

Protecting Sensitive Attributes in Automated Trust Negotiation

William H. Winsborough
Network Associates Laboratories
15204 Omega Drive
Suite 300
Rockville, MD 20850-4601
william_winsborough@nai.com

Ninghui Li
Department of Computer Science
Stanford University
Gates 4B
Stanford, CA 94305-9045
ninghui.li@cs.stanford.edu

ABSTRACT

Exchange of attribute credentials is a means to establish mutual trust between strangers that wish to share resources or conduct business transactions. Automated Trust Negotiation (ATN) is an approach to regulate the flow of sensitive attributes during such an exchange. Recently, it has been noted that early ATN designs do not adequately protect the privacy of negotiating parties. While unauthorized access to credentials can be denied, sensitive information about the attributes they carry may easily be inferred based on the behavior of negotiators faithfully adhering to proposed negotiation procedure. Some proposals for correcting this problem do so by sacrificing the ability to effectively use sensitive credentials. We study an alternative design that avoids this pitfall by allowing negotiators to define policy protecting the attribute itself, rather than the credentials that prove it. We show how such a policy can be enforced. We address technical issues with doing this in the context of trust management-style credentials, which carry delegations and enable one attribute to be inferred from others, and in the context where credentials are stored in a distributed way, and must be discovered and collected before being used in ATN.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General—*security and protection*; D.4.6 [Operating Systems]: Security and Protection—*access controls*; K.4.1 [Computers and Society]: Public Policy Issues—*pri-*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WPES'02, November 21, 2002, Washington, DC, USA.
Copyright 2002 ACM 1-58113-633-1/02/0011 ...\$5.00.

vacy; K.4.4 [Computers and Society]: Electronic Commerce—*security*

General Terms

Design, Management, Security

Keywords

Trust negotiation, attribute-based access control, trust management, digital credentials

1. INTRODUCTION

Traditionally, authorization has been based on the identity of the entity requesting access to a resource, either directly or through roles assigned to the entity. Many researchers have noted that this approach is cumbersome when resource and requester are not in the same security domain. An alternative is to grant resources based on characteristics of the requester that may be more relevant or more universally recognized than his¹ identity, such as age, employer, project assignment, or credit status. We call this approach attribute-based access control (ABAC). Some trust management systems, such as *RT* [4, 5], explicitly support ABAC, and others, such as SPKI/SDSI 2.0 [1, 3], can be used as ABAC systems. In ABAC systems, authorization decisions are based on attributes of the requester, which are established by digitally signed credentials through which credential issuers assert their judgments about the attributes of entities. Because these credentials are digitally signed, they can serve to introduce strangers to one another without on-line contact with attribute authorities. ABAC avoids the need for permissions to be assigned to individual requesters before the request is made. Instead, when a stranger requests access, the ABAC-enabled access mediator can make an authorization decision by combining agreements and judgments of decentralized authorities in a natural and logical way.

¹We blur the distinction between the software agent and the human or organization on whose behalf it operates.

ABAC systems depend on credentials that specify attributes of entities and/or rules for deriving entities' attributes. These attributes (such as financial or medical data) may be sensitive. The goal of a growing body of work on *automated trust negotiation (ATN)* [7, 8, 9, 10, 11, 12] is to enable resource requesters and access mediators to establish trust in one another through cautious, iterative, bilateral disclosure of credentials. In ATN, each negotiator establishes access control policies used to regulate not only the granting of system resources, but also the disclosure of credentials to negotiation opponents. The negotiation consists of a sequence of exchanges that begin by disclosing credentials that are not sensitive. As credentials flow, higher levels of mutual trust are established, and access control policies for more sensitive credentials are satisfied, enabling these credentials also to flow. In successful negotiations, credentials eventually flow to satisfy the policy of the desired resource.

Recently it has been noted [8, 9] that early ATN designs do not adequately protect the privacy of negotiating parties. In early designs, the behavior of a negotiator can reveal much about the contents of their credentials before any of those credentials are transmitted. For instance, under some negotiation system designs, when a negotiator is asked to prove a sensitive attribute, the negotiator transmits a counter-query based on access control policy associated with credentials that prove the requested attribute. Once the access control policy has been satisfied by the negotiator's opponent, the negotiator can provide the credential. Assuming the negotiator is faithfully adhering to the procedure, a counter-query is transmitted if and only if the negotiator satisfies the sensitive attribute. Thus, while the negotiator's opponent may not yet have proof of the integrity and authenticity of the attribute, the privacy of the attribute has certainly been compromised.

It may be difficult to protect information about a negotiator's sensitive attributes. While transmitting credentials can clearly disclose such information, we have seen that other behavior can do so as well. The information that an attribute is not satisfied may also be sensitive, which an opponent may be able to infer by presenting a query about the attribute that the negotiator is unable to satisfy. Furthermore, by combining pieces of information obtained through other means, a negotiation opponent can infer additional information about attributes, particularly when using credential systems that support delegation and other inference rules. In the current paper, we present an ad hoc study of some of the vulnerabilities of ATN systems to uncontrolled information flow, motivating and extending the approach to this problem taken in [9].

1.1 Mitigating Uncontrolled Attribute Flow

The fact that prior trust negotiation systems failed to protect information about the negotiator's attributes was first observed contemporaneously by Seamons et

al. [8] and by Winsborough and Li [9]. Seamons et al. identified three cases where the negotiator may wish to hide information about which credentials they do and do not hold, and presented techniques that the negotiator could use for this purpose. First, if having a certain credential is sensitive, on receiving a request for it, the negotiator can pretend not to have it. In this case, the credential can be used only if the opponent pushes credentials that satisfy the access control policy without being prompted to do so. This sacrifices the goal of ATN system *completeness*, namely that negotiation should succeed whenever possible². In general, this technique may prevent the negotiator from effectively using the credential. Second, when an ATN user wants to hide his lack of a credential, he can establish an access control policy for a credential place holder, and disclose the lack only after the policy is satisfied. This treatment of the negative case is more appealing than the first technique, but is not comprehensive. Third, if a request for credentials specifies required field values, the negotiator can prevent a probing attack by ignoring the field-value requirements when preparing his response. Like the second technique, we view this as a step in the right direction.

