Application-Sensitive Access Control Evaluation: Suitability and Cost Analysis
(Extended Version)

(Blinded for review)

***Abstract*—To date, most work regarding the formal analysis of access control schemes has focused on quantifying and comparing the expressive power of a set of schemes. Although expressive power is important, it is a property that exists in an *absolute* sense, detached from the application-specific context within which an access control scheme will ultimately be deployed. In this paper, we depart from the practice of absolute evaluation and introduce a mathematical framework aimed at assessing the suitability of existing access control schemes for a *specific* application. In our framework, an analyst first formally defines a workload representing the operational demands of the application and specifies metrics representing the pertinent costs. Reductions demonstrate how each candidate scheme—or some augmentation of it—can be used to safely implement the workload. Our framework assigns each such implementation a score based on the chosen cost metric and deems that the access control scheme with the highest scoring implementation is the one best suited for the application. We demonstrate our framework by presenting an example workload based on the requirements of a modern military, and exploring the suitability of a series of popular access control schemes for this workload.**

I. Introduction

Access control is one of the most fundamental aspects of computer security, and has been the subject of much formal study. However, existing work on the formal analysis of access control schemes has focused largely on comparing the *relative expressive power* of two or more access control schemes (e.g., [1], [2], [3], [4], [5], [6], [7], [8]). Although expressive power is an interesting and meaningful basis for comparing access control schemes, it exists only as a comparison made in absolute terms. That is, the knowledge that a scheme $\mathcal{S}$ is more expressive than another scheme $\mathcal{S}'$ provides no assurance that $\mathcal{S}$ is the best access control scheme for use within a particular real-world application context. It could be the case, for instance, that $\mathcal{S}'$ is *expressive enough* for a particular application and also has lower administrative overheads than $\mathcal{S}$ would in the same situation. As was noted in a recent NIST report, access control is not an area with "one size fits all" solutions and, as such, systems should be evaluated and compared relative to application-specific metrics [9]. This report notes a variety of possible access control quality metrics, but provides little guidance for actually applying these metrics and carrying out *practical* analyses of access control schemes.

Considering the wide availability of many diverse access control schemes and the relative difficulty of designing and building new secure systems from the ground up, an interesting topic exploration is that of *suitability analysis*. Informally, this problem can be stated as follows: *Given a description of a system's access control needs and a collection of existing access control schemes, which scheme best meets the needs of the system?* Instances of this question can arise in many different scenarios, ranging from the deployment of new applications within an organization, to the revisitation of long-standing protection mechanisms that no long perform as expected. In the former case, suitability analysis could help developers sort through the myriad available security frameworks (e.g., WPL [10], Spring [11], Shiro [12], etc.) and the multiple access control schemes embedded in each. In the latter case, suitability analysis can help an organization realign its security practices with its actual needs in an effort to avoid system abuse and circumvention (e.g., the sharing of data outside of established channels to avoid delays and other hindrances [13], [14]).

In this paper, we develop a mathematical framework and techniques to facilitate suitability analysis within the access control space. We first formalize the notion of an access control workload to abstract the required access control specific needs of an application and the expected uses of these functionalities. Analysis then consists of two orthogonal tasks: (i) demonstrating that each candidate access control scheme is capable of *safely* implementing the workload, and (ii) quantifying the costs associated the use of each candidate scheme. In this paper, we focus on one particular definition of a safe implementation [5], and leave the discussion of other possible notions of implementation to a companion paper [15]. Within this context, we develop techniques for safely extending the functionality of candidate schemes that require additional expressive power, develop guidelines for formally specifying a wide range of access control cost metrics, and investigate numerical and analytical methods for carrying out suitability analysis. In doing so, we make the following contributions:

- We present the first formal definition of an *access control workload*. This enables system administrators to clearly and concisely specify the functionalities that must be provided by access control schemes that are to be used within a given context, as well as identify the ways in which these schemes are envisioned to be exercised in practice.
- We develop formal cost metrics capable of encoding well-known operational costs (e.g., data structure management overheads [9]), implementation-specific costs (e.g., interface and state-management complexity between a candidate system and possible extensions to its TCB), as well as human-centric costs (e.g., administrative overheads).
- We describe a novel method for assessing the *suitability* of an existing access control scheme with respect to a particular workload. We first establish whether the candidate system is expressive enough to safely implement

the functionality of the workload via reduction. We then leverage the constructive nature of these reductions, the patterns of access invocations specified in the workload, and flexible cost metrics and labeling functions to explore the expected costs of deployment.
- To address issues of fragility that arise when constructing reductions between a workload and a candidate scheme, we introduce the notion of *access control auxiliary machines* (AMs). From a practical perspective, auxiliary machines represent "tweaks" that can be made to an existing scheme to increase the range of questions that it can answer. From a theoretical perspective, AMs describe the classes of enhancements to a scheme's expressive power that *do not* alter its safety properties. We formalize the properties of AMs, and demonstrate their use during suitability analysis.
- We present a detailed case study demonstrating how our framework can be used to gain insight into a realistic scenario. Namely, we investigate a workload derived from defense contractor technical reports describing changing military needs in the face of dynamic coalitions [16], [13].

The remainder of this paper is organized as follows. In Section II we discuss prior work on the formal analysis of access control schemes, and describe why it is insufficient for the problem addressed in this paper. In Section III, we describe the methods involved in specifying access control requirements and systems, including formally defining the mathematical structures we use to represent access control schemes, auxiliary machines, and cost labeling functions. In Section IV, we describe our approach to analyzing the suitability of a candidate access control scheme for a given workload. In Section V we apply our framework to a realistic case study. Finally, Sections VI and VII discuss current limitations of our approach and areas for future work.

II. Related Work

The formal study of access control schemes began with the seminal paper by Harrison, Ruzzo, and Ullman that investigated the rights leakage problem [1]. This paper formalized a general access control model and proved that determining whether a particular access right could ever be granted to a specific individual—the so-called "safety problem"—was an undecidable problem. Shortly thereafter, Lipton and Snyder showed that in a more restricted access control system, this problem was not only decidable, but decidable in linear time [2]. These two results introduced the notion that the most capable system is not always the right choice—that restricting our system can yield higher efficiency and greater ease in solving relevant security problems. This led to many results investigating the relative expressive power of various access control schemes, often leveraging some notion of (bi)simulation (e.g., [3], [4], [6], [7], [8]).

Further work by Ammann et al. [3], Chander et al. [4], and Li et al. [17] developed simulation-based frameworks for comparing the expressive power of various access control schemes. These simulation frameworks proved to be too relaxed, allowing almost any reasonable scheme to be shown equivalent to all others. To address this, Tripunitara and Li [5] developed a more restrictive notion of expressive power. Their framework supersedes the more informal notions of simulation developed in prior works by requiring the use of specific types of mappings between systems that guarantee relevant security properties are preserved under simulation; this provides a greater level of precision when ranking access control schemes in terms of their expressiveness. Unfortunately, none of these frameworks support the comparison of access control schemes with regards to their ability to perform *well* within a particular environment.

The need for application-specific evaluation of access control systems was reinforced by a recent NIST report, which states that "when it comes to access control mechanisms, one size does not fit all" [9]. The report bemoans the lack of established quality metrics for access control systems, going so far as to list more than a dozen possibilities, but stopping short of explaining how one might choose between them or evaluate established systems with respect to one's specific requirements. In this paper, we develop a formal framework for exploring exactly this problem.

Our notion of *access control auxiliary machines*—machines for expanding the expressiveness of an access control scheme without sacrificing safety—operates on similar principles to that of other work in extending existing access control schemes. Auxiliary machines are similar to, e.g., the Linux Security Modules framework developed by Wright et al. [18], which allows additional access control functions to be added to Linux via kernel modules. However, the LSM framework does not make any guarantees about the safety of the resulting system. More directly related is the work of Wang et al. [19], which looks at safely extending role-based access control schemes with delegation primitives. Our technique for extending access control schemes, by contrast, is more general and allows a broader class of extensions.

III. System and Workload Specification

Figure 1 displays the sequence of steps required to choose the most suitable access control scheme for a given application. The analyst starts by formalizing the objects of interest:

- The *workload*, which describes the access control requirements of the organization, both in terms of the operations required and how they will be used.
- The candidate *access control schemes*, formalized as state machines. These well-defined schemes are used as potential implementations of the workload's abstract access control requirements.
- The *cost labeling functions*, which assign costs to the basic operations of the candidate schemes. These functions are defined in terms of the *cost metric* that the analyst identifies as best representing the costs most critical to the analysis.

Once these artifacts have been formalized (the topic of this section), the analyst combines them to assign an overall cost to each candidate access control scheme for implementing the workload (Section IV).

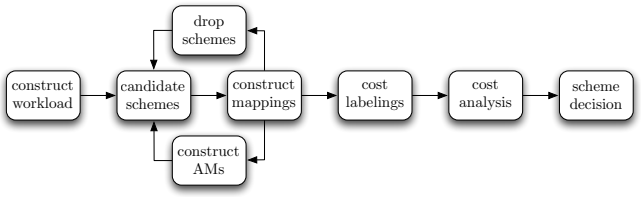


Fig. 1: The workflow of our analysis framework.

A. Access Control Scheme

An access control *system* is a state machine that (i) encodes the protection state of a system, (ii) answers questions based upon this state, and (iii) allows this state to evolve in response to commands. An access control *scheme* is a set of access control systems that differ only in how the state is allowed to evolve. In this paper, our framework is concerned with evaluating the suitability of an access control scheme for an application. Formally, we use the definition of an access control scheme first presented by Tripunitara and Li [5].

Definition 1 (Access Control Scheme): An *access control scheme* is a state-transition system $\mathcal{S} = \langle \Gamma, \Psi, Q, \vdash \rangle$, where Γ is the set of states, Ψ is the set of state transition rules, i.e., $\Psi \subseteq \wp(\Gamma \times \Gamma)$, Q is the set of boolean queries, and $\vdash : \Gamma \times Q \to \{true, false\}$ is the entailment relation. ♦

In an access control scheme $\mathcal{S} = \langle \Gamma, \Psi, Q, \vdash \rangle$, a state $\gamma \in \Gamma$ encodes the protection state of the system—i.e., all information required to make an access decision—at a given time. The queries, Q, represent the set of questions that the system can answer, including those that grant accesses to system resources. Given a query $q \in Q$, $\gamma \vdash q$ denotes that in state γ, q evaluates to $true$. A transition rule controls how the protection state evolves by dictating which states can be reached from which other states in a single step, i.e., it is a subset of $\Gamma \times \Gamma$. If a system is in state γ_i with transition rule $\psi \in \Psi$, transitioning to state γ_j is legal if and only if $\langle \gamma_i, \gamma_j \rangle \in \psi$. In practice, each transition rule is usually defined by a collection of commands (e.g., `add_subject`, `grant_right`, etc). A *set* of transition rules is consequently a subset of the power set of $\Gamma \times \Gamma$. In what follows, it will be useful to discuss the set of all possible transitions allowed by a scheme $\mathcal{S}$, i.e., $\bigcup \Psi$, which we denote with $\Phi^{\mathcal{S}}$. To illustrate the formalism, we present a simple discretionary access control (DAC) scheme.

Example 1: The DAC scheme is defined by $\mathcal{D} = \langle \Gamma^{\mathcal{D}}, \Psi^{\mathcal{D}}, Q^{\mathcal{D}}, \vdash^{\mathcal{D}} \rangle$. Each $\gamma \in \Gamma^{\mathcal{D}}$ is defined by $\langle S, O, R, M \rangle$, where S is the set of subjects, O is the set of objects, R is the set of rights, and $M : S \times O \to 2^R$ is the access matrix. $\Psi^{\mathcal{D}}$ is defined by the commands `create_object(`s`, `o`)`, `destroy_object(`s`, `o`)`, `create_subject(`s`, `s'`)`, `destroy_subject(`s`, `s'`)`, and `grant_`r_i`(`s`, `s'`, `o`)` and `revoke_`r_i`(`s`, `s'`, `o`)` for each $r_i \in R$, all of which work as expected. $Q^{\mathcal{D}}$ has all queries of the following forms: (1) "Does subject s exist?", and (2) "Does subject s have right r_i to object o?". $\vdash^{\mathcal{D}}$ is defined as follows for queries of each of the forms above: (1) $true$ if and only if $s \in S$, and (2) $true$ if and only if $r_i \in M[s, o]$. ♦

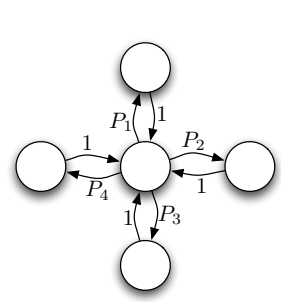
B. Access Control Workload

An access control workload describes an abstraction of the access control needs of an environment. A workload specifies both an *operational* component (the relevant operations that must be supported), as well as an *invocation* component (how those operations are expected to be used). The operational component can be viewed as the collection of high-level commands and queries that the application would like to execute, and hence can be formalized as an (abstract) access control state machine using Definition 1. We note that, while formalized in the same way, workloads and schemes differ in their intention. While a scheme represents a functioning piece of software, a workload is built by the analyst to represent the higher-level *desired* functionality of a system, without necessarily being appropriate for direct implementation.

The invocation component, on the other hand, cannot be formalized so easily. It describes the ways in which the system is typically used; i.e., the ways in which the high-level commands and queries are executed. At a minimum, the invocation component should be able to dictate the order in which transitions in the workload machine occur, the probability with which commands are executed, and which queries are asked during which paths of execution. In our framework, the invocation component is also a state transition system, but a simpler and more abstract one than the operational component. States are labeled with (sometimes empty) classes of transitions and queries—taken from the operational component of the workload—and transitions are labeled with probabilities. Each node represents events that are possible within the system, and a directed edge between two nodes represents the probability that one event will follow another.

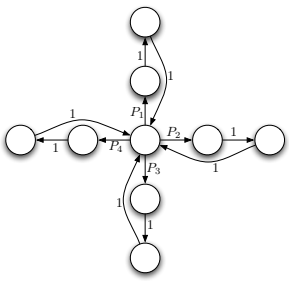
Definition 2 (Access Control Invocation): Let $\mathcal{S} = \langle \Gamma, \Psi, Q, \vdash \rangle$ be an access control scheme as defined in Definition 1. We say that $I^{\mathcal{S}} = \langle A, a_0, T \rangle$ is an *access control invocation* over the scheme $\mathcal{S}$ where (i) each state $a \in A$ is labeled with an element of $\wp(\Phi^{\mathcal{S}}) \cup \wp(Q)$, (ii) $a_0 \in A$ is the start state, and (iii) $T : A \times A \to [0, 1]$ is a probability-labeled set of transitions between states, i.e., $\forall a_i \in A : \sum_{a_j} T(a_i, a_j) = 1$. ♦

Figure 2 shows several invocation structures that can be represented using this formalism (where 0 probability edges are hidden). One of the simplest examples, specifying a probability distribution over system commands such as `add_user`, is depicted for 4 distinct commands in Figure 2a. The center node is the start event, each of the outer four nodes represent the execution of commands, and P_1, P_2, P_3, and P_4 give the probabilities for each command. Notice that when one of the command execution events occurs, the invocation returns to the start event with probability 1. Formally, labeling each of the outer nodes with commands amounts to labeling each of those nodes with a set of state transitions.

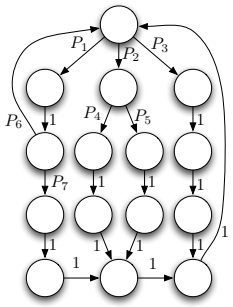


(a) Simple probability distribution

(b) Trace



(c) Transaction-based probability distribution



(d) General probabilistic system

Fig. 2: Example invocation structures.

For a more complex example, Figure 2c shows how to represent a transaction-based probability distribution. In this invocation, each "arm" describes a series of queries and transitions that make up a transaction, and each transaction happens with some probability (P_1, P_2, etc., in the figure). Note that this example is similar to the types of transaction-based workloads used for evaluating databases and distributed systems; e.g., TPC-C[1], the de facto standard in transaction processing benchmarks.

In our framework, the combination of an access control scheme and an invocation is a workload. The access control scheme describes the operations that must be supported for the application of interest, and the invocation is a Markov model that describes how those operations can be executed probabilistically.

Definition 3 (Access Control Workload): An *access control workload* is defined by $\langle \mathcal{S}, I^{\mathcal{S}} \rangle$, where the *operational component*, $\mathcal{S} = \langle \Gamma, \Psi, Q, \vdash \rangle$, is an access control scheme, and the *invocation component*, $I^{\mathcal{S}} = \langle A, a_0, T \rangle$, is an access control invocation over $\mathcal{S}$. ♦

Note that it is not always obvious how to transform an abstract description of one's desired access control policy into the machine-level specification required to specify the scheme component of a workload. We discuss this problem in Section VI.

[1]Transaction Processing Performance Council—Benchmarks, http://www.tpc.org/information/benchmarks.asp

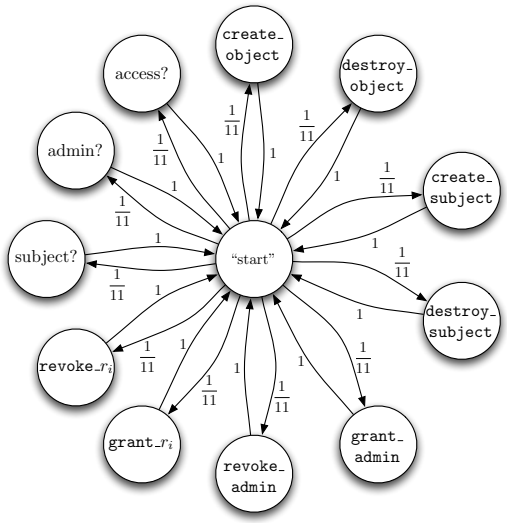


Fig. 3: The invocation, $I^{\mathcal{F}}$, from Example 2.

