Towards Quantitative Analysis of Proofs of Authorization: Applications, Framework, and Techniques

Adam J. Lee
Department of Computer Science
University of Pittsburgh
adamlee@cs.pitt.edu

Ting Yu
Department of Computer Science
North Carolina State University
yu@csc.ncsu.edu

***Abstract*—Although policy compliance testing is generally treated as a binary decision problem, the evidence gathered during the trust management process can actually be used to examine these outcomes within a more continuous space. In this paper, we develop a formal model that allows us to quantitatively reason about the outcomes of the policy enforcement process in both absolute (i.e., user to ideal case) and relative (i.e., user to user) terms. Within this framework, it becomes possible to quantify, e.g., the robustness of a user's proof of authorization to possible perturbations in the system, how close an unauthorized user is to satisfying a particular policy, and relative "top-k" style rankings of the best users to carry out a particular task. To this end, we explore several interesting classes of scoring functions for assessing the robustness of authorization decisions, and develop criteria under which these types of functions can be composed with one another. We further show that these types of functions can be extended to quantify how close unauthorized users are to satisfying policies, which can be a useful risk metric for decision making under unexpected circumstances.**

***Keywords*-access control; policy; risk; trust management;**

I. Introduction

In many widely distributed systems, enforcing access control policies based upon user identity alone is a tricky proposition, as the set of users authorized to access a given resource may not be known a priori. Over the past decade, a plethora of trust management techniques have been developed that enable *any* principal to access *any* resource, provided that she can generate one or more machine verifiable proofs that authorize the requested action (e.g., [2]–[4], [15], [22]). The structure of the resulting proof trees is implied by policy assertions made by principals within the system, while the leaves correspond to certified assertions regarding principal attributes, role membership information, or other system state. Like the vast majority of the access control literature, these systems treat policy enforcement as a binary process: a principal is either able to generate a correct proof of authorization and is thereby granted access, or else she is not.

This binary view of the authorization process has no impact on the *soundness* of these systems: any well-formed proof of authorization is guaranteed to be accepted. However, this simplified view does reduce the *precision* of the information that is gleaned from the process itself. If examined closely, proofs of authorization can reveal a great deal of information about the conditions under which some user u is granted access to a resource r. For instance, can u produce multiple proofs of authorization for the resource r? If so, how disjoint are the proofs that she can produce? Are certain "common case" proofs preferred to other, less frequently used, "exceptional case" proofs? If the trust management system being used considers reputation or QoS history (e.g., as in [7], [20]), how do values of these attributes vary between users?

These types of information allow us to put the binary decisions made by trust management systems under the microscope and consider them within a more continuous space. If properly quantified, this information can reveal a great deal about the overall state of the system. For instance, consider the following questions:

- *How robust is user u's ability to access resource r?* The existence of multiple proofs of authorization implies that u's ability to access r is robust against policy changes or state perturbations, particularly if these proofs are largely independent of one another.
- *How do users within some group compare to one another?* Simply put, a policy is a set of requirements. By evaluating how well these requirements are satisfied, it becomes possible to objectively rank a set of users to find the *best* subset for a particular task.
- *How close is an unauthorized user to satisfying a policy?* In the physical world, exceptions to policies are very often made based upon the judgement calls of qualified individuals. A quantitative valuation of the quality of a *partial* proof of authorization can be a useful metric for assessing the risk associated with granting access to an unauthorized user under exceptional circumstances.

The information needed to answer the above types of questions exists within, and is in fact collected by, most trust management systems. For instance, the credential chain discovery and resolution proof construction algorithms used by RT [23] and SecPAL [3], respectively, can be used to discover any possible proofs of authorization for a particular action at runtime. However, the baby is to some degree thrown out with the bathwater, as the internal structure of a proof of authorization is essentially ignored once its correctness has been verified. Nonetheless, this structural information is gathered by existing trust management systems and can be used to answer the above types of questions.

We note that others have designed various types of quantitative trust management systems in the past (e.g., [8], [30]). However, these works tend to focus on degrees of uncertainty, rather than on providing a means of assessing the relative quality of valid proofs of authorization. Our goal in this paper is to articulate a formal understanding of these types of analyses, and to develop practical mechanisms for analyzing proofs of authorization generated by trust management systems. In addressing this challenge, we make the following contributions:

- **Formal Model.** We begin by developing a unified intellectual framework for the quantitative analysis of proofs of authorization. The scaffolding provided by this framework allows us to derive a set of *necessary* properties that must be possessed by authorization scoring functions, as well as a set of *desirable*, though not strictly requisite, properties. We then provide an instantiation of this framework for analyzing RT_0 policies, which is used to make the discussion in the remainder of the paper concrete.
- **Scoring Functions.** As there is no prototypical distributed system, we examine situations in which the principal evaluating proofs of authorization has access to (i) perfect information about the state of the system; (ii) perfect information about the state of her security domain, but only limited external information; and (iii) only the information that she uncovers during the proof construction process, which may in fact be simplified by other principals in the system (e.g., complex proofs may be collapsed into simple signed assertions). In every case, we show that interesting classes of useful authorization scoring functions do in fact exist and can be used to reliably assess proofs of authorization in absolute (i.e., user to ideal case) as well as relative (i.e., user to user) terms.
- **Functional Composition.** In certain circumstances, it may be undesirable to use a single authorization scoring function when considering a collection of proofs of authorization. For instance, a principal may wish to use a fine-tuned function for scoring the portions of proofs generated within her domain, but more general structural assessments of the portions of a proof generated outside of her domain. To this end, we identify the conditions under which two or more authorization scoring functions can be composed without altering the properties of the individual functions comprising the composition. We then show that the authorization scoring functions defined in this paper can be safely composed with one another.
- **Quantification of Risk.** We develop extensions to our authorization scoring functions that enable the scoring and ranking of *incomplete* proofs of authorization. This allows administrators to assess how close a particular user is to satisfying a policy, and can serve as a primitive risk metric that can guide the decision making process when access exceptions need to be made during emergencies or other unexpected circumstances.

In addition to motivating the above types of analysis, we show that these techniques are not just of theoretical interest, but also practical to carry out in existing trust management systems. To the best of our knowledge, the framework and analysis presented in this paper represent the first comprehensive effort towards a quantitative analysis of proofs of authorization.

We begin by presenting several motivating scenarios for this work in Section II. Section III outlines the intellectual framework within which our discussion will take place and identifies several necessary criteria for authorization scoring functions, as well as other desirable (but not strictly necessary) properties. Section IV discusses several classes of scoring functions, examines the overheads of evaluating these functions, and proves that they satisfy the criteria outlined in Section III. In Section V, these techniques are extended to properly assess the partial proofs generated by unauthorized principals. Directions for extensions to the basic techniques presented in this paper and other future work are discussed in Section VI. Finally, we discuss related work in Section VII and present our conclusions in Section VIII.

II. Potential Applications

In this section, we consider a number of potential use cases that look beyond the standard binary yes/no analysis of proofs of authorization. These scenarios show that a more quantitative analysis is useful for considering the proofs generated by a single user, or the collection of proofs that can be generated by a group of users; identify the users who best satisfy a particular policy, as well as those who are the least qualified; and has uses in both offline analysis and online proof generation.

Group Formation. The ability to score proofs of authorization enables the calculation of top-k style rankings of users within a system based upon the quality of the proofs that they can generate. For instance, consider the case in which an administrator is organizing a long-lived working group. Given a scoring function that lends preference to users who can generate multiple and largely independent proofs of authorization, the administrator can find the k most robust users to be members of this long-lived group.

Individual Analysis. A quantitative analysis of proofs of authorization is also useful for considering the quality proofs generated by a single user. For example, consider an anomaly-based intrusion detection system that checks the actions taken by authorized users. Such systems often suffer from high false positives, and may require heavy human intervention to separate the good from the bad. The ability to analyze the quality of the proofs used to initiate the actions in question can help guide the human investigator toward the most likely *actual* threats first. For instance, a proof scoring function that gives preference to short chains of delegations may help investigators identify users gaining access to a resource by means of long chains of poorly maintained policy delegations.

Offline Policy Analysis. Authorization scoring can also be useful during offline policy analysis. By considering both the identities of role members and the scores of their proofs, policy administrators may identify weak points in policies that enable large numbers of users to satisfy a policy through poorly-maintained sub-clauses or improper definitions. This long term mining/clustering of (activity, proof score) pairs may help identify policies in need of revision.

Degree of Non-Membership. Authorization scoring can also be useful for assessing the quality of the incomplete proofs produced by unauthorized individuals. In the physical world, blanket policies are often subject to exceptions during unexpected circumstances. These exceptions are the result of risk/reward analyses weighing the benefits of permitting an exception against the potential costs associated with potential abuses. The ability to score partial proofs of authorization can provide a type of risk metric that quantifies how close a partial proof is to being complete, relative to the metrics embodied by the scoring function used. In the event that no user is authorized to take a particular action, this technique can also be used to help locate the principal who would require the least privilege elevation in order to complete the task; that is, this can help minimize the risk associated with completing some necessary operation.

