

Safety in Automated Trust Negotiation

WILLIAM H. WINSBOROUGH

University of Texas at San Antonio

and

NINGHUI LI

Purdue University

Exchange of attribute credentials is a means to establish mutual trust between strangers wishing to share resources or conduct business transactions. Automated Trust Negotiation (ATN) is an approach to regulate the exchange of sensitive information during this process. It treats credentials as potentially sensitive resources, access to which is under policy control. Negotiations that correctly enforce policies have been called “safe” in the literature. Prior work on ATN lacks an adequate definition of this safety notion. In large part, this is because fundamental questions such as “what needs to be protected in ATN?” and “what are the security requirements?” are not adequately answered. As a result, many prior methods of ATN have serious security holes. We introduce a formal framework for ATN in which we give precise, usable, and intuitive definitions of correct enforcement of policies in ATN. We argue that our chief safety notion captures intuitive security goals. We give precise comparisons of this notion with two alternative safety notions that may seem intuitive, but that are seen to be inadequate under closer inspection. We prove that an approach to ATN from the literature meets the requirements set forth in the preferred safety definition, thus validating the safety of that approach, as well as the usability of the definition.

Categories and Subject Descriptors: K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls, information flow controls*

General Terms: Algorithms, Design, Security, Theory, Verification

Additional Key Words and Phrases: Access control, attribute-based access control, automated trust negotiation, credentials, safety, strategy

A preliminary version of this paper appeared in *Proceedings of 2004 IEEE Symposium on Security and Privacy* under the title “Safety in Automated Trust Negotiation”, @2004 IEEE.

Some of this work was performed while the first author was at Network Associates Laboratories in Rockville, MD 20850, while he was at the Center for Secure Information Systems, George Mason University, Fairfax, VA 22030-4444, and while the second author was at the Department of Computer Science, Stanford University in Stanford, CA 94305.

Authors’ addresses: William H. Winsborough, Department of Computer Science, University of Texas at San Antonio, 6900 North Loop 1604 West, San Antonio, TX 78249-0667, USA; email: wwinsborough@acm.org. Ninghui Li, Department of Computer Sciences, Purdue University, 656 Oval Drive, West Lafayette, IN 47907-2086, USA; email: ninghui@cs.purdue.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2006 ACM 1094-9224/06/0800-0352 \$5.00

ACM Transactions on Information and System Security, Vol. 9, No. 3, August 2006, Pages 352–390.

1. INTRODUCTION

In Attribute-Based Access Control (ABAC) systems, access control decisions are based on attributes of requesters. These attributes may be arbitrary properties of the requester, such as roles he assumes in his home organization, rights to access specific resources, licenses or degrees held, age, *etc.*, and are often documented by digitally signed credentials. A principal proves that it has an attribute by showing an appropriate set of relevant credentials. Because attributes (such as financial or medical status) may be sensitive, they need protection. The goal of a growing body of work on *automated trust negotiation (ATN)* [Hess et al., 2002; Seamons et al. 2001, 2002; Winsborough and Li 2002a, 2002b; Winsborough et al. 2000; Winslett et al. 2002; Yu et al. 2000; Yu and Winslett 2003b; Yu et al. 2003] is to enable resource requesters and access mediators to establish trust in one another through cautious, iterative, bilateral disclosure of credentials. The distinguishing characteristic of ATN that differentiates it from most other trust establishment schemes (e.g., [Bonatti and Samarati 2000; Herzberg et al. 2000]) is that credentials themselves are treated as protected resources.

Prior work on ATN lacks an adequate notion of security. Fundamental questions such as “what needs to be protected in ATN?” and “what are the security requirements?” are not adequately answered. The main purpose of this paper is to answer some of these questions by providing a formal ATN framework with precise and appropriate security definitions. Let us illustrate the deficiencies of security definitions in existing ATN work.

In most ATN frameworks, each negotiator establishes access control (AC) policies to regulate the disclosure of credentials to negotiation opponents. A typical description in the literature of the safety requirement for AC-policy-based ATN is the one given in Yu et al. [2003]: “given a sequence $Q = \{C_1, \dots, C_n\}$ of disclosures of protected resources, if each C_i is unlocked at the time it is disclosed, then we say Q is a *safe disclosure sequence*.” Here, *unlocked* means that the AC policy for the credential is satisfied by credentials disclosed previously by the other party, and a credential is one kind of resource. This deceptively simple requirement turns out to be far from adequate in ensuring that an ATN system protects the privacy of sensitive attributes. Several groups of researchers have noted [Seamons et al. 2002; Winsborough and Li 2002b; Yu and Winslett 2003a] that although early ATN designs satisfy the safety requirement for AC policies, they nonetheless fail to adequately protect the privacy of negotiators. Thus, it is recognized that a problem exists with ATN’s traditional notion of safety. The problem stems from the fact that the traditional notion is satisfied by ATN designs in which, although a sensitive credential itself is not transmitted until its associated AC policy is satisfied, the behavior of a negotiator reveals a great deal about the contents of these credentials. Indeed, most ATN designs do just that. When a negotiator is asked to prove a sensitive attribute, the negotiator’s behavior depends on whether it has the attribute or not. By observing the negotiator’s behavior, the negotiator’s opponent can infer whether the negotiator has a sensitive attribute or not. Thus, in the case where simply having or not having the attribute is private information, while the negotiator’s

opponent may not yet have proof of the authenticity of the attribute, the privacy of the attribute has certainly been compromised. In Seamons et al. [2002], some ad hoc solutions are proposed. For example, it is suggested that instead of transmitting the AC policy, a negotiator having a sensitive attribute could simply behave as though he did not, and just wait, hoping the opponent will happen to send enough credentials to satisfy the AC policy.

Trust negotiation is of little value if participants must mislead one another to protect sensitive information, since this would make many negotiations fail unnecessarily. Yet most prior negotiation techniques allow a negotiator's opponent to gain advantage just in case the negotiator is honest. As we show in this paper, one of the few existing ATN strategies that is immune from this problem is the *eager strategy* [Winsborough et al. 2000]. In it, each party transmits all credentials whose access control policies have already been satisfied, whether these credentials are related to the eventual negotiation goal or not. In the eager strategy, when a negotiator does not receive a given credential from the opponent, it does not know whether this is because the opponent does not have the credential, or because the negotiator simply has not satisfied the opponent's AC policy for that credential. However, because the eager strategy does not focus the exchange on credentials that are relevant to the authorization decision at hand, it is impractical for many scenarios.

In Winsborough and Li [2002b], an approach was proposed for focusing the credential exchange while simultaneously protecting sensitive attributes of negotiators. The approach is based on the notion of an *acknowledgement policy* ("ack" policy, for short). An ack policy resembles an AC policy, although it is associated with an attribute, rather than with a credential proving the attribute. The key difference from AC policy is that one can associate an ack policy with an attribute one does not have. This makes it possible to provide the ack policy without, by doing so, indicating whether one satisfies the associated attribute. Note that it is not possible to provide an AC policy for an attribute without indicating that one satisfies the attribute, because AC policies are established only for attributes that one satisfies. The intuitive goal of ack policies is that no one should learn through negotiation whether or not a negotiator N possesses an attribute without first satisfying its ack policy. This intuitive notion of safe enforcement of ack policies was not formalized in previous work using the concept Winsborough and Li [2002a, 2002b]. Therefore, it was impossible to prove that a given strategy using ack policies is safe.

The present goal is to articulate a suitable definition of this notion that is precise, usable, and intuitive. The definition should be precise and usable so that one can prove safety of negotiation strategies using the definition. The definition should also be intuitive in that ATN systems satisfying it should fulfill our expectations that sensitive attributes of the negotiator be protected from unintended disclosure. This goal is in keeping with the research tradition in information security and cryptography of finding security definitions for numerous problems and protocols that are suitably precise, usable, and intuitive.