One might characterize the attitude taken in the techniques proposed by Seamons et al. as, "When you have something to hide, pretend otherwise until you know enough about your negotiation opponent." The attitude reflected in [9] and the one we explore here can be summarized as, "Discuss sensitive topics only with appropriate opponents." The general principle is that certain issues, such as medical or financial data, are private in nature, and even if a negotiator does not feel that his current status with respect to one of them is sensitive, he should protect the data anyway. Otherwise, when his status changes (e.g., his health deteriorates), or a different issue is raised, the mere fact that he protects the new data strongly suggests that his status is now unfavorable. Thus, when asked about a private matter, the response indicated by the attitude taken here is that the negotiator should always protect the answer as if he might have something to hide. This attitude has the drawback of being more cautious than might seem necessary in some cases. For instance, it says that a negotiator should protect answers to queries about his health, even if he is healthy. However, it can prevent opponents that have no business knowing the negotiator's health status inferring when he gets sick, and it can be turned into an ATN procedure that does not compromise completeness.

In [9], a notion called *acknowledgment policy* is introduced (Ack policy, for short). An Ack policy is es-

²We assume that negotiators do not push all authorized credentials, as is done in the eager strategy of [10]. Most ATN systems exchange policy content in the form of counter queries for the purpose of focusing disclosures on credential that are relevant to the current negotiation.

tablished to protect an attribute. It complements the protection of credentials by means of access control policies³. In this context, an attribute is just some property that the credential issuer says an entity satisfies. A negotiator that is not shown by credentials to satisfy the attribute is presumed not to satisfy the attribute. A negotiator *acknowledges* satisfying or not satisfying an attribute by answering a query about it only after the other party satisfies the Ack policy governing the attribute. This change in emphasis from protecting credentials to protecting attributes enables the negotiator to respond to a query in a way that does not depend on his credentials.

There are marked distinctions of this approach. Ack policy is organized in the same way whether or not the negotiator has the sensitive attribute, thereby avoiding multiple special cases. Moreover, Ack policy protects attributes that are proven by a combination of credentials, rather than focusing on the credentials themselves. This higher level of abstraction is particularly significant when dealing with a credential system that supports delegation and other inference rules, such as any of the trust management systems listed above. In these cases, attributes can be deduced from one another. Finally, in the decentralized environment where ATN has its role, the need to discover credentials introduces additional challenges in preventing unauthorized inference of attribute information. It is not clear how techniques based solely on credential access control policies could be extended to this context.

1.2 Contribution and Organization

The focus in [9] was on detailed specification of negotiation procedures that use Ack policies. Many factors constraining the design of Ack policy and its enforcement mechanisms were not mentioned and only a partial treatment was given of the danger that Ack policies may be breached by means of deductive inference. Additionally, while credential discovery techniques [5] were used to ensure needed credentials are available to the negotiation process, the interaction between the discovery problem and the protection of sensitive attribute information was not examined.

In the current paper, we give a comprehensive treatment of the problem of enforcing Ack policy. We thoroughly examine and address the problem of preventing unauthorized inference of sensitive attributes in a context where the underlying logic of delegation credentials enables one attribute to be inferred from another. We study the impact on Ack policy enforcement of the need to discover credentials in a decentralized environment. This clearly affects the negotiator's ability to use the credential to prove his attributes. But it is also needed for the negotiator to be able to hide the fact that he

³Although we give them little attention below, access control policies remain useful for protecting credential contents other than attributes, such as the signatures that can be used to verify integrity and authenticity.

does not satisfy a given attributes. We additionally examine the motivation for the design structure of Ack policy. Several factors constrain our ability to enforce Ack policy on arbitrary attributes and combinations of attributes. Finally, we identify some open problems concerning the use of credentials that are not universally available and the vulnerability such use may create.

The paper is organized as follows. Section 2 presents background from prior work concerning Ack policy enforcement. Section 3 discusses specification of Ack policy in the context of what might be considered the simplest realistic credential language. Section 4 is the core of the paper, presenting the significant challenges raised by simple delegation credentials for Ack policy enforcement, and showing how they can be overcome. Integral to this is managing the problem of credential discovery. Sections 5 and 6 extend the language to support respectively conjunction and attribute-based delegation within delegation credentials. There we consider the impact of these new credential language features on the structure and enforcement of Ack policy. Section 8 concludes.

2. BACKGROUND

In this section we state the goal of Ack policy and outline the approach to enforcing it that was taken in [9]. The credential language used there is RT_0 . This language is slightly simpler than the one we discuss in the following sections (having no fields or constraints), but is otherwise the same.

The goal of access control policies is simple. Credentials should not flow until AC policies are satisfied. Traditionally, the safety requirement of ATN is that the above property is guaranteed.

The goal of Ack policies is that no one should learn through negotiation whether or not a negotiator N satisfies an attribute without first satisfying its Ack policy. This means that until N 's negotiation opponent shows that it satisfies the Ack policy for a sensitive attribute, N 's behavior and the credentials N transmits must give no indication of whether N satisfies the attribute. This is the safety requirement for Ack policy enforcement in ATN.

When using a credential language that allows issuers to assert the attributes only of entities specifically identified in the credential, the enforcement of Ack policies is straightforward. When credentials carry rules that can be used to infer one attribute from another, the problem is trickier. The following example illustrates the use of such credentials.

EXAMPLE 1. *A fictitious Web publishing service, EPub, offers a discount to preferred customers of its parent organization, EOrg. EOrg considers students of the university StateU to be preferred customers. StateU delegates the authority over identifying students to RegistrarB, the registrar of one of StateU's campuses. RegistrarB then issues a credential to Alice stat-*

ing that Alice is a student. These are represented by four RT_0 credentials:

$$\begin{aligned} \text{EPub.discount} &\leftarrow \text{EOrg.preferred} & (1) \\ \text{EOrg.preferred} &\leftarrow \text{StateU.student} & (2) \\ \text{StateU.student} &\leftarrow \text{RegistrarB.student} & (3) \\ \text{RegistrarB.student} &\leftarrow \text{Alice} & (4) \end{aligned}$$

The credential “EPub.discount \leftarrow EOrg.preferred” can be read as: if EOrg assigns some entity the attribute “preferred”, EPub assigns the entity the attribute “discount”. The four credentials above form a chain, pictured in Figure 1, proving that Alice is entitled to a discount. ■

The Trust Target Graph (TTG) [9] method organizes the process of ATN in terms of the construction of a graph (the TTG) that is shared between the two negotiators. Nodes in the TTG represent queries (called *trust targets* in [9]) made by one negotiator or the other. Each trust target corresponds to a unique node (thereby avoiding redundant computations). Edges are of various types, and represent corresponding kinds of dependencies. One of the most important kinds of edge is an implication edge, which originates at a query that if shown would entail satisfaction of the destination query. Another kind, a control edge is based on an Ack policy (or an access control policy). It originates at a query that if satisfied would permit further information to flow concerning the destination query.