The above examples clearly show that there is no universal proof scoring function, but rather the function to be used will depend largely on application context. At the same time, it is clear that not every function would make for a good authorization scoring function. To this end, we now develop a formal framework within which we consider this problem. We then identify both *necessary* properties for authorization scoring functions, as well as other *desirable* properties.

III. Intellectual Framework

In this section, we first present a general framework to model the types of information available to trust management systems for the purpose of quantitative analyses. We then materialize this framework in the context of RT_0 trust management system. The remainder of the paper is built upon this materialized framework. Although RT_0 is a relatively simple system, we note that it is representative of a larger class of Datalog-based trust management systems and provides a concrete substrate for the analysis of authorization policies. Later in the paper, we discuss how our mechanisms can be extended to work within richer policy languages.

A. General Framework

Trust management systems are designed to make authorization decisions in a decentralized environment where principals from multiple security domains may interact. Each domain autonomously specifies its security policy regarding access to its local resources. In particular, to facilitate cross-domain collaboration, the security policy of one domain may also grant resource access to principals from other domains. This is typically done through various forms of delegations. In general, a trust management system can be modeled to include the following components:

- $\mathcal{P}$: a set of principals. In most trust management systems, a principal is represented by her public key.
- $\mathcal{S}$: a set of resources. The concept of resources can be used to encode a variety of things whose availability needs to be controlled, such as privileges, capabilities, or memberships to roles. Each resource s is owned by a principal, denoted as $owner(s)$.
- $\mathcal{C}$: a set of credentials, each of which is issued by some principal $p \in \mathcal{P}$. A credential can be abstracted to be in the form $s \leftarrow q$, while $s \in \mathcal{S}$ and q is an opaque policy construction, which shall be materialized by specific trust management systems. This credential indicates that this statement controls access to resource s. Here we require that only the principal $owner(s)$ can issue credentials about resource s. It is possible that multiple credentials control access to a resource s. Note that the concept of credentials here is not restricted to traditionally digitally signed certificates. As long as a statement in the above form can be authenticated and verified (i.e., it is actually issued by the owner of s), it can be considered a valid credential. Also, as typical in most trust management systems, credentials include not only statements about the properties of a principal but also inference rules. This will become clear when we present RT_0 as an example trust management system.
- F: an inference scheme which takes as input a principal, a set of credentials (which include both facts and inference rules as mentioned above), and a target resource, and determines whether the target resource is available to the principal, i.e., $F : \mathcal{P} \times \mathcal{S} \times 2^{\mathcal{C}} \rightarrow \{TRUE, FALSE\}$.

Note that traditional centralized authorization systems can also be represented by the above model (when all the credentials are issued by a single principal). However, trust management systems in general include credentials issued by multiple principals. Similarly, the inference scheme should also have the capability to infer over credentials from different principals to reach a decision.

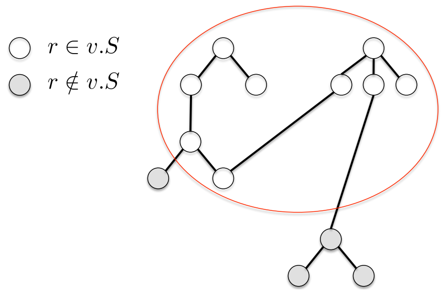

Figure 1. Graphical representation of a system view.

The protection state of a trust management system can be completely described by the (countable) sets of principals $\mathcal{P}$, credentials $\mathcal{C}$, and resources $\mathcal{S}$. In reality, the sets $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{S}$ are time dependent and should thus be parameterized by the time at which the system is examined. In this paper, we will assume that policy evaluation is conducted in a consistent manner (e.g., see [18]), and thus obviate the need for considering time explicitly.

The existence of disclosure policies [25], the overheads of state collection, and other practical constraints are likely to prevent any one principal from discovering the sets $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{S}$ in their entirety. As a result, it is necessary to represent the information that a given principal does have regarding the (partial) state of the system. To accomplish this, we define functions $res : \mathcal{C} \rightarrow \mathcal{S}$ and $ac : \mathcal{S} \rightarrow 2^{\mathcal{C}}$, and then formally define a principal's *view* of the system.

$$res(s \leftarrow q) \mapsto s \tag{1}$$

$$ac(s) \mapsto \{c \in \mathcal{C} \mid res(c) = s\} \tag{2}$$

Definition 1 (View): The *view* that some principal $p \in \mathcal{P}$ has of the protection state of a trust management system is defined as a three tuple $v_p = \langle S \subseteq \mathcal{S}, C \subseteq \mathcal{C}, \mathcal{A} \rangle$, where for each $s \in S$, $ac(s) \subseteq C$, and $\mathcal{A}$ is the abstraction of any auxiliary information that p has about the system.

Intuitively, res identifies the resource that a credential protects, while ac returns all the credentials that protect a resource s. Definition 1 then states that for every principal $p \in \mathcal{P}$, there are some resources that p has complete knowledge about how they are protected, and that p may also have partial information about other resources. For instance, if p is the security administrator for some domain, she is likely to know (at least) all of the credentials that protect resources in that domain, as well as some credentials from other domains. Figure 1 shows an intuitive graphic representation of a view v, in which nodes may be interpreted as resources and edges represent direct authorization as well as delegations among accesses to resources. White nodes represent resources in $v.S$, while grey nodes represent resources not in $v.S$.

A principal's auxiliary information $\mathcal{A}$ might encode knowledge about the number of principals having access to some resources, the importance of each credential in the context of specific applications, or the correlation between accessibility of different resources. For simplicity, in the rest of our discussion, we assume that for any given principal, $\mathcal{A}$ is fixed. That is, given two views created by the same principal, these two views will have the same auxiliary information $\mathcal{A}$. Meanwhile, the views of different principals may have different auxiliary information $\mathcal{A}$.

Given a view $v = \langle S, C, \mathcal{A} \rangle$, one may wonder—since all credentials protecting resources in S are in C—why S is necessary to be included in v. One subtlety is that the principal not only knows all credentials protecting resources in S, but also knows that this set is complete (i.e., no other credentials protecting $s \in S$ exist in the system). Therefore, it is possible that two views have the same credential set C but different resource set S, and they do not correspond to the same knowledge of the system. When comparing two views, we need to consider both S and C.

Definition 2 (Dominates): Given two views $v_1 = \langle S_1, C_1, \mathcal{A}_1 \rangle$ and $v_2 = \langle S_2, C_2, \mathcal{A}_2 \rangle$, we say v_1 *dominates* v_2 if $S_2 \subseteq S_1$, $C_2 \subseteq C_1$ and $\mathcal{A}_1 = \mathcal{A}_2$.

The above definition of views has three interesting cases: $S = \emptyset$ (no information), $S = \mathcal{S}$ (omniscient information), and $S \subset \mathcal{S}$ (partial information). As the first case is to some degree degenerate, our goal in the remainder of this paper is to develop mechanisms for quantitatively scoring the quality of proofs of authorization given only partial information about the state of the system.

Definition 3 (Authorization Scoring Function): A *authorization scoring function* in a trust management system is defined to be $\mathsf{score} : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}$, where $\mathcal{V}$ is the domain of views in a trust management system, and $\mathcal{T}$ is some total ordered domain.

Intuitively, given a principal p, a target resource s, and a view v, $\mathsf{score}(p, s, v)$ measures the "strength" of p's proofs of authorization for s. Note that our definition of authorization scoring functions does not necessarily require p to be authorized to access s according to the underlying inference scheme F. Therefore, it is meaningful to discuss the strength of a principal's access even when she is not authorized to access the resource. Note also that v is not the view of p. Rather, it is the view of the principal who initiates the evaluation of an authorization scoring function. For example, if Alice wants to know how strongly Bob has access to a resource s, then Bob's trust score is calculated as $\mathsf{score}(Bob, s, v_{Alice})$. In other words, Alice's knowledge of the system (i.e., her view) affects Bob's trust score.

Clearly, the meaning of access strength largely depends on specific applications that a trust management system supports. Therefore, it may not always make sense to discuss whether one scoring function is better than another. In fact, it is unlikely that a single authorization scoring function is sufficient for analysis purpose. Instead, a system might adopt multiple scoring functions, each of which reflects a different aspect of a user's authorization proofs. For example, a system may employ two scoring functions: one evaluating the number of independent proofs of authorization for a given resource, and another evaluating how "direct" the proof of authorization is. Given this interpretation, we can easily extend Definition 3 to include scoring functions that output a partially ordered vector of values, rather than a single value from a total order domain. In other words, an m-dimensional scoring function can always be represented as m scoring functions, each of whose output is from a different dimension of the vector respectively. For simplicity, in this paper we focus on scoring functions whose outputs are from a single total order domain, as defined in Definition 3.