The approach we take in this paper is to formalize the following intuition about safe enforcement of ack policies: unless N 's negotiation opponent satisfies the ack policy for a sensitive attribute, N 's behavior in the negotiation

must give no indication of whether N possesses any credentials documenting the sensitive attribute. As we will see, the details of the safety condition are somewhat intricate, and simply preventing an adversary from determining specific attributes is inadequate. In light of this notion of safety, ack policies are exactly the sort of policies that are required to enable negotiators to behave safely while presenting to one another information about interdependencies among candidate credential disclosures: only by using policies that do not depend significantly on his attributes can a negotiator present policy content governing disclosure of his attributes without leaking information about his likely attributes.

Accommodating diverse credential systems requires effort. In particular, We seek a notion of safety that can be supported by systems in which credentials can represent delegations of authority. Such credential systems support a limited form of deduction, which means we must prevent safety being breached through deductive inference. The threat of probabilistic inference also influences the selection of an appropriate safety condition.

Let us outline our safety definition. We first formalize the ability of an adversary to distinguish between one negotiator and another. For each negotiator N and each adversary M , there is a set U of attributes whose ack policies are not satisfied by M . We define a strategy to be safe if any other negotiator N' who differs from N only in credentials that prove attributes in U is indistinguishable from N by M based on ATN. We discuss other definitions that capture similar, but different intuitions about safety, showing they are strictly weaker and inadequate in various respects. Our notions of safety have important similarities to the notions of noninterference [Goguen and Meseguer 1982] and nondeducibility [Sutherland 1986], which we discuss in Section 6.

Although we conclude that AC policies are inadequate for protecting information about whether or not a principal has a given attribute, they may nonetheless be useful for protecting other information in credentials. For instance, they may be adequate when having the attribute is not sensitive, but a given attribute *value* is sensitive. Everyone has a blood type, but one's particular blood type might be somewhat sensitive. If a credential containing one's blood type is protected by an AC policy, no information is leaked by disclosing that one has such a credential, so long as the contents of the credential are not revealed. Thus, we also provide a precise definition of safe enforcement of AC policies.

The contributions of this paper are as follows:

1. A formal framework for trust negotiation and a precise definition of safety for enforcement of ack policies in that framework.
2. Proof that the eager strategy is safe based on this formal definition.
3. A formal analysis of the relationship between our safety definition and two alternative definitions that also seem intuitive.
4. An analysis that shows why our first safety definition is preferable to the two alternatives mentioned above.
5. A family of strategies based on the Trust Target Graph (TTG) protocol [Winsborough and Li 2002b] that supports flexibility in the search for a successful negotiation.

6. A proof that all strategies in this family are safe according to the formal definition. This validates both the usability of the definition and the design of the TTG framework.
7. A precise definition of safety for AC policies that can be used with cryptographic ATN protocols.

The rest of this paper is organized as follows. In Section 2, we discuss in detail why previous notions of safety are inadequate. Section 3 presents contributions 1–4 from the list above; Section 4 reviews the TTG protocol and presents contributions 5 and 6. In Section 5, we discuss deficiencies of previous safety definition for AC policies and give our definition (contribution 7). We discuss related work in Section 6 and conclude in Section 7. The appendix contains proofs of theorems.

2. PRIOR UNSATISFACTORY NOTIONS OF SAFETY

Most existing negotiation strategies are safe according to the limited definitions laid out for them by their designers. However, as we show in this section, they are not safe in the sense of protecting the content of credentials, which is arguably the central goal of ATN: if credential content did not need protection, requesters could simply push all their credentials to the access mediator for evaluation. This leads us to the inevitable conclusion that the definition of safety set forth in prior work is inadequate, thereby motivating our introduction of adequate definitions in Section 3. Consider the following illustration of unsafe behavior exhibited by typical strategies.

Example 1. Bob obtains a credential from the Internal Revenue Service (IRS) documenting his low-income status. This credential is useful, for example, when Bob interacts with a nonprofit organization that offers a service preparing free living wills over the Internet for people with low incomes. Suppose Bob uses an AC policy recommended by the IRS for protecting this credential, which says that Bob will show his IRS.lowIncome credential to organizations that document they are registered with the IRS as nonprofits. Bob uses his ATN-enabled browser to contact an ATN-enabled service provided by a nonprofit to obtain a living will and Bob's browser and the service's access mediator will negotiate successfully.

Another Web user, Alice, does not have a low-income credential. Alice and Bob each visit the web site of an unfamiliar real estate service, SwampLand.com. When Alice and Bob each request information about listed properties, the SwampLand access mediator initiates a negotiation requesting Alice and Bob prove they have low-income status, which is not an appropriate requirement. If Alice and Bob use a typical ATN strategy, such as the TrustBuilder1-Relevant Strategy [Yu et al. 2003], this request induces Bob to present his AC policy for his low-income credential. The aim in doing so is to prompt SwampLand to present a nonprofit credential, should it have one. This enables Bob, for instance, to determine whether SwampLand is authorized to receive Bob's low-income credential without SwampLand having to transmit all its credentials to Bob. By contrast, the same request for a low-income credentials causes the negotiation with Alice to fail, since Alice does not have the requested credential

and, therefore, has no AC policy she can present, since AC policies are defined only for credentials one actually has. SwampLand.com can easily observe the difference between Alice and Bob's behaviors and deduce Bob's low-income status, even though Bob's AC policy indicates he does not want to share that information with for-profit companies. Granted, SwampLand.com does not obtain proof that Bob is low-income. However, this should provide Bob little comfort in using the ATN strategy, as SwampLand.com's unauthorized inferences are accurate just in case he adheres to the protocols faithfully. Similarly, SwampLand.com can deduce that Alice does not have the credential, although Alice also may not wish this to be disclosed.

The unsafe behavior illustrated in this example is not an artifact of the details of the TrustBuilder strategy, but rather characterizes most ATN strategies. It occurs because these strategies transmit AC policies, or information derived from them, in an effort to focus exchanges on credentials that are relevant to enabling the negotiation to succeed. This focus aims to reduce message size and other resource utilization, as well as to avoid distributing sensitive information needlessly. Assuming ATN strategy should not fail when success is possible, the competing goal of protecting sensitive attributes and this goal of focused disclosure seem to be at odds with one another. This is because of the nature of AC policies, namely, that they are associated only with attributes that the negotiator satisfies. A negotiator cannot consistently reply to credential requests by transmitting AC policies without revealing which credentials he has and which he does not have. Although it may be possible to obfuscate this information to some degree by replying in a less consistent manner [Seamons et al. 2002], doing so tends to cause negotiation to fail unnecessarily. An alternative is to introduce a form of policy that can be associated with an attribute whether or not the negotiator satisfies it. In keeping with Winsborough and Li [2002a, 2002b], we call such policies *ack* policies. We now use a simple example to show how *ack* policies can be used to stop information leakage.

Example 2. Continuing the scenario from Example 1, Bob adopts the IRS's recommended *ack* policy, which says that he will discuss the matter of low-income status only with nonprofit organizations registered with the IRS. Alice does not have the low-income status credential, but also considers information about her income status sensitive, so she also adopts the same *ack* policy.

There is a simple negotiation strategy according to which when Alice and Bob each visit the web site of SwampLand.com, and both are asked to prove they have low-income status, both Alice and Bob then ask SwampLand.com to prove that it is nonprofit. Therefore, SwampLand.com only learns that both Alice and Bob considers their income status sensitive, but not whether they are low-income or not.

