A Unified Scheme for Resource Protection in Automated Trust Negotiation

Ting Yu
Department of Computer Science
University of Illinois at Urbana-Champaign
tingyu@cs.uiuc.edu

Marianne Winslett
Department of Computer Science
University of Illinois at Urbana-Champaign
winslett@cs.uiuc.edu

Abstract

Automated trust negotiation is an approach to establishing trust between strangers through iterative disclosure of digital credentials. In automated trust negotiation, access control policies play a key role in protecting resources from unauthorized access. Unlike in traditional trust management systems, the access control policy for a resource is usually unknown to the party requesting access to the resource, when trust negotiation starts. The negotiating parties can rely on policy disclosures to learn each other's access control requirements. However, a policy itself may also contain sensitive information. Disclosing policies' contents unconditionally may leak valuable business information or jeopardize individuals' privacy. In this paper, we propose UniPro, a unified scheme to model protection of resources, including policies, in trust negotiation. UniPro improves on previous work by modeling policies as first-class resources, protecting them in the same way as other resources, providing fine-grained control over policy disclosure, and clearly distinguishing between policy disclosure and policy satisfaction, which gives users more flexibility in expressing their authorization requirements. We also show that UniPro can be used with practical negotiation strategies without jeopardizing autonomy in the choice of strategy, and present criteria under which negotiations using UniPro are guaranteed to succeed in establishing trust.

1 Introduction

Computer systems traditionally have had closed, centrally managed security domains. Every entity that can take actions within such a system has one or more identities in that domain. The system grants or denies an entity's requests to access certain resources according to its access control policies and the authenticated identities of the requester. The underlying assumption is that entities in the system already know each other. Therefore, trust can be easily established based on each other's identity. Further, without obtaining a local identity, an entity will not be able to interact with the system and gain access to the system's resources.

As we move towards a globally internetworked infrastructure, open systems like the Internet provide an environment where two or more parties who are virtually strangers to each other can make connections and do business together. Such interactions often involve release of sensitive information and remote access to a party's local resources. Mutual trust between the two parties is crucial in such an environment. Obviously, establishing trust based on identity is not a feasible approach. Parties may come from different security domains and they often will not have any preexisting relationship. Therefore, identity information such as user names and passwords, or identity certificates, is usually inadequate to determine whether or not a party should be trusted. Instead, the *properties* of the participants, e.g., employment status, group membership, citizenship, will be most relevant. Most current security approaches in the Internet are still identity-based, which requires a new client to pre-register with a service, in order to obtain a local login, capability, or credential before requesting that service. Such practices are usually offered in a "take-it-or-leave-it" fashion, i.e., the clients either *unconditionally* disclose their information to the server, or do not get the service at all. There is little chance for the clients to apply their own access control policies for their information, and decide accordingly whether the server is trustworthy enough so that sensitive information can be disclosed.

The approach of automated trust negotiation differs from traditional identity-based access control systems mainly in the following aspects:

1. Trust between two strangers is established based on parties' properties, which are proven through disclosure of digital credentials. A digital credential is a verifiable, unforgeable, digitally signed assertion by a credential issuer about the properties of the parties mentioned in the credential. A credential often contains a public key of one or more of the parties it mentions, so that those parties can prove that the credential de-

scribes them. Digital credentials can be implemented via X.509 certificates [8] or private credentials [4], for example.

2. Every party can define access control policies to control outsiders' access to their sensitive resources. These resources can include services accessible over the Internet, roles in role-based access control systems, credentials, policies, and capabilities in capability-based systems.

3. In the approaches to trust negotiation developed so far, two parties establish trust directly without involving trusted third parties, other than credential issuers[1]. Since both parties have access control policies, trust negotiation can employ a peer-to-peer architecture, where a client and server are treated equally. Instead of a one-shot authorization and authentication, trust is established incrementally through a sequence of bilateral credential disclosures. Less sensitive credentials are disclosed first. Later on, when a certain level of trust has been established, more sensitive credentials can be disclosed.

For example, suppose Alice wants to purchase some medicine from an online drug store, www.cheapmedicine.com. Because this is the first time that Alice has placed an order at this drug store, CheapMedicine requires Alice to present a certified prescription and a valid digital credit card. For Alice, both the prescription and her credit card contain sensitive private information. So she may in turn require the drug store to show a valid pharmacy license issued by the Food and Drug Administration and a membership certificate issued by the Better Business Bureau.

As we can see from the above example, access control policies play a key role in trust negotiation. Unlike traditional decentralized systems, where access control policies are either publicly visible (such as in traditional file systems) or completely hidden (such as the configuration of a firewall for a local network), one's access control policies may need to be dynamically disclosed during the process of trust establishment. In many situations, either because the trust between two strangers is so limited, or because of the nature of the ongoing transactions, disclosing the contents of an access control policy to a stranger may leak valuable business information or jeopardize one's privacy. For example, suppose that an online store gives a special discount to employees of its business partners. If the policy, which contains the list of its business partners, is shown to arbitrary requesters for the service, then by merely looking at the policy, an outsider will know who is partnering with the store.

To give access control policies the protection they need, this paper proposes UniPro, a Unified Resource Protection Scheme for trust negotiation. In section 2, we introduce desiderata for protection of sensitive access control policies, and show how previously proposed approaches measure up to the desiderata. In section 4, we present UniPro and show that unlike previous proposals, UniPro treats access control policies as first-class resources, provides fine-grained control over policy disclosure, and clearly distinguishes between policy disclosure and policy satisfaction. We also prove that UniPro generalizes previous proposals, and explain why in spite of this, we believe that other ways of protecting policies still have an important role to play in trust negotiation. Because each party should have wide autonomy in how it conducts a trust negotiation—autonomy limited only by the need to interoperate correctly with another party during negotiation—section 5 discusses the impact of UniPro on the theory of strategy interoperability, and the tradeoff between the greater freedom in policy protection that UniPro provides and the potential for unnecessary negotiation failure. We also explain how we expect these unnecessary failures to be avoided in practice. Section 7 concludes the paper and briefly discusses possible future work.

2 Sensitive Policies and Their Protection

Realistic access control policies tend to contain sensitive information, because the details of Alice's policy P for disclosure of credential C tend to give hints about C's contents. For example, if there is any information in C that is so sensitive that Alice chooses to control its disclosure very tightly, then it is possible to guess at the nature of that information by looking at P: is her parole officer allowed to see C? Her welfare case worker? The local HIV, cancer, or mental health clinic? An Enron employees' stock sale web page whose policy spelled out exactly who could sell stock and when might have raised many eyebrows. More generally, a company's internal and external policies are part of its corporate assets, and it will not wish to indiscriminately broadcast its policies in their entirety.

