a family of g-SIS models and show that these models satisfy the above-mentioned properties using the NuSMV model checker.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—Access controls; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—Unauthorized access

General Terms

Security

Keywords

Groups, Security Properties, Linear Temporal Logic

1. INTRODUCTION AND MOTIVATION

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. 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 develops formal models for Group-Centric Secure Information Sharing or g-SIS [18, 20]. 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 or merger and acquisition) to participation in a shared activity (such as a semester long class) to a dynamic coalition (such as a mission-oriented group driven towards completion of a particular task). The subscription metaphor speaks to 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 systematically investigated. 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. 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: Typically committee 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 after that discussion has concluded. The conversation that occurred during her absence is not accessible to her. In a different setting, such conversations may be recorded in a smart board and made available to her on return.

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 Here access to content may depend upon when the subscription began and the terms of subscription.

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. In addition to Level 1 services, Level 2 ($12/year) 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 lose all access on cancelling their subscription. Level 4 ($17/year) subscribers can view all articles that they had access to before leaving, even after canceling subscription.

Secure multicast: In secure multicast [27], typically new members joining the group cannot access content distributed prior to their join time. Similarly, members leaving the group can no longer access any new content. Thus access is dependant on when the users join and leave the group.

Clearly, the "secure meeting room" metaphor suggests a smaller-scale information sharing scenario whereas "subscription service" 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 a 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 "multiple groups" with possibly overlapping users. 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 users at a higher level dominate those at the lower levels. Another common relationship is that of mutual exclusion where the same user is prohibited from joining conflicting groups.

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 users, Add and Remove for objects. Users 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 user never loses access to objects authorized prior to joining the group. Similarly, in Restorative Join, the joining user may regain access to objects authorized during past membership period. 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., [8]), where they 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.

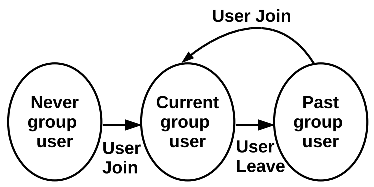


Figure 1: User Membership States.

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 [14, 15, 21, 30, 31, 33]. 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 paper. Without a basic understanding of group operation semantics, 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 in terms of Linear Temporal Logic (LTL) [22], by specifying various properties. We first specify the Core Properties (Provenance, Persistence, etc.) that characterize a group in g-SIS. The core properties *must* be satisfied by any g-SIS model. Next we identify the semantics of a collection of group operations and thereby specify Membership and Membership Renewal Properties. These properties are based on specific variations of group operations: Join, Leave, Add and Remove. Unlike core properties, they need not be satisfied by all g-SIS models but suggest some useful operational semantics for many applications. Finally, we specify the authorization behavior for a family of g-SIS models and show using model checking that these models satisfy the aforementioned properties. In this paper, we confine our attention to the particular authorization behavior with respect to read access in a single group using LTL. Our future work involves extensions to write operations and multiple groups.

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 family of g-SIS specifications and show by using model checking that the specifications satisfy the Core Properties. We also revisit our metaphors and discuss how the group operations can express many scenarios. In section 5, 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" [5, 11, 24]. 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 [3, 13, 23, 25] in the 1980's and Digital Rights Management in the 1990's and 2000's. XrML [1] and ODRL [2] are recent examples of policy languages developed for this purpose. Group-Centric sharing differs in that it advocates bringing the users and objects together to facilitate sharing. We envision that Dissemination 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. At a pragmatic level, we believe Group-Centric and Dissemination-Centric are significantly different approaches to information sharing.

The use of groups in access control goes back to earliest OS's (e.g., [28]) and is now commonplace in modern OS's and directory services such as LDAP [40]. In [36], the authors discuss user authorization schemes for dynamically defining membership in groups. [29] discusses special kind of group hierarchies. In [7], 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 [4], the authors use a role-based delegation framework for specifying policies for resource and information sharing within and across organizations. The modern concept of Role-Based Access Control [34] can be viewed as an evolution of access control to simplify administration in organizations bringing in additional concepts such as hierarchies and constraints [32].

Finally, information sharing challenges have also been considered in the context of Dynamic Coalition Problem (see [16, 26, 39] for example) which is concerned with the challenges involved when a coalition is dynamically formed, for example, in response to a crisis. Our work differs from others in that we focus on the operation semantics for the Group-Centric SIS problem. Further, formal specification of g-SIS properties using LTL enables us to automate verification using model checking. To the best of our knowledge, this is the first effort towards developing a formal model for the Group-Centric approach to SIS. The policies considered by Secure Multicast [27] are probably the closest but it will be evident later that g-SIS subsumes such policies.