B. *Properties of Authorization Scoring Functions*

The properties of authorization scoring functions can be divided into two categories: *necessary* and *desirable* properties. Necessary properties are those that we argue all authorization scoring functions should possess. Otherwise, they will appear to be semantically incorrect (or at least counter-intuitive), and thus are unlikely to be used to support meaningful applications. Desirable properties are those that will make authorization scoring functions more practical, e.g., properties that might optimize the evaluation of these functions. Unless explicitly stated otherwise, all views in our discussion belong to the principal initiating the evaluation of authorization scoring functions, which also means that all the views will have the same auxiliary information $\mathcal{A}$.

We have identified the following necessary properties.

Deterministic. A scoring function should be deterministic. This property is directly required from Definition 3. Although a user's view may not be complete, this and other uncertainty can be captured by the auxiliary information. The final output of an authorization scoring function should be definite so that it is easy to interpret when comparing the scores of different principals.

Simple ordering. Given two principals A and B, if A is authorized to gain access to a resource s while B is not, then A's authorization score should be no lower than that of B in terms of access to s. Intuitively, an authorized principal should be able to produce higher quality proofs than an unauthorized one. Formally, let score be an authorization scoring function and C be all the credentials contained in a view v. If $F(A,s,C) = TRUE$ and $F(B,s,C) = FALSE$, then we should have $\mathsf{score}(A,s,v) \geq \mathsf{score}(B,s,v)$.

Authorization relevant. Though the specific ways to score the trust of principals depend on applications, it should nevertheless tie back to how a principal is authorized to access a resource. Therefore, if a credential is not relevant to authorization decisions for a principal, it should not affect the authorization score of that principal either. To formally describe this property, we first have the following definitions.

Definition 4 (Proof of Access): We say a set C of credentials is a *proof* for a principal p to access resource s in a trust management system if $F(p,s,C) = TRUE$. If no proper subset of C is a proof, then we say that C is a *minimal proof.*

Definition 5 (Relevant Credentials): Given a set C of credentials, let $C_1, \ldots, C_n$ be all the minimal proofs for a principal p to access a resource s. We say a credential $c \in C$ is *relevant* to p's access to s if there exists a C_i, $1 \leq i \leq n$, such that $c \in C_i$. Otherwise, we say c is *irrelevant.*

Let $v_1 = \langle S, C_1, \mathcal{A}\rangle$ and $v_2 = \langle S, C_2, \mathcal{A}\rangle$ be two views where C_1 and C_2 have the same minimal proofs for p to access s. We say an authorization scoring function score is *authorization relevant* if $\mathsf{score}(p,s,v_1) = \mathsf{score}(p,s,v_2)$. In other words, if the difference between two views only includes irrelevant credentials, they should not affect a principal's authorization score.

In addition to the above necessary properties, we have also identified a set of desirable properties:

Interpretable. The score returned by a scoring function should have some reasonable meaning about certain security aspects of a principal's authorization. In other words, from an authorization scoring function, we should not only be able to say a principal's authorization score is a particular value, but also be able to interpret this score in terms of some useful features (e.g., robustness or direct vs. delegated authorization) of its authorization in a system.

Bounded. We say an authorization scoring function is *bounded* if its output domain is a bounded total ordered domain. Given a bounded authorization scoring function, we can possibly identify an "ideal" principal in terms of access to a resource s. By comparing a principal with this ideal principal, we may further observe interesting features of individual principals as well as the trust management system as a whole.

Monotonic. Most trust management systems are monotonic in the sense that discovering more credentials may possibly make a principal change from being unauthorized to authorized, but not vice versa. One benefit of this property is that once a proof of authorization is established, a system may just stop further evaluation. A similar property may also be desirable for the evaluation of authorization scoring functions. To this end, we say that an authorization scoring function score is *monotonic* if for any two views v_1 and v_2 such that v_2 dominates v_1, we have $\mathsf{score}(p,s,v_1) \leq \mathsf{score}(p,s,v_2)$.

Note that we classify monotonicity to be a desirable, rather than necessary, property. To understand why, consider that one natural measurement of authorization robustness is the ratio between the number of existing minimal proofs and the total number of all possible minimal proofs in a system. In this case, adding new credentials may actually increase the number of possible minimal proofs (e.g., by introducing new delegation paths) but not increase the number of minimal proofs for a particular principal. This would cause a decrease in this principal's trust score. Although this is a reasonable authorization scoring function, it is not monotonic in the general case.

Given the above properties, one may wonder whether we can define practical functions that satisfy the above properties. To make the remainder of this paper more concrete, we now materialize the above framework through RT_0. Based on the materialized concrete system, we then present the design of two interesting classes of authorization scoring functions and analyze their properties.

C. RT_0 Overview

RT_0 is the most basic language in the RT family of trust management languages [22]. As in all of the RT languages, principals are identified by means of identity certificates. RT_0 roles are then defined as strings identifying the name of the role and cannot be parameterized. Policy statements in RT_0 are expressed as one or more *role definition credentials* signed by the author of the role definition. There are four basic types of role definition credentials in RT_0:

- **Simple Member.** A role definition of the form $A.R \leftarrow B$ encodes the fact that principal A considers principal B to be a member of the role $A.R$.
- **Simple Containment.** A role definition of the form $A.R \leftarrow B.R_1$ encodes the fact that principal A defines the role $A.R$ to contain all members of principal B's role $B.R_1$.
- **Linking Containment.** A role definition of the form $A.R \leftarrow A.R_1.R_2$ is called a *linked role*. This defines the members of $A.R$ to contain all members of $B.R_2$ for each B that is a member of $A.R_1$.
- **Intersection Containment.** The role definition $A.R \leftarrow B_1.R_1 \cap \cdots \cap B_n.R_n$ defines $A.R$ to contain the principals who are members of each role $B_i.R_i$ where $1 \leq i \leq n$.

These four basic types of role definitions can be used to define a wide range of access control policies. For example, the following RT_0 role definitions express a policy requiring that entities accessing a given resource be graduate students in one of a university's technical departments.

$$Univ.auth \leftarrow Univ.techDept.gradStudent$$
$$Univ.techDept \leftarrow CS$$

If a principal, Alice, could provide a credential proving the statement $CS.gradStudent \leftarrow Alice$, then she could satisfy the above policy.

D. A Materialized Model Using RT_0

We now materialize the general framework using RT_0. It is not hard to see that the general framework can also be easily materialized through other trust management systems such as SD3, KeyNote, and SDSI/SPKI. We focus on RT_0 due to its simplicity, as well as its influence in the research of decentralized trust management.

Clearly, the set $\mathcal{P}$ of principals required by the framework can be defined as the set of all principals in the RT_0 system. The set $\mathcal{S}$ of resources can then be defined as the set of all RT_0 roles defined by all principals in $\mathcal{P}$. Given a credential $C = A.R \leftarrow body$, principal A is said to be the owner of $A.R$. Since each credential must be digitally signed, a principal may only define her own roles. As a result, a principal A is trivially aware of all definitions for any role $A.R$. The inference scheme F in RT_0 is well defined through its set semantics for role memberships [23].

Given the set of all RT_0 roles $\mathcal{R}$ and the set of all RT_0 credentials $\mathcal{C}$, the view of a principal in an RT_0 system can be defined in accordance with Definition 1, as follows.

Definition 6 (Views in RT_0): The *view in RT_0* for a principal $p \in \mathcal{P}$ is defined as a three tuple $v_p = \langle R \subseteq \mathcal{R}, C \subseteq \mathcal{C}, \mathcal{A} \rangle$, where for each $A.R \in R$, all the credentials with head $A.R$ are in C, and $\mathcal{A}$ defines any auxiliary information that p has about the system.

IV. Scoring Role Members

In this section, we show that it is indeed possible to develop authorization scoring functions that meet the requirements set forth in Section III-B. Recall from Section II that there is no single authorization scoring function that makes sense to use in every scenario. To this end, we explore several example constructions that may be used under a variety of assumptions regarding the information available to the evaluator. Note that the scoring functions that we present below are by no means meant to be the *only* functions that trust management systems should use. Instead, they just serve as representative functions that may be useful for the quantitative analysis of authorization proofs. To simplify our discussion, we consider scoring functions of the form $\mathsf{score} : \mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ in this section.