Typically based on the access control policy of a requested resource, the root of the TTG is a query whose satisfaction entails negotiation success. The negotiators take turns adding edges to the graph, which remains a connected graph in which the root is the unique sink. At the end of its turn, a negotiator transmits the changes it has made so that the opponent can update its own graph accordingly, thus achieving agreement in the two graphs at the beginning of each turn.

EXAMPLE 2. Alice is cautious about whom she tells that she is a university student. Her Ack policy for StateU.student requires recipients of this information to be members of the Better Business Bureau (BBB). EPub can prove this by using the following credential:

$$\text{BBB.member} \leftarrow \text{EPub} \quad (A)$$

The TTG constructed during a negotiation between Alice and EPub is shown in Figure 2. ■

The addition of an implication edge to the TTG is very similar to a resolution step in logic programming, reducing the problem of solving one query to that of solving another. When a negotiator adds an implication edge on a query that he did not pose, he must also transmit credentials proving that if he satisfies the source query, he will have satisfied the destination query. These credentials enable the opponent to verify the implication. Thus the TTG summarizes the negotiators’ state of knowledge of what has been shown and what

remains to be shown about each query posed during the negotiation. Each negotiator uses the TTG to compute for himself which queries have been satisfied.

Negotiators also keeps track of queries that have failed, thereby enabling efficient detection of negotiation failure. A negotiator can mark it “fully-processed,” indicating that it is done adding children to a query. This means that the TTG contains complete knowledge of that query, information that can be used by the opponent to determine that a query has failed.

Negotiators introduce control edges at queries posed to them concerning sensitive attributes. The origin of such an edge is a counter query based on the Ack policy governing the attribute in the destination query. The counter query must be satisfied before the negotiator will add implication edges to the protected query, or mark it fully-processed.

Of course, the most obvious way that a negotiator discloses information about his attributes is by transmitting credentials that prove those attributes hold. The TTG method ensures that the Ack policies of such attributes are satisfied before this happens by solving queries through a step-by-step, resolution-like process, reducing queries about one attribute to queries about attributes that can be proven more directly. At each step in the process, if the query created concerns a sensitive attribute, the negotiator enforces the appropriate Ack policy before moving on.

3. WHEN CREDENTIALS DO NOT CARRY DELEGATIONS

In this section, we study issues related to Ack policies in the (simpler) case in which the credential language does not have delegation. We do this in two steps, first considering a language that does not have fields, then considering the case when fields are added.

3.1 A Very Simple Language

To start, we consider a very simple language. In this language, a credential has the form “ $A.R \leftarrow B$ ”, in which A and B are entities, and R is a attribute name. This credential means that A asserts that B has the attribute R , or, in other words, B has the attribute $A.R$. The credential is said to *define* $A.R$. Throughout the paper, when multiple credentials define the same attribute, the effect is that of disjunction: the credentials can be used independently to show the attributes of various negotiators.

A policy rule is a conjunction of attributes: “ $A_1.R_1 \wedge A_2.R_2 \wedge \dots \wedge A_k.R_k$ ”. To satisfy such a policy rule, a negotiating party needs to prove that it has the attribute $A_j.R_j$ for each j such that $1 \leq j \leq k$. A query asking whether an entity N satisfies such a policy is written $\langle A_1.R_1 \wedge A_2.R_2 \wedge \dots \wedge A_k.R_k \stackrel{?}{\leftarrow} N \rangle$. In the TTG method of negotiation, such a query is decomposed into separate subqueries, one for each attribute: $\langle A_1.R_1 \stackrel{?}{\leftarrow} N \rangle \dots \langle A_k.R_k \stackrel{?}{\leftarrow} N \rangle$.

EPub.discount $\xleftarrow{(1)}$ EOrg.preferred $\xleftarrow{(2)}$ StateU.student $\xleftarrow{(3)}$ RegistrarB.student $\xleftarrow{(4)}$ Alice.

Figure 1: A credential chain showing that Alice is authorized for EPub’s student discount.

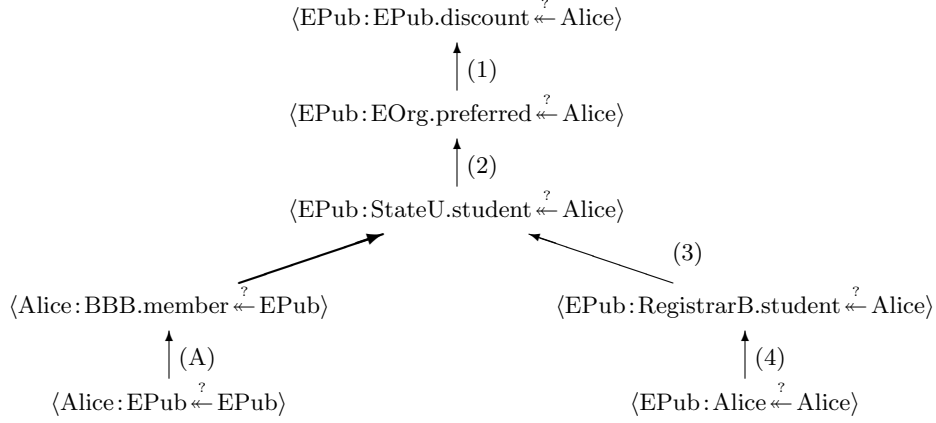


Figure 2: Trust Target Graph created during negotiation between Alice and EPub. The thicker edge on the left represents a control edge, corresponding to the dependency given by Alice’s Ack policy. Implication edge (A) must be added by EPub, and the credential justifying it must be transmitted to Alice before Alice adds implication edges (3) and (4). Technically, the difference between a trust target and a query is that a trust target has an additional (first) component that specifies which party posed the query.