Example 2: Consider an environment that grants users discretionary control over their own resources, but allows administrators to have full access to any object. As a first approximation, analysts expect each action supported by the system to be equally likely to occur. For this workload, $W_{\mathcal{F}} = \langle \mathcal{F}, I^{\mathcal{F}} \rangle$, the invocation $I^{\mathcal{F}}$ is shown in Figure 3. $\mathcal{F}$ is similar to the DAC scheme from Example 1, but also maintains a set of administrators that have full access to each object in the system. It is defined by $\langle \Gamma^{\mathcal{F}}, \Psi^{\mathcal{F}}, Q^{\mathcal{F}}, \vdash^{\mathcal{F}} \rangle$. Each $\gamma \in \Gamma^{\mathcal{F}}$ is defined by $\langle S, A, O, R, M \rangle$, where S is the set of subjects, $A \subseteq S$ is the set of administrator subjects, O is the set of objects, R is the set of rights, and $M : S \times O \to 2^R$ is the access matrix. $\Psi^{\mathcal{F}}$ is defined by the commands `create_object(`s`, `o`)`, `destroy_object(`s`, `o`)`, `create_subject(`s`, `s'`)`, `destroy_subject(`s`, `s'`)`, `grant_admin(`s`, `s'`)`, `revoke_admin(`s`, `s'`)`, and `grant_`r_i`(`s`, `s'`, `o`)` and `revoke_`r_i`(`s`, `s'`, `o`)` for each $r_i \in R$, which perform the expected operations. $Q^{\mathcal{F}}$ has all queries of the following forms: (1) "Does subject s exist?", (2) "Is subject s an administrator?", and (3) "Does subject s have right r_i to object o?". $\vdash^{\mathcal{F}}$ is defined as follows for queries of each of the forms above: (1) $true$ if and only if $s \in S$, (2) $true$ if and only if $s \in A$, and (3) $true$ if and only if $r_i \in M[s, o] \vee s \in A$.

C. Cost Labeling Function

A cost labeling function dictates how expensive each command or query is for a candidate access control scheme. Given a cost labeling function and a means of translating workload commands/queries into the corresponding commands/queries of a candidate scheme, we can compute the cost of implementing the workload within that scheme.

An important part of a cost labeling function is choosing a relevant cost metric. This metric should be representative

of the "problem" (i.e., what types of cost the analyst cares about) while also enabling the definition of a cost labeling function for each candidate scheme. For example, while the metric "operational cost per day" may be representative of access control evaluation goals in industry, it is hard to assign costs in this metric for every system transition and query. A metric such as "average administrative personnel-hours spent per access control operation," on the other hand, is more easily quantified while still enabling the same types of analyses.

In this paper, we make no commitment to any particular cost metrics but rather develop an analysis framework that operates on any metric satisfying a number of simple properties. A cost metric must include a set of elements representing the costs, an associative and commutative operator that combines two costs to produce another cost (e.g., addition), and a *partial order* for comparing costs. Formally, this means that a cost metric is any *ordered abelian semigroup*.

Definition 4 (Cost Metric): A *cost metric*, $\mathbf{G} = \langle G, +, \leq \rangle$, is an ordered abelian semigroup under the binary operation $+$ and the partial order $\leq$. ♦

Definition 4 can be used to encode a variety of interesting access control metrics, including several of those noted in a recent NIST report on the assessment of access control schemes [9]. For example, costs like "steps required for assigning and dis-assigning user capabilities" and "number of relationships required to create an access control policy" can be represented using the cost metric $\langle \mathbb{N}, +, \leq \rangle$. Our notion of metric is general enough to represent many other types of costs as well. Metrics for human work such as "personnel-hours per operation" and "proportion of administrative work to data-entry work" can be represented using the cost metrics $\langle \mathbb{Z}^+, +, \leq \rangle$ and $\langle \mathbb{Z}^+ \times \mathbb{Z}^+, +, \leq \rangle$, respectively. Maximum memory usage can be represented using $\langle \mathbb{N}, \max, \leq \rangle$. In our case study (Section V-C1), we discuss several other cost metrics.

A common desire is for an analyst to evaluate an access control scheme using several different cost metrics in parallel. Thus, we define a *vector* of cost metrics.

Definition 5 (Vector of Cost Metrics): Given cost metrics $\mathbf{N}_1 = \langle N_1, +_1, \leq_1 \rangle$, $\mathbf{N}_2 = \langle N_2, +_2, \leq_2 \rangle$, ..., $\mathbf{N}_i = \langle N_i, +_i, \leq_i \rangle$, let $\mathbf{M} = \langle M, +_*, \leq_* \rangle$ be the *vector* of cost metrics $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$, where:

- $M = N_1 \times N_2 \times \cdots \times N_i$.
- Given $a_1, b_1 \in \mathbf{N}_1$, $a_2, b_2 \in \mathbf{N}_2$, ..., $a_i, b_i \in \mathbf{N}_i$, $\langle a_1, a_2, \ldots, a_i \rangle +_* \langle b_1, b_2, \ldots, b_i \rangle = \langle a_1 +_1 b_1, a_2 +_2 b_2, \ldots, a_i +_i b_i \rangle$.
- Given $a_1, b_1 \in \mathbf{N}_1$, $a_2, b_2 \in \mathbf{N}_2$, ..., $a_i, b_i \in \mathbf{N}_i$, $\langle a_1, a_2, \ldots, a_i \rangle \leq_* \langle b_1, b_2, \ldots, b_i \rangle$ if and only if $a_1 \leq_1 b_1 \wedge a_2 \leq_2 b_2 \wedge \ldots \wedge a_i \leq_i b_i$. ♦

Definition 5 gives a simple way of combining several metrics. As the following theorem states, a vector of cost metrics is also a cost metric, enabling the analyst to use a combination of metrics within our analysis framework. We prove Theorem 1 in Appendix C-A.

TABLE I: Cost labeling table for the DAC scheme from Example 3.

action	cost
`create_object`	1
`destroy_object`	2
`create_subject`	2
`destroy_subject`	2
`grant_`r_i	2
`revoke_`r_i	2
`s_exist?`	1
`access?`	1

Theorem 1: Given cost metrics $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$ and their vector, $\mathbf{M}$, $\mathbf{M}$ is a cost metric.

Once a metric is chosen, the analyst must next model how each candidate access control scheme accrues costs using that metric. This requires assigning costs associated with each change in system state and the computation of each query's answer. Such an assignment is a *cost labeling function*.

Definition 6 (Cost Labeling Function): Given access control scheme $\mathcal{S} = \langle \Gamma^{\mathcal{S}}, \Psi^{\mathcal{S}}, Q^{\mathcal{S}}, \vdash^{\mathcal{S}} \rangle$ and cost metric $\mathbf{G} = \langle G, +, \leq \rangle$, a *cost labeling function for $\mathcal{S}$ in $\mathbf{G}$* is defined as $\ell^{\mathcal{S}}_{\mathbf{G}} : \Phi^{\mathcal{S}} \cup Q^{\mathcal{S}} \to G$. ♦

While, formally, a cost labeling function is often infinite (since the number of states and queries is often infinite), in practice, the cost of any two instances of the same system command or any two instances of the same query form are often the same or are functions of the size of the state. Thus, cost labeling functions can usually be described in simple, finite terms.

Example 3: Recall the DAC scheme from Example 1. As a metric for cost analysis, we might choose to measure the number of data structure lookups, measured within the cost metric $\langle \mathbb{N}, +, \leq \rangle$, using the usual notions of addition and less-than. For example, if answering a query only requires information from a single cell of an array, the cost is 1, while queries that require information about each user will have a cost proportional to the number of users. Table I lists the cost for each command and query. For example, executing `grant_`r_i requires checking that the executing user can grant this right (1 lookup), and entering r_i in the correct cell of the matrix (1 lookup) for a total cost of 2 operations. ♦

IV. Suitability Analysis

Figure 4 is a graphical representation of the suitability analysis process. Once the analyst has formally specified the workload, candidate schemes, and cost labeling functions (described in Section III), these objects are used to assign an overall cost to each access control scheme. This task is comprised of two steps. In the first (security analysis), the analyst constructs mappings from the workload to each of the candidate access control schemes that demonstrate how the workload's operations can be carried out by the scheme's operations (while preserving the safety properties of

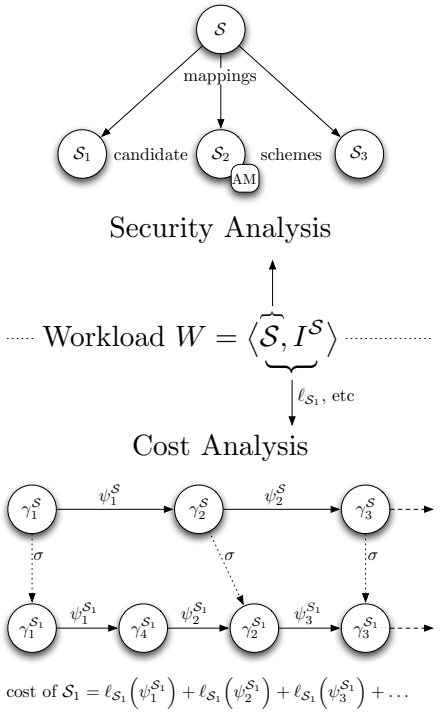


Fig. 4: Overview of the analysis framework.

the workload). For each scheme which does not admit such a mapping, the analyst can choose to augment the scheme with an access control auxiliary machine that enables such a mapping, if possible. In the second step (cost analysis), the analyst computes (approximations of) the expected cost of implementing the workload on each of the schemes and deems the scheme that is assigned the least cost to be the one most suitable for the application.

A. Preliminaries

Once the analyst selects an appropriate set of candidate access control schemes, she must verify each scheme's ability to safely execute the operations required by the workload. To do so, the analyst demonstrates the existence of mappings from the workload's operational component (which, recall, is itself is an access control scheme) to each of the candidate access control schemes. The mappings provide a translation from the workload's operations to (sequences of) operations from each candidate scheme. Moreover, these mappings are used to guarantee that the safety properties of the workload are preserved in each candidate scheme. In this paper, we leverage the notion of safe implementations first developed in [5], and leave the orthogonal issue of exploring the use of other types of implementation to our companion paper [15].

More precisely, a mapping is said to be *strongly security preserving* if it preserves all *compositional security analysis* instances [5]. A compositional security analysis instance is a generalization of the simple safety analysis first formalized in [1] to arbitrary boolean combinations of queries. A mapping σ from scheme $\mathcal{A}$ to scheme $\mathcal{B}$ translates state/transition rule pairs from $\mathcal{A}$ to state/transition rule pairs from $\mathcal{B}$, and also translates the queries from $\mathcal{A}$ to queries from $\mathcal{B}$. If σ preserves compositional security analysis, pairs of corresponding states across the mapping answer all compositional security analysis instances in the same way.

Included in [5] is a type of mapping that is useful in performing security analysis: the *state-matching reduction*. If γ is a state-matching reduction from $\mathcal{A}$ to $\mathcal{B}$, then pairs of corresponding states across the mapping answer all boolean combinations of queries in the same way. Furthermore, for every state γ from $\mathcal{A}$ and every state γ' from $\mathcal{A}$ reachable from γ, $\sigma(\gamma')$ is reachable from $\sigma(\gamma)$ in $\mathcal{B}$.

A mapping is a state-matching reduction if and only if it is strongly security-preserving. However, state-matching reductions are easier to construct, since they are based on structural properties, while proving a strongly security-preserving mapping requires the direct analysis of all security analysis instances.

Definition 7 (State-Matching Reduction [5]): Given two access control schemes $\mathcal{A} = \langle \Gamma^{\mathcal{A}}, \Psi^{\mathcal{A}}, Q^{\mathcal{A}}, \vdash^{\mathcal{A}} \rangle$ and $\mathcal{B} = \langle \Gamma^{\mathcal{B}}, \Psi^{\mathcal{B}}, Q^{\mathcal{B}}, \vdash^{\mathcal{B}} \rangle$, and a mapping from $\mathcal{A}$ to $\mathcal{B}$, $\sigma : \left(\Gamma^{\mathcal{A}} \times \Psi^{\mathcal{A}}\right) \cup Q^{\mathcal{A}} \to \left(\Gamma^{\mathcal{B}} \times \Psi^{\mathcal{B}}\right) \cup Q^{\mathcal{B}}$, we say that two states $\gamma^{\mathcal{A}}$ and $\gamma^{\mathcal{B}}$ are *equivalent* under the mapping σ when for every $q^{\mathcal{A}} \in Q^{\mathcal{A}}$, $\gamma^{\mathcal{A}} \vdash^{\mathcal{A}} q^{\mathcal{A}}$ if and only if $\gamma^{\mathcal{B}} \vdash^{\mathcal{B}} \sigma\left(q^{\mathcal{A}}\right)$. A mapping σ from $\mathcal{A}$ to $\mathcal{B}$ is said to be a *state-matching reduction* if for every $\gamma^{\mathcal{A}} \in \Gamma^{\mathcal{A}}$ and every $\psi^{\mathcal{A}} \in \Psi^{\mathcal{A}}$, $\langle \gamma^{\mathcal{B}}, \psi^{\mathcal{B}} \rangle = \sigma\left(\langle \gamma^{\mathcal{A}}, \psi^{\mathcal{A}} \rangle\right)$ has the following two properties:

1) For every state $\gamma_1^{\mathcal{A}}$ in scheme $\mathcal{A}$ such that $\gamma^{\mathcal{A}} \overset{*}{\mapsto}_{\psi^{\mathcal{A}}} \gamma_1^{\mathcal{A}}$, there exists a state $\gamma_1^{\mathcal{B}}$ such that $\gamma^{\mathcal{B}} \overset{*}{\mapsto}_{\psi^{\mathcal{B}}} \gamma_1^{\mathcal{B}}$ and $\gamma_1^{\mathcal{A}}$ and $\gamma_1^{\mathcal{B}}$ are equivalent under σ.
2) For every state $\gamma_1^{\mathcal{B}}$ in scheme $\mathcal{B}$ such that $\gamma^{\mathcal{B}} \overset{*}{\mapsto}_{\psi^{\mathcal{B}}} \gamma_1^{\mathcal{B}}$, there exists a state $\gamma_1^{\mathcal{A}}$ such that $\gamma^{\mathcal{A}} \overset{*}{\mapsto}_{\psi^{\mathcal{A}}} \gamma_1^{\mathcal{A}}$ and $\gamma_1^{\mathcal{A}}$ and $\gamma_1^{\mathcal{B}}$ are equivalent under σ. ♦

Theorem 2 (Rephrased, from [5]): Given two schemes $\mathcal{A}$ and $\mathcal{B}$, and a mapping, σ, from $\mathcal{A}$ to $\mathcal{B}$, σ is a state-matching reduction if and only if it is strongly security-preserving; that is, every compositional security analysis instance in $\mathcal{A}$ is true if and only if the image of the instance under σ is true in $\mathcal{B}$.

B. Security Analysis

Recall from Section III that the operational component of an access control workload is itself an access control scheme. As such, we immediately have the following corollary establishing the utility of Tripunitara and Li's reduction framework for determining whether a given scheme is capable of safely implementing the operations required by a workload. For brevity, we often refer to a mapping from the operational component of workload $W_{\mathcal{S}}$ to scheme $\mathcal{A}$ as simply a mapping from $W_{\mathcal{S}}$ to $\mathcal{A}$.

Corollary 3: Given workload $W_{\mathcal{S}} = \langle \mathcal{S}, I^{\mathcal{S}} \rangle$ and access control scheme $\mathcal{A}$, a mapping σ from $\mathcal{S}$ to $\mathcal{A}$ preserves all compositional security analysis instances of $W_{\mathcal{S}}$ in $\mathcal{A}$ if and only if σ is a state-matching reduction.

Definition 8 (Safe Implementation): Given an access control workload, $W_{\mathcal{S}} = \langle \mathcal{S}, I^{\mathcal{S}} \rangle$, access control scheme, $\mathcal{A}$, and a mapping σ from $\mathcal{S}$ to $\mathcal{A}$, $\mathcal{A}$ and σ represent a *safe implementation* of $W_{\mathcal{S}}$ if and only if σ is a state-matching reduction. ♦

If there is a state-matching reduction from the workload of interest to each of the candidate access control schemes, then all of the schemes admit safe implementations of the workload, and the analyst can begin cost analysis. But it is possible that one or more schemes may not safely implement the workload. For example, despite the similarities between the DAC scheme $\mathcal{D}$ introduced in Example 1 and the workload $W_{\mathcal{F}}$ introduced in Example 2, it is impossible to define a state-matching reduction from $W_{\mathcal{F}}$ to $\mathcal{D}$. When a scheme fails to admit a safe implementation of a workload, the analyst must choose either to eliminate that scheme from the analysis or to augment it with an *access control auxiliary machine* that enables a the construction of a safe implementation. Note, though, that it is not always possible to build an AM to enable the construction of a given mapping; we will explore such an example in Section V-B.

Intuitively, an access control auxiliary machine (AM) adds protected state, transitions for that state, and queries that can be answered using a combination of the old and the new state. More formally, an AM is an access control scheme whose entailment relation can also depend on the state of the scheme being augmented.

Definition 9 (Access Control Auxiliary Machine): An *access control auxiliary machine* for augmenting an access control scheme with states Γ_0 is a state-transition system $\langle \Gamma, \Psi, Q, \vdash \rangle$, where Γ is the set of states, $\Psi \subseteq \wp(\Gamma \times \Gamma)$ is the set of state transition rules, Q is the set of boolean queries, and $\vdash\colon \langle \Gamma_0 \times \Gamma \rangle \times Q \to \{true, false\}$ is the entailment relation. ♦

Augmenting an access control scheme with an auxiliary machine is achieved by computing the cross product of the states of the two machines, the cross product of the transition rules of the two machines, the union of the queries, and roughly the union of the two entailment relations of the two machines. The entailment relations are combined such that all the original queries are answered by the original entailment relation, and all of the new queries are answered by the new relation.