A. One Step Lookahead

Perhaps the simplest case within which to consider the authorization scoring problem is that in which the principal A evaluating some other principal's membership in the role $A.R$ has no information about roles other than $A.R$. In terms of the framework presented in Section III, the system view used by principal A can be defined by the tuple $v_{A.R} = \langle \{A.R\}, ac(A.R), \mathcal{A} \rangle$. This view gives the principal A access to *exactly* the set of credentials defining the role $A.R$. Role memberships defined in terms of simple membership credentials ($A.R \leftarrow P$) can be evaluated locally

Algorithm 1 A simple recursive scoring scheme.

```
1: **Function** $\mathsf{score}(p \in \mathcal{P}, A.R \in \mathcal{R}, v \subseteq \mathcal{V}) : \mathbb{R}$
2: // Filter credentials and initialize storage vector
3: $C = \{c_i \mid c_i \in v.C \wedge \mathsf{head}(c) = A.R\}$
4: Discard all $c_i \in C$ of the form $A.R \leftarrow P', P' \neq P$
5: $\overline{s} = [1, 0, \ldots, 0]$ // vector in $\mathbb{R}^{|C|+1}$
6:
7: **for all** $c_i \in C$ **do**
8:     $\overline{w_i} = v.\mathcal{A}.\mathsf{weight}(c_i)$ // weight vector for $c_i$
9:     **if** $c_i = A.R \leftarrow P$ **then**
10:        $\overline{t} = [1, 1]$
11:    **else if** $\mathsf{body}(c_i) = B_1.R_1 \cap \cdots \cap B_k.R_k$ **then**
12:        $\overline{t} = [1, B_1.\mathsf{score}(p, B_1.R_1), \ldots, B_k.\mathsf{score}(p, B_k.R_k)]$
13:    **else if** $\mathsf{body}(c_i) = A.R_1.R_2$ **then**
14:        Find $B \subseteq A.R_1$ such that $\forall B_j \in B : P \in B_j.R_2$
15:        $\overline{t} = [1, max_{B_j \in B}(B_j.\mathsf{score}(p, B.R_2))]$
16:    **if** $\overline{t}$ contains any 0 entries **then**
17:        $\overline{s}[i] = 0$
18:    **else**
19:        $\overline{s}[i] = \overline{t} \cdot \overline{w_i}$
20:
21: // Get master weight vector and combine all weights
22: $\overline{w} = v.\mathcal{A}.\mathsf{weight}(A.R)$
23: **return** $\overline{s} \cdot \overline{w}$
```

by A, while information regarding roles other than $A.R$ is obtained by recursively issuing requests to the principals defining these roles.

This naive authorization scoring scheme can be viewed as a simplification of the RT proof construction process that only builds proofs of height 1. Specifically, the root of each proof tree is a node representing the role $A.R$ and the leaves of the proof tree are credentials asserting a simple membership in the roles directly used to define membership in $A.R$. This allows principals to hide the details of exactly *how* membership in roles that they define is determined from other principals in the system by replacing sub-proofs encoding proof structure with simple membership credentials asserting that membership has been verified. This is similar in spirit to the proofs of authorization constructed in the Minami-Kotz distributed proof system [24], which hide the structure of a proof from unauthorized users.

Scoring Construction. Evaluating role memberships given only this limited information can be viewed as a process similar to the recursive resolution of DNS queries or the lazy proof construction process used within the Grey distributed proof system [2]. Algorithm 1 shows how such an authorization scoring function can be designed. This naive algorithm assumes that each credential c_i defining some role $A.R$ is associated with a weight vector $\overline{w_i}$ described in A's auxiliary information $\mathcal{A}$. The first entry of this vector is a constant factor and the remaining entries are scaling factors for the scores computed for each role in the body of c_i. the score for a given credential is then computed as the linear combination represented by the dot product of this vector with the vector of scores gathered recursively for each role in the body of c_i.[1] We require that every such $\overline{w_i}$ contains only non-negative entries and that $||\overline{w_i}||_1 = 1$.

We further assume that each role $A.R$ is associated with another weight vector $\overline{w}$, the first entry of which is, again, a constant factor. The second entry in this vector is a scaling factor that is associated with principals who are defined through simple membership to be a member of $A.R$. The remaining entries in $\overline{w}$ are scaling factors for the scores computed for each credential c_i that defines membership in $A.R$ The final score for the role $A.R$ is then computed as the linear combination represented by the dot product of $\overline{w}$ and the scores calculated for each c_i defining $A.R$. Again, $\overline{w}$ is assumed to be encoded in the additional information $\mathcal{A}$ maintained by the principal A, and we require that $\overline{w}$ contains only non-negative entries and that $||\overline{w}||_1 = 1$.

Example. To more concretely demonstrate the scoring function defined in Algorithm 1, consider the following RT_0 role definitions:

$$Univ.auth \leftarrow CS.student \cap ACM.member \qquad (3)$$

$$Univ.auth \leftarrow Univ.techDept.gradStudent \qquad (4)$$

This policy states that computer science students who are ACM members, and graduate students within technical departments at $Univ$ to be members of the role $Univ.auth$. Assume that the weight vector for credential (3) is defined as $[0, 0.7, 0.3]$, which gives more weight to the role defined by the CS department at $Univ$ than to the role defined by the ACM. Further, assume that the master weight vector for $Univ.auth$ is defined as $[0, 0.5, 0.25, 0.25]$, where the entries in this vector reflect a constant factor of 0, and the weights assigned to simple members of $Univ.auth$, credential (3), and credential (4), respectively. This implies that simple membership in $Univ.auth$ is strongly preferred over proofs that involve delegation to other principals, and that both types of delegated proofs are given equal preference.

Properties. Although extremely simple to implement, this naive scoring function can be shown to satisfy a number of the properties identified in Section III-B. In particular, we have the following theorem:

Theorem 1: The function $\mathsf{score} : \mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ defined in Algorithm 1 satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties.

Proof: To prove the above claim, we address each property one at a time:

- **Deterministic.** Note that the score function does not make use of any randomized information. Provided that system policies and weight vectors do not change, two invocations of $\mathsf{score}(p, A.R, v)$ will always return the same value.
- **Simple Ordering.** The check on Line 16 of Algorithm 1 ensures that non-zero scores are only recorded for members of a role, while non-members of a role

[1]The constant factor can be used to adjust the "baseline" score for a proof of access generated using a given credential. Setting this term to zero scores the proof using only the scores returned for each sub-proof.

always receive a zero score. This trivially gives us the simple ordering property.

- **Authorization Relevant.** Line 3 of Algorithm 1 ensures that only credentials defining the role $A.R$ will be considered. This implies that all credentials considered by Algorithm 1 are *relevant* to proving membership in the role $A.R$ by Definition 5. Therefore, irrelevant credentials will not influence the score computed by Algorithm 1.
- **Bounded.** Because we require that $||\overline{w}||_1 = 1$ for all weight vectors used by Algorithm 1, the maximum score that can be computed for a set of proofs of authorization is bounded above by 1. Since these vectors must also contain only non-negative entries, the minimum score that can be computed for a set of proofs of authorization is bounded below by 0.
- **Monotonic.** Since no weight vector $\overline{w_i}$ used by Algorithm 1 can contain negative entries, learning additional role memberships for a principal cannot decrease the score calculated by Algorithm 1. As a result, the score function is monotonic.

Given the above arguments, we can conclude that the function $\mathsf{score} : \mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ defined in Algorithm 1 satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties. ■

We note that the *interpretation* of the value returned by the score function defined by Algorithm 1 is entirely determined by the weight vectors used. For instance, consider the scenario in which the master weight vector for $A.R$ is defined as $[0, \frac{1}{n}, \ldots, \frac{1}{n}]$ and the weight vectors for each credential c_i are defined as $[1, 0, \ldots, 0]$. In this case, a higher score implies that more proofs of authorization have been produced. Specifically, a score of $\frac{k}{n}$ implies that a principal has produced k proofs of membership for $A.R$; if the score computed is 1, then all possible proofs have been produced.

B. Deep Structural Information

At the other end of the information disclosure spectrum lie the system models used in the credential chain discovery and resolution-based proof construction algorithms used by the RT [23] and SecPAL [3] systems, respectively. Essentially, these systems make two similar assumptions: (i) principals can discover all information needed to construct a proof of authorization at runtime, and (ii) any information in the system can be freely disclosed between principals (i.e., there are no credential release policies in place). Together, these assumptions imply that principals need not know anything about the protection state of the system initially, but can learn any *relevant* information that they need to know in order to build a proof (or proofs) of authorization on-demand. In the notation from Section III, this means that a principal's initial view can be defined as $v = \langle \emptyset, \emptyset, \emptyset \rangle$.

Under the above assumptions, it becomes clear that the class of scoring functions described in Section IV-A is undesirable for at least two reasons. First, the computation of proof scores must proceed iteratively: credentials must first be discovered, then be analyzed offline by a human to construct appropriate weight vectors for each role, and finally be used to compute the score associated with a given proof. Second, this class of scoring functions essentially ignores the *overall* structure of a given proof by making scoring decisions *one level* at a time. For these reasons, we now explore an alternate class of authorization scoring functions that better make use of the information that is available during the proof construction process.

Scoring Construction. Note that the structure of an RT_0 proof of authorization is entailed by the set of credentials used to construct the proof. We have particular interest in the set of *minimal* proofs for the role $A.R$. Given a function $\mathsf{sets} : 2^{\mathcal{C}} \times \mathcal{R} \rightarrow 2^{2^{\mathcal{C}}}$ to enumerate the set of minimal proofs discovered for a particular role[2], we can define an authorization scoring function as follows:

$$\mathsf{score}(p, A.R, v) = \sum_{i=1}^{|\mathsf{sets}(v.C, A.R)|} \frac{1}{2}^{i} \qquad (5)$$

At a high level, the interpretation of this scoring function is that if $\mathsf{score}(p_1, A.R, v) < \mathsf{score}(p_2, A.R, v)$, principal p_2 can generate a larger number of proofs of authorization for role $A.R$ than principal p_1. An interesting side effect of this function is that the contribution that each proof of authorization makes to the final score decreases exponentially. Intuitively, this makes sense, as the difference between one and two proofs of authorization is more significant than, e.g., the difference between eight and nine proofs. While interesting, the scoring function defined in Equation 5 makes only limited use of the information encoded in the structure of each proof that is discovered.