3. FOUNDATIONS FOR G-SIS

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

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 U is a finite set of users, O is a finite set of objects, and R is a finite set of actions, such as read and write. In this paper, we do not introduce subjects in the model which will be needed in more elaborate models. The current model is simple enough that the authorization of subjects need not be distinguished from that of the users who control them. Let P be a set of predicates over sorts U, 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 (we suggest that the language represented here should be a sub-language of any g-SIS language designed in the future.). 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 $u \in U$, $o \in O$, $r \in R$, $\text{join}_i(u), \text{add}_i(o), \ldots, \text{Authz}(u, o, r) \in \mathcal{F}$.

Formally, a state is a function from variable-free atomic 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 subformulas that state, for example, some type of join event occurs. It is therefore convenient to introduce the following shorthands:

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

Note that Join holds in a state just in case the transition to the state is a Join event. The properties we consider treat the authorization a user has to access an object independently of actions involving other users 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 $\forall u \in U. \forall o \in O. \text{Authz}(u, o, r) \rightarrow (\text{Join}(u) \wedge (\neg(\text{Leave}(u) \vee \text{Remove}(o))\ \mathcal{S}\ \text{Add}(o)))$. Note that Join, Leave, Add, Remove and Authz, all refer to the same pair of u 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.

Well-Formed Traces.

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

A. An object cannot be Added and Removed and a user cannot Join and Leave at the same time.[1]

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

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

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

[1]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.

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.

C. If a user u joins a group, u cannot join again unless u 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.

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

The language we defined above allows us to develop g-SIS specifications which formally define the precise conditions under which authorization can hold. A g-SIS specification is syntactically correct if it is stated in terms of joins, leaves, adds and removes and satisfies the well-formedness constraints. We formally define the requirements for the syntactic correctness of a g-SIS specification.

DEFINITION 3.1 (SYNTACTIC CORRECTNESS). *A g-SIS specification is* syntactically correct *if it is of the form:*

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

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

3.2 Core g-SIS Properties

We begin with the Core properties, all of which must be satisfied by any g-SIS specification.[2] 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 authorization in g-SIS. If need be, these properties can be extended to models involving additional aspects (e.g. attributes).

1. Persistence Properties: These properties consider the conditions under which authorization may change.

Authorization Persistence: When a user u is authorized to access an object o, it remains so at least until a group event involving u or o occurs.[3]

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

[2]Note that each of these formulas defines a *safety* property [37] 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.

[3]As we will see later, authorization may no longer hold when certain variations of enabling events (e.g. Join, Add) occur.

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

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

A generalized statement of these properties may be "Authorization does not change unless an authorization changing event occurs." With this generalization, we believe persistence property is required of all access control systems. The following properties are more specifically targeted at g-SIS. They seek to recognize the additional authorizations enabled by group membership vis-à-vis non-membership.

2. Authorization Provenance: Intuitively, a user will not be authorized to access an object until a point at which both the user and object are simultaneously group members. This is formalized in φ_2 below:

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

Two important observations can be made from formula φ_2. First, if Authz holds in a given state then there was an overlapping period of membership between the user and object at least once in a given trace. Next, authorization to access an object cannot begin for the *first time* during a user's non-membership period.

3. Bounded Authorization: These properties require that authorizations not increase during non-membership periods of users and objects (note that authorizations may decrease). Authorizations that hold during non-membership period should have held at the time of Leave or Remove.

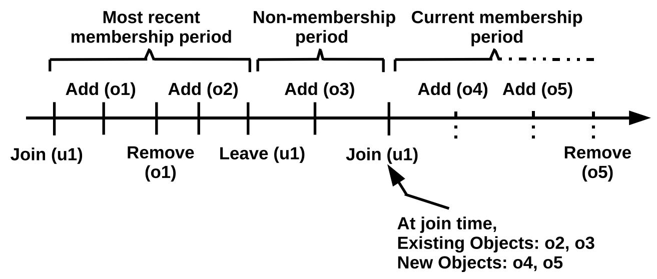
Bounded User Authorization: The set of all objects that a user can access during non-membership periods is bounded at Leave time. This set cannot grow until the user rejoins.

