On the Suitability of Dissemination-centric Access Control Systems for Group-centric Sharing

William C. Garrison III
bill@cs.pitt.edu

Yechen Qiao
yeq1@cs.pitt.edu

Adam J. Lee
adamlee@cs.pitt.edu

Department of Computer Science
University of Pittsburgh
Pittsburgh, Pennsylvania 15260

ABSTRACT

The Group-centric Secure Information Sharing (g-SIS) family of models has been proposed for modeling environments in which group dynamics dictate information-sharing policies and practices. This is in contrast to traditional, dissemination-centric sharing models, which focus on attaching policies to resources that limit their flow from producer to consumer. The creators of g-SIS speculate that it may not be strictly more expressive than dissemination-centric models, but that it nevertheless has pragmatic efficiency advantages in group-centric scenarios [12]. In this paper, we formally and systematically test these characteristics of an access control system's suitability for a scenario—expressiveness and cost—to evaluate the capabilities of dissemination-centric systems within group-centric workloads. We show that several common dissemination-centric systems lack the expressiveness to meet all security guarantees while implementing the wide range of behavior that is characteristic of the g-SIS models, except via impractical, convoluted encodings. Further, even more efficient implementations (admissible under relaxed security requirements) suffer from high storage and computational overheads. These observations support the practical and theoretical significance of the g-SIS models, and provide insight into techniques for evaluating and comparing access control systems in terms of both expressiveness and cost.

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

Keywords

Information sharing; Access control; Suitability analysis

CODASPY'14, March 3–5, 2014, San Antonio, Texas, USA.

ACM 978-1-4503-2278-2/14/03 $15.00.
http://dx.doi.org/10.1145/2557547.2557566

1. INTRODUCTION

Group-centric Security Information Sharing (g-SIS) [11,12] is a modeling paradigm and class of access control models that has been proposed for sharing environments in which users and resources are brought together in groups to facilitate collaboration and efficient exchange of information. Its creators contrast it with the traditional, *dissemination-centric* modeling paradigms that are currently used in access control and information sharing. In dissemination-centric sharing, emphasis is placed on attaching policies (and/or attributes that determine policy) to resources as they are created or made available. These policies then restrict which consumers can access the resources. In g-SIS, on the other hand, users are granted access to resources based on their temporal membership in groups—e.g., if object o is added to a group that a user u is a member of, u will be granted access to o. The rules for users who join later, objects that are removed, and users who leave are determined by the particular parameterization of g-SIS used.

Although g-SIS seems to represent its motivating scenarios rather elegantly, there has, to date, been no fully-functional implementation of the g-SIS models. This may partially be due to the creators' hypothesis that group- and dissemination-centric techniques yield the same theoretical set of capabilities (i.e., that the set of dissemination-centric models is, collectively, equal in expressiveness to the set of group-centric models) [12]. Even assuming that this hypothesis holds, g-SIS seems pragmatically better-suited to group-centric workloads than systems such as role-based access control built with dissemination-centric uses in mind (in both ease of implementation and efficiency). Thus, in this work we evaluate the expressiveness hypothesis and other questions regarding the capabilities of dissemination-centric access control systems within the context of group-centric sharing workloads. Specifically, we investigate the following questions:

1. Which systems based on the g-SIS models can be *safely* implemented within, or simulated by, dissemination-centric access control systems?
2. How *strong* are the security properties that can be guaranteed by dissemination-centric systems when implementing workloads based on the g-SIS models?
3. How *efficiently* can dissemination-centric systems implement workloads based on the g-SIS models?
4. What practically-interesting instantiations of the g-SIS models *cannot* be safely and efficiently implemented by dissemination-centric systems?

While investigating these questions, we formalize several instantiations of the g-SIS models. Some are based on mathematical extrema in the space of g-SIS instantiations and are intended to represent a diverse cross-section of the capabilities of the g-SIS models. Others are based on realistic use cases to which g-SIS seems particularly well-suited. We then employ *parameterized expressiveness* [9]—a fine-grained, parameterized generalization of simulation-based expressiveness techniques—to answer questions #1 and #2. To evaluate question #3, we utilize a Monte Carlo simulation technique to generate traces of group-based sharing actions. These actions are then simulated in g-SIS systems, and they are translated into equivalent action sequences in dissemination-centric access control systems. The costs of executing these traces are recorded and compared. Finally, we address question #4 by interpreting and analyzing the results of questions #1–3, discussing the various failings of the use of dissemination-centric techniques in group-centric environments.

Our analysis provides new insights into the relationship between group-centric and dissemination-centric sharing, and represents the first in-depth analysis into the use of g-SIS. We support the notion that g-SIS is a practically significant proposal by demonstrating the inability of traditional systems to satisfy many of its models safely and efficiently. More fundamentally interesting, to the best of our knowledge, our analysis represents the first systematic examination and comparison of access control systems based on both their theoretical capabilities (i.e., relative expressive power) and more pragmatic notions of quantitative efficiency (i.e., implementation costs). We believe this style of analysis has the potential to answer many practical questions that arise when examining an application's access control needs, and that our demonstration of these techniques toward understanding the impact of g-SIS supports this claim by example.

The rest of this paper is structured as follows. In Section 2, we introduce g-SIS and expressiveness analysis. In Section 3, we describe the specific g-SIS instantiations that we will be analyzing. In Section 4, we explain our parameterized expressiveness analysis and interpret its results. In Section 5, we discuss our techniques for cost analysis via Monte Carlo simulation, and present the results of this analysis. We discuss our results and reason about the drawbacks of utilizing dissemination-centric techniques for group-centric workloads in Section 6. Finally, we discuss other related work in Section 7 and conclude in Section 8.

2. BACKGROUND AND PRIOR WORK

We now discuss the immediately relevant prior work. We first overview g-SIS, the group-centric secure information sharing paradigm that inspires our workloads. We then provide an overview of prior work on access control expressiveness analysis, with a particular focus on the technique used in this work: parameterized expressiveness. A discussion of other related work is deferred to Section 7.

2.1 The g-SIS Models

The g-SIS models encompass a wide range of access control systems and behavior, and seem to subsume other group- and role-based access control and information sharing systems [16, 18,21]. The motivating scenarios which inspired g-SIS include periodical subscriptions and secure message rooms. It is conceptually simpler to model such scenarios within g-SIS than by using dissemination-centric access control; that g-SIS is inherently more capable of representing such scenarios is a claim that we make and support through the present work.

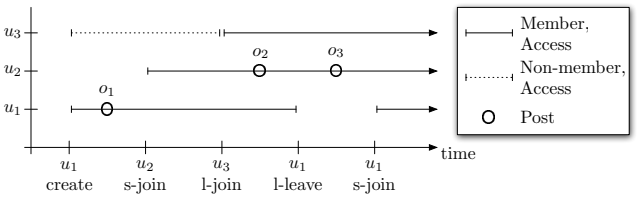


Figure 1: Example accesses in a single group in g-SIS

A major distinguishing feature of g-SIS is its preservation of a full membership record for groups. A graphical depiction of such a record is shown in Figure 1. Users can join and leave groups, and objects can be added and removed from groups. The log of these events is used to decide whether a user can access an object. The basic operations each have numerous variants, ranging from *strict* to *fully liberal*. The semantics of these variants depends on the type of action.

Users who perform a strict join to a group receive access only to objects added after they join, whereas a fully liberal join grants immediate access to all existing objects. Note that, in Figure 1, user u_2 performs a strict join whereas u_3 performs a liberal join, meaning that u_3 has access to o_1 while u_2 does not. A strict leave rescinds all of the user's accesses within the group; a liberal leave allows the user to retain access. Performing a strict add of a message to a group grants only current members access; a liberal add grants future members access as well. In Figure 1, if o_3 is added using strict add, then u_1 could not access it even after a liberal join. Finally, a strict remove rescinds access to the removed object from all users in the group, while a liberal remove allows users to retain access.

The application of these operations to the motivating scenarios (subscriptions, secure messaging) are, thus, fairly obvious. A subscription service can provide a base level of service with no access to back issues and no continued access after canceling. For an additional fee, users can access back issues (via liberal join) or maintain access to their issues after canceling (via liberal leave). Secure messaging can make various uses of combinations of strict and liberal actions: users can liberal leave before a discussion with which they have a conflict of interest and strict join at its conclusion; objects to which new members should not gain automatic access can be strict added while others are liberal added.

Restrictions on g-SIS in this paper. In this work, we primarily focus on strict and purely-liberal variants of actions, such as those described above. However, g-SIS also supports variants that lie between these extremes. For example, in some systems a liberal (but not *fully* liberal) leave might allow a user to retain only *some* of their accesses when leaving. The semantics of these variants are application-dependent. In this work, liberal actions are assumed to be fully liberal unless otherwise specified. Finally, g-SIS supports variants of join with different behavior for users who have left and re-joined a group. Lossy join may revoke some or all permissions retained from a past liberal leave; lossless join will not revoke any permissions. Restorative join may re-grant some or all permissions revoked by a past leave; nonrestorative join will not re-grant any permissions. In this work, we focus on lossless, nonrestorative joins unless otherwise specified.