2.1 Why Ack Policies Are Practical

It has been argued [Yu and Winslett 2003a] that the use of *ack* policies is unworkable because people who feel they have nothing to hide with respect to a given attribute will not bother to use an *ack* policy for that attribute, thereby casting suspicion on those who do. However, in most cases, a negotiator wishing

to protect some of his sensitive attributes by using ack policies also needs to enforce ack policies on some attributes about which he has nothing to hide. Otherwise the fact that he protects the attribute probably indicates that he has something to hide about whether he has that attribute. In particular, if a negotiator has some attributes that it discloses to a given opponent and some other attributes that it does not, the opponent can determine whether, among those attributes for which the opponent is authorized, the negotiator uses ack policies for attributes about which he has nothing to hide. If there are no such ack policies, the opponent can infer that the negotiator likely has something to hide with respect to other attributes with which the negotiator associates ack policies.

Thus, in the typical case, the negotiator wishes to protect some attributes about which he has nothing to hide and, therefore, little interest in designing ack policies. In such cases, if there were a straightforward mechanism for obtaining suitable ack policies for potentially sensitive attributes, the simplest course of action would be always to use them. After all, if exceptions were to be made, they would have to be specified.

We argue that defining appropriate ack policies and making them available should be part of attribute vocabulary design. Using some mix of natural and formal language, the vocabulary designer can be expected to explain the attributes he names. Characterizing the appropriate recipients of the named information can be viewed as part of that explanation. This answers the question of where ack policies come from.

To ensure that ack policies can be collected as needed, we can take advantage of a mechanism that also provides a nice solution to the problem of avoiding unintended collisions among attribute names. Name collisions can be prevented by making each attribute name include a reference, such as a URL, to a document describing the attribute vocabulary of which it should be interpreted as being a part. Names containing different vocabulary references cannot be misinterpreted as being the same. It is natural that the vocabulary description should include a description of appropriate recipients of information about attributes in the vocabulary. We argue that the latter description should be expressed as a formal policy that can be automatically retrieved by a negotiator. Using the suggested scheme for disambiguating the intended vocabulary, any request that a negotiator prove he has a certain attribute must include a reference to the vocabulary document. It, therefore, includes a reference to an ack policy recommended by the vocabulary designer—the premier expert in the meaning of the attribute. The negotiator can simply retrieve the ack policy and use it as his own in the remainder of the negotiation. By using the designer-recommended policy, the negotiator obtains not only convenience, but uniformity in his behavior with respect to that of other negotiators.

One minor drawback of the scheme is that if a negotiator retrieves an ack policy during a negotiation, this network activity may be observable by the adversary. A simple solution to this problem, however, is to retrieve the ack policy for every attribute the adversary inquires about during negotiation.

2.2 Safety and Delegation

Ensuring safe enforcement of ack policies is tricky. One difficulty comes from the fact that credentials may contain rules for deriving principals' attributes. Such rules are necessary in credential systems that express delegation of authority, as is almost essential in decentralized environments and is common in most access control languages designed for distributed access control. When credentials may contain delegations, having one attribute may imply having another attribute. Suppose, for instance, that a credential asserts that anyone who has attribute t_1 also has attribute t_2 . The following two kinds of inference can then be made.

- forward positive inference: If the opponent M knows that N has attribute t_1 , then M infers that N also has attribute t_2 (i.e., *modus ponens*).
- backward negative inference: If the opponent M knows that N does not have attribute t_2 , then M infers that M does not have t_1 either (i.e., *modus tollens*).

Furthermore, sometimes the only way of having the attribute t_2 is by having attribute t_1 . In that case M can perform the following two kinds of inference as well.

- backward positive inference: If M knows that N has attribute t_2 , then M infers that N also has attribute t_1 .
- forward negative inference: If M knows that N does not have attribute t_1 , then M infers that N does not have attribute t_2 either.

Because of the possibility of these (and maybe other) inferences, it is not obvious what the precise safety requirement for ack policies should be. Although previous work develops techniques to try to defend against these inferences, it is not clear whether these techniques satisfy the intended security requirements, since such requirements have not been defined in a precise way.

3. A FORMAL FRAMEWORK FOR TRUST NEGOTIATION

In this section, we present a formal framework for automated trust negotiation and precise definitions for safety in this framework. In Section 3.1, we set up the framework and in Section 3.2 we give the definition of the safety requirement for a negotiation strategy. In Section 3.3 we show that the eager strategy satisfies this safety requirement. In Section 3.4, we discuss two alternative safety notions that appeal to different intuitions and show that they are weaker than the definition in Section 3.2. We also present reasons why we ultimately dismiss each of these alternatives as inadequate. In Section 4, we apply the framework to a credential system that supports delegation, extending a strategy from the literature to obtain a family of strategies that satisfy our safety notion.

3.1 The Framework

An *ATN system* is a 7-tuple $\langle \mathcal{K}, \mathcal{T}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \text{Resource}, \text{Policy} \rangle$ whose elements are as follows:

- \mathcal{K} is a countable set of *principals*. Each principal is identified with a public key. We use K , possibly with subscripts, to denote principals.
- \mathcal{T} is a countable set of *attributes*. Each attribute t is identified by a pair containing an attribute authority (which is a principal) and an attribute name (which is a finite string over some standard alphabet).
- \mathcal{E} is a countable set of *credentials* that could legally be issued. We use e for members of \mathcal{E} and E for finite subsets of \mathcal{E} .
- $S : \mathcal{E} \rightarrow \mathcal{K}$ is a function; $S(e) \in \mathcal{K}$ is called the *subject* of credential e . If $S(e) = K$, e is a *credential for K* .
- $T : \mathcal{E} \rightarrow 2^{\mathcal{T}}$ is a function such that each $T(e)$ is finite and nonempty. A credential e *proves* that $S(e)$ *has* or *possesses* the attributes in $T(e)$. For each K and each set E of credentials for K , the set of attributes *induced* by E is $T(E) = \bigcup_{e \in E} T(e)$.
- Resource is a countable set of resources.
- Policy denotes the set of positive propositional logical formulas in which the propositions are attributes in \mathcal{T} . These formula are called *policies* and we use ϕ to denote one policy. If E is a set of credentials having subject K and if $T(E) \models \phi$, we say that ϕ is satisfied by K .

Possession of attributes in \mathcal{T} may be considered sensitive and the goal of ATN is to protect such information. In our framework, one credential may prove that its subject possesses more than one attribute. In addition to supporting credentials that explicitly aggregate attributes, this feature will be useful when we introduce delegation, in Section 4.

Notice that whether or not a credential proves possession of an attribute is independent of the presence of other credentials. In particular, given a set of credentials E , if no credential $e \in E$ proves an attribute t by itself, the combination of credentials in E does not prove t either. We can and do allow conjunctions of attribute to be required in policies. For instance, one can authorize a discount for principals with a valid university student ID and an ACM membership credential, but we do not enable one to define or protect the derived attribute of being a student member of the ACM, based on these two credentials. The student attribute or the ACM-member attribute must be protected if being a student member of ACM is sensitive.

In this model, a participant in the ATN system is characterized by a finite *configuration* G , which is given by $G = \langle K_G, E_G, \text{Ack}_G, \text{AC}_G \rangle$. The elements of G are as discussed below. (We drop the subscripts when G is clear from context.) We denote the set of all configurations by Configuration .

- K is the principal *controlled* by the participant; this means that the participant has access to the private key that corresponds to K , enabling the participant to prove itself to be the (presumably unique) entity controlling the key.
- $E \subset \mathcal{E}$ is the set of credentials that are assumed to have been issued for K in the configuration G .
- $\text{Ack} : \mathcal{T} \leftrightarrow \text{Policy}$ is a partial function mapping a finite subset of attributes in \mathcal{T} to policies. Attributes in the preimage of Ack are called sensitive attributes.