The negotiator, N , specifies acknowledgement policy for a sensitive attribute, $A.R$, by defining a set of policy rules. Any opponent, O , must then prove that he satisfies at least one policy rule in the set before N will give O any answer to the query $\langle A.R \xleftarrow{?} N \rangle$.

3.2 Adding Attribute Fields

The addition of fields and field values to attributes significantly increases expressive power of the credential and policy languages. For example, fields can be used to carry account numbers, a person’s date of birth, or the name of a project an employee is assigned to. In this language, a credential still has the form $A.R$; however, R now has internal structures. We call R an attribute term; it has the form $r(f_1 = a_1, \dots, f_n = a_n)$, in which r is an attribute name, and each f_j , for $1 \leq j \leq n$, is a field name and a_j is a constant.

A policy rule has the form $A_1.R_1 \wedge A_2.R_2 \wedge \dots \wedge A_k.R_k : X_1 \in S_1 \wedge \dots \wedge X_m \in S_m$, in which each R_j is an attribute term, each field is either a constant or a variable, each X_j for $1 \leq j \leq m$, is a variable that appears in $A_1.R_1 \wedge \dots \wedge A_k.R_k$, and each S_j for $1 \leq j \leq m$ is a value set constraining the legal values of X_j . For more details, see [4].

Technically, an Ack policy is a policy rule associated with a specific attribute, $A.R$. In practice, N may wish to specify one Ack policy for several issuers of R , listing each of the authorities to which this Ack policy applies. One such entry for R could be designated the default Ack policy, which applies to all known issuers of the attribute. Such issuers may include those identified by

the user, as well as those identified by the system. However, in general not all issuers are known, and we will see in the following sections that Ack policy associated with unknown issuers can be unenforceable when the language supports delegation.

4. DELEGATION OF AUTHORITY: RULES IN CREDENTIALS

We now consider a language that supports delegation of authority. Such credentials are typically issued in a decentralized way, and must be discovered and collected prior to being used in ATN. We summarize how this can be done, discussing the impact of our approach on controlling unauthorized inference of sensitive attributes. We then study the latter problem in detail, followed by a summary of the steps needed to prepare for safe negotiation. (Credential collecting steps can be done before ATN begins, and no dynamic discovery of credentials during the negotiation is needed.) Finally, we touch briefly on additional hazards.

4.1 The Language and the Issues

In addition to credentials like $A.R \xleftarrow{?} B$, we also have credentials like $A.R \xleftarrow{?} B_1.R_1$. We call the former type-1 credentials, and the latter type-2 credentials. The meaning of $A.R \xleftarrow{?} B_1.R_1$ is that A asserts that each entity satisfying $B_1.R_1$ also satisfies $A.R$.

To be more precise, a type-2 credential has the form $A.R \xleftarrow{?} B_1.R_1 : X_1 \in S_1 \wedge \dots \wedge X_m \in S_m$, in which every variable that appears in R also appears in R_1 , and each X_j for $1 \leq j \leq m$ appears in R_1 . Again, for more

details, see [4]. Ack policies are again associated with individual attributes whose issuers are known, and the syntax of policy rules remains the same as in Section 3.2.

Note that a delegation of authority over an attribute is represented by a type-2 credential; type-2 credentials play an important role in the trust management approach, where they are quite common. Adding delegations within credentials introduces the following problems.

- Distributed discovery of credentials: possession of an attribute $A.R$ may be proven through a chain of credentials. In a decentralized environment, credentials are issued and stored in a distributed manner. Consequently, negotiators must contend with the problem of discovering and collecting credentials for the purpose of proving having an attribute.
- Deductive inferencing: when it is known that $A.R \leftarrow B_1.R_1$, information about satisfaction of $A.R$ may imply information about satisfaction of $B_1.R_1$, and vice versa.
- A negotiator's ability or inability to produce a type-2 credential can reveal certain information about the negotiator's attributes. It may indicate whether the negotiator has been authorized to have a copy of the type-2 credential. Moreover, it indicates whether the negotiator knows of the attribute's existence, which proves to be a limiting factor in our ability to defend against breaches of Ack policy.

4.2 Negotiating with Credentials Whose Storage Is Distributed

The environment we target is inherently decentralized. Negotiators having no prior relationship or knowledge of one another seek to establish a level of mutual trust by exchanging credentials. These credentials are issued and initially stored in a distributed manner, and must be discovered and collected for the purpose of showing an entity's attributes. We now consider the following basic problem: which negotiator can realistically be expected to discover which credentials? In addition to being a fundamental issue in the organization of the TTG ATN method, the problem of credential discovery has direct bearing on negotiators' ability to enforce Ack policy in the presence of delegation credentials. We now summarize a storage type system and a notion of well-typed credentials that guarantee credentials proving attributes can be discovered even when they are stored in a distributed way. For additional information on the type system, refer to [5].

Each credential is assumed to be stored either by its issuer (an *issuer-stored* credential) or by its subjects (a *subject-stored* credential), where subject and issuer are defined as follows. For a type-1 credential, $A.R \leftarrow B$, we call A the *issuer*, and B the *subject*. For a type-2

credential $A.R \leftarrow B.R_1$, we call A the *issuer* and B the *subject*.

Each attribute name has two types: an issuer-side type and a subject-side type. Attributes (e.g., $A.R = A.r(\cdot \cdot \cdot)$) inherit these types from their attribute names (r). On the issuer side, the possible types are issuer-(traces-)none, issuer-(traces-)def, and issuer-(traces-)all, in order of increasing strength. If R is issuer-def or issuer-all, each credential of the form $A.R \leftarrow e$ is required to be issuer-stored. (In the current section, e is always an attribute or an entity. Other attribute expressions, introduced in the following sections, have types that are determined from their constituent attribute terms, which will be discussed below as they are introduced.) The significance for credential discovery is that knowing the issuer A , a search process can retrieve $A.R \leftarrow e$, finding the issuer of e . If R is issuer-all, the well-typing rule for credentials additionally requires e to be issuer-all. This means that a backward search process can iterate the credential retrieval step to discover all entities that satisfy the attribute and the credentials that prove it⁴.