2.2 Expressiveness Analysis

Prior Work. We first evaluate the ability of dissemination-centric approaches to implement group-centric workloads in terms of theoretical capability, or *expressiveness*. The expressiveness of an access control system describes the set of policies that it can represent. Many notions of expressiveness have been studied in the literature [1,3,5,7,9,13,14,17,19,20, 22], and today there is much ambiguity in saying simply, "$\mathcal{Y}$ is more expressive than $\mathcal{Z}$." Some notions only concern whether one system can represent all the same sets of authorizations as another [5], while others reduce a system's storage to an abstracted structure to show how one can subsume another structurally [1]. Somewhere in between, other notions of expressiveness define a set of queries (including authorizations) that must be preserved [9,22]. Some may only concern properties of the states [3], but most go further, enforcing preservation of state reachability [5,7,17,19]: i.e., "not only are all $\mathcal{Z}$'s states expressible in $\mathcal{Y}$, $\mathcal{Y}$'s mechanism for transforming states allows us to build them." Some notions even go so far as to enforce back-reachability, by requiring that any state in $\mathcal{Y}$ representing a state in $\mathcal{Z}$ should not be able to be transformed into a state that the original in $\mathcal{Z}$ could not be [1,22]. There are several other dimensions in which expressiveness notions differ, including the use of strong simulation (a state change in $\mathcal{Z}$ must be simulated in a single state change in $\mathcal{Y}$) vs. weak simulation (a state change in $\mathcal{Z}$ can be simulated in a number of changes in $\mathcal{Y}$).

Parameterized Expressiveness. Given this state of affairs, and our lack of tools for reconciling various notions of expressiveness, *parameterized expressiveness* is an attractive approach [9]. Parameterized expressiveness is an access control expressiveness framework created to generalize other expressiveness notions. This framework has much in common with other techniques, defining systems based on state machines and building various types of simulations between them. However, rather than commit to a single set of properties preserved by those simulations, the set of security guarantees to be preserved is a parameter of the analysis. This allows the notion of expressiveness used in a particular analysis to match the requirements of the application within which the systems are candidates for use.

This focus on expressiveness within the context of a particular application allows expressiveness to be defined with respect to *workloads*, which are descriptions of specific application environments. This allows an analyst to be more precise about expressiveness in a practical sense, with a focus on the capability to represent the policies that are needed by an application, rather than considering the ability to represent a superset of another system's policies. Whereas other expressiveness notions allow an analyst to make statements such as, "System $\mathcal{Y}$ admits a simulation of form S of $\mathcal{Z}$," parameterized expressiveness makes statements such as, "If system $\mathcal{Z}$ can implement workload $\mathcal{W}$ while preserving the set of properties $\mathcal{G}$, then $\mathcal{Y}$ can implement $\mathcal{W}$ while preserving $\mathcal{G}$," which is more closely aligned with the practical concerns associated with a system's expressiveness. For these reasons, parameterized expressiveness is a good fit for our goals.

PE Formalisms. We now summarize key structures used during parameterized expressiveness analyses. Interested readers can refer to [9] for full details.

- An access control *model* defines how protection state is encoded and interpreted. Given model $\mathcal{M} = \langle \mathcal{S}, \mathcal{R}, \mathcal{Q}, \vdash \rangle$, $\mathcal{S}$ is a set of states, $\mathcal{R}$ is a set of authorization requests, and $\mathcal{Q}$ is a set of queries. $\mathcal{Q}$ includes at least $auth(r)$ for each request r, and the entailment relation $\vdash$: $\mathcal{S} \times \mathcal{Q} \to \{\text{TRUE}, \text{FALSE}\}$ defines which queries are true in each state. We sometimes denote $\mathcal{S}$ as $States(\mathcal{M})$ and $\mathcal{Q}$ as $Queries(\mathcal{M})$. A *theory*, an assignment for each query in state x, is denoted $Th(\text{x})$. The subset of $Th(x)$ consisting of only $auth$ queries is $Auth(x)$. The set of all possible theories in $\mathcal{M}$ is denoted $Th(\mathcal{M})$.
- An access control *system* refines a model by defining methods for transforming the state. A system is thus a state machine $\langle \mathcal{M}, \mathcal{L}, next \rangle$ which defines its parent model $\mathcal{M}$, a set of labels $\mathcal{L}$ (representing access control commands), and the transition function $next : States(\mathcal{M}) \times \mathcal{L} \to States(\mathcal{M})$. We sometimes denote the set of states, theories, and queries in model $\mathcal{M}$ of system $\mathcal{Y}$ as $States(\mathcal{Y})$, $Th(\mathcal{Y})$, and $Queries(\mathcal{Y})$, respectively.
- An access control *workload* describes the access control demands of an application. This is accomplished by describing an idealized access control system encoding required functionality, and the set of allowable traces of labels through that system. A workload $\mathcal{W} = \langle \mathcal{A}, \mathcal{T} \rangle$ thus defines access control system $\mathcal{A}$ and set of traces $\mathcal{T}$. Each trace is a pair $\langle s_1, \tau \rangle$, where $s_0 \in States(\mathcal{A})$ is the initial state and $\tau = \ell_1 \circ \ell_2 \circ \cdots$ is a sequence of labels in $\mathcal{A}$: $\forall i, \ell_i \in Labels(\mathcal{A})$.

Given these formalisms, we now define the particular systems and workloads inspired by g-SIS, with respect to which we will evaluate dissemination-centric systems.

3. INSTANTIATIONS OF G-SIS

In this section, we describe the g-SIS systems and workloads that form the basis of our analyses in Sections 4 and 5. We describe these systems using the formalism described above to facilitate our analyses.

3.1 The g-SIS$_0$ Model

The g-SIS$_0$ model defines the state representation and queries for our g-SIS systems. It defines structures for storing the records for all combinations of the basic g-SIS strict and liberal actions (join, leave, add, remove). The authorization request is defined for non-restorative joins. In the g-SIS$_0$ model, states are comprised of the following fields.

- Sets S, O, G, and T of subjects, objects, groups, and times
- $>_T$, the total order on T
- $Time \in T$, the current time
- $StrictJoin \subseteq S \times G \times T$, the record of strict joins
- $LiberalJoin \subseteq S \times G \times T$, the record of liberal joins
- $StrictLeave \subseteq S \times G \times T$, the record of strict leaves
- $LiberalLeave \subseteq S \times G \times T$, the record of liberal leaves
- $StrictAdd \subseteq O \times G \times T$, the record of strict adds
- $LiberalAdd \subseteq O \times G \times T$, the record of liberal adds
- $StrictRemove \subseteq O \times G \times T$, the record of strict removes
- $LiberalRemove \subseteq O \times G \times T$, the record of liberal removes

$$
\begin{array}{l}
authForward(s,o,g) \triangleq \exists t_1,t_2.(\\
\quad Join(s,g,t_1) \wedge \\
\quad Add(o,g,t_2) \wedge \\
\quad t_2 > t_1 \wedge \\
\quad \forall t_3.(\\
\quad\quad Leave(s,g,t_3) \Rightarrow (t_1 > t_3 \vee t_3 > t_2) \wedge \\
\quad\quad StrictLeave(s,g,t_3) \Rightarrow t_2 > t_3 \wedge \\
\quad\quad StrictRemove(o,g,t_3) \Rightarrow t_2 > t_3 \\
\quad) \\
)
\end{array}
\qquad
\begin{array}{l}
authBackward(s,o,g) \triangleq \exists t_1,t_2.(\\
\quad LiberalJoin(s,g,t_1) \wedge \\
\quad LiberalAdd(o,g,t_2) \wedge \\
\quad t_1 > t_2 \wedge \\
\quad \forall t_3.(\\
\quad\quad Remove(o,g,t_3) \Rightarrow (t_2 > t_3 \vee t_3 > t_1) \wedge \\
\quad\quad StrictLeave(s,g,t_3) \Rightarrow t_1 > t_3 \wedge \\
\quad\quad StrictRemove(o,g,t_3) \Rightarrow t_1 > t_3 \\
\quad) \\
)
\end{array}
$$

$$auth(s,o,g) \triangleq authForward(s,o,g) \vee authBackward(s,o,g)$$

Figure 2: The authorization procedure for g-SIS$_0$.

The g-SIS$_0$ model defines queries $Member(s,g)$ for whether subject s is currently a member of group g and $Assoc(o,g)$ for whether object o is currently associated with group g. These are answered in the obvious way. Authorization requests are answered as described in Figure 2. Here, *authForward* applies in cases where the user joined the group before the object was added, and *authBackward* applies when the user joined after the object was added.