For sensitive attribute t , $\text{Ack}[t]$ is called the *ack policy* of t (in G). Ack can associate an ack policy with an attribute whether K possesses the attribute or not.

- $\text{AC} : \text{Resource} \mapsto \text{Policy}$ is a partial function mapping a finite subset of resources to policies. Resources in the preimage of AC are resources the participant has.

Example 3. Consider the scenario described in Examples 1 and 2. Bob's configuration is $G_B = \langle K_B, E_B, \text{Ack}_B, \text{AC}_B \rangle$, in which K_B is Bob's public key, E_B contains one credential that proves K_B has IRS. low-Income, and $\text{Ack}_B(\text{IRS.lowIncome}) = \text{IRS.nonprofit}$. Alice's configuration is $G_A = \langle K_A, E_A, \text{Ack}_A, \text{AC}_A \rangle$, in which $E_A = \emptyset$ and $\text{Ack}_A(\text{IRS.lowIncome}) = \text{IRS.nonprofit}$.

While we assume that each principal (public key) is controlled by, at most, one entity, it is possible that one entity controls several principals. The reader may wonder why in that case our notion of configuration includes just one principal. Why not let a negotiator use several principals in a negotiation? The reason we do not is that it makes it difficult to ensure that the principals all correspond to a single entity, so it opens the door to colluding entities obtaining resources they should not have. This is prevented in our model.

Before a trust negotiation process starts, the two negotiators establish a secure connection and authenticate the principals they each control. In addition to ensuring that sensitive information is not disclosed to any third parties who may be monitoring the communications, this also ensures that one can be confident that credentials revealed by the other participant indeed belong to the participant. One way to achieve this is for the two parties to establish a TLS/SSL connection using self-signed certificates.

A negotiation process starts when one participant (called the *requester*) sends a request to another participant (called the *access mediator*) requesting access to some resource. The access mediator identifies the policy protecting that resource and then starts the negotiation process. The negotiation process is modeled formally as a pair of sequences of message, each sequence being defined by one negotiator. The negotiation proceeds by two negotiators taking turns extending these sequences, thus modeling message exchange. Each negotiator maintains a local state during the negotiation process. Internal structure of the messages and local states are opaque in the abstract framework described in the current section. However we assume there are two distinguished states: *success*, and *failure*. A negotiation process fails when one of the two negotiators enters into the *failure* state. (In practice, a negotiator might send a message notifying the opponent about the failure; for technical convenience, we choose not to include such a message in the model here.) A negotiation process succeeds when the access mediator enters into the *success* state. A negotiation process stops when it succeeds or when it fails.

A negotiation strategy determines the structure of states and what actions a negotiator takes in a negotiation process. More specifically, a *negotiation strategy* is a 5-tuple $\text{strat} = \langle Q, \mathcal{M}, \text{rstart}, \text{start}, \text{reply} \rangle$ whose elements satisfy the following:

- Q is a countable set of states. We use q (possibly with superscripts and subscripts) to denote a state.
- \mathcal{M} is a countable set of *messages*. We use m and a for messages, possibly with subscripts.
- The function $\text{rstart} : \text{Configuration} \times \mathcal{K} \rightarrow Q$ defines the initial state of the requester, given his configuration G and the requester principal K_O . This state is $\text{rstart}(G, K_O) = q$, in which $q \notin \{\text{success}, \text{failure}\}$.
- The function $\text{start} : \text{Configuration} \times \text{Resource} \times \mathcal{K} \rightarrow Q \times \mathcal{M}$ defines how an access mediator starts a negotiation, given his configuration G , the resource requested ρ , and the requester principal K_O . It yields $\text{start}(G, \rho, K_O) = \langle q, m \rangle$. The access mediator uses q as its initial local state and, when $q \notin \{\text{success}, \text{failure}\}$, sends the message m to the requester to start the negotiation.
- The function $\text{reply} : Q \times \mathcal{M} \rightarrow Q \times \mathcal{M}$ defines each action taken by a negotiator, given the negotiator's configuration G , its current state q , and the last message m from the opponent. It yields $\text{reply}(q, m) = \langle q', m' \rangle$. The negotiator changes state to q' and (when $q' \notin \{\text{success}, \text{failure}\}$) sends m' to the other negotiator.

3.2 Safety of Ack-Policy Enforcement

We now define what it means when we say a negotiation strategy is safe. Intuitively, a strategy is safe if the ack policies are correctly enforced when using the strategy. What does it mean to say that a negotiator N 's ack policies are correctly enforced? The definition we will present uses the following intuition: no adversary M , using observations it can make in negotiation processes with N , can make any inference about credentials proving the attributes of N it is not entitled to know (i.e., attributes whose ack policies are not satisfied by M).

To make the above intuition precise, we first model the ability of adversaries. An *adversary* is given by a set of principals it controls and a set of credentials for each of the principals. This models the ability of entities controlling different principals to collude. (We want a notion of safety that precludes colluding principals from inferring information that none of them is authorized for by pooling their observations about how the negotiator behaves with each of them.) We assume each such set contains all credentials potentially available to the principal for use in trust negotiation. (If an adversary controls a principal that is an attribute authority of an attribute t , then credentials about t are available to the adversary.) We assume that an adversary only interacts with a participant N through trust negotiation. We allow the adversary M to initiate negotiation with N , by sending N a request, as well as to wait for N to initiate a negotiation process by sending a request to M . An adversary is limited by the credentials available to it, which determine the attributes possessed by the principals it controls. We assume that it is infeasible to forge signatures without knowing the private keys.

The next definition introduces several concepts in a top-down fashion. It begins by giving the main definition of indistinguishability of configurations and then gives definitions for terms used in the main definition. The definition

formalizes the observations an adversary M can make by engaging in negotiations with a negotiator that uses a given strategy. We capture this in terms of M 's ability to determine that the negotiator's actual configuration G is not some other configuration G' . When M cannot do this, G and G' are said to be indistinguishable. The definition then formalizes the actions that an attacker can take and the response it induces in a negotiator that has a certain configuration and uses a certain strategy. The goal of the definition is to enable us to articulate the intuition that if an adversary cannot distinguish between two configurations, one of which has an attribute and the other of which does not, then the adversary cannot infer whether the negotiator has the attribute or not. This will be made precise below.

Definition 3.1. Indistinguishability, Attack Sequence, Induced Reaction Sequence. Given an adversary M , a negotiation strategy strat , and two configurations G and G' , G and G' are *indistinguishable under strat by M* if for every attack sequence seq that is feasible for M , the reaction sequence induced by seq from G is the same as the reaction sequence induced by seq from G' .

In the following, we define *feasible attack sequences* and the *reaction sequences they induce*. There are two forms of attack sequence, requester and responder. A *requester attack sequence* has the form $[K_A, \rho, a_1, a_2, \dots, a_k]$, in which K_A is a principal, ρ is a resource, and a_1, a_2, \dots, a_k are messages. This corresponds to the case in which the adversary uses K_A , a principal it controls, to request access to resource ρ , and then sends a_1, a_2, \dots, a_k one by one in the negotiation. Given a configuration G and a strategy $\text{strat} = \langle Q, q_0, \text{start}, \text{reply} \rangle$, the *reaction sequence induced by a requester attack sequence* $[K_A, \rho, a_1, a_2, \dots, a_k]$ is the sequence of messages: $[m_1, m_2, \dots, m_\ell]$ such that there exists a sequence of states $[q_1, q_2, \dots, q_\ell]$ that satisfies the following conditions:

1. $\langle q_1, m_1 \rangle = \text{start}(G, \rho, K_A)$
2. For all $i \in [2, \ell]$, $\langle q_i, m_i \rangle = \text{reply}(q_{i-1}, a_{i-1})$
3. For all $i \in [1, \ell - 1]$, $q_i \notin \{\text{success}, \text{failure}\}$
4. Either $\ell = k + 1$ or both $1 \leq \ell \leq k$ and $q_\ell \in \{\text{success}, \text{failure}\}$ (in the latter case, the negotiation ends before the complete attack sequence is used)

A *responder attack sequence* has the form $[K_A, a_1, a_2, \dots, a_k]$, in which K_A is a principal and a_1, a_2, \dots, a_k are messages. This corresponds to the case that the negotiator sends a resource request to the adversary, who responds by sending the messages of the attack sequence. Given a configuration G and a strategy $\text{strat} = \langle Q, \text{rstart}, \text{start}, \text{reply} \rangle$, a *reaction sequence induced by a responder attack sequence* $[K_A, a_1, a_2, \dots, a_k]$ is the sequence of messages $[m_1, m_2, \dots, m_\ell]$ such that there exists a sequence of states $[q_0, q_1, \dots, q_\ell]$ that satisfies the following conditions:

1. $q_0 = \text{rstart}(G, K_A)$
2. For all $i \in [1, \ell]$, $\langle q_i, m_i \rangle = \text{reply}(q_{i-1}, a_i)$

3. For all $i \in [1, \ell - 1]$, $q_i \notin \{\text{success}, \text{failure}\}$
4. Either $\ell = k + 1$ or both $1 \leq \ell \leq k$ and $q_\ell \in \{\text{success}, \text{failure}\}$

Observe that an attack sequence may induce different reaction sequences when interacting with different strategies. In particular, these reaction sequences may be of different lengths and some may be shorter than the attack sequence. To simplify presentation, we choose to define an attack sequence as an object that exists independently of the possible reaction sequences it induces.

Given an adversary M , an attack sequence seq is *feasible* for M if K_A is controlled by M and the only credentials included in seq are those in credentials available to M . Feasibility formalizes the notion that M cannot forge signatures as part of computing seq .

The notion of indistinguishability given in Definition 3.1 is suitable only for deterministic negotiation strategies. Furthermore, the way a feasible attack sequence is defined limits strategies that can be considered to those that verify possession of a credential by seeing the digital signature in the credential and verifying that the signature is valid. These limitations do not affect the development of this paper, as the strategies that we analyze in this paper all satisfy the above requirements.

A more general way of defining indistinguishability is to follow the definition of indistinguishability in the cryptographic literature, see, e.g. Goldreich [2001]. In this approach, each negotiation strategy is modeled as a Probabilistic Polynomial-time Interactive Turing Machine (PPITM), which takes a configuration as its private input. A negotiation process is modeled as a joint computation between two PPITMs. Given an adversary M , a distinguisher A based on M is a PPITM that takes M as private input, interacts with a negotiation strategy, and outputs either 0 or 1. We use the notation $A(M)[S(G)]$ to denote the output of A when given M as private input and interacting with the strategy S , which is given G as its private input. We say that two configurations G and G' are indistinguishable under a strategy S if for any distinguisher A based on M :

$$| \Pr[A(M)[S(G)] = 1] - \Pr[A(M)[S(G')] = 1] |$$

is negligible in the security parameter, where the security parameter can be taken as the minimal length of public keys used by the attribute authorities, and the probability is taken over the coin choices of A and S .

Observe that indistinguishability under Definition 3.1 implies the above cryptographic notion of indistinguishability, assuming that the signature schemes used in the credential are secure. If G and G' are indistinguishable under strat in the sense of Definition 3.1, then the only way to observe any difference at all between interacting with G and G' is by forging a credential the adversary does not have, which can be successfully carried out only with negligible probability.

Definition 3.2. Unacknowledgeable Attribute Set. Given a configuration G and an adversary M , we say that an attribute t is *unacknowledgeable* to M if no principal controlled by M possesses attributes that satisfy $\text{Ack}_G(t)$. We define $\text{UnAcks}(G, M)$ to be the set of attributes that are unacknowledgeable to M .

Intuitively, ATN should not enable an adversary M to learn any information about $\text{UnAcks}(G, M)$ that M would not otherwise be able to learn. Given such a set of unacknowledgeable attributes, the negotiator's credentials can be divided into those that can be released to M and those that cannot.

Definition 3.3. Releasable and Unreleasable Credentials. Given a set of credentials E and a set of unacknowledgeable attributes U , the set of *unreleasable credentials* consists of those that define unacknowledgeable attributes, and is given by $\text{unreleaseable}(E, U) = \{e \in E \mid T(e) \cap U \neq \emptyset\}$. The remaining elements of E are *releasable credentials*: $\text{releaseable}(E, U) = E - \text{unreleaseable}(E, U) = \{e \in E \mid T(e) \cap U = \emptyset\}$.

Equipped with this terminology, we can now state that if U is the set of attributes that must not be acknowledged to M and if two negotiators using the same strategy have the same set of releasable credentials with respect to U , then they should behave the same from the point of view of M . To put it another way, a strategy is *credential-combination-hiding* if configurations that differ only in unreleasable credentials are indistinguishable. We now formalize this intuition in the central definition of the paper, which requires that an ATN strategy hide all information about credentials representing unacknowledgeable attributes.

Definition 3.4. Credential-Combination Hiding. A negotiation strategy strat is *credential-combination-hiding safe* if for every pair of configurations $G = \langle K, E, \text{Ack}, \text{AC} \rangle$ and $G' = \langle K, E', \text{Ack}, \text{AC} \rangle$, and every adversary M , if $\text{releaseable}(E, \text{UnAcks}(M, G)) = \text{releaseable}(E', \text{UnAcks}(M, G))$, then G and G' are indistinguishable under strat by M .

One aspect of Definition 3.4 that differs from prior notions of safety is that it is concerned only with the attributes M has, and not with the ones M proves in the negotiation. This simplifies matters and is entirely justified because our objective is to ensure that information flow is authorized, not that it is matched by a compensatory flow in the reverse direction. For instance, if a negotiator has cached (valid) opponent credentials from a previous negotiation, there is no need to require those credentials to be retransmitted.

Nevertheless, since a safe strategy must function correctly with any adversary M , M must in general prove its attributes before N reveals attributes that depend on them. If M 's attributes and N 's attributes are mutually interdependent, then it may be there is no safe strategy that would allow these attributes to be shown, even though N and M both possess the attributes required to permit them each to reveal the attributes in question. The strategies we discuss in this paper are unable to support using attributes that participate in dependence cycles with attributes of the adversary. In many cases such cyclic dependences can be handled by using cryptographic protocols, such as those harnessed for use in ATN in Li et al. [2005]. Such protocols can, for instance, transmit messages that provide positive proof to an adversary that the negotiator has a certain attribute just in case the adversary has the attributes required to be eligible for that information. However, these are nondeterministic protocols. Thus, to extend our notion of safety to support such protocols, we

would need to generalize our notion of indistinguishability, as outlined above in our discussion of that topic. It remains open whether cryptographic protocols can “break all cycles,” that is, can provide a safe strategy that always enables negotiators to reveal attributes that the adversary is authorized to know.

Definition 3.4 does not protect against every possible form of information leakage during ATN, especially when an attacker has certain knowledge about the distribution of credentials. For example, the attacker may know that entities having a credential e are likely to have another credential that proves the possession of a sensitive attribute that is not in $T(e)$. Similarly, an attacker may know that entities having a credential e are less likely to possess certain sensitive attributes. We speculate that our model could be extended to address this threats. For instance, one approach to remedy the threat identified above is to replace $T(e)$ in Definition 3.3 by a larger set, $\overline{T}(e)$. Whereas $T(e)$ is the set of attributes proven by credential e , $\overline{T}(e)$ is the set of attributes that e suggests the principal either has or does not have. This will enlarge the set of unreleasable credentials and strengthen the resulting safety property.