Example 1. [11] Suppose a web page's access control policy states that in order to access documents of a project in the site, a requester should present an employee ID issued either by Microsoft or by IBM. If such a policy can be shown to any requester, then one can infer with high confidence that this project is a cooperative effort of the two companies.

Example 2. (Inspired by examples in [2]) Coastal Bank's loan application policy says that a loan applicant must be a customer of the bank who is not on the bank's bad-customer list. If this policy is fully disclosed to a requester, then an outsider can easily learn who the

[1] Mobile devices that do not have enough power to negotiate directly are exceptions to this rule.

bank's bad customers are, which is very sensitive business information.

In the first example, the correlation between two constraints in a single policy reveals certain properties of the protected resource. The policy in the second example directly refers to sensitive local information that should be protected. One obvious way to prevent such information leakage is to selectively disclose part of a policy. For instance, in Example 1, we can first ask the requester to show an employee ID. After receiving the credential, we check whether it is issued by either Microsoft or IBM. Similarly, in Example 2, Coastal Bank may only ask for a customer ID and perform the check on the bad customer list after the credential is received. However, this approach may cause a dispute between the two negotiating parties. If we simply remove those sensitive constraints when disclosing a policy, Alice may disclose some credentials and believe that Coastal Bank's policy has been satisfied, while Coastal Bank believes the opposite because the sensitive constraints of the bank's policy are not satisfied.

A key observation is that since policies may be sensitive and need to be protected from unauthorized disclosure, there is no essential difference between policies and other resources, from the point of view of resource protection. That means that when we protect a sensitive policy, we can protect it in the same way as any other resource. Therefore a resource protection scheme that satisfies the following desiderata is desirable.

1. When Alice requests access to Bob's resource R, Bob may disclose R's access control policy P to Alice so that Alice can learn how to gain access to R. Once Alice has disclosed credentials that satisfy P, Bob will grant Alice access to R. One underlying assumption in this process is that the two parties have the same understanding of the semantics of policies. When one party believes that a policy has been satisfied by disclosed credentials, the other party should believe the same. Otherwise, a dispute may arise even though the two parties negotiate trust in good faith. We call this the *satisfaction-agreement* assumption.

2. Protection of sensitive policies should be as flexible as for any other kind of resource, allowing any kind of constraint that is used for protecting other resources. Because different parts of a policy may be sensitive in different ways, the policy protection approach should allow fine-grained control of the protection applied to each part of a policy.

3. Let R be a resource with access control policy P. The resource protection scheme should decouple the protection of R and P. R's accessibility should only depend on P's satisfaction. Whether P is disclosed or not should not affect R's accessibility. In other words, the scheme should be able to model the situation where the client presents the right credentials, satisfies P and gains access to R, without becoming eligible to see the actual contents of P during a negotiation. Taken to the extreme, this will allow servers to offer private services whose existence and access control policies are never made public, but whose location and policies can be pushed privately to selected clients.

4. Based on the disclosures made so far, plus the local resources and their policies, a negotiation strategy suggests the next message that a party should send to the other negotiation participant [17]. Two strategies are said to be *interoperable* if by adopting them respectively, two parties can always establish trust whenever their policies theoretically allow trust to be established. The design of interoperable strategies is an important issue in research on trust negotiation, and the resource protection scheme should not become an obstacle to interoperation. The resource protection scheme must allow a wide variety of practical negotiation strategies that will interoperate correctly with one another.

5. The resource protection scheme must allow a human-friendly interface for policy capture and maintenance. We believe that the challenge of policy capture and maintenance is perhaps the biggest obstacle to widespread deployment of systems that are open to access by outsiders. Businesses and individuals must have confidence in their policies, but policies are hard to write and will require updates in a changing world. While these issues are the focus of work by other researchers [5, 3, 12, 13] and are largely outside of the scope of the current paper, we believe that they must be kept in mind when designing a unified resource protection scheme.

3 Related Work

A general introduction to trust negotiation and related trust management issues can be found elsewhere [16]. As described in detail there, a number of trust negotiation systems and supporting middleware have been proposed and/or implemented in a variety of contexts (e.g., [1, 2, 7, 6, 9, 10, 11]). Of these, only two [2, 11] support sensitive access control policies; the remainder assume that policies can be freely disclosed. We will discuss these two schemes in detail.

3.1 Service Accessibility Rules

Bonatti and Samarati [2] proposed a framework for regulating service access and information release on the web.

The framework is targeted at environments where no single centralized security domain exists. Their framework contains a policy language for access control specification, and a filtering mechanism to identify the relevant policies for a negotiation. A service accessibility rule in their model is composed of two parts: a prerequisite rule and a requisite rule. Service prerequisite rules state the conditions that a requester must satisfy before she can be considered for the service. Service requisite rules state sufficient conditions for obtaining access to the service, i.e., if a requester satisfies a requisite rule, she will be entitled to access the service. To protect both the server and client's privacy, the model enforces a specific ordering between a service's prerequisite and requisite rules. The server will not disclose a requisite rule until after the requester satisfies a corresponding prerequisite rule. Prerequisite rules can contain constraints that are hidden from strangers.

For instance, to get a special offer from an auto insurance company, suppose that a requester is first required to show a valid driver's license which contains basic personal information such as gender, age and address. Based on such information, the insurance company will determine whether the requester is likely to be in a high risk group. How the company makes its decision should not be disclosed to the requester. If the requester does not belong to a high risk group, the insurance company will require disclosure of the title of the vehicle to verify that the requester is actually the owner of the insured vehicle. By using the model in [2], we have the following rules:

1. $service\_prereqs(special\_offer()) \leftarrow credential(drivers\_license(gender = X, age = Y, issuer =$“DMV”$)) \mid high\_risk(gender = X, age = Y) = false$. (The constraint $high\_risk(gender = X, age = Y) = false$ is hidden when the prerequisite rule is disclosed.)

2. $service\_reqs(special\_offer()) \leftarrow credential(vehicle\_title(owner = X, issuer =$“DMV”$)), credential(drivers\_license(name = Y)), X = Y$.

For simplicity, we have omitted several details in the above service rules. In particular, issuers of these credentials should be principals that the server trusts, and the requester should authenticate to the owner of each credential. We would expect the credential issuer to be identified not by a string, but by a public key lookup. Further, a party must verify that each credential it receives during trust negotiation has contents that correspond to its signature. Also, trust negotiation should be conducted over a secure channel to prevent certain attacks. We will also omit these important details in all later examples in this paper.

By explicitly indicating whether a policy is a prerequisite or requisite rule, a requester can be made to understand the consequences of disclosing particular credentials. Thus, the satisfaction-agreement assumption holds in this model.