3.2 Extrema Systems

Top, bottom, and role-like g-SIS are systems of the g-SIS$_0$ model. Each of these systems contains a subset of the full set of actions supported in g-SIS$_0$, and represents a different extreme in terms of resulting behavior.

Top g-SIS Top g-SIS contains only strict actions. Since all adds and joins are strict, there is no need for *authBackward*. Newer users to a group always have a subset of accesses of older users, users who leave retain no permissions to group-associated objects, and objects that are removed are no longer accessible by group members. We describe top g-SIS in full in Appendix A.

Bottom g-SIS Bottom g-SIS contains only liberal actions. Thus, a subject is granted access to an object as long as they belonged to a group at the same time at some point (currently or in the past), however briefly. Access to added objects is granted to current and new users, but once an object is removed no new users are granted access. Thus, new users tend to have fewer accesses than older users.

Role-like g-SIS Finally, role-like g-SIS is an approximation of a role-based access control system within g-SIS. It allows liberal join and add actions and strict leave and remove. Thus, all current members have access to all current objects, but users who leave lose all access, and objects that are removed are revoked from all users.

3.3 Workloads

In addition to the above extrema systems, we also study several more realistic parameterizations that reflect how g-SIS might be used in practice. Conceptually, these lie somewhere in the g-SIS spectrum between the extrema systems defined above, and represent real-world usages of group-centric techniques. We formalize these as workloads.

PC This is an instance of the "secure message room" example use case of g-SIS [11, 12] that is defined to model academic program committee discussions. It is also based on the g-SIS$_0$ model, and includes commands for liberal joining PC groups, as well as resigning via strict leave. The workload requires conflict-of-interest handling, so members can liberal leave before a discussion with which they have a COI, and strict join after it concludes. All discussion is liberal added. Traces restrict execution of this workload to sequential phases. In the creation phase, program chairs create PC groups. In the joining phase, PC members join PC groups. In the discussion phase, PC members discuss (add objects to groups) and execute COI patterns.

PSP The Playstation Plus premium gaming service [15] uses temporal constraints to decide accesses, and is thus a natural fit for modeling in g-SIS. PSP uses a g-SIS model with an extension over g-SIS$_0$: the *auth* query supports *restorative* joins. Subscribers liberal join, and strict leave when canceling. If a user cancels and later joins again, she is re-granted access to all objects she had before leaving (except those which have been strict removed). Managers liberal add promotions (free games and discounts). When a promotion is complete, free games are liberal removed (users who are members at the time a free game is available may continue to access it as long as they are a member), while discounts are strict removed and thus become inaccessible to all users. Trace restrictions allow users to subscribe in 3-month increments. The managers add and remove several promotions each week, maintaining the same total number for each group.

4. EXPRESSIVENESS ANALYSIS

In this section, we describe the details of our expressiveness analysis using parameterized expressiveness and present a summary of the results. Full details, including full specifications of workloads, systems, implementations, and reductions, are deferred to a companion technical report [8].

A fundamental construction in parameterized expressiveness is the access control *implementation*, the set of mappings constructed to prove that a workload can be satisfied by a system. An implementation of $\mathcal{W}$ in $\mathcal{Y}$, $\langle \alpha, \sigma, \pi \rangle$, defines a state mapping $\sigma : States(\mathcal{W}) \to States(\mathcal{Y})$, a label mapping $\alpha : States(\mathcal{Y}) \times Labels(\mathcal{W}) \to Labels(\mathcal{Y})^*$, and a query mapping π which contains, for each $q \in Queries(\mathcal{W})$, a function $\pi_q : Th(\mathcal{Y}) \to \{\text{TRUE}, \text{FALSE}\}$ which maps a set of system query values to a value for the workload query q. Thus, each π_q is a procedure for answering workload query q given the value of each system query in the current state.

An access control *reduction*, then, is a set of mappings allowing us to prove that a system $\mathcal{Z}$ is at least as expressive as system $\mathcal{Y}$ with respect to a particular set of security guarantees $\mathcal{G}$. This is written $\mathcal{Y} \leq^{\mathcal{G}} \mathcal{Z}$, and indicates that any workload $\mathcal{W}$ that can be implemented in $\mathcal{Y}$ with guarantees

$\mathcal{G}$ can also be implemented in $\mathcal{Z}$ with $\mathcal{G}$. Making this type of statement is the goal of conducting parameterized expressiveness, and is a more precise and pragmatic view of a system's capabilities than other expressiveness techniques provide. A reduction from $\mathcal{Y}$ to $\mathcal{Z}$, $\langle \sigma, \pi \rangle$, defines a state mapping σ and a query mapping π where the state-mapping preserves the query-mapping ($\forall s \in States(\mathcal{Y})\ Th(s) = \pi(Th(\sigma(s)))$). The conditions put on the reduction differ based on the set of security guarantees it preserves.

4.1 Security Guarantees

In this work, we consider the following security guarantees.

Correctness Correctness is a bare minimum requirement for any implementation. Intuitively, correctness says the following: a workload state's image in a system answers mapped queries exactly as the original state answers the original queries; and the same resulting system state is reached by executing a workload action and mapping the result into the system as is reached by mapping the initial state and executing the action's image in the system. More precisely, given a workload, $\mathcal{W} = \langle \mathcal{A}, \mathcal{T} \rangle$, a system $\mathcal{Y}$, and an implementation $\langle \alpha, \sigma, \pi \rangle$, the implementation is correct if σ preserves π (i.e., for every workload state w, $Th(w) = \pi(Th(\sigma(w)))$ and α preserves σ (i.e., for every workload state w and label ℓ, $\sigma(next(w, \ell)) = terminal(\sigma(w), \alpha(\sigma(w), \ell))$).

Weak AC-Preservation This guarantee is a weaker version of AC-preservation [9]. Intuitively, AC-preservation says that $\pi_{auth(r)}$ must map authorization request r from workload state w to system state $\sigma(w)$ directly, checking whether $\sigma(w) \vdash auth(r)$. This forces the workload and system to have the same format for requests. However, this is not the case with the workloads and systems we consider in this work. Specifically, g-SIS requests ask whether a subject has access to an object in a particular group, while, e.g., $RBAC_0$ requests ask whether a subject has access to a permission. We define weak AC-preservation which captures the spirit of AC-preservation (ensures the use of the authorization procedure of the system) but that allows us to answer a workload authorization using the system's authorization procedure, even if their requests use different formats. Thus, we allow the use of a *request transformation* function f, so we can ask $auth(f(r))$ for some function f. Formally, we require the following: For any workload state w and workload request r, $\pi_{auth(r)}(Th(\sigma(w))) = \text{TRUE} \Rightarrow \sigma(w) \vdash auth(f(r))$; and, for any workload state w and system request r', $\sigma(w) \vdash auth(r') \Rightarrow \exists r.(\pi_{auth(r)}(Th(\sigma(w))) = \text{TRUE} \wedge f(r) = r')$.

Homomorphism The homomorphic property eliminates implementations that abuse system state by encoding workload state in a way that is fragile to string substitutions. Without this requirement, an implementation can, e.g., store unbounded state in a single user name by encoding whole relations as a single string. A homomorphic mapping f is one in which $f(x)[v] = f(x[v])$ for any constant string substitution $[v]$. A homomorphic implementation is one in which each mapping is homomorphic. Intuitively, this requires that data elements be opaque, and that the symbol representing any element (e.g., user, object, role) can be substituted for any other without affecting the behavior of the system.

Safety A safe implementation is one that does not grant or revoke unnecessary permissions during the execution of the image of a single workload label. That is, if executing workload label ℓ in the implementing system yields the state sequence $\langle s_1, \ldots, s_k \rangle$, then for all s_i in the sequence, $Auth(s_i) \setminus Auth(s_0) \subseteq Auth(s_k) \setminus Auth(s_0)$ and $Auth(s_0) \setminus Auth(s_i) \subseteq Auth(s_0) \setminus Auth(s_n)$. Intuitively, safety ensures that the intermediate states through which a system travels while implementing a single workload label do not add or remove granted requests except those that must be added or removed as determined by the start and end states. We consider safety for implementations only, as there is no known metatheorem for proving safety via reduction.

4.2 Dissemination-Centric Systems

We choose several dissemination-centric access control systems as candidates for implementing the group-centric workloads described in Section 3. In particular, we focus on role- and group-based models. While these access control models are dissemination-centric, they provide a level of indirection between subjects and objects that enables greater expressiveness than models based on the access matrix or access control lists [14, 16]. Comparing to group-enabled dissemination-centric access control systems enables our analysis to more directly compare the effect of the group-centric paradigm, whereas comparing to non-group-enabled systems would be more likely to highlight simply the advantage of the additional level of indirection provided by groups. Thus, we evaluate the following dissemination-centric systems.