In reality, principals may wish to leverage different notions of authorization robustness. To enable this, we first assume that the notion of robustness of interest to a principal can be encoded by a scaling function $\omega : 2^{\mathcal{C}} \times 2^{2^{\mathcal{C}}} \rightarrow [0, 1]$ that quantifies the robustness of a proof of authorization when compared with other proofs. The function ω can then be used to parameterize a function $\mathsf{osets}_\omega : 2^{\mathcal{C}} \times \mathcal{R} \rightarrow 2^{2^{\mathcal{C}} \times \mathbb{R}}$ that computes an ordered list $\langle (C_1, w_1), \ldots, (C_n, w_n) \rangle$ such that $\forall C_i \in sets(C, A.R), \exists w_i : (C_i, w_i) \in \mathsf{osets}_\omega(C, A.R)$ and $\forall (C_i, w_i), (C_j, w_j) \in \mathsf{osets}_\omega(C, A.R) : i \leq j \leftrightarrow w_i \geq w_j$. That is, osets_ω orders the proofs for a particular role in

[2]We do not comment further on the implementation details of the sets function. It is well known that this functionality can be achieved through iterative distributed search (e.g., [3], [23]) or the efficient analysis of previously gathered credentials (e.g., [19]).

decreasing order of their robustness as reported by ω.

$$\mathsf{score}(p, A.R, v) = \sum_{(C_i, w_i) \in \mathsf{osets}_\omega(v.C, A.R)} w_i \cdot \frac{1}{2}^i \qquad (6)$$

Equation 6 first computes all minimal proofs contained within the set C of credentials and then orders these proofs according to the robustness function ω. The final score is then computed as a function of both the number of proofs found, as well as the robustness ratings for each proof. Several interesting notions of proof robustness can in fact be encoded using this type of ω function:

$$\omega_{len}(C_s, \_) = \gamma^{\max_{p \in \mathsf{paths}(C_s)}(\mathsf{length}(p))} \qquad (7)$$

$$\omega_{ind}(C_s, C) = 1 - \frac{\max_{C_i \in C \setminus \{C_s\}}(|C_s \cap C_i|)}{|C_s|} \qquad (8)$$

$$\omega_{li}(C_s, C) = \alpha \cdot \omega_{len}(C_s, \_) + \beta \cdot \omega_{ind}(C_s, C) \qquad (9)$$

In particular, the function ω_{len} defined in Equation 7 computes the scaling factor for a proof C_s by first computing the longest root-to-leaf path in the proof tree entailed by the set C_s, and then raising a constant $\gamma \in [0, 1]$ to this power. Since longer paths will produce lower scaling factors, ω_{len} gives preference to shorter paths. This is sensible if the principal scoring proofs of authorization prefers to see proofs with short delegation chains.

On the other hand, the function ω_{ind} defined in Equation 8 computes the scaling factor for a proof C_s by analyzing how disjoint this proof is when compared to other proofs in $C \setminus \{C_s\}$. In particular, a higher ω_{ind} score implies a more disjoint path. This definition of robustness is important if the principal scoring proofs of authorization is concerned about changes to the policies parameterizing the system. Specifically, a principal that can produce a collection of highly disjoint proofs is less likely to be affected by one (or more) changes to the policies parameterizing the system than a principal who produces a largely overlapping collection of proofs. Finally, the w_{li} function defined in Equation 9 quantifies robustness by means of combining ω_{len} and ω_{ind} using the constants $\alpha, \beta \in [0, 1]$ chosen such that $\alpha + \beta = 1$.

Example. To demonstrate the class of scoring functions defined by Equations 5–9, we will build upon the example presented in Section IV-A. Consider the following set of RT_0 policy credentials:

$$Univ.auth \leftarrow CS.student \cap ACM.member \qquad (10)$$

$$Univ.auth \leftarrow Univ.techDept.gradStudent \qquad (11)$$

$$Univ.techDep \leftarrow CS \qquad (12)$$

$$CS.student \leftarrow CS.ugrad \qquad (13)$$

$$CS.student \leftarrow CS.gradStudent \qquad (14)$$

$$CS.gradStudent \leftarrow Alice \qquad (15)$$

$$ACM.member \leftarrow Alice \qquad (16)$$

Note that two proofs can be constructed that grant Alice access to the role $Univ.auth$, namely $C_1 = \{(10), (14), (15), (16)\}$ and $C_2 = \{(11), (12), (15)\}$. Note that the longest path in C_1 is of length 3 (i.e., $CS.student \leftarrow CS.gradStudent \leftarrow Alice$), while the longest path in C_2 is of length 2 (i.e., $Univ.techDept.gradStudent \leftarrow Alice$). If we let $C = \{C_1, C_2\}$, $v = \langle \emptyset, \{(10), \ldots, (16)\}, \emptyset \rangle$, and $\gamma = 0.9$, we have the following:

$$\omega_{len}(C_1, C) = 0.9^3 = 0.729$$

$$\omega_{len}(C_2, C) = 0.9^2 = 0.81$$

$$\omega_{ind}(C_1, C) = 1 - \frac{|C_1 \cap C_2|}{|C_1|} = 1 - \frac{1}{4} = \frac{3}{4}$$

$$\omega_{ind}(C_2, C) = 1 - \frac{|C_2 \cap C_1|}{|C_2|} = 1 - \frac{1}{3} = \frac{2}{3}$$

It follows that the scoring function defined by Equation 5 calculates $\mathsf{score}(Alice, Univ.auth, v) = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$. Similarly, the scoring function defined by Equations 6 and 7 gives us that $\mathsf{score}(Alice, Univ.auth, v) = 0.9^2 \cdot \frac{1}{2} + 0.9^3 \cdot \frac{1}{4} = 0.58725$. Lastly, the scoring function defined by Equations 6 and 8 gives us that $\mathsf{score}(Alice, Univ.auth, v) = \frac{3}{4} \cdot \frac{1}{2} + \frac{2}{3} \cdot \frac{1}{4} = \frac{13}{24} \approx 0.5417$.

Properties. The class of scoring functions described by Equations 5–9 can be shown to satisfy a number of the important properties identified in Section III-B. Since Equation 5 can trivially be represented by Equation 6 when osets is parameterized by the function $\omega(\_, \_) = 1$, we have the following theorem:

Theorem 2: The class of scoring functions $\mathsf{score} : \mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ represented by Equation 6 satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties, provided that the scaling function $\omega : 2^{\mathcal{C}} \times 2^{2^{\mathcal{C}}} \rightarrow [0, 1]$ used to parameterize the osets function is deterministic.

Proof: To prove the above claim, we will focus on each property individually:

- **Deterministic.** The ω function used to parameterize osets is the only potential source of non-determinism in Equation 6. Since ω is assumed to be deterministic, the class of scoring functions score represented by Equation 6 is therefore also deterministic.
- **Simple Ordering.** Note that the $\mathsf{osets}_\omega(C, A.R)$ function locates and rank-orders the proofs of authorization for the role $A.R$ that are contained in the set C of credentials. As a result, a principal who cannot produce *any* proofs for the role $A.R$ will always be assigned a score of 0. Further, principals who can produce one or more proofs for the role $A.R$ will always receive a score of *at least* 0. This trivially satisfies the simple ordering property.
- **Authorization Relevant.** By Definition 5, every credential in a minimal proof for some role $A.R$ is relevant

to the proof of membership in $A.R$. Since osets only ranks and returns minimal proofs for $A.R$, Equation 6 does not consider *any* irrelevant information when scoring proofs of authorization. As such, irrelevant information cannot influence the score produced by Equation 6.

- **Bounded.** Note that sum of the infinite geometric series $\sum_{i=1}^{\infty} \frac{1}{2}^i$ converges to 1. Since every term in this summation is positive, we have that $\forall_{1 \leq j} : \sum_{i=1}^{j} \frac{1}{2}^i < \sum_{i=1}^{j+1} \frac{1}{2}^i$. As a result, the sum of any sub-series of the infinite series will converge to some value in the range $[0, 1)$. Equation 6 is nothing more than such a summation in which each term is scaled by a weight $w_i \in [0, 1]$. As a result, Equation 6 is bounded below by 0 and bounded above by 1.
- **Monotonic.** To prove the monotonicity of Equation 6, we proceed by induction. We first assume that principal p has previously discovered the (ordered) collection of proofs and weights $(C_1, w_1), \ldots, (C_n, w_n)$ for the role $A.R$. The base case that we must consider is that a new pair (C_s, w_s) is discovered such that no weight w_i is less than w_s. In this case, this new pair will introduce a new term to the end of the summation calculated by Equation 6, thereby increasing principal p's score for the role $A.R$.
  Assume that (C_s, w_s) can be inserted before up to n terms in the sequence of (c_i, w_i) pairs while still preserving the monotonicity requirement. Now, assume that p has previously found proofs of authorization with the sequence of weights $S = (C_1, w_1), \ldots, (C_i, w_i), \ldots, (C_{i+n}, w_{i+n})$ and has now discovered a (C_s, w_s) pair such that $w_s > w_i$, thereby needing to be inserted before $n + 1$ terms in the sequence S. We first note that replacing (C_i, w_i) with (C_s, w) will generate a sequence S' that—when used in conjunction with Equation 6—will produce a score greater than that produced using S, since $w_s > w_i$ and all other terms are the same. By the inductive hypothesis, (C_i, w_i) can then be re-inserted before the n final terms of S' while still preserving monotonicity.