A prerequisite rule serves two functions at the same time: controlling the disclosure of the service requisite rules and controlling access to the service. The scheme does not decouple policy disclosure with policy satisfaction. Thus, desideratum 3 is not satisfied. Also, the only way to protect a sensitive policy is to keep it completely inaccessible to a requester. This makes it difficult to use service accessibility rules to express complex authorization requirements. Therefore desideratum 2 is not met. On the other hand, examples 1 and 2 can be easily expressed using service accessibility rules.

Example 3. McKinley Clinic makes its patient records available for online access. Let R be Alice's record. To gain access to R, R's policy states that a requester must either present Alice's patient ID for McKinley Clinic, or present a California social worker license and a release-of-information credential issued to the requester by Alice. Knowing that Alice's record specifically allows access by social workers will help people infer that Alice may have a mental or emotional problem. Alice will probably want to keep this constraint inaccessible to strangers. However, employees of McKinley Clinic should be allowed to see the contents of this policy. Note that this does not mean that to satisfy R's policy, a requester has to work for McKinley Clinic. Any licensed social worker can access Alice's patient record as long as the social worker has obtained a release-of-information credential from Alice and pushes the relevant credentials to the server.

In example 3, if the social worker constraint is put in a prerequisite rule, then it must be made inaccessible to everybody or accessible to everybody. On the other hand, if the constraint is put in the requisite rule, then the constraint on working for McKinley Clinic has to appear in the prerequisite rule, which means that anyone other than Alice who accesses Alice's record has to be employed by McKinley Clinic. In either case, we do not capture the original intent of the policy.

If the social worker is not working for McKinley Clinic, how could she know that she should present her social worker license and the release-of-information credential from Alice? In practice, we expect that Alice will have disclosed a version of R's policy to the social worker at the same time as Alice gave out the release-of-information credential. The social worker can cache both the credentials and the policy fragment for later use during trust negotiation. More generally, we expect advance disclosure and caching of policies to be useful whenever policies are invisible to all but a targeted audience.

3.2 Policy Graphs

The concept of policy graphs was proposed by Seamons et al. [11]. In their scheme, a resource's access control is expressed as a policy graph instead of a single policy. A policy graph is a directed acyclic graph with a single source node N_0 and a single sink node R. The sink node represents a sensitive resource, while all the other nodes represent policies. In a policy graph, if there is an edge from node N_i to N_j ($N_i \rightarrow N_j$), then we say N_i is a predecessor of N_j. The policy represented by the source node N_0 is not protected, thus can be disclosed to any requester. All other policies and resources N_k can be accessed only if there exists a directed path $(N_0, \ldots, N_j, N_k)$, such that every policy in the subpath $(N_0, \ldots, N_j)$ is satisfied by the credentials disclosed by the requester. The essential idea of policy graphs is to protect access control policies through gradual disclosure of constraints. For example, suppose a special insurance promotion is offered to graduate students whose age is over 25 or whose GPA is over 3.0. The access control policy graph may be as shown in figure 1.

During a negotiation, if a requester cannot present a valid student ID proving she is a graduate student, then the insurance company can just terminate the negotiation and does not need to disclose further constraints (P_1 and P_2). If the insurance company regards the constraints on age and GPA as sensitive information, then, to some extent, policy P_0 in the source node helps to prevent the disclosure of P_1 and P_2 to arbitrary strangers. Based on this observation, Seamons et al. claim [11] that by dividing access control requirements into layers and organizing them as a policy graph, one can effectively protect sensitive policies.

However, from the semantics of policy graphs, we can see that each policy in a policy graph not only controls its direct successor's accessibility, but also that of its indirect successors. In the above example, P_0 controls when P_1 and P_2 can be disclosed. Further, to gain access to R also requires a requester to satisfy P_0 in the first place. Thus, the concept of policy graphs couples a policy's satisfaction tightly with the policy's disclosure, which makes it difficult to use policy graphs to express certain access control requirements.

In example 1, if we treat the policy requiring an employee ID as the source node P_0 and regard constraints on the issuer of the employee ID as its successor P_1, then anyone who can produce an employee ID may be able to see the contents of P_1, which is obviously not desired. On the other hand, if we put $false$ in the policy graph as P_1's predecessor, indicating that nobody is allowed to see the contents of P_1, then no one will be able to gain access to the project documents, which is not consistent with the original access control requirements either. Similarly, policy graphs cannot be used to capture the intent of examples 2 and 3.

In summary, policy graphs preserve the satisfaction-agreement assumption: once a policy P is satisfied, a requester knows that either she gains access to R or she will see the content of one of P's successors. But policy graphs couple policy disclosure with policy satisfaction, which limits their expressiveness and violates desiderata 2 and 3.

4 A Unified Scheme for Resource Protection

The lack of centralized authority on the Internet suggests that more than one general-purpose language may become popular for expressing policies. Further, specialized sub-languages may evolve by, for example, making it easier to capture and maintain common access control constraints for certain kinds of resources. Thus our goal is to provide a general-purpose way to protect policies without dictating the choice of policy languages, beyond certain minimal requirements described later.

Of course, the examples we give must use a particular language. We will use variants of the languages adopted by Seamons et al. and Bonatti and Samarati, the latter of whom states that "basic predicates constitute the basic literals that can be used in rules [that govern] services' accessibility and [local information] disclosure". Their language includes several kinds of basic predicates. *Credential* predicates are used to specify constraints on credentials. *Declaration* predicates are used to specify constraints on unsigned information submitted by a party, such as a preferred font color. *Cert_authority* predicates are used to specify the issuers that a party trusts. *State* predicates evaluate information stored at a party or acquired during a negotiation, such as the local time of day. *Abbreviation* predicates allow a complex predicate to be defined once, then referred to thereafter by name. Built-in mathematical predicates can also be used. Our examples will use only a few of these types of predicates, and we will not address issues specific to policy language design, such as the specification of authentication requirements, the permissible use of negation in policies, and the appropriate handling of (external) functions, such as $high\_risk$, that appear in policies. Further, we will assume that the underlying language has a formal definition of what it means for Alice to satisfy a policy—more precisely, for the set of credentials disclosed by Alice to satisfy a policy. For convenience, we also assume that the underlying language includes conjunction and disjunction. We use symbols in the calligraphic font to represent policies in an underlying policy language: $\mathcal{P}$, $\mathcal{P}_1$, $\mathcal{P}_2$, etc. We begin by reifying policies.

Definition 4.1. *A* policy definition *takes the form* $P \leftrightarrow \mathcal{P}$*, where* P *is a unique ID for this policy and* $\mathcal{P}$ *is a policy in the underlying policy language. We call* $\mathcal{P}$ *the* content *of the policy, denoted as content*(P).