Definition 10 (Augmented Access Control Scheme): Let $\mathcal{S} = \left\langle \Gamma^{\mathcal{S}}, \Psi^{\mathcal{S}}, Q^{\mathcal{S}}, \vdash^{\mathcal{S}} \right\rangle$ be an access control scheme, $\mathcal{U} = \left\langle \Gamma^{\mathcal{U}}, \Psi^{\mathcal{U}}, Q^{\mathcal{U}}, \vdash^{\mathcal{U}} \right\rangle$ be an access control auxiliary machine, and $Q^{\mathcal{S}} \cap Q^{\mathcal{U}} = \emptyset$. The *augmented access control scheme* formed by augmenting scheme $\mathcal{S}$ with AM $\mathcal{U}$, is the scheme $\mathcal{S} \circ \mathcal{U} = \left\langle \Gamma^{\mathcal{S}\circ\mathcal{U}}, \Psi^{\mathcal{S}\circ\mathcal{U}}, Q^{\mathcal{S}\circ\mathcal{U}}, \vdash^{\mathcal{S}\circ\mathcal{U}} \right\rangle$ where

- $\Gamma^{\mathcal{S}\circ\mathcal{U}} = \Gamma^{\mathcal{S}} \times \Gamma^{\mathcal{U}}$
- $\Psi^{\mathcal{S}\circ\mathcal{U}} = \left\{ \psi^{\mathcal{U}} \times \psi^{\mathcal{S}} \mid \psi^{\mathcal{S}} \in \Psi^{\mathcal{S}}, \psi^{\mathcal{U}} \in \Psi^{\mathcal{U}} \right\}$ where $\psi^{\mathcal{U}} \times \psi^{\mathcal{S}}$ contains all $\langle\langle a, c\rangle, \langle b, c\rangle\rangle, \langle\langle a, c\rangle, \langle a, d\rangle\rangle$ such that $\langle a, b\rangle \in \psi^{\mathcal{U}}$, $\langle c, d\rangle \in \psi^{\mathcal{S}}$
- $Q^{\mathcal{S}\circ\mathcal{U}} = Q^{\mathcal{S}} \cup Q^{\mathcal{U}}$
- $\vdash^{\mathcal{S}\circ\mathcal{U}}\colon \Gamma^{\mathcal{S}\circ\mathcal{U}} \times Q^{\mathcal{S}\circ\mathcal{U}} \to \{true, false\}$ where, if $q \in Q^{\mathcal{S}}$, then $\left\langle \gamma^{\mathcal{S}}, \gamma^{\mathcal{U}} \right\rangle \vdash^{\mathcal{S}\circ\mathcal{U}} q \Leftrightarrow \gamma^{\mathcal{S}} \vdash^{\mathcal{S}} q$, and if $q \in Q^{\mathcal{U}}$, then $\left\langle \gamma^{\mathcal{S}}, \gamma^{\mathcal{U}} \right\rangle \vdash^{\mathcal{S}\circ\mathcal{U}} q \Leftrightarrow \left\langle \gamma^{\mathcal{S}}, \gamma^{\mathcal{U}} \right\rangle \vdash^{\mathcal{U}} q$. ♦

By observing the parts of Definition 10 defining states, queries, and entailment—$\Gamma^{\mathcal{S}\circ\mathcal{U}}$, $Q^{\mathcal{S}\circ\mathcal{U}}$, and $\vdash^{\mathcal{S}\circ\mathcal{U}}$, respectively—the following lemma becomes apparent.

Lemma 4: Given access control scheme $\mathcal{S} = \left\langle \Gamma^{\mathcal{S}}, \Psi^{\mathcal{S}}, Q^{\mathcal{S}}, \vdash^{\mathcal{S}} \right\rangle$ and access control auxiliary machine $\mathcal{U} = \left\langle \Gamma^{\mathcal{U}}, \Psi^{\mathcal{U}}, Q^{\mathcal{U}}, \vdash^{\mathcal{U}} \right\rangle$, for all $q \in Q^{\mathcal{S}}$, $\gamma \in \Gamma^{\mathcal{S}}$, and $\gamma' \in \Gamma^{\mathcal{U}}$, $\gamma \vdash^{\mathcal{S}} q$ if and only if $\langle \gamma, \gamma' \rangle \vdash^{\mathcal{S}\circ\mathcal{U}} q$.

This lemma provides the state correspondence required by the state-matching reduction as defined in Definition 7. Likewise, the reachability properties are satisfied by the transition rules, $\Psi^{\mathcal{S}\circ\mathcal{U}}$, of Definition 10. Combining these yields the following theorem, which justifies our definition for augmenting a scheme: an augmented access control scheme inherits desired security properties of the original scheme—in particular those relevant to the state-matching reduction—while also gaining expressive power from the AM.

Theorem 5: Given access control scheme $\mathcal{S} = \left\langle \Gamma^{\mathcal{S}}, \Psi^{\mathcal{S}}, Q^{\mathcal{S}}, \vdash^{\mathcal{S}} \right\rangle$ and access control auxiliary machine $\mathcal{U} = \left\langle \Gamma^{\mathcal{U}}, \Psi^{\mathcal{U}}, Q^{\mathcal{U}}, \vdash^{\mathcal{U}} \right\rangle$, there exists a state-matching reduction from $\mathcal{S}$ to $\mathcal{S} \circ \mathcal{U}$.

This theorem is proven in Appendix C-B.

Following from Theorem 5, the following corollary ensures that an access control scheme can be augmented with an AM without violating security properties that can be expressed as compositional security analysis instances.

Corollary 6: For any access control scheme $\mathcal{S}$ and access control auxiliary machine $\mathcal{U}$, there exists a mapping σ from $\mathcal{S}$ to $\mathcal{S} \circ \mathcal{U}$ such that, for any compositional security analysis instance i in $\mathcal{S}$, i is true in $\mathcal{S}$ if and only if the image of i in $\mathcal{S} \circ \mathcal{U}$ via mapping σ is true in $\mathcal{S} \circ \mathcal{U}$.

While these security properties of auxiliary machines and augmented schemes enable the analyst to use the constructs without fear of contaminating the original schemes, they do not imply that the use of AMs is without penalty. Since AMs would be implemented as additional trusted code that communicates in a secure way with the original access control software, one may be concerned if a high proportion of the total state is stored within the AM, or if a large amount of communication needs to occur between the original state and the AM state. We highlight the generality of our definition of cost metric (Definition 4) by observing that these types of concerns can easily be addressed within that definition during the cost analysis phase. Recall that a cost metric's underlying algebraic structure is the ordered abelian semigroup. This allows us to represent, for example, the maximum proportion of total state contained within the AM using the metric $\langle [0, 1], \max, \leq \rangle$. We will show another method of addressing these concerns in our case study (Section V-C), where we measure amount of I/O performed by the AM.

We now demonstrate the use of access control auxiliary machines by augmenting the DAC scheme $\mathcal{D}$ presented in Example 1 with the features needed to support the workload

$W_{\mathcal{F}}$ presented in Example 2.

Example 4: Recall that the workload $W_{\mathcal{F}}$ (Example 2) differs from the DAC scheme $\mathcal{D}$ mainly in that $W_{\mathcal{F}}$ has administrators with full rights to the system. In particular, queries of form (2) within the workload are problematic, as the DAC scheme $\mathcal{D}$ has no way of maintaining the list of administrative users. One natural attempt at fixing this problem is to create a special object within $\mathcal{D}$, rights over which indicate administrator status. Another possibility is to create a special right that administrators have over all objects. Such approaches fail to allow a safe implementation, because they invalidate security analysis instances. In particular, the query "Does subject s have right r_{admin} to object o?" is always false in the workload's scheme component, (which was why r_{admin} was chosen), but it is sometimes true in $\mathcal{D}$ if we attempt to use it in this way.

Instead, we construct an auxiliary machine that stores information about administrators and answers queries about administrators, yet ensures the original queries are answered by the original system. Let the auxiliary machine $\mathcal{G} = \left\langle \Gamma^{\mathcal{G}}, \Psi^{\mathcal{G}}, Q^{\mathcal{G}}, \vdash^{\mathcal{G}} \right\rangle$. Each $\gamma \in \Gamma^{\mathcal{G}}$ is the state defined by $\langle A, N \rangle$, where $A \subseteq S$ is the set of subjects that are administrators, and $N : A \times O \to 2^R$ is a "hidden" access matrix that keeps track of the access rights each administrative subject would revert to upon losing administrator status. Note that this AM relies upon the existence of the sets S, O, and R in the access control scheme it will be applied to. $\Psi^{\mathcal{G}}$ is defined by the commands `grant_admin(`s`, `s'`)`, `revoke_admin(`s`, `s'`)`, and `soft_grant_`r_i`(`s`, `s'`, `o`)` and `soft_revoke_`r_i`(`s`, `s'`, `o`)` for each $r_i \in R$. $Q^{\mathcal{G}}$ includes all queries of the forms: (1) "Is subject s an administrator?," and (2) "Does subject s have hidden right r_i to object o?." Finally, $\vdash^{\mathcal{G}}$ is defined as (1) $true$ if and only if $s \in S$, and (2) $true$ if and only if $r_i \in N[s, o]$.

This AM will augment the DAC scheme with the ability to keep track of which subjects are administrators, as well as which rights each one would have if they lost such status. The mapping from the workload $W_{\mathcal{F}}$ to this new scheme is then obvious, with one exception: not only does the new scheme have to keep track of this new information, it also has to answer queries of form (2) *without* changing $\vdash$ for those queries. Thus, when a subject is added to A, the mapping must copy all current accesses for that subject from M to N and then grant that subject all accesses in M. This procedure is reversed when removing an object from A, and any accesses granted to or revoked from a user in A should take place in N and have no effect on M. ♦

C. Cost Analysis

After completing security analysis, the analyst has constructed a way, for each scheme, of translating a workload command to a sequence of commands within that scheme. This translation allows us to compute the cost of executing each workload command in each of the candidate schemes, and therefore to compute the overall cost of executing the workload in each scheme as specified by the invocation. We denote the function that maps each transition in the workload W to a sequence of transitions in the the scheme $\mathcal{A}$ as $\varepsilon^{\mathcal{A}} : \Phi^W \to \Phi^{\mathcal{A}^*}$. In theory, this is enough to compute the expected cost of executing the workload in each scheme. Below we discuss the two usual approaches to expected value computation: approximation via simulation and exact.

1) Simulation: Recall that the invocation component of a workload is a state machine where each state describes the type of transition or query to be carried out in that state, and each state transition is labeled with a probability. To employ simulation for cost analysis, we choose a reasonable start state for the scheme at random, and then randomly run the invocation machine. For each transition or query that is reached, the appropriate cost is aggregated with a running total, where the notion of "aggregate" is made possible by the cost metric's semigroup operator. The simulation procedure is described in Algorithm 1, with cost labeling function ℓ and mapping σ. Note that += refers to the application of the semigroup operator.

Algorithm 1 Cost analysis simulation algorithm

```
s = random system start state
for all actions 1..limit
  n = random next node in invocation
  if n is a state transition φ
    apply φ to s
    for all φ^s ∈ ε(φ)
      cost += ℓ(φ^s)
  if n is a query q
    cost += ℓ(σ(q))
```

2) Expected Value: While simulation can be applied to a wide range of workloads, an exact expected cost is preferable when it can be computed. For cost labeling functions where, for example, the cost of each node in the invocation is constant (i.e., not dependent on the size of elements of the state), there is a static procedure for calculating the expected cost. For instance, in Example 2, we defined a workload with an invocation that represented a simple probability distribution among all commands and queries. The expected cost of running this system can be expressed algebraically because each command and query has a constant cost.

The exact expected cost of executing the implementation of workload $W = \left\langle \mathcal{S}, I^{\mathcal{S}} \right\rangle$ in scheme $\mathcal{T}$ for k steps is expressed as follows. Here, the outer summation is over each c in the transition rules, which we often use to mean the transitions brought about by a single command, but can represent any class of transitions whose costs are equal, and can thus be grouped as the label of a single node of the scheme's invocation graph. Note that Σ refers to the repeated application of the semigroup operator, which is not necessarily summation.

$$E(\text{cost}) = k \sum_{c \in \Phi^{\mathcal{S}}} \Pr(c) \left(\sum_{c^{\mathcal{T}} \in \varepsilon^{\mathcal{T}}(c)} \ell^{\mathcal{T}}\left(c^{\mathcal{T}}\right) \right) + k \sum_{q \in Q^{\mathcal{S}}} \Pr(q) \left(\ell^{\mathcal{T}}\left(\sigma^{\mathcal{T}}(q)\right) \right)$$

Of the two approaches, expected value is more desirable than simulation because it results in an exact overall cost. However, expected value calculation is often impractical. For example, if the cost of a transition or query is a function of the current state (e.g., the number of objects), simulation is a better choice because this information can be maintained and is easy to inspect. Also, for workloads based on a recorded trace of a running system, simulation is more natural, since "playing back" the recording is likely easier than calculating expected value. If, however, the invocation probabilities are simple and the command costs are constant, computing expected value may be feasible.

V. Case Study: Coalition Access Control

In this section, we present a full example of how analysis is carried out within our framework. We define a workload for a modern military access control system, based on U.S. armed forces and defense contractor technical reports [16], [13], and use reasonable approximations in place of information that is not public. Then, we conduct a hypothetical (yet realistic) analysis using the techniques described in the previous sections. Specifically, this example explains the process that the U.S. military might use to decide whether their current access control system—based on the Bell-LaPadula (BLP) scheme—is still the best choice given their evolving requirements, namely the increase reliance on short-lived coalitions.

A. Modern Military Needs

1) Background and Context: In the BLP scheme, each object is assigned a *classification* (i.e., unclassified, classified, secret, or top secret) and any number of *compartments* or *labels*. In addition, each subject is assigned a clearance to indicate the level at which they are trusted, and any number of compartments indicating their areas of "need to know." Classifications/clearances are ordered, whereas compartments are not. The *dominates* relationship (denoted $\succ$) is used to make mandatory access control (MAC) decisions, and is a partial order over security levels (a security level is a clearance or classification combined with a set of compartments). Given two security levels l_1, l_2, $l_1 \succ l_2$ if and only if the classification of l_1 is at least as high as that of l_2, and the set of compartments of l_1 is a superset of that of l_2. To be allowed to read an object, a subject's security level must dominate that object's security level. To be allowed to write, the opposite is enforced—the object must dominate the subject. In addition to this mandatory component, BLP also contains a discretionary access control matrix. Even if a subject passes the MAC requirements above, it must also have access according to the matrix.

2) Towards a Modern Military Access Control Workload: We now summarize the U.S. military's access control needs as described in a recent U.S. Air Force publication [16], and use the details of these requirements to build our example workload. This report describes a shift in the military's access control requirements, mainly as a result of a greater reliance on short-term multinational coalitions. Specifically, use of short-lived coalitions necessitates the identification and alteration of security levels for large numbers of documents, actions which require high amounts of work by the system's administrators. The report envisions a new access control system for the military, in which users and data objects are annotated with metadata descriptions of their content and attributes, and access decisions are made in a policy- or rule-based manner by using these metadata sets.

We now define our *modern military access control workload*. In the operational description of this workload, users (resp. documents) are assigned attributes (resp. tags) capturing their security-relevant properties. Logical policies are then written to grant users with certain sets of attributes access to documents with certain sets of tags. Thus, changes in the policy can affect access for a large number of users and/or documents at once. The operational description of this workload must maintain a set of users, a set of documents, relations assigning users (resp. documents) their attributes (resp. tags), and a logical policy. Commands allow transitions that create/destroy users and documents, assign/revoke tags and attributes, and alter policy. Queries allow asking whether a user exists, whether a user has a specified attribute, whether a document has a specified tag, or whether a specified user has a specified right to a specified document. We now formally define the operational component, $\mathcal{M}$, of the modern military workload.

Definition 11 (Modern Military Workload Scheme): In the *modern military workload scheme*, $\mathcal{M}$, we assume the existence of sets A, the set of possible attributes, i.e., metadata for users; T, the set of possible tags, i.e., metadata for documents; and I, the set of access rights. These components are not defined as part of the state, as they do not change. The MMW scheme is then defined as the state transition system $\mathcal{M} = \left\langle \Gamma^{\mathcal{M}}, \Psi^{\mathcal{M}}, Q^{\mathcal{M}}, \vdash^{\mathcal{M}} \right\rangle$. Each MMW state $\gamma^{\mathcal{M}} \in \Gamma^{\mathcal{M}}$ is defined by $\langle U, D, UA, DT, P \rangle$, where U is the set of users, D is the set of documents, $UA \subseteq U \times A$ is the user-attribute relation, $DT \subseteq D \times T$ is the document-tag relation, and P is the set of policy sentences, written in a formal policy language. $\Psi^{\mathcal{M}}$ is defined by the commands:

- `create_document(`u`, `d`)`
- `destroy_document(`u`, `d`)`
- `assign_tags(`u`, `d`, `T_1`)`
- `revoke_tags(`u`, `d`, `T_1`)`
- `create_user(`u`, `u'`)`
- `destroy_user(`u`, `u'`)`
- `assign_attributes(`u`, `u'`, `A_1`)`
- `revoke_attributes(`u`, `u'`, `A_1`)`
- `transform_policy(`u`, `P_+`, `P_-`)`

$Q^{\mathcal{M}}$ includes all queries of the following forms: (1) "Does user u exist?", (2) "Does user u have attribute a?", (3) "Does document d have tag t?", and (4) "Does user u have access i to document d?". $\vdash^{\mathcal{M}}$ is defined as follows for queries of each of the forms above: (1) $true$ if and only if $u \in U$, (2) $true$ if and only if $\langle u, a \rangle \in UA$, (3) $true$ if and only if $\langle d, t \rangle \in DT$, and (4) $true$ if and only if $\exists T_1 \subseteq T, A_1 \subseteq A : \forall t \in T_1, \langle d, t \rangle \in DT \wedge \forall a \in A_1, \langle u, a \rangle \in UA \wedge P$ evaluates, under its language's procedure, to a fixed point in which users with attribute set A_1 have access i to documents with tag set T_1. ♦

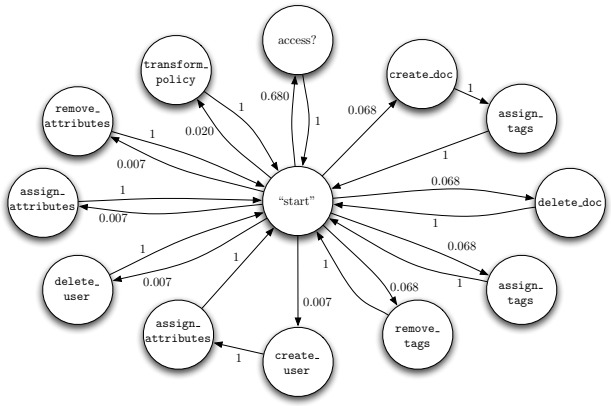


Fig. 5: $I_{\mathcal{M}}$, the invocation component of the modern military workload.