We have therefore shown that the class of scoring functions $\mathsf{score} : \mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ represented by Equation 6 satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties, provided that the scaling function $\omega : 2^{\mathcal{C}} \times 2^{2^{\mathcal{C}}} \rightarrow [0, 1]$ used to parameterize the osets function is deterministic. ■

As was discussed earlier in this section, it is also possible for the score function defined by Equation 6 to satisfy the *interpretation* property. In particular, when using the trivial scaling function $\omega(\_, \_) = 1$, a higher score implies that more paths have been found. Similarly, when other notions of robustness are encoded by representative ω functions, $\mathsf{score}(p, A.R, v) > \mathsf{score}(p', A.R, v)$ implies that principal p can produce more robust proofs of authorization that principal p', relative to the notion of robustness encoded by the particular ω function used.

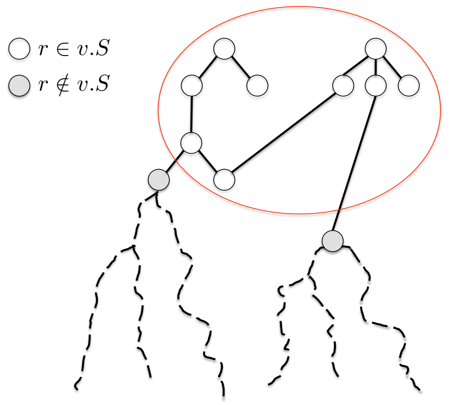

Figure 2. One possible scenario in which hybrid authorization scoring strategies could be beneficial. The oval denotes the security domain of the principal carrying out the authorization scoring process.

C. Hybrid Scoring Functions

Although the authorization scoring functions discussed in Sections IV-A and IV-B are useful, they in fact represent two opposite extremes in terms of the information available to the principal scoring proofs of authorization. The recursive scoring function described by Algorithm 1 assumes that this principal has access only to credentials defined within her domain, while the scoring function encoded by Equations 6–9 assumes that incomplete information is discovered at runtime. In many cases, the information available to the principal scoring proofs of authorization is likely to fall somewhere between these two points.

Figure 2 illustrates one such scenario. In this situation, the principal (Alice) scoring proofs of authorization is assumed to have complete knowledge of the set of roles $v_A.S$ within the domain encoded in her view v_A. However, she is also able to use, e.g., credential chain discovery techniques to discover credentials defining roles outside of her domain, with the proviso that she may not be able to uncover *every* credential defining a particular role. Given that she has complete knowledge of not only the structure of policies within her domain, but also of the *semantics* of the roles involved in these policies, Alice may wish to fine-tune her mechanism for scoring membership within these roles to reflect her "insider knowledge." For instance, she could accomplish this by developing very specific weight vectors for use in conjunction with Algorithm 1.

At the same time, rather than delegating the scoring of membership in roles outside of $v_A.S$ to the parties defining those roles—as in Algorithm 1—Alice may wish to leverage her ability to carry out distributed credential discovery processes. Given that the roles appearing on the "horizon" of Alice's domain appear within credentials defined in her domain, she is likely to be aware of their semantics (e.g., that a BestBuy membership is easier to come by than a DoD clearance). Although she may not be able to reliably interpret *every* role used to prove membership in these horizon roles, her knowledge of the horizon role itself may guide the choice the particular ω function used reason about the robustness of proofs of membership of that role. In such a scenario, Alice may wish to *sequentially compose* the scoring functions defined in Sections IV-A and IV-B.

A More General Problem. The above scenario is really a special case of a more general question: *Given two proof scoring functions that satisfy the requirements set forth in Section III-B, is it possible to combine these functions in such a way that the composite function retains the desirable characteristics of the original functions?* In order to show that this is, indeed, the case, we must first formalize the notions of *horizon*, *sequential composition*, and *order-preserving homomorphism*.

Definition 7 (Horizon): The *horizon* of a view v is defined as the set of resources $\mathsf{horizon}(v) = \{r \mid \exists c \in ac(v.S) : r \in \mathsf{body}(c) \wedge r \notin v.S\}$. That is, $\mathsf{horizon}(v)$ contains all resources mentioned in the body of policies protecting resources in $v.S$ that are not themselves in $v.S$.

Definition 8 (Sequential Composition): Assume that we have a view v, a principal p, a resource r, and two authorization scoring functions score_1 and score_2. We say that score_1 is *sequentially composed* with score_2 if there exists a resource $r' \in \mathsf{horizon}(v)$, a principal p', and a view v' such that $\mathsf{score}_2(p', r', v')$ is calculated when calculating $\mathsf{score}_1(p, r, v)$.

Definition 9 (Order-Preserving Homomorphism): Let $\mathsf{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \to \mathcal{T}_1$ and $\mathsf{score}_2 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \to \mathcal{T}_2$ be two authorization scoring functions. Let $t_1 \in \mathcal{T}_1$ (resp. $t_2 \in \mathcal{T}_2$) be a threshold such that if $\mathsf{score}_1(p, s, v) \leq t_1$ (resp. $\mathsf{score}_2(p, s, v) \leq t_2$) then p cannot access resource s. Similarly, if $\mathsf{score}_1(p, s, v) > t_1$ (resp. $\mathsf{score}_2(p, s, v) > t_2$) then p can access resource s. A function $f : \mathcal{T}_2 \to \mathcal{T}_1$ is an *order-preserving homomorphism* from $\mathcal{T}_2$ to $\mathcal{T}_1$ if and only if (i) $t \leq t_2 \to f(t) \leq t_1$, (ii) $f(t_2) = t_1$, and (iii) $t > t_2 \to f(t) > t_1$.

In the discussion above, Alice desires to sequentially compose the scoring function defined in Algorithm 1 with the scoring function defined by Equations 6–9. Furthermore, the identity function $f(x) \mapsto x$ is trivially an order-preserving homomorphism between the (identical) totally-ordered ranges of these two functions. We now have the necessary machinery to prove the following theorem:

Theorem 3: Let $\mathsf{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \to \mathcal{T}_1$ and $\mathsf{score}_2 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \to \mathcal{T}_2$ be two authorization scoring functions that satisfy the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties. If there exists an order-preserving homomorphism f between $\mathcal{T}_2$ and $\mathcal{T}_1$, then the sequential composition of score_1 with score_2 is also *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic*.

Proof: Showing that the sequential composition of score_1 with score_2 is deterministic and bounded is straightforward. Using the (deterministic) outputs of score_2 as inputs to the (deterministic) function score_1 will clearly result in a deterministic output. Furthermore, since f maps the outputs of score_2 into a range that can be considered by score_1 and score_1 is bounded, then the sequential composition of score_1 with score_2 is also bounded.

Since score_2 is authorization relevant, by Definition 5 it is not influenced by irrelevant credentials. As a result, score_1 will not be indirectly influenced by irrelevant credentials via score_2. Since score_1 is authorization relevant, it is not influenced by irrelevant credentials that it considers directly. This implies that the sequential composition of score_1 with score_2 is also uninfluenced by irrelevant credentials, and thus authorization relevant by Definition 5. By a similar argument, the sequential composition of score_1 with score_2 is also monotonic: relevant credentials considered directly by score_1 or indirectly via score_2 can only increase the output of sequential composition of score_1 with score_2.

Finally, we must address the simple ordering property. Note that the order-preserving homomorphism f allows score_1 to reliably use the value $f(\mathsf{score}_2(p', r', v'))$ to determine whether P' is authorized to access the horizon resource r'. Given that score_2 is assumed to be deterministic and to satisfy the simple ordering property, this interpretation of $f(\mathsf{score}_2(p', r', v'))$ is guaranteed to be correct. Furthermore, since score_1 is also assumed to be deterministic and to satisfy the simple ordering property, the output value $\mathsf{score}(p, r, v)$ that is computed using this information is also guaranteed to correctly reflect P's ability to access the resource r. As a result, the sequential composition of score_1 with score_2 satisfies the simple ordering property. ■

As a result of the above theorem, we immediately have the following corollary stating that the two classes of authorization scoring functions discussed earlier in this section can be sequentially composed with one another:

Corollary 1: The result of sequentially composing the authorization scoring functions defined by Algorithm 1 and Equations 6–9 using the order-preserving homomorphism $f(x) \mapsto x$ is an authorization scoring function that is also *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic*.