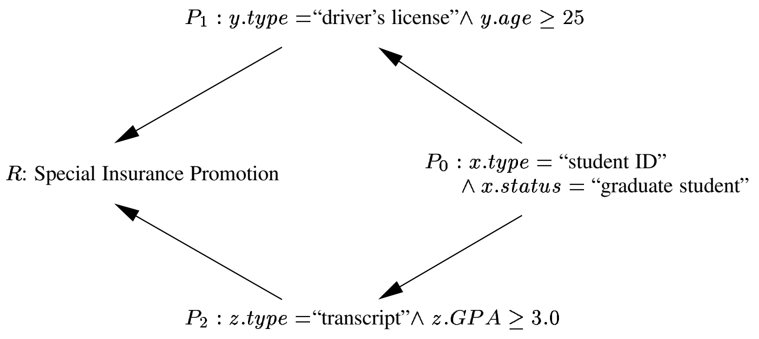
P_1 : $y.type$ ="driver's license"$\wedge$ $y.age \geq 25$
R: Special Insurance Promotion
P_0 : $x.type$ = "student ID"
$\wedge$ $x.status$ = "graduate student"
P_2 : $z.type$ ="transcript"$\wedge$ $z.GPA \geq 3.0$

Figure 1. An example of policy graphs.

Definition 4.2. *A set $\mathcal{C}$ of credentials* minimally satisfies $\mathcal{P}$ *if no proper subset of $\mathcal{C}$ satisfies $\mathcal{P}$.*

For example, consider the policy stating that the requester must be a social worker licensed by the state of California, and have a release form signed by Alice. The sets of credentials minimally satisfying $\mathcal{P}$ will contain exactly two credentials.

Definition 4.3. *Given policy definition $P \leftrightarrow \mathcal{P}$ and policy content $\mathcal{P}'$, we say a set $\mathcal{C}$ of credentials satisfies $(\mathcal{P}') \wedge P$ if $\mathcal{C}$ satisfies $(\mathcal{P}') \wedge (\mathcal{P})$. Similarly, $\mathcal{C}$ satisfies $(\mathcal{P}') \vee P$ if $\mathcal{C}$ satisfies $(\mathcal{P}') \vee (\mathcal{P})$.*

Definition 4.3 allows policy IDs to appear in policy definitions, to give finer-grained greater control over policy protection. In the remainder of the paper, policy contents will be written in this *augmented policy language*. Definition 4.3 also introduces a very simple type of policy composition. The definition can be viewed as stating what it means to satisfy two policies in the underlying language simultaneously, or to satisfy one of two policies. This kind of policy reuse and composition [3, 12, 13] is an effective way to help users design and maintain their policies. We will use it to translate policy graphs and service accessibility rules into UniPro.

Although Definition 4.3 may seem trivial at first glance, it must be formulated very carefully for underlying policy languages that allow free variables to appear in policies; conjoining two policies may force two variables to refer to the same resource or attribute value. Definition 4.3 is appropriate for the policy languages used by Seamons et al. and Bonatti and Samarati, but it may need to be tailored to meet the needs of other policy languages developed in the future.

For example, let P_1 be the ID of an airline seat upgrade policy that requires a requester to be a Platinum Frequent Flyer with a seat upgrade certificate. Then we have $P_1 \leftrightarrow x.type$ = "Platinum Frequent Flyer" $\wedge$ $x.issuer$ = "CheapAir" $\wedge$ $y.type$ = "Seat Upgrade Certificate" $\wedge$ $y.issuer$ = "CheapAir". Let P_2 be the ID of a seat upgrade policy that requires the requester to be a gate agent: $P_2 \leftrightarrow z.type$ = "Employee ID" $\wedge$ $z.issuer$ = "CheapAir" $\wedge$ $z.jobTitle$ = "Gate Agent". Now we want to have a policy ID P whose content says that a requester needs to either be a gate agent or be a Platinum Frequent Flyer with an upgrade certificate. There are several possible ways to get P through policy composition; each of these ways offers different possibilities for protecting the policies.

1. We may have $P \leftrightarrow P_1 \vee P_2$. In this case, P, P_1, and P_2 can have entirely different access control policies. For example, P may be made freely accessible, so that all can see P's internal disjunctive structure. P_1 may be accessible to all Platinum Frequent Flyers and airline employees, while P_2 is only accessible to airline employees.

2. We may have $P \leftrightarrow \mathit{content}(P_1) \vee \mathit{content}(P_2)$. In this case, anyone who can satisfy P's access control policy—say, all Platinum flyers—will be able to see the full policy for seat upgrades. This may be undesirable, as all Platinum passengers may beg the gate agent for free upgrades, once they learn about this policy.

3. We may also have $P \leftrightarrow \mathit{content}(P_1) \wedge P_2$. In this case, anyone who satisfies P's policy can learn the standard way of obtaining an upgrade. The fact that gate agents can perform a free upgrade can be hidden from everyone but airline employees, if desired.

Definition 4.4. *$R : P$ denotes that P is the ID of the access control policy for resource R. A requester is entitled to gain access to R only if she has disclosed credentials that satisfy P.*

Under UniPro (**Uni**fied Resource **Pro**tection Scheme), each resource R is protected by exactly one policy $(R : P)$, and each policy ID P has exactly one policy definition $P \leftrightarrow \mathcal{P}$. Further, the fact that R is protected by a policy with a particular ID $(R : P)$ is freely disclosable to all, because P and R are both just resource IDs, which in practice are random numbers conveying no new information to the recipient. In UniPro, policies can be protected by policies which can be protected by policies and so on, but the chain of protection must bottom out with a policy that is either always hidden or always disclosable[2]. These special policies have IDs *true* and *false*, their contents are always and never satisfied, respectively, and they are freely disclosable.

Let us revisit the examples from section 2.

Example 1. The access control policy for the project documents R is $R : P$, where $P \leftrightarrow x.type =$ “Employee ID” $\wedge$ P_1 and $P_1 \leftrightarrow x.issuer =$ “Microsoft” $\vee$ $x.issuer =$ “IBM”. We also have $P : true$ and $P_1 : false$. Since P’s policy is always satisfied, any requester can see P’s content. However, P_1’s policy is never satisfied, which means P_1’s content should not be shown to anybody. Assume Alice is an employee of Sun Microsystems and wants to access the project’s documents. After P’s content has been disclosed, she may disclose her Sun employee ID. But since Alice knows that P refers to an inaccessible policy P_1 (because $P_1 : false$ is freely disclosable), she knows that her employee ID may not be enough to satisfy P. Therefore, the satisfaction-agreement assumption holds in this situation.