For simplicity, we represent the invocation component of the modern military workload as a probability distribution augmented to include secondary actions that are triggered by certain primary actions. For example, document creation is followed by a tag assignment process. Figure 5 shows invocation $I_{\mathcal{M}}$, which completes the modern military workload, $W_{\mathcal{M}} = \langle \mathcal{M}, I_{\mathcal{M}} \rangle$.[2] As seen in Figure 5, this workload relies on ephemeral data, transient users, and the ability to automatically assign accesses based on the metadata of both users and documents, all of which are elements of the evolution of the military's access control scheme as described in the aforementioned Air Force report [16].

B. Security Analysis

We now describe the process of security analysis for our case study.

1) Candidate Schemes: We now describe our candidate schemes. One obvious choice for implementing the workload $W_{\mathcal{M}}$—simply because it is currently in use within the military—is the BLP scheme overviewed in Section V-A. BLP is formally defined in Definition 13. Given the policy-based nature of $W_{\mathcal{M}}$, another interesting choice is a tag-based parameterization of a trust management language like SD3 [20], henceforth called SD3-T. In such a system, binary relations `TAG(D, T)` and `ATTR(U, A)` can be used to assign tags (resp., attributes) to documents (resp., users). Datalog Horn clauses can then be used to encode logical access control policies. For example:

```
READ(U,D) :- ATTR(U, top_secret),
             ATTR(U, coalition_alpha),
             TAG(D, op_alpha)
```

This policy grants members of the coalition "Alpha" who possess a top secret clearance attribute access to documents about operation "Alpha." A formal definition of SD3-T is given in Definition 12.

[2]Note, again, that $I_{\mathcal{M}}$ is only one possible—not *the* definitive—invocation for the U.S. military's current system, as much of the information necessary to make the latter claim is not publicly available.

A third scheme that we will consider is role-based access control (RBAC), since it is a standardized scheme that is widely used in industry. The RBAC scheme maintains a set of subjects, set of permissions, and relations for the subject-role assignment and role-permission relation. Commands allow creating and destroying subjects and permissions, granting and revoking roles, and granting and revoking permissions. Queries allow asking if a subject exists, if a subject belongs to a specified role, or whether a subject has a specified permission. RBAC is defined formally in Definition 14.

2) Mappings and Auxiliary Machines: Now that we have defined a workload and identified candidate schemes, we can begin a formal suitability analysis. First, we use security analysis to ensure that each candidate scheme is expressive enough to be capable of executing our workload safely, as described in Section IV-B. SD3-T is the easiest of the candidate schemes with regards to security analysis.

Theorem 7: SD3-T is expressive enough to safely implement the modern military workload.

This theorem is proven in Appendix C-C.

Next, we investigate the BLP scheme. BLP is based on a one-dimensional lattice-based access control scheme (LBAC), adopting its mechanism for storing the mandatory component of the decision mechanism. While it is tempting to utilize this lattice for the access control mapping from workload W_T, we have proven that using the lattice alone is not possible.[3] Furthermore, the incompatibility is so fundamental that even an auxiliary machine will not allow LBAC to execute the workload.

Theorem 8: LBAC is not expressive enough to safely implement the modern military workload.

Corollary 9: LBAC is not expressive enough to safely implement the modern military workload, regardless of any access control auxiliary machine used to augment it.

This theorem and corollary are proven in Appendix C.

Although LBAC cannot safely implement $W_{\mathcal{M}}$ alone, BLP adds a discretionary component that pure LBAC does not have. Further, accesses are checked against the current access set (denoted b in the BLP model), rather than by using the lattice directly. As a result, the BLP scheme can be extended via an access control auxiliary machine (described in Appendix B) to safely implement the workload $W_{\mathcal{M}}$.

Theorem 10: BLP is expressive enough to safely implement the modern military workload, when augmented via the tag-based auxiliary machine, $\mathcal{N}$.

The tag-based AM, $\mathcal{N}$, is described in Definition 15, and Theorem 10 is proven in Appendix C-F.

[3]Note that, as defined, subjects and objects in LBAC are granted a single clearance/classification and a single set of compartments, used to determine both read and write privileges. We have not investigated systems in which separate security levels are granted for determining read privileges and write privileges, and whether such systems may be able to implement workload W_T.

We note, however, that the mapping used to construct the state matching reduction from the modern military workload to the augmented BLP scheme relies almost entirely on external state managed by the AM, and the interesting components of the original BLP state are set to trivially simple special cases. The impracticalities of this mapping will become apparent during the cost analysis presented in Section V-C.

The last candidate scheme that we consider is RBAC. The mapping we have defined for this scheme can be seen as a midway point between the previous two examples: SD3-T is a near-perfect fit for the workload $W_{\mathcal{M}}$, while the use of BLP requires a degenerate mapping that uses an access control auxiliary machine to maintain all of the significant state used in building the state matching reduction. To implement $W_{\mathcal{M}}$ using RBAC, we can leverage RBAC's subject-role relation to maintain the user-attribute relation information from $W_{\mathcal{M}}$, and use a AM to manage the document-tag bindings.

Theorem 11: RBAC is expressive enough to safely implement the modern military workload, when augmented via the object-tag auxiliary machine, $\mathcal{O}$.

The object-tag AM, $\mathcal{O}$, is described in Definition 16, and Theorem 11 is proven in Appendix C-G.

C. Cost Analysis

Now that we have ensured that all three candidate access control schemes possess (or can be augmented to attain) the necessary expressiveness to safely execute the workload $W_{\mathcal{M}}$, we describe the process of using simulation to conduct a cost analysis for each candidate scheme.

1) Cost Metrics: Recall that the first step in cost analysis is the definition of cost metrics. In our examination of this workload, we considered three metrics: administrative personnel-hours, number of I/O operations, and number of auxiliary machine I/O operations, all of which can be represented using the cost metric $\langle \mathbb{R}^+, +, < \rangle$.

We use administrative personnel-hours because it is relatively easy to quantify and reflects the amount of work that must be done by well-paid individuals to support the functionality of the access control scheme. Tasks such as managing system-wide policies, granting clearances, and creating and deleting users are all tasks that require work of administrator users. We use I/O operations as a means of measuring the computational performance as systems get very large. This metric becomes relevant, e.g., in the context of mappings that must manipulate complex state when executing workload operations. Lastly, we investigate auxiliary machine I/O operations to capture the effect on expanding the trusted computing base of an access control scheme. Implementing an AM atop an existing access control system would require a trusted channel for communication between the original system and the AM, a cost our final metric attempts to capture.

2) Cost Labeling Functions: The next step in cost analysis is the definition of cost labeling functions for each candidate scheme. In this case study, we generalize our cost labeling functions over state transition commands and query forms. Since our cost analysis will be carried out via simulation, rather than define the cost for each and every possible transition, we define a probability distribution of costs for each command and query form. During our simulation, we sample from these distributions as commands or queries are executed to derive per-instance costs.

We now describe cost labeling functions $\ell_A^{\mathcal{X}}$, $\ell_I^{\mathcal{X}}$, and $\ell_M^{\mathcal{X}}$ for a scheme $\mathcal{X}$ that assign costs for administrative personnel-hours, I/O operations, and AM I/O operations, respectively. Here, $\mathcal{T}$ represent the SD3-T scheme (Definition 12), $\mathcal{B}$ represent the Bell-LaPadula scheme (Definition 13), and $\mathcal{R}$ represent the role-based access control scheme (Definition 16). Table IIa shows a table of costs for each command and query in SD3-T. Table IIb and Table IIc show similar tables for BLP and RBAC, respectively. We have omitted commands that are never executed under the corresponding mapping from workload $W_{\mathcal{M}}$ (e.g., BLP commands for manipulating the lattice).[4] Some costs are relative to the size of certain components in the state, and thus our simulation must keep track of the sizes of these components as the systems evolve. Some costs are constant, while others are log-normal distributions randomly sampled by our simulator.

Our cost labeling function for I/O operations measures the cost of accessing and updating the elements of the data structures underlying the access control scheme. For the sake of simplicity, we assume constant-time access to all data elements, ensuring that we need only count the number of elements we need to read or write. For SD3-T, each command and query reads or writes a single element in one of the data structures of the access control scheme, with the exception of the access queries. We cannot generalize which policy sentences are needed without knowledge of the underlying policy language, and thus must assume all sentences are loaded. However, we can be sure that only user/attribute pairs containing the inquiring user and document/tag pairs containing the document in question will be needed. Assuming the number of sentences in the total policy will greatly outnumber the number of attributes of a single user or the number of tags of a single document, we count only the number of policy sentences. Costs are similar for BLP, with all operations costing one, with the exception of `get_`x, which costs 3 due to the need to verify both mandatory and discretionary components, costing 1 each, before changing b, the final access. The access questions in RBAC, similarly, require verifying whether each role grants the access, a procedure that costs 2 per role, plus looking up the role itself.

Figure 6 shows plots of each log-normal distribution used in Table II. Commands like `assign_tags` use the distribution $\ln \mathcal{N}\left(-2.2, 1^2\right)$ because they have a very high probability of being cheap (e.g., many tags are simple metadata like creation time, subject matter, etc.). However, occasionally these commands can be expensive (e.g., a tag could indicate security classification like "top secret"). For this distribution, over 79% of actions require less than 15 minutes of administrator

[4]Full details of these mappings are presented in Appendix C.

TABLE II: Cost labeling tables for our candidate schemes.

(a) $\mathcal{T}$, SD3-T

action, a	$\ell_A^{\mathcal{T}}(a)$	$\ell_I^{\mathcal{T}}(a)$	$\ell_M^{\mathcal{T}}(a)$
create_document	0	1	0
destroy_document	0	1	0
assign_tags	$\ln\mathcal{N}(-2.2, 1^2)$	1	0
revoke_tags	$\ln\mathcal{N}(-2.2, 1^2)$	1	0
create_user	0	1	0
destroy_user	0	1	0
assign_attributes	$\ln\mathcal{N}(0.8, 1^2)$	1	0
revoke_attributes	$\ln\mathcal{N}(-1.5, 1^2)$	1	0
transform_policy	$\ln\mathcal{N}(1.1, 1.1^2)$	1	0
u_exist?	0	1	0
attribute?	0	1	0
tag?	0	1	0
access?	0	$\|P\|$	0

(b) Augmented BLP scheme $\mathcal{B} \circ \mathcal{N}$

action, a	$\ell_A^{\mathcal{B}}(a)$	$\ell_I^{\mathcal{B}}(a)$	$\ell_M^{\mathcal{B}}(a)$
create_object	0	1	0
destroy_object	0	1	0
create_subject	0	1	0
destroy_subject	0	1	0
get_x	0	3	0
release_x	0	1	0
s_exist?	0	1	0
access?	0	1	0
assign_tags	$\ln\mathcal{N}(-2.2, 1^2)$	1	1
revoke_tags	$\ln\mathcal{N}(-2.2, 1^2)$	1	1
assign_attributes	$\ln\mathcal{N}(0.8, 1^2)$	1	1
revoke_attributes	$\ln\mathcal{N}(-1.5, 1^2)$	1	1
transform_policy	$\ln\mathcal{N}(1.1, 1.1^2)$	1	1
attribute?	0	1	1
tag?	0	1	1

(c) Augmented RBAC scheme $\mathcal{R} \circ \mathcal{O}$

action, a	$\ell_A^{\mathcal{R}}(a)$	$\ell_I^{\mathcal{R}}(a)$	$\ell_M^{\mathcal{R}}(a)$
create_object	0	1	0
destroy_object	0	1	0
grant_role	$\ln\mathcal{N}(1.5, 1^2)$	1	0
revoke_role	$\ln\mathcal{N}(-1.5, 1^2)$	1	0
grant_permission	$\ln\mathcal{N}(-2.2, 1^2)$	1	0
revoke_permission	$\ln\mathcal{N}(-2.2, 1^2)$	1	0
s_exist?	0	1	0
role?	0	1	0
access?	0	$3 \cdot \|R\|$	0
assign_tags	$\ln\mathcal{N}(-2.2, 1^2)$	1	1
revoke_tags	$\ln\mathcal{N}(-2.2, 1^2)$	1	1
transform_policy	$\ln\mathcal{N}(1.1, 1.1^2)$	1	1
tag?	0	1	1

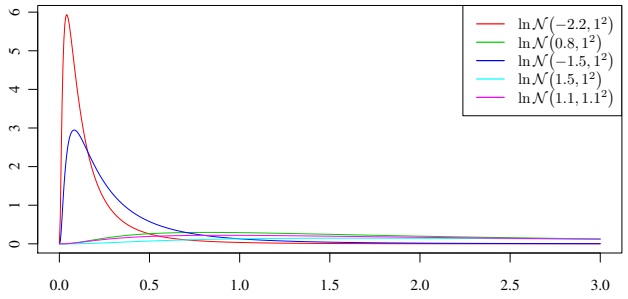


Fig. 6: Log-normal curves used in cost labeling functions.

time, and over 93% require less than 30 minutes. Commands that use the distribution $\ln\mathcal{N}(0.8, 1^2)$, on the other hand, are often expensive. The arithmetic mean of this distribution is almost 50 minutes of administrator time, with more than half requiring more than 2 hours. This category includes `assign_attributes`, since an additional attribute can often grant a wide range of accesses, and thus is likely to be heavily audited.

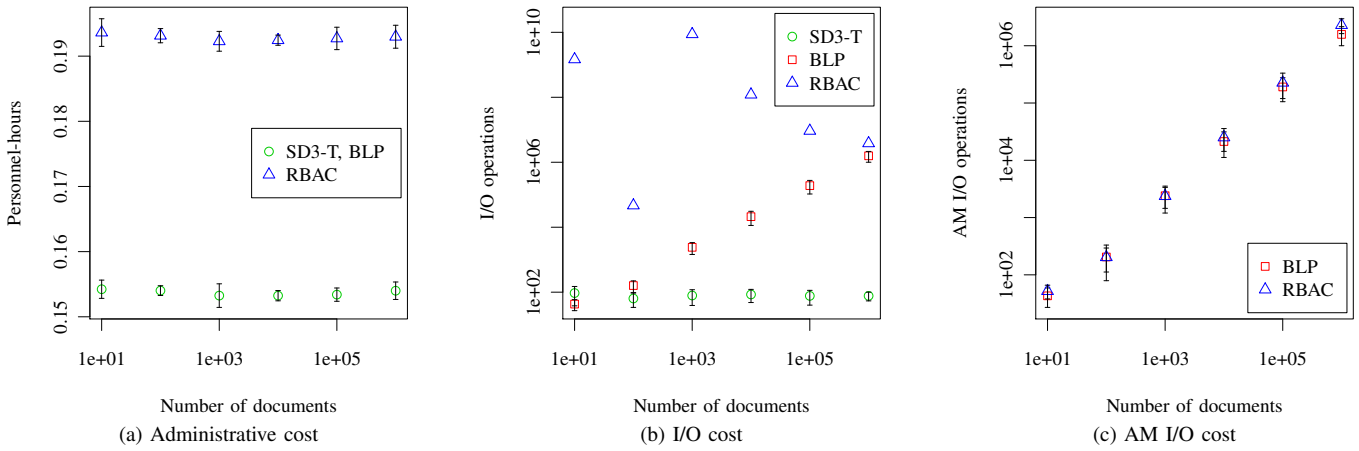
As with our invocation distributions, we do not claim that our cost labeling functions for administrative cost are perfectly accurate, but rather are reasonable estimates given the information available. Given that the number of personnel-hours the military spends on each access control task is not public information, we use reasonable values for the purpose of demonstrating our framework. For example, we know that roles are typically employment positions, such as junior security officer [21], while attributes in SD3-T encompass even simple metadata, such as hometown and year hired. Thus, on average, executing `grant_role` in RBAC will be more expensive administratively than executing `assign_attributes` in SD3-T, an assumption which is modeled by the increased scale parameter in the latter.

3) Simulation Results: Our simulator allows various parameters of the initial state—e.g., number of users, number of documents, and size of policy—to be varied prior to the start of a given run. Then, the invocation graph $I_{\mathcal{M}}$ is traversed as described in Section IV-C, tracking the costs accrued for each metric. We chose a variety of start states, then ran our simulator on each until 1,000,000 actions (commands and queries) had been executed. Figures 9a, 9b, and 9c show the average costs per operation in terms of administrator personnel-hours, I/O operations, and AM I/O operations for typical executions of our simulator. As shown in these figures, our simulation reaches steady state by 200,000 actions, much earlier than the 1,000,000 action point that was chosen for our simulation length. Figures 7 and 8 show charts for each of our cost metrics as we vary the number of documents and users, respectively, in the simulation's initial state. These charts show averages over five runs. Error bars, where shown, indicate one standard deviation in each direction.

From the point of view of the administrative personnel-hours metric, SD3-T and the augmented BLP implementation are equally efficient, while the augmented RBAC implementation was consistently around 25% more expensive, as shown in both Figures 7a and 8a. These values do not vary with either the number of documents (as shown in Figure 7a) or number of users (as shown in Figure 8a). Thus, from the perspective of this metric, RBAC loses in every case. SD3-T and BLP are equal in this metric due to the similarity in their implementations: all of the expensive actions in our augmented BLP scheme are tag-based actions within the AM.