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

The above property states that additional authorizations cannot be granted to a user after Leave time. Any object that is accessible after Leave should have been authorized at the time of Leave.

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

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



Most recent membership period
Non-membership period
Current membership period
Add (o1)
Add (o2)
Add (o3)
Add (o4)
Add (o5)
Join (u1)
Remove (o1)
Leave (u1)
Join (u1)
Remove (o5)
At join time, Existing Objects: o2, o3 New Objects: o4, o5

Figure 2: User Operations Illustration.

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

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

This property states that after a user joins a group, any object that is added subsequently should be authorized. Obviously, the user 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 (φ_2).

We believe that properties 1, 2 and 3 are truly core and foundational and will apply to sophisticated g-SIS models beyond those formalized in this paper. We suspect that property 4 may need to be relaxed in situations where there is selective access within a group. For instance, a user may be required to belong to a particular role in addition to being a group member in order to access the object. Hence adding an object may not always guarantee immediate access. Finally, whether or not additional core properties exist remains an open question. Since these core properties are required of any g-SIS model, a g-SIS specification is semantically correct only if it satisfies all of these properties. We define this below:

DEFINITION 3.2 (SEMANTIC CORRECTNESS). *A g-SIS specification γ is* semantically correct *if:*

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

Thus, in summary, a g-SIS specification must obey the well-formedness requirements (syntactic correctness, definition 3.1) and the core properties (semantic correctness, definition 3.2).

3.3 Membership Properties

In this subsection and the following, we discuss a few additional properties that are based on specific variations of group operations. Unlike the core properties, a g-SIS specification is not required to satisfy these properties. Instead, these properties define certain group operation semantics that are useful for many applications.

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

Strict Join (SJ) Vs Liberal Join (LJ): In SJ, the joining user may only access some or all of the objects added after Join time. LJ additionally allows the user to access some or all of the objects that were added prior to join time. Suppose that in figure 2 the second Join ($u1$) is an SJ. Then $u1$ can access $o4$ and $o5$ but cannot access $o2$ and $o3$. If the Join was an LJ instead of SJ, $u1$ 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 in table 2. In a g-SIS specification with LJ, there exists at least one well-formed trace for which Authz does not satisfy α_0.

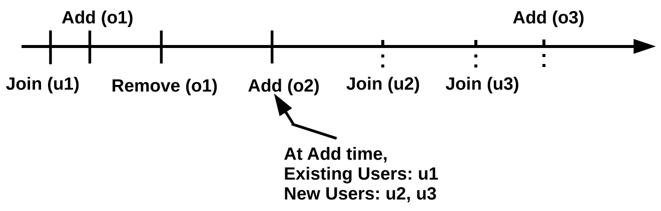


Figure 3: Object Operations Illustration.

Strict Leave (SL) Vs Liberal Leave (LL): In SL, the leaving user loses access to all objects. In LL, the leaving user may retain access to some or all of the objects authorized prior to Leave time. In figure 2, on SL, $u1$ loses access to all group objects ($o1$ and $o2$) authorized during the membership period. An LL will allow $u1$ to retain access to $o2$ (and possibly $o1$, depending on the type of Remove). leave$_i$, a type of Leave, would be admitted as SL only if it satisfies α_1 stated in table 2. 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, only some or all of the users who joined before Add time can access. In LA, the added object may also be accessed by some or all of the users that join (e.g., LJ) later. If Add ($o2$) in figure 3 is an SA, only $u1$ can access the object. Users $u2$ and $u3$, joining later, cannot access this object. But on LA current user $u1$ and future users $u2$ and $u3$ may access $o2$. add$_i$, a type of Add, would be admitted as SA only if it satisfies α_2 stated in table 2. 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 user. In LR, some or all of the users who had access at Remove time may retain access (of course users joining later are not allowed to access the removed object—this respects the Authorization Provenance core property). In figure 3, if Remove ($o1$) is an SR, every group user (including $u1$) loses access to $o1$. If Remove ($o1$) is an LR, $u1$ can continue to access $o1$. However $u2$ and $u3$ will not have access to $o1$. remove$_i$, a type of Remove, would be admitted as SR only if it satisfies α_3 stated in table 2. 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 users 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 sessions in the room and its continuity when she leaves the room again. As the name implies, these properties are applicable only to returning users and are discussed below.