Definition 3.4 also does not protect against inferences that take advantage of possible relationships between two principals’ attributes. For example, an attacker may know that if one principal K_1 has a sensitive attribute, then another principal K_2 is also likely to have the attribute. The attacker may then learn K_2 ’s attribute by negotiating with K_1 . While such inferences can be stopped if K_1 and K_2 use the same policies for protecting their attributes, Definition 3.4 does not by itself prevent such inferences being made.

3.3 Safety of The Eager Strategy

Next, we discuss the eager strategy and observe that it satisfies Definition 3.4. A negotiator using the eager strategy sends all credentials as soon as the attributes they define have their ack policies satisfied by credentials received from the opponent. The two negotiators take turns exchanging all credentials that are unlocked, i.e., that define attributes whose ack policies have been satisfied by credentials disclosed previously by the opponent. In the first transmission, the access mediator sends all credentials defining unprotected attributes. The requester then sends all credentials defining unprotected attributes or attributes whose ack policies were satisfied in the first transmission. The negotiators continue exchanging credentials in this way until either the policy governing the desired resource has been satisfied by credentials sent by the requester, in which case the negotiation succeeds, or until a credential exchange occurs in which no new credentials become unlocked, in which case the negotiation fails.

Definition 3.5 (Eager Strategy). The *eager strategy* is presented in Figure 1. It uses a state of the form $\langle G, \text{opCreds}, \text{locCreds}, K_O, \rho \rangle$, in which G is the negotiator’s configuration, opCreds and locCreds are the sets of credentials disclosed thus far by the opponent and the negotiator, respectively, K_O is the opponent’s public key, and ρ is a resource if the negotiator is an access mediator and *null* otherwise.

```

 $\mathcal{M} = \mathbf{FE}$ 
 $Q = \text{Configuration} \times \mathbf{FE} \times \mathbf{FE} \times \mathcal{K} \times \text{Resource}$ 
rstart( $G, K_O$ ) =
  startState =  $\langle G, \emptyset, \emptyset, K_O, \text{null} \rangle$ 
  return startState

start( $G, \rho, K_O$ ) =
  if  $\text{AC}_G(\rho)$  is trivially satisfied, return  $\langle \text{success}, \text{null} \rangle$ 
  publicCreds =  $\{e \in E \mid \text{each policy in } \text{Ack}_G(T(e)) \text{ is trivially satisfied}\}$ 
  startState =  $\langle G, \emptyset, \text{publicCreds}, K_O, \rho \rangle$ 
  return  $\langle \text{startState}, \text{publicCreds} \rangle$ 

reply( $\langle G, \text{opCreds}, \text{locCreds}, K_O, \rho \rangle, m$ ) =
  opCreds+1 = opCreds  $\cup$   $m$ 
  if local negotiator is resource provider (i.e.,  $\rho \neq \text{null}$ )
    and opCreds+1 proves  $K_O$  satisfies  $\text{AC}_G(\rho)$ 
    return  $\langle \text{success}, \text{null} \rangle$ 
  locCreds+1 =  $\{e \in E \mid \text{opCreds proves } K_O \text{ satisfies each policy in } \text{Ack}_G(T(e))\}$ 
   $m_{+1} = \text{locCreds}_{+1} - \text{locCreds}$ 
  if  $m_{+1} = \emptyset$  return  $\langle \text{failure}, \text{null} \rangle$ 
  return  $\langle \langle G, \text{opCreds}_{+1}, \text{locCreds}_{+1}, K_O, \rho \rangle, m_{+1} \rangle$ 

```

Fig. 1. The eager strategy. \mathbf{FE} denotes the set of finite subsets of \mathcal{E} .

THEOREM 3.6. *The eager strategy is credential-combination-hiding safe.*

The proof is found in Appendix A.

Example 4. If Alice and Bob have the configurations shown in Example 3, and each one negotiates with SwampLand.com, which has no credentials, both negotiations start with SwampLand sending an empty message and then immediately fail, with no further messages flowing. For the sake of illustration, if we assume that $K_A = K_B$, $\text{Ack}_A = \text{Ack}_B$ and $\text{AC}_A = \text{AC}_B$, then SwampLand.com obtains no basis on which to distinguish Alice from Bob.¹

It should be noted that the eager strategy does not take advantage of the distinguishing characteristic of ack policies, *viz.*, that they can be defined for attributes the negotiator does not possess and, therefore, can be revealed without disclosing whether the negotiator has the attribute. In Section 4, we present and prove safe a strategy that takes advantage of the fact that ack policies can be safely disclosed, enabling the strategy to use them to focus the exchange on relevant credentials.

3.4 Weaker Notions of Safety

In this section we discuss two weaker notions of safety that seem natural to consider, one of which in particular seemed quite appealing to us at first. However, as we explain at the end of this section, it turns out that both are inadequate. These two alternative notions of safety are strictly weaker than

¹In practice, Alice and Bob would not have the same key; the point is that SwampLand.com cannot distinguish someone who has the low-income attribute from someone who does not.

credential-combination hiding; in this section we prove their logical relationship to credential-combination hiding and to one another.

A strategy that violates Definition 3.4 may not actually enable an adversary to make any inferences about the negotiator's unacknowledgeable attributes. A violation means that there exist configurations G and $G' = \langle K_G, E', \text{Ack}_G, \text{AC}_G \rangle$ and an adversary M such that the releasable credentials of G and G' are the same, but G and G' can be distinguished by M . This means that M can infer that certain combinations of unreleasable credentials are not candidates for being the exact set held by G ; however, it does not ensure M can rule out any combination of unacknowledgeable attributes.

For example, suppose that low-income status can also be proved by multiple credential issued by the IRS. A strategy that violates Definition 3.4 may enable an adversary to rule out a negotiator's having one of these credentials, without enabling the adversary to infer that a negotiator does not have the low-income attribute.

Thus, it seems natural to consider a weaker notion of safety in which we ensure only that M cannot rule out any combination of attributes. The goal of the following weaker safety notion, which we call attribute-combination hiding, is to preclude negotiation, enabling the adversary to make any inferences that certain attribute combinations are impossible. However, when there are interdependencies among attributes, anyone familiar with the credential scheme can rule out certain attribute combinations. For instance, if every credential proving one attribute t_1 also proves another attribute t_2 , it is impossible to have t_1 but not t_2 . Therefore, the definition only precludes the adversary inferring things he does not already know.

Definition 3.7 (Attribute-Combination Hiding). A negotiation strategy strat is *attribute-combination-hiding safe* if for every configuration $G = \langle K, E, \text{Ack}, \text{AC} \rangle$, for every subset U of \mathcal{T} , and for every expressible subset U' of U , there exists a configuration $G' = \langle K, E', \text{Ack}, \text{AC} \rangle$, such that (a) $T(E') \cap U = U'$ and (b) for every adversary M such that $\text{UnAcks}(G, M) \supseteq U$, G and G' are indistinguishable under strat by M .

Given a set U of attributes, U' is an *expressible subset* of U if there exists a set of credentials E_0 such that $T(E_0) \cap U = U'$. By “exists” here, we mean hypothetically; the credentials in E_0 need never actually have been issued.

Definition 3.7 says that if N uses strategy strat , then from M 's point of view, N could have any expressible combination of attributes in U . If the definition is violated, then there is a configuration G , a set of attributes U , and a $U' \subseteq U$, such that there exists a credential set E' that agrees with U' on U (i.e., $T(E') \cap U = U'$), and every such E' is distinguishable from E_G by some adversary M with $\text{UnAcks}(G, M) \subseteq U$. In other words, M can determine that $T(E') \cap U \neq U'$, thereby ruling out U' as a candidate for the combination of unacknowledgeable attributes held by N .