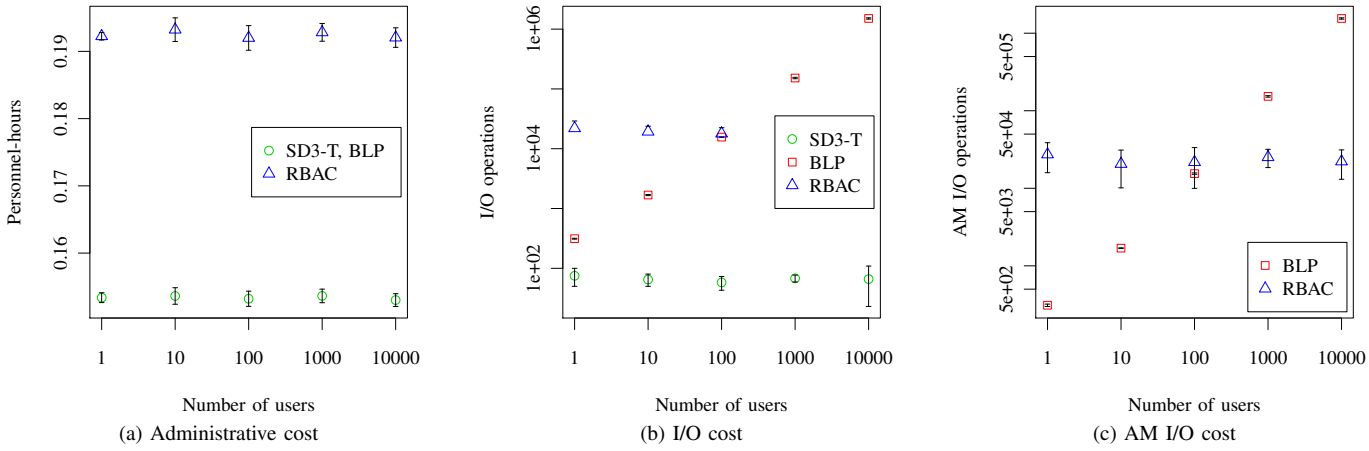
In terms of I/O operations, SD3-T is nearly always the most efficient. The cost of executing our workload remained almost constant with respect to both number of documents and number of users, as seen in Figures 7b and 8b, respectively. In contrast, BLP's I/O costs rose linearly with both number of documents and number of users, remaining lower than SD3-T's costs only in instances with very few documents, as seen in Figure 7b. RBAC's costs were even higher, as the mapping from $W_{\mathcal{M}}$ to $\mathcal{R} \circ \mathcal{O}$ relies on a number of roles exponential in the number of SD3-T attributes.[5] Thus, as seen in Figure 8b RBAC's I/O cost grew exponentially with number of users, quickly growing in many orders of magnitude once more than 100 users were created. Figure 7b shows that, with regards to number of documents, RBAC's costs are almost arbitrary, depending on the growth in number of users for any given execution, and thus randomness factors greatly overwhelmed

[5]Investigating the optimality of this mapping—and mappings in general—is subject of future work (see Section VI).



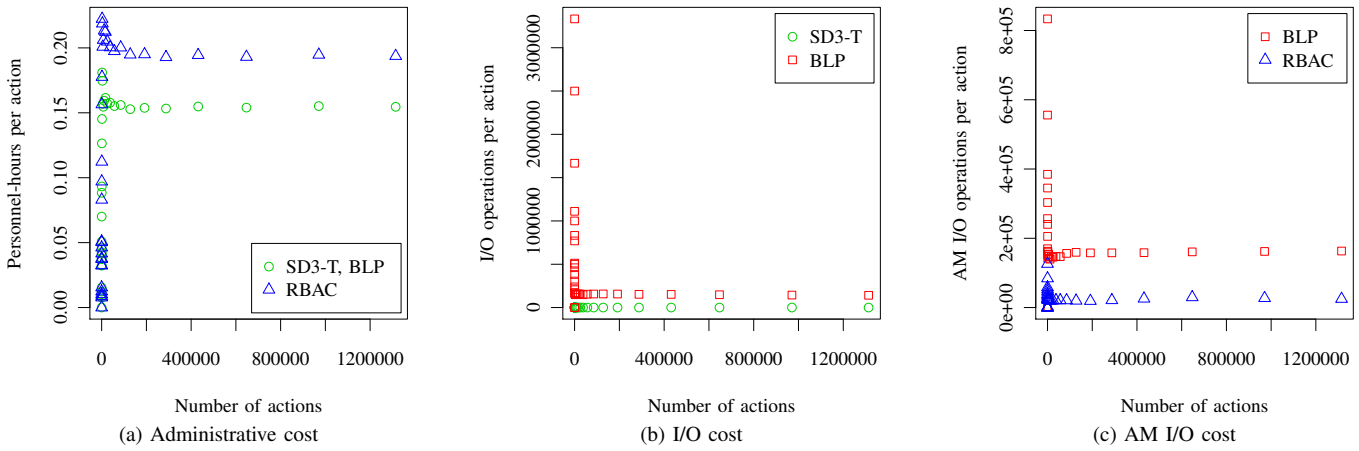
(a) Administrative cost (b) I/O cost (c) AM I/O cost

Fig. 7: Change in average cost per action as number of documents is varied. Averages over 1,000,000 actions from initial state $|P| = 150$, $|U| = 1,000$.



(a) Administrative cost (b) I/O cost (c) AM I/O cost

Fig. 8: Change in average cost per action as number of users is varied. Averages over 1,000,000 actions from initial state $|P| = 150$, $|D| = 10,000$.



(a) Administrative cost (b) I/O cost (c) AM I/O cost

Fig. 9: Examples of typical runs of our simulator, showing that the system has hit steady state much earlier than 1,000,000 actions. Initial state $|P| = 150$, $|D| = 10,000$, $|U| = 1,000$.

experimental factors.

Our final metric considers I/O costs associated with managing and querying auxiliary machine state. Since SD3-T is sufficient to execute our workload without an AM, its costs in this category are always zero. However, an interesting relationship is seen between the augmented implementations of BLP and RBAC. Previously, BLP seemed much more feasible than RBAC due to the latter's managing a number of roles exponential in the number of the workload's attributes. However, when considering only AM state, RBAC's costs were competitive with—in some cases, even lower than—those of BLP. Figure 7c shows both schemes exhibiting roughly linear growth in AM I/O costs as the number of documents increased, though BLP's costs were a bit lower. As the number of users increased, however, Figure 8c shows BLP's costs growing linearly while RBAC's remain nearly constant. Thus, for systems with greater than about 100 users, RBAC's AM I/O costs were lower than BLP's.

4) Summary: Our results show that a purely tag-based approach like SD3-T is the best choice for implementing our workload in almost all situations and metrics. This is not surprising, as our source for our workload [16] describes the military's present-day requirements almost exactly as a tag-based system. It is enlightening to note, however, that there are situations where SD3-T was not the most efficient scheme (e.g., considering I/O cost in a state with very few users, as shown in Figure 8b). This shows that the scheme that seems to best parameterize a workload is not always the most efficient way to implement that workload. Between BLP and RBAC, two established access control schemes, BLP is typically the better choice, though there are also situations where RBAC's costs are lower (e.g., considering AM I/O costs in a state with a large number of users, as seen in Figure 8c).

VI. Discussion

In this section, we will discuss limitations of our analysis framework and potential areas for future work.

A. Abstraction Levels

A recent NIST report [9] distinguishes three abstractions of an access control system: access control policies, models, and mechanisms. A policy describes the high-level requirements of a system, a model is the mathematical representation of the system, and a mechanism is the implementation that is actually deployed to enforce the policy. While most of this work is done at the intermediate abstraction stage (models), it would be preferable to have a deterministic way to transform the most abstract notion of access control into the more concrete representation that we use for our workloads, as previously mentioned in Section III-B. This would make it easier for administrators to represent the system requirements appropriately, without relying on a "correct" translation from policy to model.

B. Mapping Non-Existence

Recall from Section IV that access control auxiliary machines can be used to augment systems that are not expressive enough to build the mapping needed to satisfy the operational requirements of a workload. We present techniques for proving the non-existence of certain types of mappings in our companion paper [15]. In the specific case of state-matching reductions, there are examples of non-existence proofs [5], but these are typically harder to produce than constructive existence proofs. Thus, we often resort to informal arguments describing why we do not think there exists a state-matching reduction, as was done in Example 4. Ideally, in all cases where there does not exist the desired mapping from the workload to the candidate scheme, it would be possible to formally prove this, since such proofs give higher confidence in the necessity of using auxiliary machines during the evaluation process.

C. Mapping Optimality

The constructive nature of the reductions used to demonstrate an access control scheme's ability to implement a given workload's operational requirements lead quite naturally to the cost analysis of the scheme, as costs can be assigned to each transition taken within the mapping used by the state-matching reduction. Given an access control scheme $\mathcal{S}$ and a workload W, we therefore carry out the cost analysis of a *particular* mapping from $\mathcal{S}$ to W, rather than the *best* mapping of $\mathcal{S}$ to W. It would be useful to develop techniques for proving the optimality of a given mapping with respect to a particular cost metric. This would enable analysts to make strong claims about the (sub-)optimality of an access control scheme for a given workload without needing to justify or defend the mappings used during their analysis.

D. Types of Mappings

Recall that we base our definition of safe implementation (Definition 8) on the state-matching reduction, the strongest type of mapping studied in previous work [5]. In this paper, we limit our discussion of this choice, and instead refer interested readers to our companion paper [15]. In that paper, we focus on exploring different types of implementations for access control workloads. We present alternative logical formalizations of access control workloads and systems, and discuss methods for determining whether a system can correctly implement a workload. We also describe classifications of implementations—homomorphic, non-interfering, safe, and succinct—that offer additional guarantees beyond correctness. These alternate forms of implementation allow the analyst to choose specifically what types of properties are relevant to the particular workload being analyzed. The cost analysis techniques that we explored in this paper can apply just as easily to these other notions of implementation.

VII. Conclusions

To evaluate the suitability of existing access control schemes for a particular application, we introduced the notion of an access control workload and developed a framework for analyzing the cost of implementing that workload within various candidate schemes. A workload is a structure containing a state transition system that models the access control related

actions that must be supported within the application of interest, as well as a Markov model specifying how these actions are expected to be invoked in practice. Existing reduction techniques [5] are then used to determine if candidate schemes are expressive enough to safely implement the workload. If necessary, candidate schemes can be augmented with access control auxiliary machines—a novel contribution of our work—to enhance their expressive power without sacrificing safety in an effort to facilitate these types of reductions. The constructive nature of these reductions can be combined with the invocation portion of a workload description and analyst-defined cost metrics to carry out cost analysis that is performed either via expected value computation or by simulation. To illustrate this framework, we presented a detailed case study using a potential modern military workload derived from defense contractor technical reports. Within this context, we evaluated several candidate schemes' suitability for implementing that workload relative to a vector of cost metrics assessing data manipulation overheads, auxiliary machine complexity, and administrative user burdens. While some calculated costs validated intuition, there were instances in which the candidate scheme most similar to the workload's operational component was found to be strictly less suitable than a less similar scheme.

REFERENCES

[1] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Communications of the ACM*, vol. 19, no. 8, pp. 461–471, Aug 1976.

[2] R. J. Lipton and L. Snyder, "A linear time algorithm for deciding subject security," *Journal of the ACM*, vol. 24, no. 3, pp. 455–464, Jul 1977.

[3] P. Ammann, R. J. Lipton, and R. S. Sandhu, "The expressive power of multi-parent creation in monotonic access control models," *Journal of Computer Security*, vol. 4, no. 2/3, pp. 149–166, 1996.

[4] A. Chander, J. C. Mitchell, and D. Dean, "A state-transition model of trust management and access control," in *14th IEEE Computer Security Foundations Workshop*, Jun 2001, pp. 27–43.

[5] M. V. Tripunitara and N. Li, "A theory for comparing the expressive power of access control models," *Journal of Computer Security*, vol. 15, no. 2, pp. 231–272, 2007.

[6] S. Osborne, R. Sandhu, and Q. Munawer, "Configuring role-based access control to enforce mandatory and discretionary access control policies," *ACM Transactions on Information and System Security*, vol. 3, no. 2, pp. 85–106, May 2000.

[7] R. Sandhu, "Expressive power of the schematic protection model," *Journal of Computer Security*, vol. 1, no. 1, pp. 59–98, 1992.

[8] R. Sandhu and S. Ganta, "On testing for absence of rights in access control models," in *6th IEEE Computer Security Foundations Workshop*, Jun 1993, pp. 109–118.

[9] V. C. Hu, D. F. Ferraiolo, and D. R. Kuhn, *Assessment of Access Control Systems*. National Institute of Standards and Technology, 2006.

[10] "Microsoft Web Protection Library," http://wpl.codeplex.com.

[11] "Spring Security," http://static.springsource.org/spring-security/site/.

[12] "Apache Shiro," http://shiro.apache.org.

[13] "Horizontal integration: Broader access models for realizing information dominance," MITRE Corporation JASON Program Office, Tech. Rep. JSR-04-13, 2004.

[14] S. Sinclair, S. W. Smith, S. Trudeau, M. E. Johnson, and A. Portera, "Information risk in financial institutions: Field study and research roadmap," in *FinanceCom*, Dec 2007, pp. 165–180.

[15] Blinded, "Application-sensitive access control evaluation: Logical foundations (extended version)," http://dl.dropbox.com/u/38851982/access_control_logic_extended.pdf, Nov 2011.

[16] U.S. Air Force Scientific Advisory Board, "Networking to enable coalition operations," MITRE Corporation, Tech. Rep., 2004.

[17] N. Li, J. C. Mitchell, and W. H. Winsborough, "Beyond proof-of-compliance: security analysis in trust management," *Journal of the ACM*, vol. 52, no. 3, pp. 474–514, May 2005.

[18] C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman, "Linux security modules: General security support for the linux kernel," in *11th USENIX Security Symposium*, Aug 2002, pp. 17–31.

[19] Q. Wang, N. Li, and H. Chen, "On the security of delegation in access control systems," in *13th European Symposium on Research in Computer Security*, Oct 2008, pp. 317–332.

[20] T. Jim, "SD3: A trust management system with certified evaluation," in *IEEE Symposium on Security and Privacy*, May 2001, pp. 106–115.

[21] R. S. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 model for role-based administration of roles," *ACM Transactions on Information and System Security*, vol. 2, no. 1, pp. 105–135, Feb 1999.

APPENDIX A
FORMAL DESCRIPTION OF ACCESS CONTROL SCHEMES

Definition 12 (SD3 with Tag-Based Administrative Model, $\mathcal{T}$): The *SD3 scheme with tag-based administrative model*, or SD3-T, is defined as the state transition system $\mathcal{T} = \langle \Gamma^{\mathcal{T}}, \Psi^{\mathcal{T}}, Q^{\mathcal{T}}, \vdash^{\mathcal{T}} \rangle$. Each SD3-T state $\gamma^{\mathcal{T}} \in \Gamma^{\mathcal{T}}$ is defined by $\langle S, O, P \rangle$, where S is the set of subjects, O is the set of objects, and P is the set of policy sentences, written in SD3. $\Psi^{\mathcal{T}}$ is defined by the commands `create_object(`s`, `o`)`, `destroy_object(`s`, `o`)`, `create_subject(`s`, `s'`)`, `destroy_subject(`s`, `s'`)`, `transform_policy(`u`, `P_+`, `P_-`)`. $Q^{\mathcal{T}}$ includes all queries of the following forms: (1) "Does subject s exist?", (2) "Is policy sentence p in the policy?", and (3) "Does subject s have access i to object o?". $\vdash^{\mathcal{T}}$ is defined as follows for queries of each of the forms above: (1) $true$ if and only if $s \in S$, (2) $true$ if and only if $p \in P$, (3) $true$ if and only if P evaluates, under the SD3 resolution procedure, to a fixed point in which subject s has access i to object o. ♦

Definition 13 (Bell-LaPadula Access Control Scheme, $\mathcal{B}$): In the *Bell-LaPadula access control scheme*, we assume the existence of sets $A = \{r, w, a, e\}$, the set of access attributes, corresponding to read, write, append, and execute, respectively; C, the set of clearances or classifications; and P, the set of compartments. These components are not defined as part of the state, as they do not change. The *security level* of a subject or object is an element of $C \times 2^P$. The BLP scheme is then defined as the state transition system $\mathcal{B} = \langle \Gamma^{\mathcal{B}}, \Psi^{\mathcal{B}}, Q^{\mathcal{B}}, \vdash^{\mathcal{B}} \rangle$. $\Gamma^{\mathcal{B}}$ is defined by $\langle S, O, b, M, f_S, f_O, f_C \rangle$, where S is the set of subjects, O is the set of objects, $b \subseteq S \times O \times A$ is the current access set,[6] $M : S \times O \to 2^A$ is the access matrix (the discretionary component of BLP), $f_S : S \to C \times 2^P$ is the security level function of subjects, $f_O : O \to C \times 2^P$ is the security level function of objects, and $f_C : S \to C \times 2^P$ is the *current* security level function of subjects.[7] $\Psi^{\mathcal{B}}$ is defined using the following commands. Commands with names ending in x_i exist for each $x_i \in A$.

```
command  get-read(s, o)
    if r ∈ M[s, o] ∧ f_S(s) ⊀ f_O(o) ∧
    (s is trusted ∨ f_C(s) ⊀ f_O(o))
        b = b ∪ {⟨s, o, r⟩}

command  get-append(s, o)
    if a ∈ M[s, o] ∧
    (s is trusted ∨ f_O(o) ⊀ f_C(s))
        b = b ∪ {⟨s, o, a⟩}

command  get-execute(s, o)
    if e ∈ M[s, o]
        b = b ∪ {⟨s, o, e⟩}

command  get-write(s, o)
    if w ∈ M[s, o] ∧ f_S(s) ⊀ f_O(o) ∧
    (s is trusted ∨ f_C(s) = f_O(o))
        b = b ∪ {⟨s, o, w⟩}

command  release-x_i(s, o)
    b = b − {⟨s, o, x_i⟩}

command  give-x_i(s, s', o)
    if w ∈ M[s, o]
        M[s', o] = M[s', o] ∪ {x_i}

command  rescind-x_i(s, s', o)
    if w ∈ M[s, o]
        b = b − {⟨s', o, x_i⟩}
        M[s', o] = M[s', o] − {x_i}

command  create-object(s, o, l)
```

[6] b contains the tuple $\langle s, o, a \rangle$ for each subject s that currently has access a to object o. This set describes the access that are currently "open", and acts as a set of tokens for permissions so the system does not have to check the conditions for access each time an object is touched.

[7] f_C is used to temporarily utilize a lower security level to work around the inability to write "below" one's security level.