On the subject side, the possible types are subject-(traces-)none and subject-(traces-)all. If R is subject-all, each credential of the form $A.R \leftarrow e$ is required to be subject-stored, and e must also be subject-all. This means that a forward search process can start from an entity and discover all its subject-all attributes and the credentials that prove them. To ensure that credentials are either issuer-stored or subject-stored, the well-typing rule also requires that no attribute expression is both issuer-none and subject-none.

The typing system ensures that every minimal credential set proving N satisfies $A.R$ can be partitioned into a set of issuer-stored credentials that are reachable from A , and a set of subject-stored credentials that are reachable from N . Thus, if prior to ATN each negotiator discovers and collects each credential that is reachable by tracing forward from itself or by tracing backward from each of its policies, then every credential that is necessary to enable successful negotiation will be available to one of the two negotiators during ATN.

4.3 Assumptions

Here we assume that the above typing system is in place. A type-1 credential showing that N satisfies a subject-all attribute will in general be available only from N . Thus it is consistent with the type system to assume that access to such a fact is under N 's control. As we will see, N can also achieve control over subject-all attributes that are proved by using type-2 credentials. However, it is impossible for an entity to protect an issuer-all attribute without relying on credential issuers to do it for him. This motivates the restriction that Ack policies be assigned only to attributes that are subject-all.

⁴This search is backward with respect to the orientation of the arrow in credentials.

Similarly, we assume N will answer queries only about subject-all attributes. This ensures that N can find the credentials needed to show he satisfies the query before he actually receives the query. It is justified because the type systems ensures that the opponent, unlike N , has the ability before ATN begins to find the credentials needed to decompose a query about an issuer-def attribute into one or more queries about subject-all attributes.

In addition to searching for credentials in a forward direction starting from itself, each negotiator will conduct a similar forward search starting from each attribute it considers sensitive. Doing so enables the negotiator to prevent its opponent being able to make unauthorized inferences about sensitive attributes that it otherwise would be able to make based on the negotiator's inability to present these credentials. Note that this search for credentials is possible only beginning at attributes whose issuers are known.

We make the worst-case assumption that O has access to all type-2 credentials. Note that some attributes may be defined only by such credentials, as some issuers may delegate their authority to other parties and not issue type-1 credentials. We presume that O may know that this is the case for some issuers.

4.4 Preventing Unauthorized Inference of Sensitive Attributes

Based on the underlying logic of the credential system, there are four ways that a negotiator's opponent can infer additional attribute information by using answers to queries that the negotiator provides. In the following four subsections we present techniques that the negotiator can use to prevent each of these.

4.4.1 Backward, negative inference

When the opponent, O , knows that there is a credential $A.R \leftarrow B_1.R_1$, and that the negotiator, N , does not satisfy $A.R$, O can conclude that N does not satisfy $B_1.R_1$. We call this backward, negative inference, as it infers negative information about $B_1.R_1$ and the direction of the inference is the reverse of that of the arrow in the credential.

This form of inferential breach was identified in [9]; the TTG protocol in [9] protects against it by using a step-by-step, resolution-like process to reduce queries and reveal credentials, as outlined in section 2. When a query $\langle A.R \stackrel{?}{\leftarrow} N \rangle$ is encountered, N first enforces the Ack policy on $A.R$ through the control-edge mechanism presented in section 2. When that is satisfied, N acknowledges that it does not have any type-1 credential proving $A.R$; however, it will reveal the credential $A.R \leftarrow B_1.R_1$ and reduce the query to $\langle B_1.R_1 \stackrel{?}{\leftarrow} N \rangle$. N now enforces the Ack policy of $B_1.R_1$. After O satisfies the Ack policy of $B_1.R_1$, assuming that there are no other credentials defining $B_1.R_1$, N acknowledges this fact.

Note that N must know that the credential $A.R \leftarrow$

$B_1.R_1$ exists to be able to follow this method. It is for handling this case that N must collect credentials that can be discovered by a forward search process, starting from each attribute that N considers sensitive. The impossibility of conducting such a search without knowing the issuer of the sensitive attribute is one of the factors preventing us from being able to enforce Ack policy on attributes whose issuer is unknown.

4.4.2 Backward, positive inference

When O knows that the only credential defining $A.R$ is $A.R \leftarrow B_1.R_1$, and that N satisfies $A.R$, then O knows that N satisfies $B_1.R_1$; we call this backward positive inference. Although this form of inferential breach was not identified in [9], the strategy used in the TTG protocol to prevent backward negative inference also protects against backward positive inference. This is because, using the step-by-step ATN procedure, N will not answer the query $\langle A.R \stackrel{?}{\leftarrow} N \rangle$ without first reducing it to $\langle B_1.R_1 \stackrel{?}{\leftarrow} N \rangle$ and enforcing the Ack policy on $B_1.R_1$.

4.4.3 Forward, positive inference

When O knows that there is a credential $A.R \leftarrow B_1.R_1$, and that N satisfies $B_1.R_1$, one can conclude that N satisfies $A.R$. We call this forward, positive inference. This form of inferential breach was identified in [9], but not fully dealt with. To deal with it, first, N needs to know that $A.R \leftarrow B_1.R_1$ exists. This is handled by the assumption that N does forward search from itself. Since $A.R$ is subject-all, so is $B_1.R_1$, by the well-typing rules; furthermore, we assume in the current case that N satisfies $B_1.R_1$, so the type system ensures that N can discover this fact, and the credentials that prove it, through the forward search process. From there, N can find $A.R \leftarrow B_1.R_1$.

Now, N needs to ensure that the Ack policy for $B_1.R_1$ is at least as strong as that of $A.R$. We call this requirement *Ack policy closure*. The most straightforward way to ensure this is simply to conjoin the latter Ack policy to the former. The result is that if N enforces the Ack policy on $B_1.R_1$ before indicating whether he satisfies the attribute $B_1.R_1$, then O will not be able to use this credential to infer that N satisfies $A.R$ unless O has already satisfied the Ack policy of $A.R$. Note that N will have to enforce Ack policy closure over all credentials obtained by the forward search process described above.

4.4.4 Forward, negative inference

When O knows that the only credential defining $A.R$ is $A.R \leftarrow B_1.R_1$, and that N does not satisfy $B_1.R_1$, then O can conclude that N does not satisfy $A.R$. We call this forward, negative inference. The scenario can be generalized to the case that there are several credentials defining $A.R$.