Example 2. Coastal Bank’s loan application policy definition is $P \leftrightarrow x.type =$ “Customer ID” $\wedge$ $x.issuer =$ “Coastal Bank” $\wedge$ P_1. We also have $P_1 \leftrightarrow x.ID \notin BadCustomerList$, $P : true$, and $P_1 : false$. After a loan applicant discloses her Coastal Bank customer ID, since P_1 is a conjunct in the content of P, and the content of P_1 will never be disclosed, she should wait for Coastal Bank to decide whether P is satisfied instead of taking it for granted that she is qualified for the loan.

Example 3. Let R be Alice’s patient record. We have

1. $R : P$.
2. $P \leftrightarrow P_1 \vee P_2$ and $P : true$. Everyone can see that there are two ways to get access to Alice’s record.
3. $P_1 \leftrightarrow x.type =$ “patient ID” $\wedge$ $x.name =$ “Alice” $\wedge$ $x.issuer =$ “McKinley Clinic”, and $P_1 : true$. Everyone can see that Alice can access her own records.
4. $P_2 \leftrightarrow x.type =$ “Professional License” $\wedge$ $x.profession =$ “Social Worker” $\wedge$ $x.issuer =$ “State of California” $\wedge$ $y.type =$ “Medical Records Release” $\wedge$ $y.issuer =$ “Alice” $\wedge$ $y.institution =$ “McKinley Clinic”. Alice can also authorize social workers to look at her records.

To prevent the inappropriate disclosure of P_2’s content, we also have $P_2 : P_3$, $P_3 \leftrightarrow z.type =$ “Employee ID” $\wedge$ $z.issuer =$ “McKinley Clinic”, and $P_3 : true$. Then everyone can see that McKinley employees can see another way to gain access to Alice’s records.

Next we analyze UniPro according to the first three desiderata discussed in section 2.

1. In UniPro, if a requester submits credentials that satisfy a resource’s policy, then she is entitled to access that resource. On the other hand, because of the explicit appearance of policy IDs in a policy, a requester will be aware of parts of a policy that have not been disclosed yet. Without seeing those parts, she will not always be able to tell whether the policy has been satisfied by the credentials she has disclosed. Therefore, there will be no disagreement between two parties over whether a policy has been satisfied, given that they both understand the semantics of the underlying policy language.

2. UniPro explicitly protects policies in the same way as other resources. Users can design policies that provide fine-grained control of sensitive policies’ disclosure. In fact, by looking at a resource’s policy, $R : P$, we cannot tell whether R is a policy, a credential or a service.

3. The semantics of UniPro explicitly separates a policy’s satisfaction from its disclosure. If a resource is protected by a policy P, then as long as P is satisfied, R can be accessed. Whether P has been disclosed or not is irrelevant to R’s disclosure.

Regarding desideratum 5, probably no policy language with roots in mathematical logic will ever be regarded as user-friendly by the average corporate programmer. We view the previously proposed policy languages for trust negotiation as efforts to put a firm semantic foundation under trust negotiation, and to understand the requirements for policy languages for trust negotiation. They have served admirably in this capacity, and UniPro builds directly upon them. More generally, we view UniPro not as a replacement for other policy languages, but rather as an underlying foundation that other, more programmer-friendly, policy capture tools can be hooked into. A programmer can use the friendliest capture tool with sufficient modeling power for a new/updated policy, and then the captured policy can be translated into UniPro-style policy definitions in a well-known, general-purpose policy language, to provide portability and mutual comprehension of policies written by

[2] If a set of policies contains non-trivial cycles, i.e., if to view a non-trivial policy P, Alice must first satisfy P, then there is an equivalent set of policies with no cycles.

strangers. To show the promise of this approach, the following two theorems show how UniPro generalizes service accessibility rules and policy graphs.

Theorem 4.1. *Given a policy graph G protecting resource R, there is a set of UniPro policy definitions with the same semantics as G. More precisely, if R's UniPro policy is satisfied by a set of disclosed credentials, then there exists a path from the source node of G to an immediate predecessor of R in G, such that every policy along the path is satisfied by the set of disclosed credentials.* □

One can also present a stronger version of Theorem 4.1, to guarantee that the policy content $\mathcal{P}$ associated with a particular node in G is not disclosed unless all the policies associated with nodes along one path from the source of G to a node with content $\mathcal{P}$ have been satisfied. However, the statement of such a theorem necessarily relies on the translation of G into UniPro policies, which we do not present here due to space limitations. This omission aside, theorem 4.1 shows that UniPro is at least as expressive as policy graphs. As shown in Examples 1 and 2 in section 2, there are realistic cases whose access control requirements cannot be expressed with policy graphs. Thus UniPro generalizes policy graphs. However, as mentioned earlier, policy graphs or another graphical interface to an underlying policy language can be used as a friendly interface on top of UniPro for capturing and maintaining access control requirements that do not require all of UniPro's modeling power.

Theorem 4.2. *Given a service prerequisite rule $\mathit{service\_prereqs}(s) \leftarrow q_1, \ldots, q_n \mid p_1, \ldots, p_m$ and a service requisite rule $\mathit{service\_reqs}(s) \leftarrow q'_1, \ldots, q'_t$, where q_i $(1 \leq i \leq n)$, p_i $(1 \leq i \leq m)$, and q'_i $(1 \leq i \leq t)$ are all predicates, there is a set of UniPro policy definitions with the same semantics as the service accessibility rules. More precisely:*

1. *Service s's UniPro policy is satisfied by a set of credentials disclosed by Alice if and only if those credentials satisfy $q_1 \wedge \cdots \wedge q_n$, $p_1 \wedge \cdots \wedge p_m$, and $q'_1 \wedge \cdots \wedge q'_t$.*

2. *If the contents of any $q'_i, 1 \leq i \leq t$, are disclosed to Alice, then $q_1 \wedge \cdots \wedge q_n \wedge p_1 \wedge \cdots \wedge p_m$ are satisfied by credentials Alice has already disclosed.*

3. *The contents of p_i, $1 \leq i \leq m$, are never disclosed.* □

As shown by Example 3 in section 2, there are realistic access control requirements that cannot be expressed using service accessibility rules. Thus UniPro generalizes service accessibility rules.

5 Negotiation Strategies for UniPro

Ideally, trust should be established whenever the two parties' access control policies theoretically allow. Yu et al. [17] studied strategy interoperability under the assumptions that access control policies can be freely disclosed and autonomy in the choice of negotiation strategy is important to parties. They designed a set of strategies that guarantee that two parties can always establish trust whenever theoretically possible, if each party adopts any strategy in the set. In this section, we extend concepts proposed in [17] to study strategy interoperation when using UniPro to protect sensitive policies.