Unlike Membership Properties, these properties apply only to users and not to objects since users may have authorizations from past membership periods at re-Join time. Since the Renewal Properties are concerned about the status of such authorizations, a decision needs to be made for the user at re-Join time. In contrast, objects, being a passive entity, do not have past authorizations at re-Add time and hence renewal decisions need not be made.

Lossless Vs Lossy Join: In Lossless Join, a re-joining user does not lose authorization(s) held immediately prior to re-joining. A

Table 2: Summary of group membership semantics

Operation	Explanation	Property
Strict Join (SJ)	Only objects added after join time can be accessed	$\boldsymbol{\alpha_0} = \Box(\text{Authz} \rightarrow \blacklozenge(\text{Add} \wedge (\neg\text{Leave}\ \mathcal{S}\ \text{join}_i)))$
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	$\boldsymbol{\alpha_1} = \Box(\text{Authz} \rightarrow (\neg\text{leave}_i\ \mathcal{S}\ \text{Join}))$
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	$\boldsymbol{\alpha_2} = \Box(\text{add}_i \rightarrow (\neg\blacklozenge\text{Join} \rightarrow (\neg\text{Authz}\ \mathcal{W}\ \text{Add})))$
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	$\boldsymbol{\alpha_3} = \Box(\text{remove}_i \rightarrow (\neg\text{Authz}\ \mathcal{W}\ \text{Add}))$
Liberal Remove (LR)	Users who had access at remove time retain access	There exists a well-formed trace that does not satisfy α_3

Join operation that causes a user to lose some or all prior authorizations is called Lossy. Suppose in figure 2 $u1$ retains access to $o2$ at the time of Leave (due to LL). When $u1$ 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. Formula β_0 (table 3) characterizes Lossless Join. 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 $u1$ leaves, SL is applied and SJ is applied on re-join. A Restorative SJ in this scenario will allow $u1$ 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, $u1$ will continue to access $o1$ until the time of Leave. If LJ is applied on re-join, a Restorative LJ will allow $u1$ 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 user had a different history. However LTL does not enable one to compare different traces. The solution we take is to consider two different users within a single trace. Because the two users can have different histories with the same trace, this strategy enables us to formalize the property. We first state two formulas ρ_1 and ρ_2 which will then be used to specify the property β_1 as shown in table 3.

In formula ρ_1, users $u1$ and $u2$ both join the group at the same time by means of the same type of Join. Further, it looks at a case where $u1$ is authorized to access an object in the current state and $u2$ is not. ρ_2 says that this should also be the case in the previous state (and vice-versa). The Non-Restorative Join property is characterized by formula β_1. It states that if two users 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 $u1$ and $u2$ Join at the same time with same type, any access that is necessarily enabled by this Join for $u1$, would also be enabled for $u2$. Any additional access that $u1$ may have that $u2$ does not have could arise only because $u1$ 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 user 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 user joins a group using Restorative Join, some or all of the accesses to objects authorized during past membership period may be restored. Note that this is in addition to the authorizations that current Join may enable. The formula $\Box(\text{Join} \wedge ((\neg\text{Leave} \wedge \neg\text{Remove})\ \mathcal{S}\ (\text{Leave} \wedge \circleddash\ \text{Authz})) \rightarrow \text{Authz})$, 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 user 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 $u1$ re-joins the group. Because of SJ, only $o4$ and possibly $o5$ can be accessed. If $u1$ 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 $o3$. A Gainful Leave is useful in scenarios where an incentive is provided for a user to leave the group. This is commonplace in voluntary retirement and severance packages.

β_2 in table 3 characterizes Gainless Leave. This formula states that if the user 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 Gainless Leave property.

Non-Restorative Vs Restorative Leave: In Non-Restorative Leave, authorizations that the user 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 $u1$ left the group with LL and re-joins with Lossy SJ. In this case, $u1$ possibly loses access to both $o1$ and $o2$ at re-join time. Later on, if $u1$ leaves with Gainful LL, a Restorative Leave will allow $u1$ to re-gain access to $o1$ and $o2$ at the time of leave, but a Non-Restorative leave will not.