RBAC $RBAC_0$ is the most basic role-based access control system proposed in the RBAC standard [18]. States contain the set of users U, set of roles R, and set of permissions P, as well as relations between them: $UR \subseteq U \times R$ describes users' membership in roles, and $PA \subseteq R \times P$ describes permissions' assignment to roles. A user u is authorized to permission p if $\exists r.(\langle u, r \rangle \in UR \wedge \langle r, p \rangle \in PA)$. Labels allow adding and removing from all of U, R, P, UR, and PA.

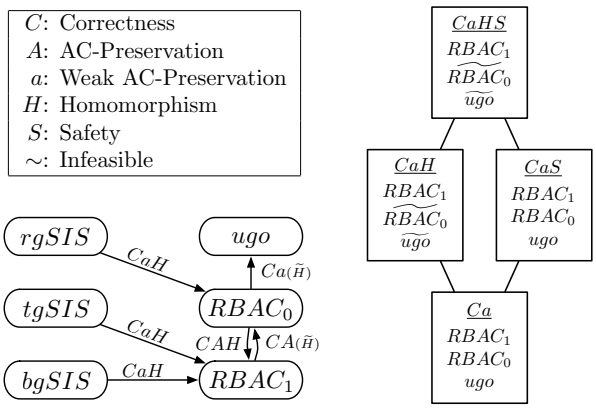
Hierarchical RBAC While $RBAC_0$ grants a level of indirection between users and permissions, $RBAC_1$ includes a hierarchical structure over roles to further extend this abstraction. $RBAC_1$ includes all state elements of $RBAC_0$ as well as the role hierarchy $RH \subseteq R \times R$, a binary relation over R whose transitive closure is the *Senior* partial order (we sometimes designate the transitive, reflexive closure $\geq$). In hierarchical RBAC, a user inherits all permissions from roles junior to roles she is explicitly assigned. That is, a user u is authorized to permission p if $\exists r_1, r_2.(\langle u, r_1 \rangle \in UR \wedge \langle r_2, p \rangle \in PA \wedge r_1 \geq r_2)$. Labels allow full manipulation of all state elements. We fully define $RBAC_1$ in Appendix A.

UNIX Permissions Finally, the *ugo* system is based on the *user, group, other* system of access control in UNIX. Thus, if $RBAC_0$ and $RBAC_1$ fill the need for a commonly-used industrial standard system, *ugo* fills the role of a common consumer system. In *ugo*, objects can be associated with an owner user and group, and permissions are then granted to the user, the group, or everyone else.

Thus, we evaluate standard, widely-deployed access control systems, in both the industrial and consumer spaces. These systems are likely candidates for a system administrator who desires to implement a group-centric workload using available and trusted access control mechanisms.

4.3 Expressiveness via System Reductions

We now present a summary of the system reductions proving expressiveness statements comparing the chosen



(a) Reductions (b) Implementations

Figure 3: Expressiveness analysis results

dissemination-centric systems and the g-SIS extrema systems. A summary of the expressiveness reductions, including a key to our shorthand for denoting the guarantees a reduction satisfies, is shown in Figure 3a. We now describe the major results depicted in this figure. A sample proof is provided in Appendix A, with the remainder deferred to a companion technical report [8].

First, we note that it is simple to construct a reduction from role-like g-SIS ($rgSIS$) to $RBAC_0$ that satisfies all considered security guarantees (correctness, weak AC-preservation, homomorphism). Role-like g-SIS has only liberal add and join operations and strict leave and remove operations, and thus its *auth* only depends on the current group members and associated documents. In this scenario, $RBAC_0$ can simply simulate groups using roles.

Theorem 1 $rgSIS \leq^{CaH} RBAC_0$.

Furthermore, $RBAC_1$ can trivially simulate $RBAC_0$ by ignoring the role hierarchy, yielding the following.

Lemma 2 $RBAC_0 \leq^{CAH} RBAC_1$.

Corollary 3 $rgSIS \leq^{CaH} RBAC_1$.

We are able to construct a reduction from top g-SIS ($tgSIS$) to $RBAC_1$ by identifying the pseudo-hierarchical structure of the authorization set in $tgSIS$: since all operations are strict, new members of a group will have a *subset* of the permissions of older members. The hierarchy is invoked by the fact that older members thus "inherit" access to all added objects, while new members only receive access to objects added after they joined. We simulate this structure in $RBAC_1$'s role hierarchy by creating a chain in RH for each group. When a g-SIS group g is created, the top of the chain, a role named g, is created in $RBAC_1$. Objects newly added to the group should be available to all users, and thus the corresponding permission in $RBAC_1$ is added to the bottom of the chain, ensuring all users in the chain will be authorized. Finally, when a new user joins group g, they create a new "view" of the g, since they are not authorized to any existing objects (due to strict join). Thus, we create in $RBAC_1$ a new role (named randomly) and link it to the bottom of g's chain.

Theorem 4 $tgSIS \leq^{CaH} RBAC_1$.

We provide a proof sketch for this theorem in Appendix A.

We build a reduction from bottom g-SIS ($bgSIS$) to $RBAC_1$ using a similar hierarchy-chain solution to $tgSIS$. Since $bgSIS$ contains all liberal actions, we still have a pseudo-hierarchy of $Auth(bgSIS)$. In this case, users who leave a group maintain access to objects from this group, so users who leave earlier have a subset of the authorizations of users who leave later (or are still members). An exception is made for removed objects, since these are not granted to new users after removal. Thus, we again create a hierarchy chain for each group. In this case, the chain grows upward. When a user is removed, a new role is created at the top of the chain and all users remaining in the group are added to this new role. Objects are added to this top role, granting access to all current members. Removed objects, being the exception to the hierarchy rule, are added to orphaned roles, along with all users who should maintain access to them.

Theorem 5 $bgSIS \leq^{CaH} RBAC_1$.

We utilize a non-homomorphic helper reduction from $RBAC_1$ to $RBAC_0$ (and expressiveness transitivity) to prove reductions from $tgSIS$ and $bgSIS$ to $RBAC_0$. This reduction, in order to store hierarchical accesses in a "flat" roleset, expands the set of roles to include a role named for every path through the hierarchy in the downward direction. Thus, if $RBAC_1$'s hierarchy says $A \geq B$, $B \geq C$, and $A \geq D$, then this is represented in $RBAC_0$ with roles $\{A, B, C, D, AB, ABC, AD, BC\}$. For every role r a user is assigned to in $RBAC_1$, she will be assigned to each role starting with r in $RBAC_0$. In the previous example, if $\langle u, A\rangle \in UR$ in $RBAC_1$, then in $RBAC_0$ this maps to $\{\langle u, A\rangle, \langle u, AB\rangle, \langle u, ABC\rangle, \langle u, AD\rangle\} \subset UR$ in $RBAC_0$.

Theorem 6 $RBAC_1 \leq^{CA} RBAC_0$.

Corollary 7 $tgSIS \leq^{Ca} RBAC_0$; $bgSIS \leq^{Ca} RBAC_0$.

We were also able to construct a *homomorphic* helper reduction from $RBAC_1$ to $RBAC_0$. This reduction makes use of an encoding technique which stores the information in the three binary relations (UR, PA, RH) of $RBAC_1$ in the two binary relations (UR, PA) of $RBAC_0$. Using this encoding, each tuple in $RBAC_1$ is stored using three to four tuples in $RBAC_0$. Thus, the resulting state encodes the required information in an unnatural, convoluted scheme which requires deeply nested looping to decode. For example, the originally straightforward authorization procedure of finding an r such that $\langle u, r\rangle \in UR$ and $\langle r, p\rangle \in PA$ must be carried out in this reduction by searching for a set of values r, v, x, y, z such that $\{\langle v, x\rangle, \langle u, v\rangle, \langle r, x\rangle\} \subseteq UR$ and $\{\langle y, z\rangle, \langle y, r\rangle, \langle z, p\rangle\} \subseteq PA$. These vast, compounding inefficiencies prevent this reduction from having any practical application. We conjecture that it is impossible to construct an asymptotically more efficient implementation than using this helper reduction while satisfying the given guarantees, which restricts us to storing the required workload state using only tuples in two relations, and using only existing constants (new constants must be information-less). We discuss this reduction further in Appendix B.

Finally, although *ugo* has the inherent disadvantage that each object is owned by only a *single* user and group, we can show $RBAC_0 \leq^{Ca}$ *ugo* since *ugo* can simulate $RBAC_0$ implementations by mapping a permission assigned to multiple roles to an object with a single group owner, which represents

all roles with authorization and includes as members all users in the $RBAC_0$ roles. Though the implementation is weakly AC-preserving, it is not homomorphic since it requires the manipulation of strings for group names.