Different parties will have different requirements for how much computation they are willing to do, how freely they disclose resources, and how interested they are in extracting information from the other party. In a free-wheeling place with decentralized control such as the Internet, parties should be free to choose whatever strategy meets their requirements. For example, parties running on portable thin clients, such as PDAs and smart cards, which only have low-end CPUs and limited memories, might prefer strategies that require little computation and memory resources but make unnecessary disclosures, while servers with sufficient computation and memory resources might like strategies that can carefully analyze the ongoing negotiation and only disclose minimal information to ensure a successful negotiation. In this section, we present two strategies for UniPro, based on a simple agreement about message format called the *UniPro protocol*.

The UniPro protocol allows three types of disclosures: one can disclose a resource, a policy ID or a relationship between a policy and a credential (a *variable assignment*). For convenience, we assume that the resource IDs used by the two parties are disjoint. For a resource disclosure,

1. [17] If the resource is a service, then the other party gains access to that service.

2. [17] If the resource is a credential, then the contents of the credential are sent to the other party. (In practice, credential content verification, parsing of the credential into a set of statements in the policy language, and any required challenge/response for authentication will take place at this point, but we do not discuss this further in this paper.)

3. [17] If the resource is a policy, then the policy definition is sent to the other party. In other words, the content of the policy is disclosed.

As one would expect, in a policy ID disclosure, a disclosure of the form $R : P$ is sent to the other party, where R is a resource ID and P is the ID of the policy for R. Note that a policy ID disclosure does not disclose the content of the policy, only the policy ID.

A variable assignment disclosure is of the form $P.x = C$, where P is a policy ID, x is a variable in *content*(P) and C is a credential ID that is *potentially relevant* (defined below) to P. If P is one of Bob's policies, then Alice sends $P.x = C$ to indicate that her credential C may be in a set of credentials that minimally satisfies *content*(P), when C is assigned to be the value of variable x in *content*(P). Intuitively, policy ID disclosures link policies to resources, while variable assignment disclosures link credentials to policies.

Definition 5.1. *Suppose $P \leftrightarrow \mathcal{P}$. Let $\mathcal{S}_\infty$ contain a subset of the policy IDs appearing in $\mathcal{P}$, and let $\mathcal{S}_\in$ be a set of credentials. Let $\mathcal{P}'$ be the result of replacing by* true *all the policy IDs in $\mathcal{P}$ that also appear in $\mathcal{S}_\infty$. If $\mathcal{P}'$ is satisfied by $\mathcal{S}_\infty$, then we say $\mathcal{S}_1 \cup \mathcal{S}_2$ is an* indirect solution set *for P. If no proper subset of $\mathcal{S}_\infty$ or $\mathcal{S}_\in$ has this property, then we say $\mathcal{S}_\infty \cup \mathcal{S}_\in$ is a* minimal indirect solution set *for P.*

Definition 5.2. *Suppose $P : \mathcal{P}$, and let C be a credential ID. If C belongs to a minimal indirect solution set to P, then we say C is* potentially relevant *to P.*

In trust negotiation using the UniPro protocol, every message that a party Alice sends is a set of the disclosures defined above. An empty message is called a *failure message*, and indicates that a party has decided to terminate the negotiation. Further, to guarantee the safety and timely termination of trust negotiation no matter what policies and resources the parties possess, the UniPro protocol requires the negotiation strategies used with it to enforce the following three conditions throughout negotiations:

1. [17] No duplicate disclosures can be made.

2. [17] A resource cannot be disclosed unless its access control policy has been satisfied.

3. A variable assignment $P.x = C$ can be disclosed only after $P \leftrightarrow$ *content*(P) has been disclosed.

A trust negotiation is triggered when Alice sends a request to access one of Bob's resources R. Upon receiving the request, Bob calls his negotiation strategy, then sends to Alice the disclosure message it outputs. Similarly, upon receiving Bob's message, Alice passes the message to her strategy and sends Bob the message suggested by her strategy. This process continues until either Alice finally satisfies R's policy and gain access to R or one party sends an empty message and terminates the negotiation.

UniPro allows portions of the content of a resource's access control policy to be hidden from a requester. To protect privacy, a requester may not want to disclose all her credentials in an attempt to satisfy those hidden constraints. So Alice may possess the right credentials to satisfy the resource's policy, but the negotiation may fail, because she cannot see the contents of a policy. This is an acceptable outcome of a negotiation—a reasonable tradeoff between privacy and access. Alice may reasonably believe that if she could gain access, she would already have obtained and cached the hidden portions of the access control policy, as mentioned in section 4. On the other hand, parties will want to adopt strategies that will guarantee a successful negotiation given "sufficient" policy visibility. Thus, when we design negotiation strategies for UniPro, there is a tradeoff between establishing trust and preserving one's privacy.

Definition 5.3. *Let R be a resource of Bob's that Alice requests to access. If there exists a sequence of resource disclosures $(R_1, \ldots, R_k = R)$ such that for all R_i, $i = 1, \ldots, k$, R_i's access control policy is satisfied by credentials in $\{R_1, \ldots, R_{i-1}\}$, then we say the sequence is a* safe disclosure sequence *culminating in R's disclosure.*

Note that if a resource R_i is disclosed in a safe disclosure sequence, that does not imply that R_i's access control policy is also disclosed in the sequence.

Definition 5.4. *[17] A strategy is a function s with input parameters $(m_1, \ldots, m_k)$, L and R, where R is the ID of the resource whose access request triggered trust negotiation, $(m_1, \ldots, m_k)$ is a sequence of nonempty disclosure messages not containing R, and L is a set of resource IDs and policies. The output of s is a disclosure message m.*

Definition 5.5. *[17] Let s_1 and s_2 be two strategies adopted by Alice and Bob, respectively. Let R be an arbitrary resource of Bob's that Alice requests to access. If Alice can always gain access to R whenever there exists a safe disclosure sequence culminating in R's disclosure, then we say s_1 and s_2 are* strongly interoperable.

Since a policy may be hidden completely from a requester, under certain situations strong interoperability will require a party to disclose all its credentials whose policies have been satisfied, which may severely jeopardize one's privacy. For example, let P be one disjunct in R's policy. If $P : false$ has been disclosed, then before Alice can give up on obtaining access to R, strong interoperability will require her to send her freely disclosable library card, AAA membership card, high school diploma, etc., to Bob in an attempt to gain access. Strongly interoperable strategies represent one extreme that concentrates only on establishing trust, with few considerations for privacy.