The above inference is justified only when $A.R$ is known not to be defined by type-1 credentials, since only then does a negative answer to $B_1.R_1$ imply a negative

answer to $A.R$. In particular, one cannot make the inference unless one is certain there is no type-1 credential $A.R \leftarrow N$. Let us assume for the moment that credential $A.R \leftarrow B_1.R_1$ is known to N before the negotiation, and that N enforces Ack policy closure with respect to this credential. In this case the Ack policy for $B_1.R_1$, which N will enforce via the step-by-step TTG procedure before acknowledging he does not satisfy $B_1.R_1$, is at least as strong as that of $A.R$. So the inference that N does not satisfy $A.R$ is not a violation. Note that this argument hinges on the assumption that N knows of credential $A.R \leftarrow B_1.R_1$.

This form of inferential breach was not discussed in [9]. It can be prevented if we assume that for each sensitive attribute, $A.R$, N discovers some way in which the satisfaction of $A.R$ could in principle be shown. Specifically, it suffices if N knows that A issues type-1 credentials defining $A.R$ (although such a credential presumably has not been issued to N). Otherwise, N needs to obtain a type-2 credential defining $A.R$, such as $A.R \leftarrow B_1.R_1$. We call such a credential a *witness*. In this case, N must enforce Ack policy closure with respect to the witness credential. In doing so, N will ensure that the Ack policy of $B_1.R_1$ is at least as strong as that of $A.R$. Before giving a negative answer to a query about $B_1.R_1$, which is necessary for O to make a negative inference about $A.R$, N will enforce the Ack policy of $B_1.R_1$, and with it the Ack policy of $A.R$.

Note that N will inductively need to know at least one way in which satisfaction of $B_1.R_1$ could be shown. Thus in general, N has to find a chain of witness credentials starting at an attribute for which it knows type-1 credentials are issued. Also note that, as in section 4.4.1, our technique for avoiding this form of breach depends on the user knowing the issuer of the sensitive attribute, in this case A .

An alternative approach. The drawback to the approach discussed here is that the negotiator has to search for credentials in a direction not supported by the type system, since sensitive attributes are typically not issuer traceable. Thus, the user must take responsibility for discovering them. We argue that this is reasonable, since the user presumably knows something about how an attribute he considers sensitive is defined. However, it is natural to consider whether there is another approach that the type system would support.

The type system supports the discovery of an entity's subject-all attributes through a forward search process that starts from the entity. It also supports searching forward from an arbitrary attribute expression to find others whose satisfaction is entailed by satisfaction of the first. Thus, when a query is posed during ATN, a negotiator can search to find sensitive attributes that are entailed by the query, as well as the credentials that show the entailment. The negotiator can now apply Ack policy closure as needed with respect to these credentials to ensure that the Ack policy he enforces on the query

is adequately strong. However, the required dynamic search process adds a significant burden, and may open the door to inference attacks based on timing. So it is not recommended.

4.5 Summary of Steps in Preparing for Negotiation

While theoretically negotiators could collect on the fly the credentials needed during negotiation, the time required for this would make it impractical. In addition to collection, it is necessary to enforce Ack policy closure with respect to collected credentials. Thus, a negotiator N should prepare for negotiation by taking the following steps beforehand:

1. Collect credentials through a forwards search process starting from N itself.
2. Determine its set of sensitive attributes and their associated Ack policies.
3. Ensure that for each sensitive attribute that N does not satisfy, N has a chain of witness credentials that together protect the attribute against forward, negative inference.
4. Enforce Ack policy closure over every credential discussed in steps 1 and 3.
5. Collect credentials through a forward search process that starts from N 's sensitive attributes (including those made sensitive in step 4).
6. Enforce Ack policy closure over every credential discussed in step 5.
7. Collect credentials through a backwards search process starting from its Ack policies and from any access control policies it enforces.

Every attribute made sensitive in step 6 will be visited by the forward search procedure during step 5. Consequently, performing an additional forward search from such an attribute would not lead to the discovery of any new credentials, making it unnecessary. Step 3 suffices to protect against forward, negative inference because any attribute made sensitive in step 4 or 6 that N does not satisfy will already have a witness credential, obtained either in step 3 or in step 5, and with respect to which Ack policy closure will be enforced in step 4 or 6, respectively.

4.6 Use Only Universally Available Delegation Credentials

The goal of ATN is to protect sensitive negotiator attributes while enabling needed credentials to flow to authorized entities. Negotiators are responsible for providing type-2 credentials that they typically must obtain for this purpose from third parties. This creates a potential for uncontrolled information flow unless such credentials are universally available. The problem is

that, if a third party credential is protected, and therefore available only to entities with certain attributes, a negotiator’s opponent can infer that the negotiator satisfies those attributes based on his ability to present the protected credential. One solution is to assume that only type-1 credentials are protected, and type-2 credentials are not. This is consistent with the goal of protecting sensitive attributes, since no attribute can be shown by using only type-2 credentials.

5. CONJUNCTIONS IN CREDENTIALS

5.1 Type-4 Credentials

We now add credentials of the form $A.R \leftarrow A_1.R_1 \wedge \dots \wedge A_k.R_k$, which are known as type-4 credentials [4]. Such a credential says that an entity satisfies $A.R$ if it satisfies each $A_j.R_j$, for $1 \leq j \leq k$. More precisely, a type-4 credential has the form $A.R \leftarrow A_1.R_1 \wedge \dots \wedge A_k.R_k : X_1 \in S_1 \wedge \dots \wedge X_m \in S_m$, in which every variable that appears in R also appears in some R_i , and each X_j for $1 \leq j \leq m$ appears in some R_i .

Policy rules remain unchanged. As in section 4, Ack policy is restricted to subject-all attributes.

5.2 Impact of Type-4 Credentials on Ack Policy and Its Enforcement

The nature of Ack policy is not changed by the introduction of type-4 credentials. In particular, Ack policy is not permitted to be associated with intersections. Enforcement of such a policy would require negotiators to record the history of queries. Moreover, even a correct enforcement mechanism would be vulnerable to attack by collusion if different opponents pool information about different attributes that each one is authorized to know about. (While we assume that entities who are entitled to attribute information do not give the information to others, entities may nonetheless collaborate to obtain information that none of them is authorized to have.)

The addition of type-4 credentials does not interfere with the TTG method preventing unauthorized backward inference. The two forms of forward inference do, however, require additional attention.