$f_O = f_O \cup \{\langle o, l\rangle\}$
$O = O \cup \{o\}$

command delete-object(s, o)
if $w \in M[s, o]$
for all s, x
$b = b - \{\langle s, o, x\rangle\}$
$M[s, o] = \{\}$
$O = O - \{o\}$

command create_subject(s, s', l)
if s is trusted to manage users
$f_S = f_S \cup \{\langle s', l\rangle\}$
$S = S \cup \{s'\}$

command destroy_subject(s, s')
if s is trusted to manage users
for all o, x
$b = b - \{\langle s', o, x\rangle\}$
$M[s', o] = \{\}$
$S = S - \{s\}$

command change-subject-current-level(s, l)
if $f_S \asymp l \wedge$
(s is trusted $\vee$ *-property would be satisfied)
$f_C(s) = l$

command change_subject_level(s, s', l)
if the following conditions are true:
- s is trusted to manage users
- for all o such that $\langle s', o, r\rangle \in b$
  $\vee \langle s', o, w\rangle \in b$, $f_C(s_i) \asymp l$
- the change would not violate *-property
  $f_C(s') = l$
  $f_S(s') = l$

command change-object-level(s, o, l)
if the following conditions are true:
- s is trusted $\wedge$ $f_C(s) \asymp f_O(o)$
  $\vee$ $f_C(s) \asymp l \wedge l \asymp f_O(o)$
- for all s_i such that $\langle s_i, o, r\rangle \in b$
  $\vee \langle s_i, o, w\rangle \in b$, $f_C(s_i) \asymp l$
- the change will not violate *-property
- s is allowed to change security level of o
  $f_O(o) = l$

$Q^{\mathcal{B}}$ includes all queries of the following forms: (1) "Does subject s exist?", and (2) "Does subject s have access a to object o?". $\vdash^{\mathcal{B}}$ is defined as follows for queries of each of the forms above: (1) $true$ if and only if $s \in S$, and (2) $true$ if and only if $\langle s, o, a\rangle \in b$. ♦

Definition 14 (Role-Based Access Control Scheme): In the *role-based access control scheme*, we assume the existence of R, the set of roles. The scheme is defined as the state transition system $\mathcal{R} = \langle \Gamma^{\mathcal{R}}, \Psi^{\mathcal{R}}, Q^{\mathcal{R}}, \vdash^{\mathcal{R}}\rangle$. Each state $\gamma^{\mathcal{R}} \in \Gamma^{\mathcal{R}}$ is defined by $\langle S, P, SR, RP\rangle$, where S is the set of subjects, P is the set of permissions, $SR \subseteq S \times R$ is the subject-role relation, and $RP \subseteq R \times P$ is the role-permission relation. $\Psi^{\mathcal{R}}$ is defined by the commands create_permission(s, p), destroy_permission(s, p), create_subject(s, s'), destroy_subject(s, s'), grant_role(s, s', r), revoke_role(s, s', r), grant_permission(s, r, p), and revoke_permission(s, r, p). $Q^{\mathcal{R}}$ includes all queries of the following forms: (1) "Does subject s exist?", (2) "Does subject s belong to role r?", and

(3) "Does subject s have permission p?". $\vdash^{\mathcal{R}}$ is defined as follows for queries of each of the forms above: (1) $true$ if and only if $s \in S$, (2) $true$ if and only if $\langle s, r\rangle \in SR$, and (3) $true$ if and only if $\exists r \in R : \langle s, r\rangle \in SR \wedge \langle r, p\rangle \in RP$. ♦

APPENDIX B
FORMAL DESCRIPTION OF ACCESS CONTROL AUXILIARY MACHINES

Definition 15 (Tag-based Auxiliary Machine for Bell-LaPadula, $\mathcal{N}$): The *tag-based auxiliary machine for the Bell-LaPadula scheme* is defined as the state transition system $\mathcal{N} = \langle \Gamma^{\mathcal{N}}, \Psi^{\mathcal{N}}, Q^{\mathcal{N}}, \vdash^{\mathcal{N}} \rangle$. Each state $\gamma^{\mathcal{N}} \in \Gamma^{\mathcal{N}}$ is defined by $\langle P, SA, OT\rangle$, where P is the set of policy sentences, $SA \subseteq S \times At$ is the subject-attribute relation,[8] and $OT \subseteq O \times T$ is the object-tag relation.[9] $\Psi^{\mathcal{N}}$ is defined by the commands `assign_tags(`s`, `o`, `T_1`)`, `revoke_tags(`s`, `o`, `T_1`)`, `assign_attributes(`s`, `s'`, `At_1`)`, `revoke_attributes(`s`, `s'`, `At_1`)`, and `transform_policy(`s`, `P_+`, `P_-`)`. $Q^{\mathcal{N}}$ includes all queries of the following forms: (1) "Does subject s have attribute a?", and (2) "Does object o have tag t?". $\vdash^{\mathcal{N}}$ is defined as follows for the queries of each of the forms above: (1) $true$ if and only if $\langle s, a\rangle \in SA$, and (2) $true$ if and only if $\langle o, t\rangle \in OT$. ♦

This AM allows us to extend the BLP scheme to maintain extra state and therefore allow us to build a state-matching reduction from $W_{\mathcal{M}}$ to a form of BLP. The mapping from $W_{\mathcal{M}}$ to this augmented BLP sets the lattice components of the BLP scheme to trivial states. All subjects and objects have the same clearance/classification, and all have empty category sets. This effectively nullifies the mandatory component of the scheme, and leaves it to be managed by b, the current access set. The auxiliary machine state, then, maintains all of the useful access control information. The queries of the form, "Does subject s have access a to object o?" (indeed, all queries) must be answered by the original procedure, in this case by checking the current access set. To maintain the answers to the access question, then, the mapping must modify b to reflect the accesses that should be allowed. Thus, after changing the policy, tags, or attributes, b must also be changed, since this data is stored within the AM and is not allowed to affect the answers to any of the queries from the original scheme. Thus, for example, changing the tags associated with an object requires iterating over all subjects to identify any whose access to that object has changed, and edit b appropriately. Similarly, when altering the policy, all subject/object pairs are verified.

Definition 16 (Object-Tag Auxiliary Machine for RBAC, $\mathcal{O}$): The *object-tag auxiliary machine for the role-based access control scheme* is defined as the state transition system $\mathcal{O} = \langle \Gamma^{\mathcal{O}}, \Psi^{\mathcal{O}}, Q^{\mathcal{O}}, \vdash^{\mathcal{O}} \rangle$.

Each state $\gamma^{\mathcal{O}} \in \Gamma^{\mathcal{O}}$ is defined by $\langle OT, L\rangle$, where $OT \subseteq O \times T$ is the object-tag relation[10], and L is the set of policy sentences[11]. $\Psi^{\mathcal{O}}$ is defined by the commands `revoke_tags(`s`, `o`, `T_1`)`, `assign_tags(`s`, `o`, `T_1`)`, and `transform_policy(`s`, `L_+`, `L_-`)`. $Q^{\mathcal{O}}$ includes all queries of the form "Does object o have tag t?". $\vdash^{\mathcal{O}}$ is defined as $true$ if and only if $\langle o, t\rangle \in OT$. ♦

This AM allows us to extend RBAC in a way that adds object-tag information, enabling a state-matching reduction from $W_{\mathcal{M}}$ to this augmented RBAC. The mapping from $W_{\mathcal{M}}$ to this augmented RBAC uses the subject-role relation to maintain the user-attribute relation information from $W_{\mathcal{M}}$, and adjusts the role-permission relation to match in such a way that queries of the form, "Does subject s have permission p?" are answered correctly within the standard RBAC policy. The document-tag information is maintained within the AM. Each subject in RBAC is roles corresponding to the *power set* of that user's attributes. Thus, if user u_1 has attributes a_1, a_2 and a_4 in $W_{\mathcal{M}}$, the mapping will map this state to one in RBAC where subject s_1 corresponding to u_1 has roles $\{a_1, a_2, a_4, a_1a_2, a_1a_4, a_2a_4, a_1a_2a_4\}$. The role-permission relation will need to be rebuilt based on any changes to the AM state, and will give *sets of attributes* their appropriate permissions. For example, if the $W_{\mathcal{M}}$ state gives users with attributes a_1 and a_3 access w to document d_2, then in RBAC RP should contain $\langle a_1a_3, p_2\rangle$, where p_2 is the permission corresponding to access w to document d_2. Thus, the number of roles is exponential in the number of attributes. Answering queries of the form, "Does subject s have permission p?" requires iterating over all roles, and thus takes time exponential in the number of attributes in $W_{\mathcal{M}}$.

APPENDIX C
PROOFS

A. Proof of Theorem 1

First, we present several requisite definitions.

Definition 17 (Abelian Semigroup): An *abelian semigroup*, $\mathbf{S} = \langle S, +\rangle$, is a set, S, together with a binary operation, $+$, that satisfies the following properties.

1) **Closure** $\forall a, b \in S, a + b \in S$
2) **Associativity** $\forall a, b, c \in S, (a + b) + c = a + (b + c)$

[8]Here, S is the set of subjects from BLP and At is external to the state as in Definition 11.
[9]Here O is the set of objects from BLP and T is external to the state as in Definition 11.
[10]Here, O and T are external to the state as in Definition 11.
[11]The set of policy sentences is denoted L rather than our typical P to avoid collisions with RBAC's set of permissions.

3) **Commutativity** $\forall a, b \in S, a + b = b + a$ ♦

Definition 18 (Partially Ordered Set): A *partially ordered set*, $\mathbf{S} = \langle S, \leq \rangle$, is a set, S, together with a binary relation, $\leq$, that satisfies the following properties.

1) **Reflexivity** $\forall a \in S, a \leq a$
2) **Antisymmetry** $\forall a, b \in S, a \leq b \wedge b \leq a \Rightarrow a = b$
3) **Transitivity** $\forall a, b, c \in S, a \leq b \wedge b \leq c \Rightarrow a \leq c$ ♦

Definition 19 (Ordered Abelian Semigroup): An *ordered abelian semigroup*, $\mathbf{S} = \langle S, +, \leq \rangle$, is a set, S, together with a binary operator, $+$, and binary relation, $\leq$, that satisfies the following properties.

1) $\langle S, + \rangle$ is an abelian semigroup
2) $\langle S, \leq \rangle$ is a partially ordered set ♦

Now, we restate the definition of a vector of cost metrics.

Definition 5 (Vector of Cost Metrics): Given cost metrics $\mathbf{N}_1 = \langle N_1, +_1, \leq_1 \rangle$, $\mathbf{N}_2 = \langle N_2, +_2, \leq_2 \rangle$, ..., $\mathbf{N}_i = \langle N_i, +_i, \leq_i \rangle$, let $\mathbf{M} = \langle M, +_*, \leq_* \rangle$ be the *vector* of cost metrics $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$, where:

- $M = N_1 \times N_2 \times \cdots \times N_i$.
- Given $a_1, b_1 \in \mathbf{N}_1$, $a_2, b_2 \in \mathbf{N}_2$, ..., $a_i, b_i \in \mathbf{N}_i$, $\langle a_1, a_2, \ldots, a_i \rangle +_* \langle b_1, b_2, \ldots, b_i \rangle = \langle a_1 +_1 b_1, a_2 +_2 b_2, \ldots, a_i +_i b_i \rangle$.
- Given $a_1, b_1 \in \mathbf{N}_1$, $a_2, b_2 \in \mathbf{N}_2$, ..., $a_i, b_i \in \mathbf{N}_i$, $\langle a_1, a_2, \ldots, a_i \rangle \leq_* \langle b_1, b_2, \ldots, b_i \rangle$ if and only if $a_1 \leq_1 b_1 \wedge a_2 \leq_2 b_2 \wedge \ldots \wedge a_i \leq_i b_i$. ♦

Theorem 1: Given cost metrics $\mathbf{N}_1 = \langle N_1, +_1, \leq_1 \rangle$, $\mathbf{N}_2 = \langle N_2, +_2, \leq_2 \rangle$, ..., $\mathbf{N}_i = \langle N_i, +_i, \leq_i \rangle$, and their vector, $\mathbf{M} = \langle M, +_*, \leq_* \rangle$, $\mathbf{M}$ is a cost metric.

Proof: All of $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$ are cost metrics. Thus, they are all abelian semigroups, and thus are all closed, associative, and commutative. Using $\mathbf{N}_1$ as an example, this implies:

1) $\forall a, b \in N_1, a +_1 b \in N_1$
2) $\forall a, b, c \in N_1, (a +_1 b) +_1 c = a +_1 (b +_1 c)$
3) $\forall a, b \in N_1, a +_1 b = b +_1 a$

Let $A, B, C \in \mathbf{M}$. By the definition of vector of cost metrics,

$$A = \langle a_1, a_2, \ldots, a_i \rangle$$

where

$$a_1 \in \mathbf{N}_1, a_2 \in \mathbf{N}_2, \ldots, a_i \in \mathbf{N}_i$$

and similarly for B and C.

By the definition of vector,

$$A +_* B = \langle a_1 +_1 b_1, a_2 +_2 b_2, \ldots, a_i +_i b_i \rangle$$

By the closure of $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$,

$$a_1 +_1 b_1 \in \mathbf{N}_1, a_2 +_2 b_2 \in \mathbf{N}_2, \ldots, a_i +_i b_i \in \mathbf{N}_i$$

$$A +_* B \in \mathbf{M}$$

Thus, $\mathbf{M}$ satisfies the property of *closure*.

By the definition of vector,

$$(A +_* B) +_* C = \langle (a_1 +_1 b_1) +_1 c_1, (a_2 +_2 b_2) +_2 c_2, \ldots, (a_i +_i b_i) +_i c_i \rangle$$

By the associativity of $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$,

$$(A +_* B) +_* C = \langle a_1 +_1 (b_1 +_1 c_1), a_2 +_2 (b_2 +_2 c_2), \ldots, a_i +_i (b_i +_i c_i) \rangle$$

$$(A +_* B) +_* C = A +_* (B +_* C)$$

Thus, $\mathbf{M}$ satisfies the property of *associativity*.

By the definition of vector,

$$A +_* B = \langle a_1 +_1 b_1, a_2 +_2 b_2, \ldots, a_i +_i b_i \rangle$$

By the commutativity of $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$,

$$A +_* B = \langle b_1 +_1 a_1, b_2 +_2 a_2, \ldots, b_i +_i a_i \rangle$$

$$A +_* B = B +_* A$$

Thus, $\mathbf{M}$ satisfies the property of *commutativity*.

Since $\mathbf{M}$ satisfies closure, associativity, and commutativity, $\langle M, +_* \rangle$ is an abelian semigroup.

All of $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$ are cost metrics. Thus, they are all partially ordered sets, and thus are all reflexive, antisymmetric, and transitive. Using $\mathbf{N}_1$ as an example, this implies:

1) $\forall a \in N_1, a \leq_1 a$
2) $\forall a, b \in N_1, a \leq_1 b \wedge b \leq_1 a \Rightarrow a = b$
3) $\forall a, b, c \in N_1, a \leq_1 b \wedge b \leq_1 c \Rightarrow a \leq_1 c$

Let $A, B, C \in \mathbf{M}$. By the definition of vector of cost metrics,

$$A = \langle a_1, a_2, \ldots, a_i \rangle$$

where

$$a_1 \in \mathbf{N}_1, a_2 \in \mathbf{N}_2, \ldots, a_i \in \mathbf{N}_i$$

and similarly for B and C.

By the reflexivity of $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$,

$$a_1 \leq_1 a_1, a_2 \leq_2 a_2, \ldots, a_i \leq_i a_i$$

$$A \leq_* A$$

Thus, $\mathbf{M}$ satisfies the property of *reflexivity*.

Assume $A \leq_* B \wedge B \leq_* A$. By the definition of vector,

$$a_1 \leq_1 b_1 \wedge b_1 \leq_1 a_1 \wedge a_2 \leq_2 b_2 \wedge b_2 \leq_2 a_2 \wedge \ldots \wedge a_i \leq_i b_i \wedge b_i \leq_i a_i$$

By the antisymmetry of $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$,

$$a_1 = b_1 \wedge a_2 = b_2 \wedge \ldots \wedge a_i = b_i$$

$$A = B$$

$$A \leq_* B \wedge B \leq_* A \Rightarrow A = B$$

Thus, $\mathbf{M}$ satisfies the property of *antisymmetry*.

Assume $A \leq_* B \wedge B \leq_* C$. By the definition of vector,

$$a_1 \leq_1 b_1 \wedge b_1 \leq_1 c_1 \wedge a_2 \leq_2 b_2 \wedge b_2 \leq_2 c_2 \wedge \ldots \wedge a_i \leq_i b_i \wedge b_i \leq_i c_i$$

By the transitivity of $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_i$,

$$a_1 \leq_1 c_1 \wedge a_2 \leq_2 c_2 \wedge \ldots \wedge a_i \leq_i c_i$$

$$A \leq_* C$$

$$A \leq_* B \wedge B \leq_* C \Rightarrow A \leq_* C$$

Thus, $\mathbf{M}$ satisfies the property of *transitivity*.

Since $\mathbf{M}$ satisfies reflexivity, antisymmetry, and transitivity, $\langle M, \leq \rangle$ is a partially ordered set.

Since $\langle M, +_* \rangle$ is an abelian semigroup and $\langle M, \leq_* \rangle$ is a partially ordered set, $\mathbf{M} = \langle M, +_*, \leq_* \rangle$ is an ordered abelian semigroup.

Thus, $\mathbf{M}$ is a cost metric. □

B. Proof of Theorem 5

Theorem 5: Given access control scheme $\mathcal{S} = \langle \Gamma^{\mathcal{S}}, \Psi^{\mathcal{S}}, Q^{\mathcal{S}}, \vdash^{\mathcal{S}} \rangle$ and access control auxiliary machine $\mathcal{U} = \langle \Gamma^{\mathcal{U}}, \Psi^{\mathcal{U}}, Q^{\mathcal{U}}, \vdash^{\mathcal{U}} \rangle$, there exists a state-matching reduction from $\mathcal{S}$ to $\mathcal{S} \circ \mathcal{U}$.

Proof: By construction. Presented is a mapping, and proof that the mapping satisfies the two properties for it to be a state-matching reduction.

Let $\mathcal{Q} = \mathcal{S} \circ \mathcal{U}$.

The mapping, σ, needs to be able to map every $\langle \gamma, \psi \rangle$ in $\mathcal{S}$ to $\sigma(\langle \gamma, \psi \rangle) = \langle \gamma^{\mathcal{Q}}, \psi^{\mathcal{Q}} \rangle$ in $\mathcal{S} \circ \mathcal{U}$, as well as every q in $\mathcal{S}$ to $\sigma(q) = q^{\mathcal{Q}}$ in $\mathcal{S} \circ \mathcal{U}$.

Let $\sigma(\langle \gamma, \psi \rangle) = \langle \langle \gamma, \gamma_* \rangle, \psi \rangle$, where γ_* is an arbitrary auxiliary machine-state for AM $\mathcal{U}$. That is, let the AM component of the state be arbitrary, but maintain the original scheme component of the state. In addition, allow only those transitions that are valid in the original scheme. We can assume without loss of generality that this is possible—if it is not, simply allow whichever AM transitions are necessary; the mapping will not utilize them, and no queries from the original scheme will be affected by these transitions, by Definition 10.