Theorem 8 *$RBAC_0 \leq^{Ca}$ ugo.*

Corollary 9 *$rgSIS \leq^{Ca}$ ugo; $tgSIS \leq^{Ca}$ ugo; $bgSIS \leq^{Ca}$ ugo.*

4.4 Expressiveness via Implementations

We now present a summary of the implementations of group-centric workloads in dissemination-centric systems. A summary of these implementations and their corresponding strengths is shown in Figure 3b. As in the previous section, we now describe the major results of these implementations.

The PC workload uses liberal join for users joining a program committee group and strict leave for resignation (permanent leave). Liberal leave is used for conflicts-of-interest (temporary leave), and strict join is used to re-join after a COI. We implement this workload in $RBAC_1$ using techniques from the reductions of both $tgSIS$ and $bgSIS$ in $RBAC_1$. Like in our reduction from $tgSIS$, each group uses a hierarchy chain building downward, adding a node (and thus a new "view" of the group) each time a user executes a strict join and assigning newly added objects to the bottom role of the chain. Like in the reduction from $bgSIS$, we use the orphan role concept, in this case for users who liberal leave; the departing user and permissions she should continue to be authorized to are added to a new role. We implement u strict leaving g by removing u from all roles connected to g, and u liberal joining g by assigning u directly to g (so she inherits permission to all current objects). This implementation is correct, weakly AC-preserving, homomorphic, and safe.

Theorem 10 *There exists a correct, weakly AC-preserving, homomorphic, and safe implementation of PC in $RBAC_1$.*

The PSP workload supports liberal (restorative) join, strict leave, liberal add, and both strict and liberal remove. Rather than use a role hierarchy chain, this reduction uses a single role for each group that is assigned to all current members and objects in the group. Two types of orphans are used, one for objects that are liberally removed (along with the users who should remain authorized to the object), and one for strict leave, to support the restorative join operation. On a strict leave of u from g, we create in $RBAC_1$ an orphan role pair r, s, where $s \geq r$, and assign u to r and all of u's permissions from g to s. Since u is in a role junior to the permissions, she is no longer authorized to them. On a re-join to the group, we simply assign u to s, re-enabling u's access to these permissions. This implementation is correct, weakly AC-preserving, homomorphic, and safe.

Theorem 11 *There exists a correct, weakly AC-preserving, homomorphic, and safe implementation of PSP in $RBAC_1$.*

We use helper reductions from the previous section to establish correct, weakly AC-preserving implementations of PC and PSP in $RBAC_0$ and *ugo*. We independently prove these implementations are safe, since there is no known meta-theorem for using reductions to prove safety.

Corollary 12 *There exist correct, weakly AC-preserving, and safe implementations of PC & PSP in $RBAC_0$ & ugo.*

4.5 Summary of Results

Observing the results of Figures 3a and 3b, it is clear that dissemination-centric systems are able to meet basic security guarantees when operating within group-centric scenarios. $RBAC_1$ is the most successful, simulating the extrema systems and workloads with all security guarantees. $RBAC_0$ was able to implement $rgSIS$ with strong guarantees, but for other g-SIS parameterizations (those with multiple "views" of a single group), $RBAC_0$ had to sacrifice homomorphism to admit feasible implementations. Finally, *ugo* was also able to satisfy all workloads and systems, but (due to each object being associated with only a single group) did not admit any homomorphic implementations.

We note that we also considered the π-system, a g-SIS system defined over the g-SIS_0 model with support for all action varieties [11], but were unable to construct a reduction from π-system to (or implementation of $\langle \pi gSIS, \mathcal{T} \rangle$ in) any dissemination-centric system that was AC-preserving. Thus, although there was some success among dissemination-centric systems in implementing *specific* parameterizations of group-centric workloads, these systems do not admit as readily implementations of the *fully expressive* form of g-SIS without the sacrifice of basic security guarantees.

5. COST ANALYSIS

Now that we have a clear picture of each dissemination-centric system's expressiveness with respect to group-centric scenarios (which, recall, reflects their theoretical capability), we investigate a second dimension of these systems' suitability to this set of workloads: efficiency and costs. To consider all of the candidate systems in practical contexts, we evaluate correct, weak AC-preservation implementations, disregarding the homomorphic requirement, which some systems can not always satisfy feasibly (see Section 4.3). We conduct cost analysis via Monte Carlo simulation driven by the structures built during expressiveness analysis.

5.1 Trace Generation

Recall from Section 2.2 that in a workload $\mathcal{W} = \langle \mathcal{A}, \mathcal{T} \rangle$, the set $\mathcal{T}$ describes the permissible traces through the ideal access control system $\mathcal{A}$. In cost analysis, we generate random traces from $\mathcal{T}$, execute these traces, and record various costs accrued during execution. Of course, just as it is infeasible to explicitly enumerate the set of all permissible traces, it is typically difficult to sample meaningful traces uniformly at random from this set. Thus, for the purpose of simulation, we specify a stochastic parameterization of this set of traces. In particular, we articulate distributions from which the components of the workload's initial state (e.g., number of users, number of objects, etc.) are drawn, and specify probabilistic models for the type and frequency of actions taken by active entities within the system.

During a simulation run, we first generate an initial state by sampling from the appropriate distributions. We then inspect this state to determine the set of *actors* that will execute labels on that state. Actors can be human users, daemons, or other entities that act on the access control system. We construct state machines that describe the order in which individual actors will execute labels and queries[1], and build

[1]Since queries can not alter the state, they are irrelevant to traces as they are used in expressiveness analysis. However, in cost analysis, we include both labels and queries in traces.

constrained workflows that describe actor cooperation. We execute all of the actors' state machines in parallel, with the constraints placed by the workflows and past actions, to generate traces of actions for each actor. The individual actor traces are then interleaved to produce global traces.

Once these traces are generated in terms of workload labels and queries, we translate them into traces of system actions for each of the implementing systems. This is made simple thanks to the expressiveness analysis described in Section 4. When simulating the implementation $\langle \alpha, \sigma, \pi \rangle$, we map initial workload states to system states with σ and workload labels to system labels with α. Finally, π acts as a set of procedures for answering queries using the mapped system state.

We generate initial states and traces to analyze each of our group-centric scenarios as follows.

Program Committee To simulate the PC workload, we select an initial state with 25–75 users. Traces are generated in three phases. First, PC groups are created. Next, PC members join groups. Finally, discussion occurs, and users post objects and execute conflict-of-interest workflows. Traces simulate an eight month cycle, overall.

Playstation Plus Initial states in PSP have 20–100 users and 2–5 regions (subscription groups), with 50–400 objects distributed between them, each representing a current promotion (free game or discount). Traces model users changing membership and administrators adding and removing objects to the regions. Each trace models a period of one year.

Extrema Systems To carry out cost analysis of top, bottom, and role-like g-SIS, we must define usage models from scratch, since these systems are not part of workloads. We generate initial states with 25–85 users, and a number of managers between 5 and 1/4 the number of regular users. In traces, managers create groups and sometimes delete posts (e.g., those that violate terms of service). Normal users join groups and share objects, both newly-created and existing (re-shares). Traces model three days to one week of heavy activity, with the average user posting multiple times per day and joining a new group every two days, on average.

5.2 Cost Measures

While the type of expressiveness analysis carried out by an analyst is defined by a set of security guarantees that must be upheld, the type of cost analysis is parameterized by the costs to be examined. There are numerous forms of cost measures, from the storage needed to maintain state, to the administrative overhead of executing labels, to the computational cost of evaluating queries. Some are named in a recent NIST report [10], which points out the need for a variety of "evaluation metrics" since no one measure answers what is necessary across all applications.

We investigate costs representing storage requirements (maximum state size during a run, number of roles); amount of data read/written (average I/O per label, proportion of state changed per label); degree to which atomicity of label execution is violated (number of stutter steps); and other application-specific measures of "misuse" of the implementing systems (average number of permission-assignments per role). To investigate the values of these measures, we plot them against properties of the trace (e.g., number of users, maximum number of objects) and against the workload's own performance within the scenario (e.g., workload I/O, maximum workload state size) for comparison purposes.

5.3 Selected Results

We carried out a comprehensive cost analysis of the implementations described in Section 4 using a purpose-built Monte Carlo simulator that our team developed. Our cost analysis uncovered a variety of clear drawbacks to implementing group-centric workloads with dissemination-centric systems. We present several demonstrative examples in Figure 4, but note that the remaining results do not inspire us to draw conclusions that are substantially different from those presented here. Note that each subfigure reports on the result of 200 runs of the workload being simulated.