Forward, positive inference. For each type-4 credential, $A.R \leftarrow A_1.R_1 \wedge \dots \wedge A_k.R_k$, such that $A.R$ is sensitive, to prevent unauthorized forward, positive inference, N must ensure that the Ack policies of the $A_j.R_j$ are together at least as strong as the Ack policy of $A.R$. In this case, ensuring Ack policy closure is a more interesting problem than with type-2 credentials. We do not give a general solution here; however, it appears a solution is readily achievable, for instance by using Lambda Prolog [6].

Forward, negative inference. Suppose N wants to use $A.R \leftarrow A_1.R_1 \wedge \dots \wedge A_k.R_k$ as a witness credential for preventing forward, negative inference

of $A.R$. Then N needs to make the Ack policy of each $A_j.R_j$ as strong as that of $A.R$, which is easily done by conjoining the latter to the former. This ensures that the Ack policy of $A.R$ is enforced before O learns that N does not satisfy $A_j.R_j$.

6. ATTRIBUTE-BASED DELEGATION

6.1 Type-3 Credentials

We now add credentials of the form $A.R \leftarrow A.R_1.R_2$, which are known as type-3 credentials [4]. Such a credential says that the set of entities satisfying $A.R$ contains the set satisfying $B.R_2$ for every B satisfying $A.R_1$. This enables A to delegate authority to B based not on B ’s identity, but based on B ’s attribute, $A.R_1$. We call $A.R_1.R_2$ a *linked attribute*.

To be more precise, a type-3 credential has the form $A.R \leftarrow A.R_1.R_2 : X_1 \in S_1 \wedge \dots \wedge X_m \in S_m$, in which every variable that appears in R also appears in R_1 or in R_2 , and each X_j for $1 \leq j \leq m$ also appears in R_1 or in R_2 . A policy rule now has the form $e_1 \wedge e_2 \wedge \dots \wedge e_k ; X_1 \in S_1 \wedge \dots \wedge X_m \in S_m$, in which each e_j is either an attribute $A_j.R_j$ or a linked attribute, $A_j.R_{1,j}.R_{2,j}$, each field is either a constant or a variable, each X_j for $1 \leq j \leq m$, is a variable that appears in $e_1 \wedge \dots \wedge e_k$, and each S_j for $1 \leq j \leq m$ is a value set constraining the legal values of X_j .

6.2 Extending Ack Policy to Protect Sensitive Linked Attributes

We can allow Ack policy to be associated with $A.R_1.R_2$ provided the expression is subject-all⁵. Such a policy protects knowledge of whether there exists a B such that N satisfies $B.R_2$. Note that the meaning is not to protect knowledge of each $B.R_2$ for which B satisfies $A.R_1$. In particular, when B is unknown to N , N will not protect the fact that N does not satisfy $B.R_2$.

- To defend against unauthorized forward, positive inference of $A.R_1.R_2$, N must discover all B such that N satisfies $B.R_2$. Then N must determine which of these B ’s satisfy $A.R_1$. For each such B , N must ensure that the Ack policy of $B.R_2$ is as strong as that of $A.R_1.R_2$.
- To defend against unauthorized forward, negative inference, N must find some B in $A.R_1$ that defines $B.R_2$, and N must determine how $B.R_2$ can be defined (i.e. by type-1 credentials or by some witness credential that N must discover). This is similar to the case discussed in section 4.4.4.
- To defend against unauthorized backward (negative or positive) inference, N must perform forward search and credential collection from each sensitive linked attribute, $A.R_1.R_2$. When N satisfies $A.R_1.R_2$, N has to do this anyway as part of

⁵This form of Ack policy is new and was not discussed in [9].

performing forward search from himself, in order to enable all satisfied attributes to be shown.

6.3 Negotiating with Linked Attributes

Suppose the opponent O issues a query $\langle A.R_1.R_2 \stackrel{?}{\leftarrow} N \rangle$. Here we extend the approach specified in [9] to handle sensitive linked attributes. There are three cases.

When R1 is issuer-all, O can find all B satisfying $A.R_1$, so O reduces the query to one of the form $\langle B.R_2 \stackrel{?}{\leftarrow} N \rangle$ for each such B .

When R1 is issuer-def, O does not reduce $A.R_1$ because in general doing so could cause the query size to grow without bound. Instead, O reduces the query $\langle A.R_1.R_2 \stackrel{?}{\leftarrow} N \rangle$ to a new form of query, $\langle ?X.R_2 \stackrel{?}{\leftarrow} N \rangle$, where $?X$ is a placeholder for an arbitrary, unspecified issuer. N is being asked whether he satisfies $B.R_2$ for any B . N expands $\langle ?X.R_2 \stackrel{?}{\leftarrow} N \rangle$ by adding a new edge connecting it to $\langle B.R_2 \stackrel{?}{\leftarrow} N \rangle$ for each $B.R_2$ that N is aware of (which happens if N considers $B.R_2$ sensitive or if N has a credential defining $B.R_2$)⁶. N can now enforce any Ack policy associated with $B.R_2$.

When R1 is subject-all, N processes the query. If N considers $A.R_1.R_2$ sensitive, N can protect the linked attribute by adding a control child that must be satisfied before N will reduce the query. Once this control child, if any, is satisfied, N reduces the query to $\langle ?X.R_2 \stackrel{?}{\leftarrow} N \rangle$, as in the above case, thereby continuing the step-by-step, resolution-like process whereby backward breach of Ack policy is prevented.

6.4 Impact on the Type System

For a given credential like $A.R \leftarrow A.R_1.R_2$, we call A both the subject and the issuer. As presented in [5], a linked attribute $A.R_1.R_2$ is issuer-all when both R_1 and R_2 are issuer-all, and subject-all when both R_1 and R_2 are subject-all. $A.R_1.R_2$ is issuer-def if R_1 is issuer-def and R_2 is subject-all, or R_1 is issuer-all and R_2 is issuer-def.

6.5 Impact on Defense against Breach

The step-by-step, resolution-like negotiation procedure, extended as outlined above, continues to prevent unauthorized backward inference of attributes.