Let $\sigma(q) = q$. That is, maintain all queries.

Let γ_0 be a start state in $\mathcal{S}$. Produce $\gamma_0^{\mathcal{Q}}$ in $\mathcal{S} \circ \mathcal{U}$ using σ. Given γ_k such that $\gamma_0 \overset{*}{\mapsto}_{\psi} \gamma_k$, we show that there exists $\gamma_k^{\mathcal{Q}}$ such that $\gamma_0^{\mathcal{Q}} \overset{*}{\mapsto}_{\psi^{\mathcal{Q}}} \gamma_k^{\mathcal{Q}}$ where, for all q, $\gamma_k^{\mathcal{Q}} \vdash q^{\mathcal{Q}}$ if and only if $\gamma_k \vdash q$.

Let $\gamma_k^{\mathcal{Q}} = \sigma(\gamma_k)$. By Definition 10, $\forall q \in Q_{\mathcal{S}}, \gamma_i \in \Gamma_{\mathcal{S}}, \gamma_i \vdash q$ if and only if, $\forall \gamma_i^{\mathcal{U}} \in \Gamma_{\mathcal{U}}, \langle \gamma_i, \gamma_i^{\mathcal{U}} \rangle \vdash q$. Thus, $\gamma_k^{\mathcal{Q}} \vdash q^{\mathcal{Q}}$ if and only if $\gamma_k \vdash q$.

Therefore, we have proven property (1) for the state-matching reduction.

We prove that property (2) for a state-matching reduction is satisfied by our mapping also by construction. Let $\gamma_0^{\mathcal{Q}}$ be the start-state in $\mathcal{S} \circ \mathcal{U}$ corresponding to γ_0, the start-state in $\mathcal{S}$. Then, if $\gamma_k^{\mathcal{Q}}$ is a state reachable from $\gamma_0^{\mathcal{Q}}$ and $q^{\mathcal{Q}}$ is a query in $\mathcal{S} \circ \mathcal{U}$ whose corresponding query in $\mathcal{S}$ is q, we construct γ_k, a state in $\mathcal{S}$ reachable from γ_0 as follows. If $\gamma_k^{\mathcal{Q}} = \langle \gamma_k^{\mathcal{S}}, \gamma_k^{\mathcal{U}} \rangle$, let $\gamma_k = \gamma_k^{\mathcal{S}}$. That is, discard the AM state and maintain the main scheme state. By Definition 10, by an argument similar to above, $\gamma_k \vdash q$ if and only if $\gamma_k^{\mathcal{Q}} \vdash q^{\mathcal{Q}}$.

Therefore, we've proven property (2) for state-matching reductions, and proven that our mapping σ is a state-matching reduction. □

C. Proof of Theorem 7

Theorem 7: SD3-T is expressive enough to safely implement the modern military workload.

Proof: Let $W_{\mathcal{M}} = \langle \mathcal{M}, I_{\mathcal{M}} \rangle$ be the modern military workload, and $\mathcal{T}$ the SD3-T scheme. SD3-T is expressive enough to safely implement the modern military workload if and only if there exists a state-matching reduction from scheme $\mathcal{M}$ to scheme $\mathcal{T}$.

We present a mapping from $\mathcal{M}$ to $\mathcal{T}$ and proof that the mapping satisfies the two properties for it to be a state-matching reduction.

The mapping, σ, needs to map every $\langle \gamma, \psi \rangle$ in MMW to $\sigma(\langle \gamma, \psi \rangle) = \langle \gamma^{\mathcal{T}}, \psi^{\mathcal{T}} \rangle$ in SD3-T, as well as every q in MMW to $\sigma(q) = q^{\mathcal{T}}$ in SD3-T.

$$
\begin{aligned}
\text{Let } \sigma(\gamma) = \gamma^{\mathcal{T}} = & \langle S_\gamma, O_\gamma, P_\gamma \rangle \text{ where} \\
S_\gamma = & U \\
O_\gamma = & D \\
P_\gamma = & \{\forall \langle u, a \rangle \in UA, \text{“}\texttt{ATTR(}u\texttt{, }a\texttt{) :- ;”}\} \cup \\
& \{\forall \langle d, t \rangle \in DT, \text{“}\texttt{TAG(}d\texttt{, }t\texttt{) :- ;”}\} \cup \\
& \{\forall p \in P, translate(p)\}
\end{aligned}
$$

Here, $translate(p)$ refers to translating the policy sentence p to an equivalent SD3-formatted sentence, e.g., “I`(U, D) :- TAG(D, T), ATTR(U, A);`”, which grants users with attribute A right I to documents with tag T.

If q is of the form, “Does user u exist?”, let $\sigma(q) = q^{\mathcal{T}}$ be the corresponding SD3-T query of the form, “Does subject s exist?”. If q is of the form, “Does user u have attribute a?”, let $q^{\mathcal{T}}$ = “Is policy sentence “`ATTR(`u`, `a`)`” in the policy?”. If q is of the form, “Does document d have tag t?”, let $q^{\mathcal{T}}$ = “Is policy sentence “`TAG(`d`, `t`)`” in the policy?”. If q is of the form, “Does user u have access i to document d?”, let $q^{\mathcal{T}}$ be the corresponding SD3-T query of the form, “Does subject s have access i to object o?”.

Let γ_0 be a start state in MMW. Produce $\gamma_0^{\mathcal{T}}$ in SD3-T using σ. Given γ_k such that $\gamma_0 \overset{*}{\mapsto}_{\psi} \gamma_k$, we show that there exists $\gamma_k^{\mathcal{T}}$ such that $\gamma_0^{\mathcal{T}} \overset{*}{\mapsto}_{\psi^{\mathcal{T}}} \gamma_k^{\mathcal{T}}$ where, for all q, $\gamma_k^{\mathcal{T}} \vdash q^{\mathcal{T}}$ if and only if $\gamma_k \vdash q$.

Consider the case where $\gamma_k = \gamma_0$, then let $\gamma_k^{\mathcal{T}} = \gamma_0^{\mathcal{T}}$. Now, consider each possible query q. If q is of the form, “Does user u exist?”, $q^{\mathcal{T}}$ is of the form “Does subject s exist?”. These queries are equal due to the direct mapping between the elements of

sets S_γ and U. If q is of the form, "Does user u have attribute a?", $q^\mathcal{T}$ is of the form, "Is policy sentence "`ATTR(u, a)`" in the policy?". Thus, $q^\mathcal{T}$ is true if and only if this sentence is added to P_γ. This sentence is added if and only if $\langle u, a\rangle \in UA$, or if and only if q is true as well. If q is of the form, "Does document d have tag t?", $q^\mathcal{T}$ is of the form, "Is policy sentence "`TAG(d, t)`" in the policy?". Thus, $q^\mathcal{T}$ is true if and only if this sentence is added to P_γ. This sentence is added if and only if $\langle d, t\rangle \in DT$, or if and only if q is true as well. If q is of the form, "Does user u have access i to document d?", $q^\mathcal{T}$ is of the form, 'Does subject s have access i to object o?". Due to the direct translation of the MMW policy into SD3, these queries will evaluate to equal truth value. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^\mathcal{T} \vdash q^\mathcal{T}$.

Next, consider some arbitrary γ_k reachable from γ_0. We construct $\gamma_k^\mathcal{T}$ this is reachable from $\gamma_0^\mathcal{T}$ and that answers every $q^\mathcal{T}$ in the same way that γ_k answers q, as follows. Consider each state transition in the sequence $\gamma_0 \mapsto_\psi \gamma_1 \mapsto_\psi \ldots \mapsto_\psi \gamma_k$ in the MMW system. If the state transition in MMW is the execution of `create_document(u, d)`, we execute `create_object(u, d)`. If the state transition in MMW is the execution of `destroy_document(u, d)`, we execute `destroy_object(u,, d)`. If the state transition in MMW is the execution of `assign_tags(u, d, T1)`, we execute `transform_policy(u, P1, {})`, where P_1 contains "`TAG(d, t)`" for each $t \in T_1$. If the state transition in MMW is the execution of `revoke_tags(u, d, T1)`, we execute `transform_policy(u, {}, P1)`, where P_1 contains "`TAG(d, t)`" for each $t \in T_1$. If the state transition in MMW is the execution of `create_user(u, u')`, we execute `create_subject(u, u')`. If the state transition in MMW is the execution of `destroy_user(u, u')`, we execute `destroy_subject(u, u')`. If the state transition in MMW is the execution of `assign_attributes(u, u', A1)`, we execute `transform_policy(u, P1, {})`, where P_1 contains "`ATTR(u', a)`" for each $a \in A_1$. If the state transition in MMW is the execution of `revoke_attributes(u, u', A1)`, we execute `transform_policy(u, {}, P1)`, where P_1 contains "`ATTR(u', a)`" for each $a \in A_1$. If the state transition in MMW is the execution of `transform_policy(u, P+, P-)`, we execute `transform_policy(u, P1, P2)`, where P_1 contains $translate(p)$ for each $p \in P_+$, and P_2 contains $translate(p)$ for each $p \in P_-$.

Now, consider each possible query q. If q is of the form, "Does user u exist?", $q^\mathcal{T}$ is of the form "Does subject s exist?". These queries are equal due to the direct mapping between the elements of sets S_γ and U being maintained in the above procedure. If q is of the form, "Does user u have attribute a?", $q^\mathcal{T}$ is of the form, "Is policy sentence "`ATTR(u, a)`" in the policy?". Thus, $q^\mathcal{T}$ is true if and only if this sentence is added to P_γ. This sentence is added if and only if $\langle u, a\rangle \in UA$, or if and only if q is true as well. If q is of the form, "Does document d have tag t?", $q^\mathcal{T}$ is of the form, "Is policy sentence "`TAG(d, t)`" in the policy?". Thus, $q^\mathcal{T}$ is true if and only if this sentence is added to P_γ. This sentence is added if and only if $\langle d, t\rangle \in DT$, or if and only if q is true as well. If q is of the form, "Does user u have access i to document d?", $q^\mathcal{T}$ is of the form, 'Does subject s have access i to object o?". Due to the direct translation of the MMW policy into SD3, these queries will evaluate to equal truth value. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^\mathcal{T} \vdash q^\mathcal{T}$.

Therefore, we've proven property (1) for state-matching reductions.

We prove that property (2) for a state-matching reduction is satisfied by our mapping also by construction. Let $\gamma_0^\mathcal{T}$ be the start-state in SD3-T corresponding to γ_0, the start-state in MMW. Then, if $\gamma_k^\mathcal{T}$ is a state reachable from $\gamma_0^\mathcal{T}$ and $q^\mathcal{T}$ is a query in SD3-T whose corresponding query in MMW is q, we construct γ_k, a state in MMW reachable from γ_0 as follows. For each s in S, we execute `create_user(s, s)`. For each o in O, we execute `create_document(s, o)`. For each sentence in P of the form, "`ATTR(u, a) :- ;`", we execute `grant_attributes(u, u, {a})`. For each sentence in P of the form, "`TAG(d, t) :- ;`", we execute `grant_tags(u, d, {t})`. For each other sentence p in P, we execute `transform_policy(u, {`$detranslate(p)$`}, {})`, where $detranslate(p)$ is the reverse procedure of $translate(p)$.

Now, consider each possible query q. If q is of the form, "Does user u exist?", $q^\mathcal{T}$ is of the form "Does subject s exist?". These queries are equal due to the direct mapping between the elements of sets S_γ and U being maintained in the above procedure. If q is of the form, "Does user u have attribute a?", $q^\mathcal{T}$ is of the form, "Is policy sentence "`ATTR(u, a)`" in the policy?". Thus, $q^\mathcal{T}$ is true if and only if this sentence is added to P_γ. This sentence is added if and only if $\langle u, a\rangle \in UA$, or if and only if q is true as well. If q is of the form, "Does document d have tag t?", $q^\mathcal{T}$ is of the form, "Is policy sentence "`TAG(d, t)`" in the policy?". Thus, $q^\mathcal{T}$ is true if and only if this sentence is added to P_γ. This sentence is added if and only if $\langle d, t\rangle \in DT$, or if and only if q is true as well. If q is of the form, "Does user u have access i to document d?", $q^\mathcal{T}$ is of the form, 'Does subject s have access i to object o?". Due to the direct translation of the SD3 policy into MMW, these queries will evaluate to equal truth value. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^\mathcal{T} \vdash q^\mathcal{T}$.

Therefore, we've proven property (2) for state-matching reductions, and proven that our mapping σ is a state-matching reduction. □

D. Proof of Theorem 8

Theorem 8: LBAC is not expressive enough to safely implement the modern military workload.

Proof: Let $W_\mathcal{M} = \langle \mathcal{M}, I_\mathcal{M}\rangle$ be the modern military workload, and $\mathcal{L}$ the LBAC scheme. LBAC is expressive enough to safely implement the modern military workload if and only if there exists a state-matching reduction from scheme $\mathcal{M}$ to scheme $\mathcal{L}$. We prove, by contradiction, that there exists no state-matching reduction from $\mathcal{M}$ to $\mathcal{L}$.

Let $\gamma_1^{\mathcal{M}}$ be a state within MMW where user u_1 has read access to document d_1 and append access to document d_2, and user u_2 conversely has append access to d_1 and read access to d_2. Note that this system is trivial to represent in $\mathcal{M}$. For an equivalent state to exist in LBAC, the lattice rules require that

$$f_C(u_1) \asymp f_O(d_1) \asymp f_C(u_2) \asymp f_O(d_2) \asymp f_C(u_1)$$

Letting Cl_x be the clearance/classification component of f_x and Co_x be the compartment component, and using the definition of $\asymp$, the dominate relationship, we can show that

$$Cl_C(u_1) \geq Cl_O(d_1) \geq Cl_C(u_2) \geq Cl_O(d_2) \geq Cl_C(u_1)$$
$$Co_C(u_1) \supseteq Co_O(d_1) \supseteq Co_C(u_2) \supseteq Co_O(d_2) \supseteq Co_C(u_1)$$

and thus

$$Cl_C(u_1) = Cl_O(d_1) = Cl_C(u_2) = Cl_O(d_2)$$
$$Co_C(u_1) = Co_O(d_1) = Co_C(u_2) = Co_O(d_2)$$

However, this would grant both u_1 and u_2 full rights over both d_1 and d_2, which violates the initial state we defined. Thus, there exists no state in LBAC that is equivalent to the described state in MMW, which violates property (1) for a state-matching reduction. Thus, LBAC is not expressive enough to represent the types of states needed to build a state-matching reduction from $W_{\mathcal{M}}$. □

E. Proof of Corollary 9

Corollary 9: LBAC is not expressive enough to safely implement the modern military workload, regardless of any access control auxiliary machine used to augment it.

Proof: Let $W_{\mathcal{M}} = \langle \mathcal{M}, I_{\mathcal{M}} \rangle$ be the modern military workload, $\mathcal{L}$ the LBAC scheme, $\mathcal{U}$ an arbitrary access control auxiliary machine, and $\mathcal{Q} = \mathcal{L} \circ \mathcal{U}$. LBAC is expressive enough to safely implement the modern military workload if and only if there exists a state-matching reduction from scheme $\mathcal{M}$ to scheme $\mathcal{Q}$. We prove, by contradiction, that there exists no state-matching reduction from $\mathcal{M}$ to $\mathcal{Q}$.

By Definition 10, $\forall q \in Q^{\mathcal{L}}, \gamma \in \Gamma^{\mathcal{L}}, \gamma' \in \Gamma^{\mathcal{Q}}, \gamma \vdash^{\mathcal{L}} q \Leftrightarrow \langle \gamma, \gamma' \rangle \vdash^{\mathcal{Q}} q$. Thus, the definition of $\asymp$ remains unchanged in $\mathcal{Q}$ from its state in $\mathcal{L}$. Let $\gamma_1^{\mathcal{M}}$ be the state described in the proof of Theorem C-D. By the same argument as in that proof, $\gamma_1^{\mathcal{M}}$ is easily represented in MMW, but no corresponding state exists in $\mathcal{Q} = \mathcal{L} \circ \mathcal{U}$. Thus, regardless of any AM, LBAC can not safely implement workload $W_{\mathcal{M}}$. □

F. Proof of Theorem 10

Theorem 10: BLP is expressive enough to safely implement the modern military workload, when augmented via the tag-based auxiliary machine, $\mathcal{N}$, as described in Definition 15.

Proof: Let $W_{\mathcal{M}} = \langle \mathcal{M}, I_{\mathcal{M}} \rangle$ be the modern military workload, $\mathcal{B}$ the BLP scheme, and $\mathcal{N}$ the tag-based auxiliary machine described in Definition 15. The augmented BLP scheme $\mathcal{B} \circ \mathcal{N}$ is expressive enough to safely implement the modern military workload if and only if there exists a state-matching reduction from scheme $\mathcal{M}$ to scheme $\mathcal{B} \circ \mathcal{N}$.

We present a mapping from $\mathcal{M}$ to $\mathcal{B} \circ \mathcal{N}$ and proof that the mapping satisfies the two properties for it to be a state-matching reduction.

The mapping, σ, needs to map every $\langle \gamma, \psi \rangle$ in MMW to $\sigma(\langle \gamma, \psi \rangle) = \langle \gamma^{\mathcal{B}}, \psi^{\mathcal{B}} \rangle$ in $\mathcal{B} \circ \mathcal{N}$, as well as every q in MMW to $\sigma(q) = q^{\mathcal{B}}$ in $\mathcal{B} \circ \mathcal{N}$.

$$\begin{aligned}
\text{Let } \sigma(\gamma) = \gamma^S =& \langle S_\gamma, O_\gamma, b_\gamma, M_\gamma, f_{S\gamma}, f_{O\gamma}, f_{C\gamma}, P_\gamma, SA_\gamma, OT_\gamma \rangle \text{ where} \\
S_\gamma =& U \\
O_\gamma =& D \\
b_\gamma =& build_b(\gamma) \\
M_\gamma =& \{\forall u \in U, d \in D, i \in I \mid \langle u, d, i \rangle\} \\
\forall s \in S, f_{S\gamma}(s) =& \langle c_0, \{\} \rangle \\
\forall o \in O, f_{O\gamma}(o) =& \langle c_0, \{\} \rangle \\
\forall s \in S, f_{C\gamma}(s) =& \langle c_0, \{\} \rangle \\
P_\gamma =& P \\
SA_\gamma =& UA \\
OT_\gamma =& DT
\end{aligned}$$

Here, $build\_b(\gamma)$ refers to a procedure that iterates over all $\langle u, d\rangle \in U \times D$ and enables in b any accesses that are allowed within the policy, P.