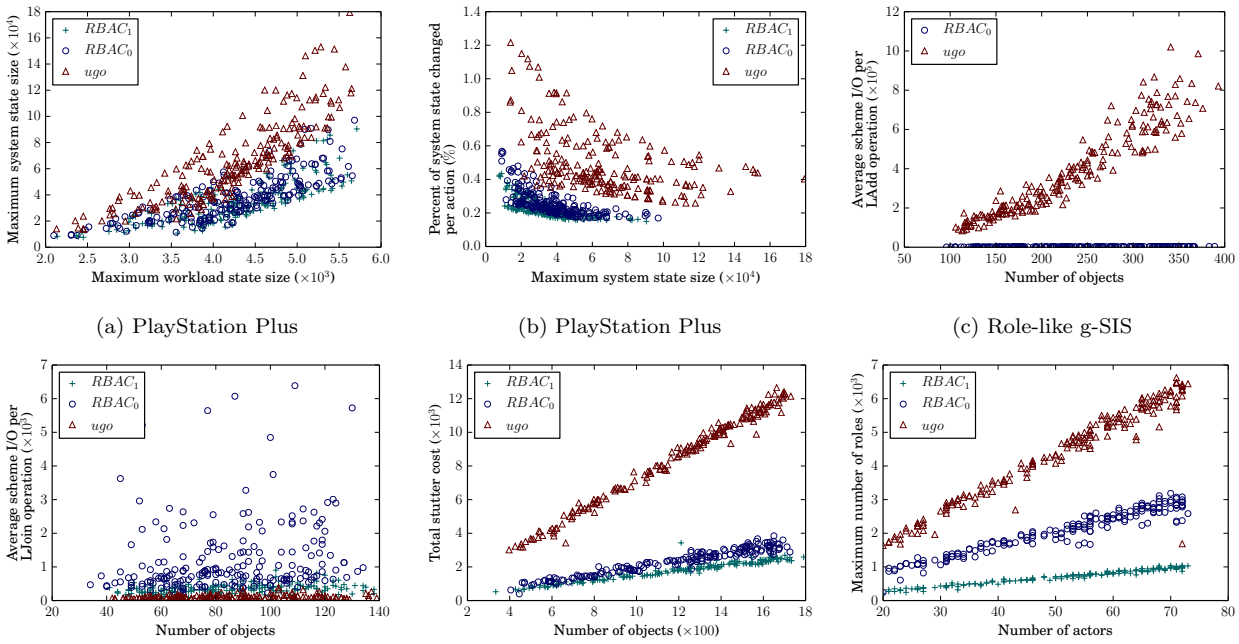
Storage measures. Figure 4a shows that, in the PSP workload, the amount of storage required in each implementing system is superlinear in the size of the workload. This is mostly caused by the blow-up in number of roles/groups required to safely implement the group-centric workloads in systems without built-in temporal abilities. Although PSP is particularly inefficient to implement in dissemination-centric systems, only $rgSIS$ can be implemented using state size comparable to the original workload size even in *ugo*.

Figure 4b shows another aspect of storage, the proportion of the state that is changed on average per simulated (workload) action, again when implementing PSP. This figure shows that our implementation in *ugo* is particularly inefficient. This is largely due to the cached authorization table that must be maintained in *ugo*, making this system a poor choice in scenarios where writes are costly. This pattern is seen across all workloads, and implementations with lower state size generally have the highest proportion of state changing (up to 10% per action), indicating that even those with (relatively) low storage requirements are re-writing large amounts of data to simulate each action.

It is clear that, to support large numbers of users in groups with high object flux, not even hierarchical roles are an efficient replacement for time-aware groups.

I/O measures. Figure 4c shows the I/O cost (in number of state elements accessed) for simulating liberal add operations in $rgSIS$. Although $RBAC_0$ is able to simulate role-like g-SIS fairly naturally, *ugo* lacks the ability to natively grant multiple groups access to an object. This missing capability is necessary in group-centric workloads, and thus we must simulate it in *ugo* by assigning each object to a single group and assigning users to these special, semantics-less groups as needed. The extra overhead of iterating over the the Member relation to extract the information from $RBAC_0$'s UR and PA and rebuild the cached authorization table is evident in Figure 4c from the superlinear increase in I/O needed to add objects to groups as the number of total objects in the system increases. By comparison to $RBAC_0$'s simple lockstep implementation, *ugo*'s is much more inefficient. The consumer-grade *ugo* system is not practical within even the most simple group-centric workloads.

Figure 4d demonstrates that high I/O is not restricted to implementations in *ugo*. This figure shows I/O for (liberal) joining groups in $bgSIS$. Recall that this is the operation that triggers the role hierarchy chain to expand in $RBAC_1$'s simulation of $bgSIS$, and thus as expected demands high I/O. Specifically, $RBAC_1$ I/O per join is about 1/4 the total I/O of all commands executed in $bgSIS$ in an average full simulation, and $RBAC_0$ regularly exceeds the workload's full I/O. Although $bgSIS$ in particular has expensive implementations of liberal join, each g-SIS workload (except $rgSIS$ which is



(a) PlayStation Plus (b) PlayStation Plus (c) Role-like g-SIS

(d) Bottom g-SIS (e) Program Committee (f) Program Committee

Figure 4: Group-centric cost analysis results

efficiently implemented in $RBAC_0$) has at least one action which causes this characteristically high I/O.

We study the amount of "stuttering" per trace in Figure 4e, which is a description of the number of extra operations that must be executed in implementing systems to simulate single workload actions. We see a drastic increase in stuttering while implementing PC, (especially in *ugo*) as more objects are added to the system. This quantifies the loss of atomicity of operations, and allows us to understand the increasing frequency with which the data structures must be locked to guarantee the desired security properties.

Role abuse. Finally, we measure the maximum number of roles created to accommodate the PC workload. It has been said that role-based systems lose their administrative value when the number of roles exceeds the number of users [23]. In Figure 4f, we compare the number of subjects in the PC workload to the number of roles (groups for *ugo*) in the corresponding state of the implementing systems. Roles vastly outnumber users, starting with $RBAC_1$'s role hierarchy chain for each group, and getting worse in both $RBAC_0$ and *ugo*, each requiring more roles than the previous in order to guarantee weak AC-preservation.

Summary. We have shown a number of measures for which dissemination-centric systems $RBAC_1$, $RBAC_0$, and *ugo* prove to be very inefficient in implementing group-centric workloads with strong security guarantees. Even the most space-efficient implementations use much more state storage than an equivalent g-SIS parameterization, and require much more I/O to operate. The number of roles created is often many times the number of users and several times the number of objects. With the exception of implementing role-like g-SIS in RBAC, all implementations also cause large amounts of stuttering, or non-atomic sequences of labels to simulate a single workload action. Thus, it seems that in most practical scenarios one must heavily compromise security guarantees or suffer vastly inefficient implementations in order to utilize dissemination-centric systems in the group-centric context.

6. DISCUSSION AND FUTURE WORK

6.1 Dissemination-centric vs. Group-centric

We set out to evaluate the hypothesis stated by the creators of g-SIS [12]: that the group-centric class of models is equal in collective expressiveness to the dissemination-centric class, but that they are pragmatically different approaches and thus should complement, rather than substitute for, one another. We found in our experiments that these approaches do indeed yield pragmatically dissimilar systems, and that even on a theoretical expressiveness level may not be equivalent.

First, in expressiveness analysis, we displayed the reduction from $rgSIS$ to $RBAC_0$, the simplest role-based system. It was not surprising that the reduction achieved strong security guarantees ($rgSIS$ was, after all, modeled after role-based systems). However, when we noticed that $tgSIS$ (and, to a lesser extent, $bgSIS$) admitted a hierarchical set of authorizations within each group, and that this allowed the hierarchical $RBAC_1$ to implement it just as strongly, we realized that $rgSIS$ is not unique—other parameterizations of g-SIS could be safely implemented using dissemination-centric systems.

This pair of strong implementations shows that in some cases a g-SIS parameterization and a dissemination-centric system can provide the same theoretical capabilities, in part because of structural similarities between how the group- and dissemination-centric counterparts manage internal state. There does not seem to be anything special about these pairs that leads us to believe they are unique. However, in other cases we find the dissemination-centric system unable to

fully match the g-SIS system, e.g. $RBAC_0$ and $tgSIS$. Thus, although we cannot count out the possibility that there is *some* traditional system with the same capabilities as a given g-SIS system, this is not a claim we can confirm based on our investigation of several commonly-used traditional systems and several natural g-SIS parameterizations.

To address the pragmatic differences between dissemination- and group-centric sharing, we carried out cost analyses of the implementations that we developed. Implementations that were bad fits in expressiveness analysis provided continuing evidence of their poor fit in cost analysis. State storage was much higher in these systems than in the ideal systems of the workloads. Executing workload actions often necessitated many stuttering steps in the implementing system, required higher I/O within the implementing system, and changed a high proportion of state for each action. However, these poorly-matched implementations were not alone—even the strongly secure, relatively simple implementation of $tgSIS$ in $RBAC_1$ had inefficiencies that became evident during cost analysis. Though $RBAC_0$ could not feasibly satisfy the homomorphic guarantee when implementing $tgSIS$ due to lacking a hierarchy, $RBAC_1$ required as much of a state space explosion as $RBAC_0$. Even though it had much lower I/O cost than $RBAC_0$, $RBAC_1$ required orders of magnitude greater I/O than in $tgSIS$ to execute its procedure for simulating a strict join.