Consider $A.R \leftarrow A.R_1.R_2$ for which $A.R$ is sensitive. To protect against unauthorized forward, positive inference of $A.R$, N must protect $B.R_2$ as strongly as N protects $A.R$, for each $B.R_2$ that N satisfies and for which B satisfies $A.R_1$. N can find all such $B.R_2$ by searching

⁶If it is later shown that such a query succeeds, O will add a corresponding edge connecting $\langle A.R_1 \stackrel{?}{\leftarrow} B \rangle$ to $\langle A.R_1.R_2 \stackrel{?}{\leftarrow} N \rangle$. See [9] for details.

from N itself because $B.R_2$ is subject-all. (That $B.R_2$ is subject-all follows by the well-typing rules for credentials from the fact that $A.R$ is sensitive, and hence subject-all.)

If N decides to use $A.R \leftarrow A.R_1.R_2$ as a witness credential for preventing forward, negative inference of $A.R$, then N needs to find a B such that he knows how $B.R_2$ can be shown. In other words, either B must issue facts or N must have a type-2 credential $B.R_2 \leftarrow D.R_3$ such that, inductively, N knows how $D.R_3$ can be shown.

6.6 Query Hiding

An idea not discussed in [9] is for N to hide the children he adds to queries like $\langle ?X.R_2 \stackrel{?}{\leftarrow} N \rangle$ when those children correspond to sensitive attributes. N just introduces the node into the graph without labelling them. Then N adds control edges. When O satisfies the control children, N can add the label. This technique avoids giving O a complete list of the R_2 attributes that N considers sensitive. O can still find out whether any given $B.R_2$ is sensitive by posing $\langle B.R_2 \stackrel{?}{\leftarrow} N \rangle$ (for instance, as a control child on some query about O) and seeing whether N adds a control child to it. Thus, query hiding can slow, but not stop information flow concerning what attributes are sensitive.

6.7 Pitfalls of Proving Attributes of Others

To show he satisfies $A.R_1.R_2$, N must show that some B satisfies $A.R_1$. An issue that has not to our knowledge been raised concerning ATN is that N 's ability to prove an attribute of B 's may reveal something about N 's attributes. If B protects the credentials needed to show an attribute, and N produces those credentials, N shows he satisfies B 's policy for access to them.

This problem can be side stepped if we assume that O does not pose and N does not answer queries of the form $\langle A.R_1.R_2 \stackrel{?}{\leftarrow} N \rangle$ where $A.R_1$ is considered sensitive by issuers of attributes based on R_2 . In most cases this seems reasonable. Finding a more comprehensive solution remains an open problem.

7. RELATED WORK

In this section we further discuss related work. Automated trust negotiation was introduced by Winsborough et al. [10], who introduced two negotiation strategies, an eager strategy in which negotiators disclose each credential as soon as its access control policy is satisfied, as well as a strategy in which negotiators disclose credentials only after exchanging sufficient policy content to ensure that a successful outcome is ensured. The former strategy has the problem that many irrelevant credentials may be disclosed; the latter, that negotiators reveal which credentials they hold in an uncontrolled way, by transmitting access control policy content for them. Yu et al. [11] introduced an efficient strategy that explicitly requires negotiators to reveal to strangers

which credentials they hold. Seamons et al. [7] introduced structures called policy graphs for protecting policy content, and strategies enforcing policy graphs. Yu et al. [12, 13] developed families of strategies that can interoperate in the sense that negotiators can use different strategies within the same family. This previous work does not address the problem of protecting information about whether you hold a credential or satisfy an attribute.

The Platform for Privacy Preferences Project (P3P) [2] enables user agents to automate decisions about disclosure of user data based on what data is requested and the Web site privacy policies. The assumption is the Web site does not consider its privacy practices to be sensitive, so any negotiation is relatively simple.

8. CONCLUSION

We have presented techniques that extend the ATN procedures given in [9] so as to fully enforce Ack policy. These techniques prevent breach of Ack policy by means of deductive inference in the underlying logic of the credential language. We have also considered other potential means of breach, and shown how they can be prevented. Throughout, we have considered the implications of the necessity to discover credentials in the decentralized context where ATN has its role. In addition, we have explained the factors that have determined which attribute structures are permitted to have Ack policy associated with them. Specifically, we have explained why Ack policy can be associated neither with a conjunction nor with an attribute whose issuer is not known. Finally, we have identified the open problem of whether credentials issued to and protected by third parties can safely be used in ATN.

Acknowledgments

This work is supported by DARPA through SPAWAR contracts N66001-01-C-8005 and N66001-00-C-8015. We thank the anonymous reviewers for their helpful suggestions.

9. REFERENCES

- [1] Dwaine Clarke, Jean-Emile Elie, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.
- [2] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The platform for privacy preferences 1.0 (P3P1.0). World Wide Web Consortium Recommendation, April 2002.
- [3] Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. SPKI certificate theory. IETF RFC 2693, September 1999.
- [4] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, May 2002.
- [5] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management. To appear in *Journal of Computer Security*. Extended abstract appeared in *Proceedings of the Eighth ACM Conference on Computer and Communications Security*, 2001.
- [6] Gopalan Nadathur. A proof procedure for the logic of hereditary harrop formulas. *Journal of Automated Reasoning*, 11:115–145, 1993.
- [7] Kent E. Seamons, Marianne Winslett, and Ting Yu. Limiting the disclosure of access control policies during automated trust negotiation. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS'01)*, February 2001.
- [8] Kent E. Seamons, Marianne Winslett, Ting Yu, Lina Yu, and Ryan Jarvis. Protecting privacy during on-line trust negotiation. In *2nd Workshop on Privacy Enhancing Technologies*. Springer-Verlag, April 2002.
- [9] William H. Winsborough and Ninghui Li. Towards practical automated trust negotiation. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002)*, pages 92–103. IEEE Computer Society Press, June 2002.
- [10] William H. Winsborough, Kent E. Seamons, and Vicki E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, pages 88–102. IEEE Press, January 2000.
- [11] Ting Yu, Xiaosong Ma, and Marianne Winslett. Prunes: An efficient and complete strategy for trust negotiation over the internet. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS-7)*, pages 210–219, November 2000.
- [12] Ting Yu, Marianne Winslett, and Kent E. Seamons. Interoperable strategies in automated trust negotiation. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS-8)*, pages 146–155. ACM Press, November 2001.
- [13] Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security (TISSEC)*, 6(1), February 2003. To appear.