Definition 5.6. *Let $(R_0, \ldots, R_n = R)$ be a safe disclosure sequence culminating in R's disclosure, and suppose $R : P$. We say the sequence is* visible *if the following two conditions are satisfied:*

1. *Either $P \leftrightarrow true$ or there exists k, $0 \leq k \leq n-1$, such that $R_k = P$, i.e., R's policy is disclosed in the sequence.*

2. *Let $P \leftrightarrow \mathcal{P}$. There exists a minimal indirect solution set $\mathcal{S}$ for P such that for every policy ID P_i in $\mathcal{S}$, policy P_i is disclosed in the sequence and $content(P_i)$ is satisfied by credentials disclosed in the sequence.*

We say the sequence is consistently visible *if every prefix of the sequence is also visible.*

If there is a consistently visible disclosure sequence culminating in the disclosure of the requested resource R, then two strangers will be able to find out at least one way to establish trust, without being stymied by hidden constraints. This enables a good balance between trust establishment and privacy protection.

Definition 5.7. *Let s_1 and s_2 be two strategies adopted by Alice and Bob, respectively. Let R be an arbitrary resource of Bob's that Alice requests to access. If Alice can always gain access to R whenever there exists a safe, consistently visible disclosure sequence of Alice's and Bob's resources that culminates in R's disclosure, then we say that s_1 and s_2 are* weakly interoperable.

Next we present two strategies that work with UniPro policies. The first one is called the *Unified Eager Strategy*, and it does not carefully analyze what disclosures are useful for establishing trust. Instead, it sends all safe disclosures to the other party. If two parties both adopt this strategy, then strong interoperability can be achieved. Pseudocode for this strategy is shown in figure 2(a).

The second strategy is called the *Unified Relevant Strategy*. It does analyze the ongoing negotiation and tries to identify those disclosures that are relevant to the current negotiation. Further, it does not try to satisfy undisclosed policies. Only weak interoperability can be achieved if this strategy is adopted by one of the negotiation participants. Definition 5.8 formally defines the concept of relevance. Figure 2(b) shows the pseudocode of the Unified Relevant Strategy.

Definition 5.8. *Given a resource R, we define resources that are* syntactically relevant *to R as follows.*

1. *R is syntactically relevant to R.*

2. *Let P' be a policy ID. If P' appears in $content(P)$ and P is syntactically relevant to R, then P' is syntactically relevant to R.*

3. *Let P be a policy syntactically relevant to R. If credential C is potentially relevant to P, then C is syntactically relevant to R.*

Syntactic relevance traces credentials through the chains of policies leading back to R. Therefore, a policy may be syntactically relevant to R even if its ID does not appear in R's policy. Similarly, a credential may be syntactically relevant to R even if it is not potentially relevant to R's policy.

Theorem 5.1. *The Unified Eager Strategy is strongly interoperable with itself.* □

Theorem 5.2. *The Unified Relevant Strategy is weakly interoperable with itself.* □

Theorem 5.3. *The Unified Eager Strategy and the Unified Relevant Strategy are weakly interoperable.* □

By combining the two strategies given in figure 2, we can obtain a new Combined Strategy that is strongly interoperable with itself but does not make as many disclosures as the Unified Eager Strategy. Upon receiving a new message from the other party, the Combined Strategy first calls the Unified Relevant Strategy. If the message returned by the Unified Relevant Strategy is not empty, then the message is sent to the other party. Otherwise, the Combined Strategy sends the message suggested by the Unified Eager Strategy. Since the Combined Strategy does not send an empty message unless the Unified Eager Strategy suggests to do so, it is not hard to prove that the combined strategy is strongly interoperable with itself.

6 Discussion

In this section we discuss several issues which are closely related to UniPro and its deployment and are relevant for future trust negotiation research.

Policy design and analysis Access control policies are a cornerstone of trust negotiation. Organizations and individuals must have faith in their resources' policies, or they will be afraid to open their systems to access from outside. To instill confidence in policies, the person in charge of the security of a resource will need a policy editor, access to canned, composable, and reusable policies (e.g., the definitions of a non-profit company, a full-time student at an accredited university, a minority-owned business); and ways to analyze, understand, and test policies before they are deployed. Security managers will want tools for regression testing (comparison of policy coverage under old and new sets of policies), consistency checking, and comparison of policies against higher-level specifications (when available). Without such tools, many organizations will be unable to change their security policies for fear that alterations will create security holes

```
The Unified Eager Strategy ($\mathcal{M}$, $L$, $R$)
    $\mathcal{M} = (m_1, \ldots, m_k)$: a sequence of safe disclosure messages.
    $L$: the local resources of this party.
    $R$: the resource to which access was originally requested.
Output:
    A message $m$.
Pre-condition:
    $R$ has not been disclosed and $m_k$ is not a failure message.

$D = \bigcup_{1 \leq i \leq k} m_i$;
$m = \emptyset$;
For every resource $R' \in L$
    $m = m \cup \{R' : P'\}$, where $P'$ is $R'$'s access control policy;
    if $P'$ is satisfied by the credentials in $D - L$
        then $m = m \cup \{R'\}$;
For every variable $x$ in every policy $P \in (D - L)$
    for every credential $C \in L$
            set $m = m \cup \{P.x = C\}$;
$m = m - D$;
return $\{m\}$;
```

(a) Pseudocode for the Unified Eager Strategy

```
The Unified Relevant Strategy ($\mathcal{M}$, $L$, $R$)
    $\mathcal{M} = (m_1, \ldots, m_k)$: a sequence of safe disclosure messages.
    $L$: the local resources of this party.
    $R$: the resource to which access was originally requested.
Output:
    A message $m$.
Pre-condition:
    $R$ has not been disclosed and $m_k$ is not a failure message.

$D = \bigcup_{1 \leq i \leq k} m_i$;
$m = \emptyset$;
For every resource $R' \in L$ that is syntactically relevant to $R$
    $m = m \cup \{R' : P'\}$, where $P'$ is $R'$'s access control policy;
    if the credentials in $D - L$ satisfy $P'$, under the variable bindings for $P'$ given in $D$,
        then $m = m \cup \{R'\}$;
For every policy $P \in (D - L)$ that is syntactically relevant to $R$,
    for every set $S_1$ of policy IDs, set $S_2 = \{C_1, \ldots, C_n\}$ of credentials in $L$, and
    set $x_1 = C_1, \ldots, x_n = C_n$ of variable bindings,
    such that $S_1 \cup S_2$ is a minimal indirect solution set for $P$ under those variable bindings,
        let $m = m \cup \{P.x_1 = C_1, \ldots, P.x_n = C_n\}$;
$m = m - D$;
return $\{m\}$;
```

(b) Pseudocode for the Unified Relevant Strategy

Figure 2. Two negotiation strategies for UniPro.

that leave them vulnerable. These concerns arise in automated trust negotiation and also in the growing number of other computational realms that rely on policies.