Notice the importance in Definition 3.7 of fixing U as a lower bound on $\text{UnAcks}(G, M)$ before picking a G' that works for all M satisfying the bound. It would not be adequate to find a different G' for each M , because then colluding attackers could pool their knowledge to rule out the various G' 's.

One might object that attribute–combination hiding is still too strong. In our running example, Bob could use a strategy that violates this requirement without enabling Swampland to determine whether Bob has the low-income attribute. Violation means only that Swampland can rule out Bob having a certain combination of attributes. For instance, Swampland might be able to infer that Bob is not both low-income and over 65 years old without being able to determine which attribute Bob does not have.

The following still weaker notion, which we call attribute hiding, only prevents the adversary learning whether specific attributes are satisfied. It says that an adversary cannot determine through ATN whether or not the negotiator has any given unacknowledgeable attribute.

Definition 3.8 (Attribute Hiding). A negotiation strategy strat is *attribute-hiding safe* if, for every configuration $G = \langle K, E, \text{Ack}, \text{AC} \rangle$ and every attribute t , there exists a $G' = \langle K, E', \text{Ack}, \text{AC} \rangle$ that differs from G in t (i.e., $t \in T(E) - T(E')$ or $t \in T(E') - T(E)$) and, for every adversary M , if t in $\text{UnAcks}(G, M)$, G' is indistinguishable from G by M .

A violation of attribute hiding means that some M can use ATN to determine whether or not N satisfies a particular unacknowledgeable attribute, which is clearly something that any reasonable safety definition must preclude. The following theorem verifies that both credential-combination hiding and attribute–combination hiding do so.

THEOREM 3.9. *The relative strength of the safety definitions is as follows:*

- (1) *If strat is credential-combination-hiding safe, then it is attribute–combination-hiding safe.*
- (2) *If strat is attribute-combination-hiding safe, then it is attribute-hiding safe.*

The proof is found in Appendix B.

Attribute hiding by itself is not sufficient as a safety requirement, because it does not preclude the adversary M inferring that N does not have a certain combination of attributes. For example, a strategy could be attribute-hiding safe while enabling the adversary to infer N has either a CIA credential or an NSA credential, so long as M cannot determine which of these is the case. Since even this imprecise information clearly may be damaging, this makes attribute hiding an unacceptable standard for ATN safety. This problem is prevented by attribute–combination hiding, illustrating that it is strictly stronger than attribute hiding.

The problem with attribute–combination hiding is revealed when we consider the probabilistic inference of attributes. Assume that the opponent has some prior knowledge about the probability that each *credential combination* occurs; the opponent can easily infer information about the probability that each *attribute combination* occurs. Given a set U of unacknowledgeable attributes, safety should mean that after any number of negotiations, the opponent has no basis on which to improve his estimate of the probability that the negotiator has any given attribute combination in U . To make this more concrete, suppose that

several configurations each induce a given set of unacknowledgeable attributes U' and that all but one of them are distinguishable from the negotiator's actual configuration G . This does not violate the requirement of attribute-combination hiding. However, it does mean that the opponent can rule out many configurations. Thus, for instance, if the one indistinguishable configuration is very rare, the adversary can learn that N 's unacknowledgeable attributes are very unlikely to be exactly U' . In the CIA and NSA credential example above, learning that the negotiator *probably* does not have a certain combination (e.g., none) of the credentials can be detrimental, even if that knowledge is not entirely certain. Credential-combination hiding does not have this problem because all configurations with the same releasable credentials are indistinguishable, so it does not permit the opponent to rule out any of the configurations that induce U' .

4. APPLYING THE MODEL WITH DELEGATION

We now discuss the application of our model to credential systems that support forms of delegation common in trust-management languages. Delegation credentials enable decentralization of authority over attributes and support administrative scalability. They are essential to the traditional trust-management approach to authorization [Blaze et al. 1996], where they allow a single attribute, such as an access right, to be delegated from one principal to another. However, they can be more general Li et al. [2002, 2003], specifying that having attribute t_1 implies having attribute t_2 . Here the authority on t_2 is delegating to the authority on t_1 some control over who satisfies t_2 .

In this section, we present the TTGstrat family of ATN strategies. Unlike with the eager strategy, negotiators exchange information about their ack policies so as to focus their credential disclosures on credentials that are relevant to the negotiation. In TTGstrat, credentials, ack policies, and AC policies are all expressed using the language RT_0 Li et al. [2002, 2003], which supports delegation. TTGstrat is based on the trust-target-graph (TTG) approach of Winsborough and Li [2002b], which it generalizes in that the search for successful branches in the negotiation structure is more flexible.

The main result of this section is that TTGstrat strategies provide credential-combination hiding. This supports our contention that Definition 3.4 is a useful definition of safety for ATN. It also shows that the use of ack policies enables negotiators to safely focus one another's credential disclosures on relevant credentials, even when using a credential system that supports delegation.

In terms of the framework given in Section 3.1, when the credential system supports delegation, we capture this by presuming that the credentials directly represented in the model are those that assign attributes directly to principals specified in the credential. These are the only credentials that appear in the configuration of a negotiator. In the environment, there is also a set L of delegation credentials that do not belong to a specific negotiator, since they can be used in many proofs that show various principals have a given attribute. In general, a delegation credential $\ell \in L$ asserts that one attribute implies another attribute. Thus, to handle the inferencing problems raised in Section 2,

when there are delegation credentials, $\text{init}(G)$ will return a \overline{G} in which $\text{Ack}_{\overline{G}}$ protects attributes more strongly than does Ack_G .

4.1 Confidentiality Assumptions

We make the simplifying assumption that all delegation credentials are available to the negotiator. If we assume only that delegation credentials are available to principals that satisfy the attributes defined in the credentials, negotiators cannot safely protect attributes they do not have. When having t_1 implies having t_2 , it is not possible to hide not having t_1 unless one also hides not having t_2 , so the negotiator must be aware of the implication. Thus, it appears to be inherent that a negotiator cannot effectively negotiate while protecting all information about an attribute without knowing whether it is at least possibly related to other attributes he may be asked about in the course of the negotiation.

The assumption that delegation credentials are available is typically justified when attributes are characteristics of subjects or roles that they occupy within their organizations. For instance, it is unlikely to be private information that a university delegates to its registrar authority for identifying students. However, when attributes are capabilities to access specific resources, there may be times when delegation of those capabilities are sensitive. If the negotiator does not have access to all delegation credentials, but has an upper bound for the set, he can still negotiate safely. However, if this is done, negotiation may fail in some cases where it would succeed if the negotiator had perfect knowledge of the delegation credentials. For instance, although a negotiator may not know it, it may be that an attribute representing a given permission can depend on other attributes representing the same permission, but cannot depend on attributes representing something else. Without having this information, safety would require the negotiator to protect all attributes as strongly as it does the permission. Thus, it seems that our assumption can be relaxed only at the cost of having some negotiations fail that would otherwise succeed.

4.2 The Language for Credentials and Policies

We first describe the language for credentials and policies. The language is a subset of RT_0 Li et al. [2002, 2003]. Credentials, ack policies, and AC policies are all expressed using statements in this language.

Constructs of RT_0 include principals, attribute names, and attributes. An *attribute name* is a string over a standard alphabet. An *attribute* takes the form of a principal followed by an attribute name, separated by a dot, e.g., $K.r$ and $K_1.r_1$. (In Li et al. [2002, 2003], attributes are called roles.)

There are three types of statements in our subset of RT_0 , each corresponding to a different way of defining attributes. For consistency, we maintain the naming convention for these three statement types used in Li et al. [2002, 2003] where further discussion of the intuition behind these statement forms can be found:

- *Type-1:* $K.r \leftarrow K_0$

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

Fig. 2. A credential chain showing that Alice is authorized for EPub’s student discount.