Thus, we believe we have validated the second point in the hypothesis. Although certain dissemination-centric systems are *able* to implement group-centric workloads, it does not mean they *should*—even when they are theoretically capable, they are not necessarily pragmatically suitable.

6.2 Beyond Expressiveness

Although expressive power analysis has long been the measuring stick for understanding and ranking access control systems in the literature (e.g., [1, 3, 5, 7, 9, 13, 14, 17, 19, 20, 22]), the analysis conducted in this paper indicates that expressiveness alone does not always tell the whole story. For instance, recall that $RBAC_1$ was able to implement many interesting g-SIS workloads while maintaining strong security guarantees. However, the complexity required for these implementations to maintain this set of properties resulted in loss of atomicity when executing certain actions, increased state size and state management overheads, and (ultimately) a loss of the elegance of the original workload. From a theoretical perspective, $RBAC_1$ was expressive enough to encode a variety of group-centric workloads; from a practical perspective, these implementations are less than ideal.

We believe that this work represents the first comprehensive analysis of expressiveness and cost within the context of access control systems. Further, the results obtained by this analysis are significant in that they provide a concrete data point indicating the potential dangers of relying too heavily on any one measure of access control suitability when examining the needs of an application. It would be worthwhile to develop a generalized framework for carrying out the types of analysis conducted in this paper while supporting a wide variety of expressiveness and cost metrics.

6.3 Towards an Expressiveness Taxonomy

One goal of parameterized expressiveness [9] is to allow one to choose the notion of expressiveness that best matches the workload in question. Although it is often easy to decide whether to require a particular PE security guarantee, PE has not yet enabled the community to break down existing notions of expressiveness into their component properties. For example, it is not known whether there is any combination of parameterized expressiveness properties that yields expressiveness statements equivalent to those made by, e.g., the state matching reduction [22]. For this reason, we identify as another area of future work the continued investigation of PE techniques and guarantees, hoping to gain knowledge of both the properties of and relationships between expressiveness notions as well as deeper, more fundamental aspects of access control and state machine simulations.

7. OTHER RELATED WORK

The need for more general techniques for evaluating access control systems was discussed in a recent NIST report, which states that "when it comes to access control mechanisms, one size does not fit all" [10]. The report bemoans the lack of established quality metrics for access control systems and lists numerous possibilities. Several of these metrics relate to state size, number of actions committed, and other quantitative measures. The report stops short of explaining how one might choose between these metrics, or how to effectively evaluate systems with respect to these metrics. In this work, we perform what we believe is the first comprehensive access control evaluation using both expressiveness and quantitative measures, and thus make a first step toward realizing the evaluation mechanisms this NIST report hopes for.

For inspiration in generating access control traces (see Section 5.1), we turn to trace generation work in other domains. In the field of disk benchmarking, Ganger [6] observed that interleaved workloads provided the most accurate approximation of recorded traces. Thus, mechanisms for representing access control workloads must be capable of simulating the interleaved actions of multiple actors. This view is reinforced by the design of IBM's SWORD workload generator for stream processing systems [2, 4]. This work also points out that synthetic workloads need to replicate both volumetric and contextual properties of an execution environment in order to provide an accurate indication of a system's performance within that environment. Thus, we conjecture that access control workloads may also benefit from expressing not only volumetric statistics such as number of documents created, but also contextual statistics such as the type of content in created documents.

8. CONCLUSION

In this work, we examined the capabilities of popular dissemination-centric access control systems to operate within group-centric workloads. We formalized several group-centric workloads as instantiations of g-SIS, a family of information sharing models that has been formalized in temporal logic but not yet implemented. We then conducted a two-phase analysis that we believe to be the first of its kind. We first evaluated whether the dissemination-centric systems are expressive enough to implement the group-centric workloads, assessing the strength of these implementations by examining the security guarantees they can preserve. We then conducted a cost analysis, investigating more pragmatic metrics that provide insight into the efficiency of these systems when implementing group-centric workloads.

We found that while RBAC with role hierarchy was able to implement the workloads that we considered with strong security guarantees, a more basic variant of RBAC without role hierarchies could only implement one of our workloads without compromising the guarantees to be upheld. Further, we found that standard UNIX-style user-group-other permissions could not implement any of our group-centric workloads while upholding all required security guarantees. In cost analysis, we found that, with limited exceptions, even those implementations upholding strong security properties suffered from inefficiencies in state size, I/O, and atomicity of operations. These results indicate that g-SIS is a practically significant proposal that elegantly satisfies a class of workloads that existing access control techniques struggle with. More fundamentally, these results demonstrate the need for access control evaluation techniques and frameworks that allow not only theoretical expressiveness analysis but also the more pragmatic and quantitative cost analysis.

Acknowledgements. This work was supported in part by the National Science Foundation under awards CNS–0964295 and CNS–1228697. The authors also thank Timothy L. Hinrichs for insightful discussion during the development of our analysis techniques.

9. REFERENCES

[1] Paul Ammann, Richard J. Lipton, and Ravi S. Sandhu. The expressive power of multi-parent creation in monotonic access control models. *JCS*, 4(2/3), 1996.

[2] Kay S. Anderson, Joseph P. Bigus, Eric Bouillet, Parijat Dube, Nagui Halim, Zhen Liu, and Dimitrios E. Pendarakis. Sword: scalable and flexible workload generator for distributed data processing systems. In *Winter Simulation Conference*, 2006.

[3] Elisa Bertino, Barbara Catania, Elena Ferrari, and Paolo Perlasca. A logical framework for reasoning about access control models. *TISSEC*, 6(1), 2003.

[4] Eric Bouillet, Parijat Dube, David George, Zhen Liu, Dimitrios E. Pendarakis, and Li Zhang. Distributed multi-layered workload synthesis for testing stream processing systems. In *Winter Simulation Conference*, 2008.

[5] Ajay Chander, Drew Dean, and John C. Mitchell. A state-transition model of trust management and access control. In *CSFW*, 2001.

[6] Gregory R. Ganger. Generating representative synthetic workloads: An unsolved problem. In *International CMG Conference*, 1995.

[7] Srinivas Ganta. *Expressive Power of Access Control Models Based on Propagation of Rights*. PhD thesis, George Mason University, 1996.

[8] William C. Garrison III, Yechen Qiao, and Adam J. Lee. On the suitability of dissemination-centric access control systems for group-centric sharing: Full proofs. `http://www.cs.pitt.edu/~adamlee/pubs/2014/garrison2014proofs.pdf`, 2013.

[9] Timothy L. Hinrichs, Diego Martinoia, William C. Garrison III, Adam J. Lee, Alessandro Panebianco, and Lenore Zuck. Application-sensitive access control evaluation using parameterized expressiveness. In *CSF*, 2013.

[10] Vincent C. Hu, David F. Ferraiolo, and D. Rick Kuhn. *Assessment of Access Control Systems*. NIST, 2006.

[11] Ram Krishnan, Jianwei Niu, Ravi S. Sandhu, and William H. Winsborough. Group-centric secure information-sharing models for isolated groups. *TISSEC*, 14(3), 2011.

[12] Ram Krishnan, Ravi Sandhu, Jianwei Niu, and William H. Winsborough. A conceptual framework for group-centric secure information sharing. In *ASIACCS*, 2009.

[13] Qamar Munawer and Ravi S. Sandhu. Simulation of the augmented typed access matrix model (atam) using roles. In *INFOSECU99*, 1999.

[14] Sylvia L. Osborn, Ravi S. Sandhu, and Qamar Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *TISSEC*, 3(2), 2000.

[15] Playstation plus. `http://us.playstation.com/psn/playstation-plus`.

[16] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), 1975.

[17] Ravi S. Sandhu. Expressive power of the schematic protection model. *Journal of Computer Security*, 1(1), 1992.

[18] Ravi S. Sandhu. Rationale for the RBAC96 family of access control models. In *ACM Workshop on Role-Based Access Control*, 1995.

[19] Ravi S. Sandhu and Srinivas Ganta. On testing for absence of rights in access control models. In *CSFW*, 1993.

[20] Ravi S. Sandhu and Qamar Munawer. How to do discretionary access control using roles. In *ACM Workshop on Role-Based Access Control*, 1998.

[21] Andrew Sciberras. Lightweight directory access protocol (LDAP): Schema for user applications. Technical Report RFC 4519, eB2Bcom, 2006. `http://www.rfc-editor.org/rfc/rfc4519.txt`.