If q is of the form, "Does user u exist?", let $\sigma(q) = q^{\mathcal{B}} =$ "Does subject s exist?", with $s = u$. If q is of the form, "Does user u have attribute a?", let $q^{\mathcal{B}} =$ "Does subject s have attribute a?", with $s = u$. If q is of the form, "Does document d have tag t?", let $q^{\mathcal{B}} =$ "Does object o have tag t?", with $o = d$. If q is of the form, "Does user u have access i to document d?", let $q^{\mathcal{B}} =$ "Does subject s have access a to object o?", with $s = u$, $a = i$, and $o = d$.

Let γ_0 be a start state in MMW. Produce $\gamma_0^{\mathcal{B}}$ in $\mathcal{B} \circ \mathcal{N}$ using σ. Given γ_k such that $\gamma_0 \overset{*}{\mapsto}_{\psi} \gamma_k$, we show that there exists $\gamma_k^{\mathcal{B}}$ such that $\gamma_0^{\mathcal{B}} \overset{*}{\mapsto}_{\psi^{\mathcal{B}}} \gamma_k^{\mathcal{B}}$ where, for all q, $\gamma_k^{\mathcal{B}} \vdash q^{\mathcal{B}}$ if and only if $\gamma_k \vdash q$.

Consider the case where $\gamma_k = \gamma_0$, then let $\gamma_k^{\mathcal{B}} = \gamma_0^{\mathcal{B}}$. Now, consider each possible query q. If q is of the form, "Does user u exist?", $q^{\mathcal{B}}$ is of the form, "Does subject s exist?". These queries are equal due to the direct mapping between elements of sets S_γ and U. If q is of the form, "Does user u have attribute a?", $q^{\mathcal{B}}$ is of the form, "Does subject s have attribute a?". These queries are equal due to the direct mapping between the elements of sets SA_γ and UA. If q is of the form, "Does document d have tag t?", $q^{\mathcal{B}}$ is of the form, "Does object o have tag t?". These queries are equal due to the direct mapping between the elements of sets OT_γ and DT. If q is of the form, "Does user u have access i to document d?", $q^{\mathcal{B}}$ is of the form, "Does subject s have access a to object o?". The procedure $build\_b$ ensures that these queries are equal by representing the resolution of the MMW policy in BLP's b. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^{\mathcal{B}} \vdash q^{\mathcal{B}}$.

Next, consider some arbitrary γ_k reachable from γ_0. We construct $\gamma_k^{\mathcal{B}}$ this is reachable from $\gamma_0^{\mathcal{B}}$ and that answers every query $q^{\mathcal{B}}$ in the same way that γ_k answers q, as follows. Consider each state transition in the sequence $\gamma_0 \mapsto_\psi \gamma_1 \mapsto_\psi \ldots \mapsto_\psi \gamma_k$ in the MMW system. If the state transition in MMW is the execution of `create_document(`u`, `d`)`, we execute `create_object(`u`, `d`)`. If the state transition in MMW is the execution of `destroy_document(`u`, `d`)`, we execute `destroy_object(`u`, `d`)`. If the state transition in MMW is the execution of `assign_tags(`u`, `d`, `T_1`)`, we execute `assign_tags(`u`, `d`, `T_1`)`, followed by iterating over all $s \in S$, using `get_`$\star$ for any new accesses to the affected object. If the state transition in MMW is the execution of `revoke_tags(`u`, `d`, `T_1`)`, we execute `revoke_tags(`u`, `d`, `T_1`)`, followed by iterating over all $s \in S$, using `release_`$\star$ for any accesses to the affected object that are no longer valid. If the state transition in MMW is the execution of `create_user(`u`, `u'`)`, we execute `create_subject(`u`, `u'`)`. If the state transition in MMW is the execution of `destroy_user(`u`, `u'`)`, we execute `destroy_subject(`u`, `u'`)`. If the state transition in MMW is the execution of `assign_attributes(`u`, `u'`, `A_1`)`, we execute `assign_attributes(`u`, `u'`, `A_1`)`, followed by iterating over all $o \in O$, using `get_`$\star$ for any new accesses the affected subject now has. If the state transition in MMW is the execution of `revoke_attributes(`u`, `u'`, `A_1`)`, we execute `revoke_attributes(`u`, `u'`, `A_1`)`, followed by iterating over all $o \in O$, using `release_`$\star$ for any accesses the affected object no longer has. If the state transition in MMW is the execution of `transform_policy(`u`, `P_+`, `P_-`)`, we execute `transform_policy(`u`, `P_+`, `P_-`)`, followed by iterating over all $\langle o, s\rangle \in O \times S$, using `get_`$\star$ for any accesses that are newly allowed and `release_`$\star$ for any accesses that are newly disallowed.

Now, consider each possible query q. If q is of the form, "Does user u exist?", $q^{\mathcal{B}}$ is of the form, "Does subject s exist?". These queries are equal due to the direct mapping between elements of sets S_γ and U being maintained in the above procedure. If q is of the form, "Does user u have attribute a?", $q^{\mathcal{B}}$ is of the form, "Does subject s have attribute a?". These queries are equal due to the direct mapping between the elements of sets SA_γ and UA being maintained in the above procedure. If q is of the form, "Does document d have tag t?", $q^{\mathcal{B}}$ is of the form, "Does object o have tag t?". These queries are equal due to the direct mapping between the elements of sets OT_γ and DT being maintained in the above procedure. If q is of the form, "Does user u have access i to document d?", $q^{\mathcal{B}}$ is of the form, "Does subject s have access a to object o?". The procedures following changes to DT, SA, and P ensure that these queries are equal by representing the resolution of the MMW policy in BLP's b. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^{\mathcal{B}} \vdash q^{\mathcal{B}}$.

Therefore, we've proven property (1) for state-matching reductions.

We prove that property (2) for a state-matching reduction is satisfied by our mapping also by construction. Let $\gamma_0^{\mathcal{B}}$ be the start-state in $\mathcal{B} \circ \mathcal{N}$ corresponding to γ_0, the start-state in MMW. Then, if $\gamma_k^{\mathcal{B}}$ is a state reachable from $\gamma_0^{\mathcal{B}}$ and $q^{\mathcal{B}}$ is a query in $\mathcal{B} \circ \mathcal{N}$ whose corresponding query in MMW is q, we construct γ_k, a state in MMW reachable from γ_0 as follows. For each s in S, we execute `create_user(`u`, `s`)`. For each o in O, we execute `create_document(`u`, `o`)`. For each $\langle s, a\rangle \in SA$, we execute `assign_attributes(`$u$`, `$s$`, `$\{a\}$. For each $\langle o, t\rangle \in OT$, we execute `assign_tags(`$u$`, `$o$`, `$\{t\}$`)`. For each $\langle s, o, a\rangle \in b$, we execute `transform_policy(`$u$`, `$\{p\}$`, `$\{\}$`)`, where p is a policy sentence that grants subject s right a to object o (using the subject's and object's IDs as attribute and tag).

Now, consider each possible query q. If q is of the form, "Does user u exist?", $q^{\mathcal{B}}$ is of the form, "Does subject s exist?". These queries are equal due to the direct mapping between elements of sets S_γ and U being maintained in the above procedure. If q is of the form, "Does user u have attribute a?", $q^{\mathcal{B}}$ is of the form, "Does subject s have attribute a?". These queries are equal due to the direct mapping between the elements of sets SA_γ and UA being maintained in the above procedure. If q is of the form, "Does document d have tag t?", $q^{\mathcal{B}}$ is of the form, "Does object o have tag t?". These queries are equal due to the direct mapping between the elements of sets OT_γ and DT being maintained in the above procedure. If q is of the form, "Does user u have access i to document d?", $q^{\mathcal{B}}$ is of the form, "Does subject s have access a to object o?". The above procedure

ensures that these queries are equal by granting, in MMW, precisely the accesses granted in $\mathcal{B} \circ \mathcal{N}$. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^{\mathcal{B}} \vdash q^{\mathcal{B}}$.

Therefore, we've proven property (2) for state-matching reductions, and proven that our mapping σ is a state-matching reduction. □

G. Proof of Theorem 11

Theorem 11: RBAC is expressive enough to safely implement the modern military workload, when augmented via the object-tag auxiliary machine, $\mathcal{O}$, described in Definition 16.

Proof: Let $W_{\mathcal{M}} = \langle \mathcal{M}, I_{\mathcal{M}} \rangle$ be the modern military workload, $\mathcal{R}$ the RBAC scheme, and $\mathcal{O}$ the object-tag auxiliary machine described in Definition 16. The augmented RBAC scheme $\mathcal{R} \circ \mathcal{O}$ is expressive enough to safely implement the modern military workload if and only if there exists a state-matching reduction from scheme $\mathcal{M}$ to scheme $\mathcal{R} \circ \mathcal{O}$.

We present a mapping from $\mathcal{M}$ to $\mathcal{R} \circ \mathcal{O}$ and proof that the mapping satisfies the two properties for it to be a state-matching reduction.

The mapping, σ, needs to map every $\langle \gamma, \psi \rangle$ in MMW to $\sigma(\langle \gamma, \psi \rangle) = \langle \gamma^{\mathcal{R}}, \psi^{\mathcal{R}} \rangle$ in $\mathcal{R} \circ \mathcal{O}$, as well as every q in MMW to $\sigma(q) = q^{\mathcal{R}}$ in $\mathcal{R} \circ \mathcal{O}$.

$$
\begin{aligned}
\text{Let } \sigma(\gamma) = \gamma^{\mathcal{R}} =& \langle S_\gamma, P_\gamma, SR_\gamma, RP_\gamma, OT_\gamma, L_\gamma \rangle \text{ where} \\
S_\gamma =& U \\
P_\gamma =& D \times I \\
SR_\gamma =& \{\langle u, r \rangle \mid u \in U, r \in \wp(\{a \mid \langle u, a \rangle \in UA\})\} \\
RP_\gamma =& build\_rp(\gamma) \\
OT_\gamma =& DT \\
L_\gamma =& P
\end{aligned}
$$

Here, $build\_rp(\gamma)$ refers to the procedure that resolves the MMW policy, P, and adds to RP the pair $\langle r, p \rangle$ when permission p is granted to users with the attribute set represented in r (note that roles are sets of attributes, and thus subjects are assigned the powerset of their set of attributes).

If q is of the form, "Does user u exist?", let $\sigma(q) = q^{\mathcal{R}} =$ "Does subject s exist?", with $s = u$. If q is of the form, "Does user u have attribute a?", let $q^{\mathcal{R}} =$ "Does subject s belong to role r?", with $s = u$ and $r = \{a\}$. If q is of the form, "Does document d have tag t?", let $q^{\mathcal{R}} =$ "Does object o have tag t?", with $o = d$. If q is of the form, "Does user u have access i to document d?", let $q^{\mathcal{R}} =$ "Does subject s have permission p?", with $p = \langle d, i \rangle$.

Let γ_0 be a state state in MMW. Produce $\gamma_0^{\mathcal{R}}$ in $\mathcal{R} \circ \mathcal{O}$ using σ. Given γ_k such that $\gamma_0 \overset{*}{\mapsto}_{\psi} \gamma_k$, we show that there exists $\gamma_k^{\mathcal{R}}$ such that $\gamma_0^{\mathcal{R}} \overset{*}{\mapsto}_{\psi^{\mathcal{R}}} \gamma_k^{\mathcal{R}}$ where, for all q, $\gamma_k^{\mathcal{R}} \vdash q^{\mathcal{R}}$ if and only if $\gamma_k \vdash q$.

Consider the case where $\gamma_k = \gamma_0$, then let $\gamma_k^{\mathcal{R}} = \gamma_0^{\mathcal{R}}$. Now, consider each possible query q. If q is of the form, "Does user u exist?", $q^{\mathcal{R}}$ is of the form, "Does subject s exist?". These queries are equal due to the direct mapping between elements of sets S_γ and U. If q is of the form, "Does user u have attribute a?", $q^{\mathcal{R}}$ is of the form "Does subject s belong to role r?". These queries are equal since each subject in RBAC is assigned a set of roles that equal the powerset of the corresponding user's set of attributes in MMW. If q is of the form, "Does document d have tag t?", $q^{\mathcal{R}}$ is of the form, "Does object o have tag t?". These queries are equal due to the direct mapping between the elements of sets OT_γ and DT. If q is of the form, "Does user u have access i to document d?", $q^{\mathcal{R}} =$ is of the form, "Does subject s have permission p?". The procedure $build\_rp$ ensures that these queries are equal by representing the resolution of the MMW policy in RBAC's RP. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^{\mathcal{R}} \vdash q^{\mathcal{R}}$.

Next, consider some arbitrary γ_k reachable from γ_0. We construct $\gamma_k^{\mathcal{R}}$ this is reachable from $\gamma_0^{\mathcal{R}}$ and that answers every query $q^{\mathcal{R}}$ in the same way that γ_k answers q, as follows. Consider each state transition in the sequence $\gamma_0 \mapsto_{\psi} \gamma_1 \mapsto_{\psi} \ldots \mapsto_{\psi} \gamma_k$ in the MMW system. If the state transition in MMW is the execution of `create_document(u, d)`, we execute `create_permission(s, p)` for each $p = \langle d, i \rangle$. If the state transition in MMW is the execution of `destroy_document(u, d)`, we execute `destroy_permission(s, p)` for each $p = \langle d, i \rangle$. If the state transition in MMW is the execution of `assign_tags(u, d, T1)`, we execute `assign_tags(s, o, T1)`, with $s = u$ and $o = d$, followed by `grant_permission(s, r, p)` for each role (attribute-set) that, given the documents new tags, is granted new permission p. If the state transition in MMW is the execution of `revoke_tags(u, d, T1)`, we execute `revoke_tags(s, o, T1)`, with $s = u$ and $o = d$, followed by `revoke_permission(s, r, p)` for each role (attribute-set) that, given the documents loss of tags, loses permission p. If the state transition in MMW is the execution of `create_user(u, u')`, we execute `create_subject(s, s')`, with $s' = u'$. If the state transition in MMW is the execution of `destroy_user(u, u')`, we execute `destroy_subject(s, s')`, with $s' = u'$. If the state transition in MMW is the execution of `assign_attributes(u, u', A1)`, we execute `grant_role(s, s', r)` for each new role in $\wp(\{a \mid \langle u, a \rangle \in UA\})$. If the state transition in MMW is the execution of `revoke_attributes(u,`

u', A_1), we execute `revoke_role(`s, s', r`)` for each role removed from $\wp(\{a \mid \langle u, a\rangle \in UA\})$. If the state transition in MMW is the execution of `transform_policy(`u, P_+, P_-`)`, we execute `transform_policy(`s, L_+, L_-`)`, followed by `grant_permission(`s, r, p`)` for each role (attribute-set) that, under the new resolution of the policy, is granted new permission p, and `revoke_permission(`s, r, p`)` for each role which loses permission p.

Now, consider each possible query q. If q is of the form, "Does user u exist?", $q^{\mathcal{R}}$ is of the form, "Does subject s exist?". These queries are equal due to the direct mapping between elements of sets S_γ and U being maintained in the above procedure. If q is of the form, "Does user u have attribute a?", $q^{\mathcal{R}}$ is of the form "Does subject s belong to role r?". These queries are equal since each subject in RBAC is assigned a set of roles that equal the powerset of the corresponding user's set of attributes in MMW. If q is of the form, "Does document d have tag t?", $q^{\mathcal{R}}$ is of the form, "Does object o have tag t?". These queries are equal due to the direct mapping between the elements of sets OT_γ and DT being maintained in the above procedure. If q is of the form, "Does user u have access i to document d?", $q^{\mathcal{R}} =$ is of the form, "Does subject s have permission p?". The resolution of the policy is represented in RP, in the above procedure, ensuring that these queries are equal. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^{\mathcal{R}} \vdash q^{\mathcal{R}}$.

Therefore, we've proven property (1) for state-matching reductions.

We prove that property (2) for a state-matching reduction is satisfied by our mapping also by construction. Let $\gamma_0^{\mathcal{R}}$ be the start-state in $\mathcal{R} \circ \mathcal{O}$ corresponding to γ_0, the start-state in MMW. Then, if $\gamma_k^{\mathcal{R}}$ is a state reachable from $\gamma_0^{\mathcal{R}}$ and $q^{\mathcal{R}}$ is a query in $\mathcal{R} \circ \mathcal{O}$ whose corresponding query in MMW is q, we construct γ_k, a state in MMW reachable from γ_0 as follows. For each $s \in S$, we execute `create_user(`u, s`)`. For each $p \in P$, we execute `create_document(`u, p`)`. For each $\langle s, r\rangle \in SR$, we execute `assign_attributes(`$u$, s, $\{r\}$`)`. For each $\langle o, t\rangle \in OT$, we execute `assign_tags(`$s$, o, $\{t\}$`)`. For each $s \in S, e \in P$ such that s is granted permission e, we execute `transform_policy(`u, $\{p\}$, $\{\}$`)`, where p is a policy sentence that grants the user corresponding to s access to the document that corresponds to e (using the user's and document's IDs as attribute and tag).

Now, consider each possible query q. If q is of the form, "Does user u exist?", $q^{\mathcal{R}}$ is of the form, "Does subject s exist?". These queries are equal due to the direct mapping between elements of sets S_γ and U being maintained in the above procedure. If q is of the form, "Does user u have attribute a?", $q^{\mathcal{R}}$ is of the form "Does subject s belong to role r?". These queries are equal due to the direct mapping between elements of the sets SR_γ and UA being maintained in the above procedure. If q is of the form, "Does document d have tag t?", $q^{\mathcal{R}}$ is of the form, "Does object o have tag t?". These queries are equal due to the direct mapping between the elements of sets OT_γ and DT being maintained in the above procedure. If q is of the form, "Does user u have access i to document d?", $q^{\mathcal{R}} =$ is of the form, "Does subject s have permission p?". The above procedure ensures that these queries are equal by granting, in MMW, precisely the accesses granted in $\mathcal{R} \circ \mathcal{O}$. Thus, $\gamma_k \vdash q$ if and only if $\gamma_k^{\mathcal{R}} \vdash q^{\mathcal{R}}$. □