Cached policies In UniPro, a relevant policy might not be disclosable during trust negotiation. Therefore the target audience for such policies needs to get them through off-line channels and cache them in their local hosts. In our discussion on strategies so far, we assume no cached policies are used during trust negotiation. When such an assumption is relaxed, it becomes important to study how to identify a cached policy that is related to an ongoing trust negotiation, when to disclose a cached policy, and what impact it will have on strategy interoperability.

Privacy protection The motivation of UniPro is to prevent unauthorized information flow during policy disclosures. Another form of information leaks happens when Alice responds to Bob's request. For example, suppose Bob tells Alice that, in order to gain access to a resource R, Alice needs to show her driver's license to prove her age is over 21. In a typical trust negotiation, if Alice can satisfy this policy, she may respond by telling Bob the access control policy for her driver's license. Otherwise, she will send a failure message and terminate the negotiation. However, in this case, by simply observing Alice's response, Bob may infer whether Alice's age is over 21, without really gaining access to her driver's license. Winsborough and Li [15, 14] proposed the concept of attribute acknowledgment policies (ack policies) to prevent such information leaks. An ack policy controls when a party can inform others whether or not he/she possesses a certain attribute. Therefore, a credential's access control policy will not be disclosed until the corresponding ack policy has been satisfied. In this way, ack policies provide one layer of protection for sensitive access control policies, although the protection is not as flexible as required to satisfy desideratum 2. A related question is whether UniPro can be used to achieve the goals of ack policies. Alice can use a $false$ access control policy to hide the fact that she does not possess a particular credential C. She could also use a more complex access control policy to ensure that the fact that she does not possess C is only revealed to authorized parties. However, access control policies do not provide as flexible means of protecting this kind of information as ack policies do. Since the protection provided by UniPro and ack policies is largely orthogonal, it will be interesting to investigate how to integrate the two schemes without sacrificing local autonomy in the choice of negotiation strategies.

7 Summary and Conclusion

To give access control policies the protection they need during trust negotiation, we have proposed UniPro, a Unified Resource Protection Scheme. The development of UniPro was guided by a set of desiderata for protection of sensitive access control policies, including the need for two parties to be able to agree on whether a certain set of credentials satisfies a particular policy, protection as powerful as that provided for any other resource, the decoupling of the protection provided for a policy P and the protection that P provides for another resource R, the ability to support human-friendly front ends for policy capture and maintenance, support for a wide variety of interoperable trust negotiation strategies, and fine-grained control over what parts of a policy are protected in what manner. UniPro provably generalizes previous approaches to protecting policies in trust negotiation, and moves us closer to satisfying the desiderata. Further, UniPro does not dictate a particular choice of policy language, allowing UniPro to take advantage of human-friendly interface tools for policy capture and maintenance as such tools are developed, and also to take advantage of advances in the design of policy languages for trust negotiation.

While UniPro provides new levels of freedom in policy protection during trust negotiation, such as the ability to offer a service whose existence and policies are known only to preselected clients, this freedom comes at a price. In particular, a trust negotiation between Alice and Bob can fail because Alice cannot see one of Bob's policies and is unwilling to blindly disclose her credentials in the hope of satisfying a policy that she cannot see. UniPro exposes this tradeoff, allowing policy designers to choose between greater privacy for their policies and an increased chance of negotiation failure. Where greater privacy is chosen, Bob or a third party may decide to push a hidden policy to Alice before she ever tries to access Bob's service, so that Alice can cache the policy and know how to access Bob's service in the future. We have extended the concept of strategy interoperability to apply to trust negotiations under UniPro, and proved interoperability guarantees for two new negotiation strategies that are compatible with UniPro. Our future work includes a closer investigation of interoperable strategies for UniPro.

Acknowledgments

This research was sponsored by DARPA through Space and Naval Warfare Systems Center San Diego grant number N66001-01-18908 (BYU) and AFRL contract numbers F33615-01-C-1805 (BYU) and F30602-97-C-0336 (NAI Labs). We also thank the anonymous reviewers for their helpful comments.

References

[1] M. Blaze, J. Feigenbaum, and A. D. Keromytis. KeyNote: Trust Management for Public-Key Infrastructures. In *Security Protocols Workshop*, Cambridge, UK, 1998.

[2] P. Bonatti and P. Samarati. Regulating Service Access and Information Release on the Web. In *Conference on Computer and Communications Security*, Athens, Nov. 2000.

[3] P. Bonatti, S. Vimercati, and P. Samarati. A modular approach to composing access control policies. In *ACM Conference on Computer and Communication Security*, Athens, Greece, Nov. 2000.

[4] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. The MIT Press, 2000.

[5] D. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Policy Specification Language. In *2nd International Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, Jan. 2001.

[6] A. Herzberg, J. Mihaeli, Y. Mass, D. Naor, and Y. Ravid. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.

[7] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. Seamons, and B. Smith. Advanced Client/Server Authetication in TLS. In *Network and Distributed System Security Symposium*, San Diego, CA, Feb. 2002.

[8] International Telecommunication Union. *Rec. X.509 - Information Technology - Open Systems Interconnection - The Directory: Authentication Framework*, Aug. 1997.

[9] W. Johnson, S. Mudumbai, and M. Thompson. Authorization and Attribute Certificates for Widely Distributed Access Control. In *IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1998.

[10] N. Li, W. Winsborough, and J. Mitchell. Distributed Credential Chain Discovery in Trust Management. In *Conference on Computer and Communication Security*, Philadelphia, PA, Nov. 2001.

[11] K. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Network and Distributed System Security Symposium*, San Diego, CA, Feb. 2001.

[12] D. Wijesekera and S. Jajodia. Policy Algebras for Access Control - The Propositional Case. In *ACM Conference on Computer and Communication Security*, Philadelphia, PA, Nov. 2001.

[13] D. Wijesekera and S. Jajodia. Policy Algebras for Access Control - The predicate Case. In *ACM Conference on Computer and Communication Security*, Washington, DC, Nov. 2002.

[14] W. Winsborough and N. Li. Protecting Sensitive Attributes in Automated Trust Negotiation. In *ACM Workshop on Privacy in the Electronic Society*, Washington, DC, Nov. 2002.

[15] W. Winsborough and N. Li. Towards Practical Automated Trust Negotiation. In *3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, California, June 2002.

[16] M. Winslett, T. Yu, K. Seamons, A. Hess, J. Jarvis, B. Smith, and L. Yu. Negotiating Trust on the Web. *IEEE Internet Computing Special Issue on Trust Management*, 6(6), Nov. 2002.

[17] T. Yu, M. Winslett, and K. Seamons. Interoperable Strategies in Automated Trust Negotiation. In *ACM Conference on Computer and Communication Security*, Philadelphia, PA, Nov. 2001.