It is also interesting to note that Theorem 3 tells us that principals can make use of any number of authorization scoring functions to compute authorization scores for each of the resources along the horizon of their domain. More concretely, in the scenario described by Figure 2, Alice can use a different authorization scoring function for each grey node on the horizon of her domain:

Corollary 2: Let $\mathsf{score} : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1$ be a authorization scoring function that satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties and let v be a view. The result of sequentially composing score with an arbitrary any number of other *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* authorization scoring functions along $\mathsf{horizon}(v)$ is an authorization scoring function that is also *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic*.

Lastly, the ability to sequentially compose two authorization scoring functions inductively gives us the ability to sequentially compose any number of such functions:

Corollary 3: Let $\mathsf{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1, \ldots, \mathsf{score}_n : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_n$ be proof scoring functions that satisfy the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties, and let $f_{n-1} : \mathcal{T}_n \rightarrow \mathcal{T}_{n-1}, \ldots, f_1 : \mathcal{T}_2 \rightarrow \mathcal{T}_1$ be order-preserving homomorphisms mapping between the ranges of these functions. The result of sequentially composing $\mathsf{score}_1, \ldots, \mathsf{score}_n$ using $f_1, \ldots, f_{n-1}$ is also authorization scoring function that is also *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic*.

V. Scoring Non-Members of a Role

The ability to construct machine verifiable proofs of authorization is one of the greatest strengths of trust management approaches to authorization. However, the unyielding reliance on these rigid proofs is also a considerable pitfall. Since the proof construction process is guided by the policies within the system, these policies must exhaustively cover *every* possible scenario in which access to some resource should be granted. As a result, these systems can fail during emergencies or other unexpected circumstances simply because policy administrators did not consider some particular case when writing the policies for their domain.

When this type of failure is manifest in the physical world, policies are often overridden in accordance with judgement calls made by qualified individuals. However, making these types of decisions in an automated system can be tricky. The ability to assess how close some unauthorized principal is to being granted access to a resource could, to a limited extent, help to automate this process. At the very least, this can act as a primitive risk metric that helps classify users who are "close" to satisfying a policy from those who are "far" from satisfying the policy. This, in turn, could be useful for helping to guide the decision making of a human auditor who can be brought into the process. In this section, we explore how these types of decisions can be made within the context of the framework presented in this paper.

Scoring Construction. To describe one possible mechanism for scoring the non-members of a particular role, we present modifications to the authorization scoring function described by Equations 6–9. Intuitively, the scoring construction presented in this section builds upon the earlier construction by also considering *partial* proofs. In order to identify partial proofs, we need to first understand what types of proofs may be produced for a particular role. We formalize this notion through the use of *canonical* proofs:

Definition 10 (Canonical Proof of Authorization): A *canonical proof of authorization* for a principal p and a role $A.R$ is a minimal set of credentials C from the universe of all possible credentials such that $F(p, A.R, C) = TRUE$. We denote by $\mathsf{csets}(p, A.R)$ the set of all canonical proofs for the principal p and the role $A.R$.

Given a canonical proof C for a principal p and a role $A.R$, it is entirely possible that some subset C' of the credentials in C are not actually defined within the system. The existence of the canonical proof C simply implies that *if* each credential $c \in C$ is discovered, then p will be granted access to $A.R$. We also note that, a user need not be omniscient in order to materialize $\mathsf{csets}(p, A.R)$: this set of canonical proofs is actually uncovered as a side-effect of the RT credential chain discovery process [23]. Given $\mathsf{csets}(p, A.R)$, we can then construct the set of all partial proofs regarding p's membership in $A.R$ using the function $\mathsf{psets} : \mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow 2^{\mathcal{C}} \times 2^{\mathcal{C}}$ defined below:

$$\begin{aligned} \mathsf{psets}(p, A.R, v) = \{(C_p, C_c) \mid C_c \in \mathsf{csets}(p, A.R) \quad (17) \\ \wedge C_p = v.C \cap C_c \wedge C_p \neq C_c\} \end{aligned}$$

The above function computes all partial matches between the set of credentials in $v.C$ and the canonical proofs in $\mathsf{csets}(p, A.R)$. Given this set of (partial, canonical) proof pairs, we can now evaluate the *quality* of a partial proof. To do this, we introduce the functions $\mathsf{leaves} : 2^{\mathcal{C}} \rightarrow 2^{\mathcal{C}}$ and $\psi : 2^{\mathcal{C}} \times 2^{\mathcal{C}} \rightarrow [0, 1]$:

$$\mathsf{leaves}(C) = \{c \in C \mid c \text{ of the form } A.R \leftarrow p\} \quad (18)$$

$$\psi(C_p, C_c) = \frac{|\mathsf{leaves}(C_p \cap C_c)|}{|\mathsf{leaves}(C_c)|} \quad (19)$$

The leaves function extracts the simple membership credentials from a set C of credentials. The ψ function then calculates the quality of a partial proof by calculating the percentage of the simple membership credentials in the corresponding canonical proof that are contained in the partial proof. To see why credentials other than simple memberships are ignored by ψ, consider the chain of delegation $A.R \leftarrow B_1.R_1 \leftarrow \cdots \leftarrow B_n.R_n$. The canonical proof for principal p in role $A.R$ would contain n simple containment credentials and one simple membership credential (i.e., the credential $B_n.R_n \leftarrow p$). Note that of these $n+1$ credentials, only one is specific to p. If n is large, then p would appear "close" to being a member of $A.R$ even though she does not meet *any* of the requirements for membership. The ψ function eliminates this type of bias.

If we let $\mathsf{opsets}(p, A.R, v)$ represent the set $\{(w, C_p, C_c) \mid (C_p, C_c) \in \mathsf{psets}(p, A.R, v) \wedge w = \psi(C_p, C_c)\}$ sorted in decreasing order of w_i, we can then define an authorization scoring function capable of scoring non-members of roles:

$$\phi(x) = \begin{cases} 1 & \text{if } x \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

$$\begin{aligned} \mathsf{score}(p, A.R, v) = \phi(|\mathsf{sets}(v.C)|) \\ + \alpha \sum_{(w_i, C_i) \in \mathsf{osets}_\omega(v.C, A.R)} w_i \cdot \frac{1}{2}^i \\ + \beta \sum_{(w, C_p, C_c)_i \in \mathsf{opsets}(p, A.R, v)} w \cdot \frac{1}{2}^i \end{aligned} \tag{21}$$

Under the constraint that $\alpha + \beta = 1$, the range of the above scoring function is $[0, 2)$. Furthermore, this function has the properties that (i) $\mathsf{score}(p, A.R, v) \geq 1 \leftrightarrow P$ is a member of $A.R$ and (ii) $\mathsf{score}(p, A.R, v) < 1 \leftrightarrow P$ is *not* a member of $A.R$.

Example. To briefly demonstrate the class of scoring functions defined in this section, consider the following set of RT_0 policy credentials:

$$Univ.auth \leftarrow CS.student \cap ACM.member \tag{22}$$

$$Univ.auth \leftarrow Univ.techDept.gradStudent \tag{23}$$

$$Univ.techDep \leftarrow CS \tag{24}$$

$$CS.student \leftarrow CS.ugrad \tag{25}$$

$$CS.student \leftarrow CS.gradStudent \tag{26}$$

$$CS.ugrad \leftarrow Bob \tag{27}$$

In this scenario, it is clear that Bob cannot prove membership in the $Univ.auth$ role. However, there exists one canonical proof that overlaps with the simple memberships that Bob possesses: $C_c = \{(22), (25), (27), ACM.member \leftarrow Bob\}$. Bob's corresponding partial proof is $C_p = \{(22), (25), (27)\}$. Since $\psi(C_p, C_c) = \frac{1}{2}$, we have that $\mathsf{score}(Bob, Univ.auth, v) = \frac{\beta}{4}$. Due to the fact that $\beta \leq 1$, Bob's score for the role $Univ.auth$ falls below the membership threshold of 1.

Properties. The class of scoring functions described in this section possesses the same properties as each of the classes of scoring functions described in Section IV. Namely, we have the following theorem:

Theorem 4: The class of non-member scoring functions $\mathsf{score} : \mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ represented by Equations 17–21 satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties, provided that the scaling function $\omega : 2^{\mathcal{C}} \times 2^{2^{\mathcal{C}}} \rightarrow [0, 1]$ used to parameterize the osets function is deterministic.

We omit the details of the proof of Theorem 4, as it essentially mirrors the proof of Theorem 2. We further note that the authorization scoring function described by Algorithm 1 can also be modified to score the non-members of a particular role. We do not discuss these modifications, however, as they present little novelty beyond that which has already been discussed in this section.

VI. Discussion

This work is a first step towards quantitative analysis of trust management proofs of authorization. Many interesting directions can be explored based on the general framework and the current design of authorization scoring functions.