Table 3: Summary of group membership renewal semantics

Operation	Explanation	Property
Lossless Join	Authorizations prior to join time is not lost	$\boldsymbol{\beta_0} = \Box((\text{Join} \wedge \neg\text{Remove} \wedge \circleddash\ \text{Authz}) \rightarrow \text{Authz})$
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	$\rho_1 = (\text{join}_i(u1) \wedge \text{join}_i(u2) \wedge \text{Authz}(u1, o, r) \wedge \neg\text{Authz}(u2, o, r))$ $\rho_2 = \circleddash\ (\text{Authz}(u1, o, r) \wedge \neg\text{Authz}(u2, o, r))$ $\boldsymbol{\beta_1} = \forall i \Box(\rho_1 \rightarrow \rho_2)$
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	$\boldsymbol{\beta_2} = \Box((\text{Leave} \wedge (\neg\text{Join}\ \mathcal{U}\ (\text{Authz} \wedge \neg\text{Join}))) \rightarrow \circleddash\ ((\neg\text{Authz} \wedge \neg\text{Join})\ \mathcal{S}\ (\text{Authz} \wedge (\neg\text{Join}\ \mathcal{S}\ \text{Join}))))$
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	$\boldsymbol{\beta_3} = \Box(\text{Leave} \wedge \text{Authz} \rightarrow \circleddash\ \text{Authz})$
Restorative Leave	Authorizations from prior to join time may be restored	There exists a well-formed trace that does not satisfy β_3

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 regain access to objects for which she lost authorization when joining the group. Formula β_3 in table 3 characterizes Non-Restorative Leave.

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

4. THE π-SYSTEM SPECIFICATION

In this section, we discuss the construction of the specification for a family of g-SIS models, called the π-system. The π-system is a g-SIS specification that formally defines the conditions under which authorization can hold. We successfully verify using a model checker called NuSMV [12] that the π-system we develop is semantically correct—that is, it satisfies the core g-SIS properties.

We also show that the π-system satisfies a subset of Membership and Membership Renewal properties. To this end, the π-system allows any variation of Membership operations (Strict/Liberal). Thus, different users and objects may be given different types of respective operations. For example, SJ for $u1$, LJ for $u2$, SL for $u1$ and possibly LJ for $u1$. However, for Membership Renewal operations, we confine our scope to Join operations that are of type Lossless and Non-Restorative and Leave operations of type Gainless and Non-Restorative. These specific types of renewal operations are the most basic since they do not require us to treat past membership authorizations explicitly. Further, the semantics of other renewal operations are likely application dependant. For example, what exact authorizations will be disabled at Join time in Lossy Join depends on the application in question (similarly for Restorative Join, Gainful Leave and Restorative Leave). Henceforth, for convenience, we assume that SJ and LJ refer to operations of type Lossless and Non-Restorative. Similarly, SL and LL refer to Gainless and Non-Restorative operations.

REMARK 4.1. *The π-system only allows group operations of type indicated below:*

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

Furthermore, note that as per the definition of Strict Join (section 3.3 and table 2), the user may access *some or all* objects that are added after Join time. However, in the π-system, we will use the following specific interpretation of Strict and Liberal operations which effectively replaces "some or all" with "all".

- On SJ, the joining user may access *all* the objects added after Join time. Objects added before Join time may not be accessed. On LJ, the joining user may access *all* the current objects added before and after Join time.
- On SL, the users lose access to all objects. On LL, access to *all* the group objects authorized at the time of Leave may be retained.
- On SA, *all* the users who joined prior to Add time may access. Users joining later may not access. On LA, *all* the users who joined prior to and after Add time may access.
- On SR, all the users lose access to the object. On LR, *all* the users who had access at Remove time may retain access.

4.1 Construction of π-system

There are two scenarios to consider when a user requests access to an object: (i) the user Join event occurred prior to object Add event or (ii) the object Add event occurred prior to user 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 user joined the group (figure 4).

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

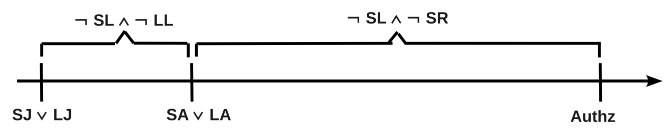


Figure 4: Formula λ_1.

Since Join occurred prior to Add, regardless of whether the object was LA'ed[4] or SA'ed or whether the user was SJ'ed or LJ'ed, the user should be authorized to access the object in both cases as per our core Availability Property (φ_5). Formula λ_1 says that the user 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 user 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 users at remove time to retain access and LL authorizes a leaving user to retain access to objects authorized during the membership period. Similarly, if the user 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 our core provenance property (formula φ_2).