[22] Mahesh V. Tripunitara and Ninghui Li. A theory for comparing the expressive power of access control models. *JCS*, 15(2), 2007.

[23] Dana Zhang, Kotagiri Ramamohanarao, Steven Versteeg, and Rui Zhang. RoleVAT: Visual assessment of practical need for role based access control. In *ACSAC*, 2009.

APPENDIX

A. EXAMPLE PROOF SKETCH

Here, we present an example proof sketch to demonstrate our techniques. Full definitions of systems, implementations, reductions, and proofs are deferred to a companion technical report [8].

First, we present $RBAC_1$, based on the system of the same name presented in [18]. States in $RBAC_1$ are comprised of the following.

- U, R, and P, the sets of users, roles, and permissions
- $UR \subseteq U \times R$, the user-role relation
- $PA \subseteq R \times P$, the role-permission relation
- $RH \subseteq R \times R$, a partially ordered role hierarchy

Requests are of the form u, p for whether user u has access to permission p. Queries include querying the

relations $UR(u,r)$, $PA(r,p)$, and $RH(r_1,r_2)$, the transitive hierarchy relation $Senior(r_1,r_2) \triangleq RH(r_1,r_2) \vee \exists r_3.(Senior(r_1,r_3) \wedge Senior(r_3,r_2))$, and the authorizations $auth(u,p) \triangleq \exists r_1,r_2.(UR(u,r_1) \wedge PA(r_2,p) \wedge (r_1 = r_2 \vee Senior(r_1,r_2)))$.

Labels are included for managing the various data structures: $addU(u)$, $delU(u)$, $addR(r)$, $delR(r)$, $addP(p)$, $delP(p)$, $assignUser(u,r)$, $revokeUser(u,r)$, $assignPermission(r,p)$, $revokePermission(r,p)$, $addHierarchy(r_1,r_2)$, and $removeHierarchy(r_1,r_2)$.

Top g-SIS ($tgSIS$) uses the g-SIS$_0$ model, but utilizes only strict versions of state elements. It includes the following labels: $addS(s)$, $delS(s)$, $addG(g)$, $delG(g)$, $addO(o)$, $delO(o)$, $strictJoin(s,g)$, $strictLeave(s,g)$, $strictAdd(o,g)$, and $strictRemove(o,g)$.

Since $tgSIS$ contains only strict actions, we can utilize a simplified $auth$ definition:

$$
\begin{aligned}
auth(s,o,g) \triangleq \exists t_1,t_2.(&\\
&StrictJoin(s,g,t_1) \wedge\\
&StrictAdd(o,g,t_2) \wedge\\
&t_2 > t_1 \wedge\\
&\forall t_3.(\\
&\quad StrictLeave(s,g,t_3) \Rightarrow t_1 > t_3 \wedge\\
&\quad StrictRemove(o,g,t_3) \Rightarrow t_2 > t_3\\
&)\\
)&
\end{aligned}
$$

Theorem 13 *There exists a reduction from top g-SIS ($tgSIS$) to $RBAC_1$ where:*

- *σ preserves π, is pseudo-injective, preserves reachability, and is homomorphic*
- *π is homomorphic and weakly AC-preserving*

Thus, $tgSIS \leq^{CaH} RBAC_1$ ($RBAC_1$ is at least as expressive as top g-SIS with respect to correctness, weak AC-preservation and homomorphism).

PROOF (SKETCH) We prove the theorem by construction—we present the reduction $\langle \sigma, \pi \rangle$, and prove it satisfies each of the properties. The state mapping, σ, stores the $tgSIS$ state in $RBAC_1$ as follows. Subjects are stored as users, each group is stored as a role, and objects are stored as permissions. We process the records (strict variety of join, leave, add, and remove) in time order, and execute the mapped action for each one. Joins require creation of a new role added at the bottom of the group's role hierarchy chain (initially, directly below the group). Leaves require iterating over the group's role hierarchy chain, removing the user from each role. To add a document to a group, the corresponding permission is assigned to the bottom role in the group's role chain. To remove an object, similar to a user leaving, iterate over all roles below the group's role and remove the object's permission from each. We define this state mapping in HPL, a minimal programming language that can only implement homomorphic mappings [9].

The query mapping, π, is defined as follows.

$$
\begin{aligned}
\pi_{Member(s,g)}(T) &= \exists r.(UR(s,r) \in T \wedge Senior(g,r) \in T)\\
\pi_{Assoc(o,g)}(T) &= \exists r.(PA(r,o) \in T \wedge Senior(g,r) \in T)\\
\pi_{auth(s,o,g)}(T) &= \exists r_1,r_2.(UR(s,r_1) \in T \wedge PA(r_2,o) \in T \wedge\\
&\qquad (r_1 = r_2 \vee Senior(r_1,r_2) \in T) \wedge\\
&\qquad Senior(g,r_1) \in T)
\end{aligned}
$$

This query mapping clearly contains no string manipulation, and is thus homomorphic.

We show that σ preserves π (for all $tgSIS$ states x, $Th(x) = \pi(Th(\sigma(x)))$) by contradiction. We assume that there is some $tgSIS$ state x and query q such that the value of q in x is the opposite of the value of $\pi(q)$ in $\sigma(x)$. We then show that, for each of the query forms of $tgSIS$, this assumption leads to contradiction, and thus that σ preserves π.

For all $tgSIS$ states x, x', if x' is reachable from x, then there exists a sequence of labels $\langle \ell_1, \ell_2, \ldots, \ell_n \rangle$ such that $terminal(x, \ell_1 \circ \ell_2 \circ \cdots \circ \ell_n) = x'$. We prove that σ preserves reachability by showing that, for any $tgSIS$ state x and label ℓ, $\sigma(next(x,\ell))$ is reachable from $\sigma(x)$ via $RBAC_1$ labels. By induction, this shows that for each intermediate $tgSIS$ state x_i between x and x', $\sigma(x_i)$ is reachable from $\sigma(x)$ and ultimately that $\sigma(x')$ is reachable from $\sigma(x)$. These actions are mapped in the same way as their corresponding records in the state mapping.

Finally, we show that σ is *pseudo-injective*, a property which allows us to show that the reduction preserves correctness. We do so by inspecting the state mapping, σ, and arguing that any two states in $tgSIS$ that map to the same $RBAC_1$ state can be treated identically by an implementation's label mapping—that is, we lose no meaningful information by mapping a $tgSIS$ state into $RBAC_1$. □

B. INFEASIBLE REDUCTION

Here, we describe the infeasible reduction from $RBAC_1$ to $RBAC_0$ mentioned in Section 4.3. The full reduction and proof are provided in the companion technical report [8].

In this reduction, we must store UR_1, PA_1, and RH_1 from $RBAC_1$ in only UR_0 and PA_0 in $RBAC_0$. We accomplish this using the following homomorphic encoding. For each $\langle u, r \rangle \in UR_1$, we generate two new constants a and b and store in UR_0 each of $\{\langle a,b \rangle, \langle u,a \rangle, \langle r,b \rangle\}$. For each $\langle r,p \rangle \in PA_1$, we generate two new constants c and d and store in PA_0 each of $\{\langle c,d \rangle, \langle c,r \rangle, \langle d,p \rangle\}$. Lastly, for each $\langle s,j \rangle \in RH_1$, we generate three new constants e, f, and g and store in PA_0 each of $\{\langle e,f \rangle, \langle f,g \rangle, \langle e,s \rangle, \langle g,j \rangle\}$.

Under this (partial) encoding, the second element of each tuple in UR_0 and the first element of each tuple in PA_0 are generated (information-less) constants. Since constants are generated to avoid collisions, there is no join over UR_0 and PA_0, which would violate AC-preservation. Finally, we add to the encoding the set of authorized requests, to fully satisfy AC-preservation. For each request $\langle u,p \rangle$ which is authorized (i.e., for each $\langle u,p \rangle$ such that $\exists s,j : \langle u,s \rangle \in UR_1 \wedge \langle j, p \rangle \in PA_1 \wedge \langle s,j \rangle \in RH_1$), we generate a new constant h and store $\langle u,h \rangle$ in UR_0 and $\langle h,p \rangle$ in PA_0.

The reduction answers queries (besides authorization requests) by extracting the relevent parts of UR_1, PA_1, and RH_1. Generated constants are identified by their positions in tuples. UR_1 tuples can be extracted from UR_0 by finding sets of three tuples which match the $\langle a,b \rangle$, $\langle u,a \rangle$, $\langle r, b \rangle$ pattern. Tuples from PA_1 and RH_1 can be extracted from PA_0 similarly.

Finally, the reduction must update the encoding after each command. For example, if user u is assigned role r, $\langle u, r \rangle$ is encoded and stored in UR_0, then each permission p_i which u gains must be determined and encoded in UR_0 and PA_0 to satisfy AC-preservation.