Extensions to Richer Policy Models. For simplicity, our discussion so far is based on RT_0. However, our general framework can easily accommodate other trust models with richer structures and semantics. One natural extension is to support RT_1, which supports parameterized views, or CTM which combines credential-based trust and reputation-based trust [20]. Clearly, parameterized views and reputations greatly expand the design space of authorization scoring functions. In particular, besides credential structures, the strength of a principal's proofs of authorization may be further refined in terms of role attributes and reputations. We may also consider a variety of aggregations of role attributes when considering disjoint minimal proofs.

Another important extension is to consider other types of auxiliary information besides weights of roles. One interesting type of auxiliary information is the correlation between roles. For example, IEEE members and ACM members might be correlated; i.e., a member of IEEE is more likely to be a member of ACM, and vice versa. If this relationship is captured in $\mathcal{A}$, then an IEEE member might be *partially trusted* to access resources that are explicitly accessible to ACM members. Essentially, these types of correlations help us better model risks when temporarily granting access to unauthorized principals.

Evaluating Top-k Style Queries. In this paper we focus on the semantics of authorization scoring functions. One important issue is to efficiently evaluate these functions. This is particularly so when answering top-k style queries. The naive approach, which first computes the authorization score of each principal and then selects k principals with the highest scores, is unlikely to be efficient in large decentralized systems. Though top-k queries have been extensively studied in database research (for a survey see [13]), it is unclear whether existing database techniques can be applied in our problem, as authorization scoring functions tend to be much more complicated than simple linear combination of multiple attributes. Also, as most trust management systems are decentralized in nature, how to divide the load of query evaluation among multiple domains may have a significantly impact on efficiency. It would be interesting to investigate the extent to which distributed query answering techniques can be adopted for trust management queries.

VII. Related Work

Digital credentials are one of the main techniques for access control in cross-domain collaboration and resource sharing. A large amount of work has been conducted in the areas of trust management [3]–[5], [22], trust negotiation [21], [28], [31], [32], and distributed proofs [2], [15], [24]. Most existing work focuses on compliance checking, i.e., given a principal and a set of credentials, determine whether that principal is authorized to access a resource. Researchers have also investigated security properties of trust management policies such as safety, availability, and liveness. None of the above work takes a quantitative approach to evaluating the strength of a principal's proofs of authorization.

Yao et al. [30] proposed an interesting point-based trust policy model, where each credential is assigned some number of points. To access a resource, one has to reveal a set of credentials whose sum of points is higher than a certain threshold. Yao et al. focus on privacy-preserving compliance checking of such policies through secure multi-party computation. By contrast, this paper aims to establish a formal framework to support a variety of inferences regarding the strength of a principal's proofs of authorization above and beyond simple compliance checking. Quantitative trust establishment is typically discussed in the context of reputation-based trust management systems (e.g., [16], [17], [29]). These systems develop computational models of reputation based on feedbacks issued between entities. Reputation models often involve different forms of aggregation of feedbacks, which are fundamentally different from credential-based trust that is the focus of this paper.

Risk-based access control [1], [8], [11], [12], [14], [27], [33] has recently been proposed as a mechanism for providing flexible authorization in complex and unpredictable environments. The key idea is to maximize productivity while bounding risk instead of eliminating risk. Some work quantifies risk through reputations, while others manually allocate "risk tokens" to principals that can later be used to exchange for access rights. On the other hand, the techniques presented in this paper can be used to automate the risk assessment process for unauthorized users, and provide robustness analysis for authorized principals.

Inferences over uncertain information have long been studied in a variety of application contexts. Techniques such as Bayesian networks [6], fuzzy logic [9] and Dempster-Shafer theory [10], [26] seek to determine the strength of a proposition based on *uncertain* facts and logical inferences. In this paper, authorization scoring functions measure the strength of proofs of authorization using *completely certain* information. That is, all credentials considered are digitally signed and thus known to be true without any level of uncertainty. Even for unknown information (e.g., credentials outside of a view) our framework relies on either inquiring a domain that can answer queries about some credentials or dynamically discovering credentials. In either case, the obtained information is certain, rather than probabilistic.

VIII. Conclusions

In this paper, we argue that the often-ignored structural information gathered during the construction of proofs of authorization has a variety of uses in user-to-user and user-to-ideal analysis tasks. To this end, we develop a framework for reasoning about the quantitative analysis of trust management proofs of authorization, and articulate sets of *necessary* requirements and *desirable* features for authorization scoring functions. Within the context of the RT_0 policy language, we develop authorization scoring functions under a variety of information availability assumptions and prove that these functions meet all of the aforementioned requirements. We further identify the conditions under which authorization scoring functions can be safely composed, and show that the functions developed in this paper can be composed with one another. Lastly, we extend our techniques to score not only complete proofs of authorization, but also the incomplete proofs generated by unauthorized principals. The score computed for incomplete proofs of authorization acts as a primitive risk metric that can be used to automate and/or guide decision making processes during crises or other unexpected circumstances.

Acknowledgments

This research was supported in part by the National Science Foundation under awards CCF-0916015, CNS-0964295, CCF-0914946 and CNS-0747247.

References

[1] B. Aziz, S. N. Foley, J. Herbert, and G. Swart. Reconfiguring role based access control policies using risk semantics. *Journal of High Speed Networks*, 15(3):261–273, 2006.

[2] L. Bauer, S. Garriss, and M. K. Reiter. Distributed proving in access-control systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 81–95, May 2005.

[3] M. Y. Becker, C. Fournet, and A. D. Gordon. Design and semantics of a decentralized authorization language. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium*, pages 3–15, 2007.

[4] M. Y. Becker and P. Sewell. Cassandra: Distributed access control policies with tunable expressiveness. In *Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 159–168, June 2004.

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the IEEE Conference on Security and Privacy*, pages 164–173, May 1996.

[6] C. Borgelt and R. Kruse. *Graphical Models: Methods for Data Analysis and Mining*. John Wiley & Sons, 2002.

[7] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in web-based social networks. *ACM Transactions in Information and System Security*. to appear.

[8] P.-C. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger. Fuzzy multi-level security: An experiment on quantified risk-adaptive access control. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2007.

[9] E. Cox. *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, Maintaining Fuzzy Systems*. AP Professional, 1994.

[10] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 38(2):325–339, 1967.

[11] N. Dimmock, J. Bacon, D. Ingram, and K. Moody. Risk models for trust-based access control(tbac). In *Proceedings of the International Conference on Trust Management (iTrust)*, pages 364–371, 2005.

[12] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies*, pages 156–162, 2004.

[13] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys*, 40(4):1–58, 2008.

[14] Horizontal integration: Broader access models for realizing information dominance. Technical Report JSR-04-132, MITRE Corporation JASON Program Office, Dec. 2004.

[15] T. Jim. SD3: A trust management system with certified evaluation. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 106–115, May 2001.

[16] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, 2007.

[17] S. Kamvar, M. Schlosser, and H. Garcia-Molina. EigenRep: Reputation Management in P2P Networks. In *Twelfth International World Wide Web Conference*, 2003.

[18] A. J. Lee and M. Winslett. Enforcing safety and consistency constraints in policy-based authorization systems. *ACM Transactions on Information and System Security (TISSEC)*, 12(2):1–33, 2008.

[19] A. J. Lee and M. Winslett. Towards an efficient and language-agnostic compliance checker for trust negotiation systems. In *Proceedings of the 3rd ACM Symposium on Information, Computer and Communications Security (ASIACCS 2008)*, pages 228–239, Mar. 2008.

[20] A. J. Lee and T. Yu. Towards a dynamic and composite model of trust. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 217–226, June 2009.

[21] J. Li, N. Li, and W. H. Winsborough. Automated trust negotiation using cryptographic credentials. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 46–57, Nov. 2005.

[22] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130, May 2002.

[23] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, 2003.

[24] K. Minami and D. Kotz. Secure context-sensitive authorization. *Journal of Pervasive and Mobile Computing (PMC)*, 1(1):123–156, 2005.

[25] K. E. Seamons, M. Winslett, and T. Yu. Limiting the disclosure of access control policies during automated trust negotiation. In *Symposium on Network and Distributed System Security (NDSS)*, Feb. 2001.

[26] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[27] N. Tuptuk and E. Lupu. Risk based authorisation for mobile ad hoc networks. In *Proceedings of the First International Conference on Autonomous Infrastructure, Management and Security*, pages 188–191, June 2007.

[28] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, Jan. 2000.

[29] L. Xiong and L. Liu. A reputation based trust model for peer-to-peer ecommerce communities. In *IEEE International Conference on E-Commerce (CEC)*, 2003.

[30] D. Yao, K. B. Frikken, M. J. Atallah, and R. Tamassia. Point-based trust: Define how much privacy is worth. In *ICICS*, pages 190–209, 2006.

[31] T. Yu, M. Winslett, and K. E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security*, 6(1):1–42, Feb. 2003.

[32] C. C. Zhang and M. Winslett. Distributed authorization by multiparty trust negotiation. In *ESORICS*, pages 282–299, 2008.

[33] L. Zhang, A. Brodsky, and S. Jajodia. Toward information sharing: Benefit and risk access control (BARAC). In *Proceedings of the 7th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 45–53, June 2006.