Scenario (ii) 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* users (that is, the users who joined the group prior to object Add). Clearly, regardless of the type of Join, the user is not authorized to access objects that were SA'ed prior to the user Join time. Thus Authz cannot hold in cases (a) and (b).

Consider cases (c) and (d) where the object is LA'ed to the group. In case (c), the object is LA'ed and the user is SJ'ed. An SJ'ed user 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 user 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 user has not been SL'ed and the object has not been SR'ed since the user 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 formally specify the π-system.

DEFINITION 4.2 (π-SYSTEM). *The π-system is given by:*

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

Note that π is a syntactically correct g-SIS specification by definition. π says that a user 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 3.1. We show that π is also semantically correct in the following subsection.

4.2 Formal Analysis

In this section, we show that the π-system entails the Core and Membership Renewal properties. Since Membership operations

[4]We use the abbreviation of the form "LA'ed" to refer to the fact that object o is Liberally Added.

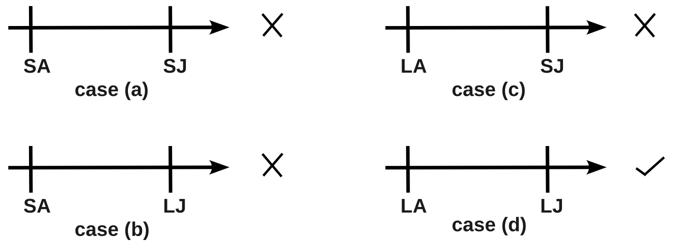


Figure 5: Cases when Add occurs prior to Join.

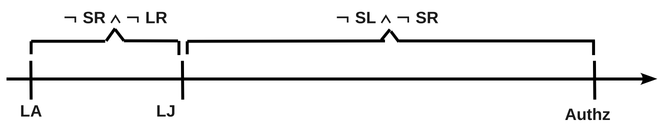


Figure 6: Formula λ_2.

are mixed (i.e., both Strict and Liberal operations are allowed), 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 users and objects in a group. We begin with the Entailment Theorem.

THEOREM 4.3 (ENTAILMENT THEOREM). *The π-system entails the Core Properties (φ_0 to φ_5) and Membership Renewal Properties (β_0 to β_3):*

$$\pi \vDash (\bigwedge_{0\le q\le 5} \varphi_q \wedge \bigwedge_{0\le r\le 3} \beta_r)$$

We utilize the model checker NuSMV [12] 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 and determines whether 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. Due to space constraints, we do not show the code in this paper. A detailed discussion of the verification can be found in our technical report [19]. Here, the NuSMV model 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 π-system in the antecedent and the Core and Membership Renewal properties as the consequent. NuSMV takes the model and the LTL formula and determines whether the formula holds in all possible traces generated by the model. The output from NuSMV in [19] shows that the LTL formula holds against the model. Thus we verify the Entailment Theorem, i.e., 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 π-system is guaranteed to be a *g-SIS* specification (i.e., it would satisfy the core properties.) For example, one can derive the Most Restrictive 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. That is, the core properties can be satisfied, namely, by the π-system, and do not conflict with each other. Further, note that the theorem proves that the π-system is semantically correct (the semantic definition 3.2 of a g-SIS specification is actually weaker). As mentioned earlier, the Most Restrictive 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 4.4 (μ-SYSTEM). *The μ-system is given by:*

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

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 4.5 (MOST RESTRICTIVE ENTAILMENT THEOREM). *The μ-system entails the Core Properties (φ_0 to φ_5), Membership (α_0 to α_3) and Membership Renewal Properties (β_0 to β_3).*

$$\mu \vDash (\bigwedge_{0 \leq j \leq 5} \varphi_j \wedge \bigwedge_{0 \leq k \leq 3} \alpha_k \wedge \bigwedge_{0 \leq l \leq 3} \beta_l)$$

We take a similar approach to that of Theorem 4.3 to prove this theorem. Again, the proof is given in [19]. 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 user and object in the group. One can derive more specifications from the π-system where some of the membership operations may vary while others remain fixed for various users and objects.

4.3 Application of π-system

We now discuss how the π-system applies to the scenarios discussed earlier in section 1.

Magazine Subscription: In general, user 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, however 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.

5. 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 extensions along three dimensions: read-write g-SIS model, multiple groups and application of Attribute-Based Access Control (ABAC) in g-SIS.

We strongly believe that information sharing, being distributed in nature, should support versioning so different users 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 with overlapping users. One natural relationship between groups is to impose a hierarchy similar to lattice based models such as Bell-LaPadula (BLP) [6], Chinese Wall [10], etc. and study how temporal interactions in g-SIS relate to these models.

Finally, we need to investigate how specifications that are purely based on temporal ordering of user and object events can be complemented with ABAC. We believe attributes such as "roles" are critical in many information sharing scenarios.

Our focus in this paper has been on specifying semantics of the group operations. In general, there could be any number of variations beyond those identified in this paper. For example, there could be post-dated join, promoting a join operation from SJ to LJ, etc. One of our goals is to identify which operations are fundamental and which are for convenience and add no new expressive power ([9, 38]). In the Policy, Enforcement and Implementation (PEI) framework of [35], the models developed in this paper are at the P layer. To implement g-SIS we need enforcement models such as in [17] possibly requiring additional properties.

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 any g-SIS specification. 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 automated techniques such as model checking. Finally, we discussed the specification of a family of g-SIS models and showed using NuSMV that the specification satisfies the g-SIS properties. Ultimately, the value of the semantics of g-SIS discussed in this paper will be determined by whether or not these are found useful in practical systems of the future.

Acknowledgments

The authors are partially supported by NSF grants IIS-0814027, IIS-0814027, CCR-0325951, CCF-0524010 and CNS-0716750, AFOSR MURI grant FA9550-08-1-0265, THECB ARP grant 010115-0037-2007 and grants from the State of Texas Emerging Technology Fund and Intel Corporation.

6. REFERENCES

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

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

[3] 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.

[4] 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.

[5] S. Bandhakavi, C. C. Zhang, and M. Winslett. Super-sticky and declassifiable release policies for flexible information dissemination control. In *Proc. of 5th ACM workshop on Privacy in electronic society*, pages 51–58, 2006.

[6] 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.

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

[8] 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.

[9] 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.

[10] D. Brewer and M. Nash. The Chinese Wall security policy. *IEEE Symp. on Security and Privacy*, pages 206–214, 1989.

[11] D. W. Chadwick and S. F. Lievens. Enforcing sticky security policies throughout a distributed application. In *Proc. of ACM Workshop on Middleware security*, 2008.

[12] 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.

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

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

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

[16] 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.

[17] R. Krishnan, J. Niu, R. Sandhu, and W. Winsborough. Stale-safe security properties for group-based secure information sharing. In *Proc. of 6th ACM workshop on Formal Methods in Security Engineering*, pages 53–62, 2008.

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

[19] R. Krishnan, R. Sandhu, J. Niu, and W. Winsborough. Formal models for group-centric secure information sharing. *UTSA CS Tech Report CS-TR-2009-002*, March 2009.

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

[21] R. Lipton and L. Snyder. A Linear Time Algorithm for Deciding Subject Security. *Journal of the ACM*, 1977.

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

[23] C. McCollum, J. Messing, and L. Notargiacomo. Beyond the pale of MAC and DAC - defining new forms of access control. *Proc. of IEEE Symp. on Sec. and Priv.*, 1990.

[24] M. Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In *Proc. of 14th International Workshop on Database and Expert Systems Applications*, pages 377–382, 2003.

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

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

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

[28] J. H. Saltzer. Protection and the control of information sharing in multics. *Comm. of ACM*, 17(7):388–402, 1974.

[29] R. Sandhu. The ntree: a two dimension partial order for protection groups. *ACM Trans. on Comp. Syst.*, 1988.

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

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

[32] R. Sandhu. Roles versus groups. In *Proc. of the first ACM Workshop on Role-based access control*, 1996.

[33] 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.

[34] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Computer*, 1996.

[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*, 2006.

[36] R. Sandhu and M. Share. Some Owner-Based Schemes with Dynamic Groups in the Schematic Protection Model. In *Proc. of IEEE Symposium on Security and Privacy*, 1986.

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

[38] 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.

[39] J. Warner, V. Atluri, R. Mukkamala, and J. Vaidya. Using semantics for automatic enforcement of access control policies among dynamic coalitions. In *SACMAT*, 2007.

[40] K. Zeilenga. Lightweight Directory Access Protocol (LDAP): Technical specification road map. *RFC 4510*.