- *Type-2*: $K.r \leftarrow K_1.r_1$
- *Type-4*: $K.r \leftarrow K_1.r_1 \cap K_2.r_2 \cap \dots \cap K_n.r_n$

A *credential* is a digitally signed type-1 statement. A *delegation credential* is a digitally signed type-2 statement. Type-4 statements can be used only for policies.

Example 5. 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 stating that Alice is a student. These are represented by four RT_0 credentials:

1. $\text{EPub.discount} \leftarrow \text{EOrg.preferred}$
2. $\text{EOrg.preferred} \leftarrow \text{StateU.student}$
3. $\text{StateU.student} \leftarrow \text{RegistrarB.student}$
4. $\text{RegistrarB.student} \leftarrow \text{Alice}$

The credential “ $\text{EPub.discount} \leftarrow \text{EOrg.preferred}$ ” is read: if EOrg assigns a principal the attribute “preferred,” then EPub assigns that principal the attribute “discount.” The four credentials above form a chain, shown in Figure 2, proving that Alice is entitled to a discount.

In our framework, a policy is a positive propositional formula in which the propositions are attributes in \mathcal{T} . In this section, such a policy is represented by a dummy attribute drawn from a set \mathcal{T}_D , a new set of attributes that is disjoint from \mathcal{T} . Dummy attributes are defined by one or more type-2 and/or type-4 statements in which the attributes to the right of the arrow are drawn from $\mathcal{T} \cup \mathcal{T}_D$. Dummy attributes are defined by statements that are assumed to be locally available to the policy enforcer. Nondummy attributes are defined by nondelegation and delegation credentials that must be provided to the policy enforcer. Each nondelegation credential $K.r \leftarrow K_0$ is stored (and protected) by its subject, K_0 .

The semantics of the above language can be defined in several equivalent ways, e.g., using sets, graphs, or logic programming rules Li et al. [2002, 2003]. Here, we present a logical semantics. In this semantics, we use one binary predicate `hasAttr`. Each statement is translated into a first-order logic sentence.

- From $K.r \leftarrow K_0$ to `hasAttr(K_0 , $K.r$)`
- From $K.r \leftarrow K_1.r_1$ to $\forall z(\text{hasAttr}(z, K.r) \Leftarrow \text{hasAttr}(z, K_1.r_1))$
- From $K.r \leftarrow K_1.r_1 \cap \dots \cap K_n.r_n$ to $\forall z(\text{hasAttr}(z, K.r) \Leftarrow \text{hasAttr}(z, K_1.r_1) \wedge \dots \wedge \text{hasAttr}(z, K_n.r_n))$

Given a set E of nondelegation (type-1) credentials that have the same subject K_0 , a set L of delegation (type-2) credentials defining nondummy attributes,

and a set P of statements defining dummy attributes (type-2 and type-4), $T(E)$ is the set of attributes $K.r$ such that the first-order theory translated from $E \cup L \cup P$ implies $\text{hasAttr}(K_0, K.r)$.

4.3 A Simplified Trust Target Graph Protocol

In this section, we introduce a simplified version of the trust-target graph protocol introduced in Winsborough and Li [2002b]. We are able to simplify it, because we are using only a subset of RT_0 . The protocol accommodates a diverse family of strategies based on TTGs. While different strategies that use the protocol may construct a wide variety of TTGs, all strategies permitted by the protocol construct TTGs that soundly demonstrate negotiator attributes.

In this protocol, a trust negotiation process involves the two negotiators working together to construct a *trust-target graph* (TTG). A TTG is a directed graph, each node of which is a trust target. Trust targets, whose syntax is given below, are queries issued by one negotiator about the other negotiator's attributes. When a requester requests access to a resource, the access mediator and the requester enter into a negotiation process. The access mediator creates a TTG containing one target, which we call the *primary target*. The access mediator then tries to process the primary target and sends the partially processed TTG to the requester. In each following round, one negotiator receives from the other new information about changes to the TTG, verifies that the changes are legal, and updates its local copy of the TTG accordingly. The negotiator then tries to process some nodes, making its own changes to the graph, which it then sends to the other party, completing the round. The negotiation succeeds when the primary target is satisfied; it fails when the primary target is failed or when a round occurs in which neither negotiator changes the graph. In the next section, we show how the TTG protocol supports the enforcement of ack policies to protect sensitive attribute information.

4.3.1 Nodes in a Trust-Target Graph. A node in a TTG is one of the three kinds of trust targets, defined as follows. Nodes are unique.

- A *attribute target* takes the form $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$, in which K_V is (a principal controlled by) one of the negotiators, $K.r$ is an attribute, and K_S is a principal. K_S is the subject of the target, which is the negotiator opposing K_V . This target means that K_V wants to see a proof of $\text{hasAttr}(K_S, K.r)$.
- An *intersection target* takes the form $\langle K_V : (K_1.r_1 \cap \dots \cap K_n.r_n) \stackrel{?}{\leftarrow} K_S \rangle$. This means that K_V wants to see the proof that $\text{hasAttr}(K_S, K_1.r_1) \wedge \dots \wedge \text{hasAttr}(K_S, K_n.r_n)$.
- A *trivial target* takes the form $\langle K_V : K_S \stackrel{?}{\leftarrow} K_S \rangle$, in which K_V is one of the negotiators, and K_S is a principal. Trivial targets provide placeholders for edges in the TTG.

In each of the above forms of targets, we call K_V the *verifier*, and K_S the *subject* of the target. Each target has a *satisfaction state*, which has one of three values: *satisfied*, *failed*, or *unknown*. Each target also has a *processing state*,

which is a pair of Boolean states: verifier- and opponent-processed. Depending on target type, these are initially either true or false. The processing state of a target can be changed by the verifier- by setting verifier-processed to true; similarly for the opponent of the verifier. When the verifier is faithfully executing the protocol, a node is *verifier-processed* if the verifier cannot process the node any further, meaning that the verifier cannot add any new child to the node via a justified edge, for instance, because he has no credentials that would justify the edge. When the opponent of the verifier is faithfully executing the protocol, a node is *opponent-processed* if the opponent cannot process the node any further. When a node is both verifier- and opponent-processed, we say that it is *fully processed*.

4.3.2 Edges in a Trust-Target Graph. Three kinds of edges are allowed in a trust-target graph, listed below. Each kind of edge has its own requirements for being justified. We use \leftarrow to represent edges in TTG's. Note that the edges of a TTG form a set, not a multiset.

- An *implication edge* takes the form $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle \leftarrow \langle K_V : \chi \stackrel{?}{\leftarrow} K_S \rangle$, in which χ is a principal, an attribute, or an intersection. An implication edge has to end at an attribute target, but can start from any target. We call $\langle K_V : \chi \stackrel{?}{\leftarrow} K_S \rangle$ an implication child of $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$. (We use similar “child” terminology for other kinds of edges.) An edge always points from the child to the parent. An implication edge is *justified* if the edge is accompanied by a credential of the form $K.r \leftarrow \chi$.
- An *intersection edge* takes the form $\langle K_V : (K_1.r_1 \cap \dots \cap K_n.r_n) \stackrel{?}{\leftarrow} K_S \rangle \leftarrow \langle K_V : K_i.r_i \stackrel{?}{\leftarrow} K_S \rangle$ where i is in $1..n$. An intersection edge is always justified.
- A *control edge* takes the form $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle \leftarrow \langle K_S : K'.r' \stackrel{?}{\leftarrow} K_V \rangle$. Control edges are used for handling acknowledgment policies. This edge would be added by the negotiator controlling K_S ; intuitively it means that, before this negotiator will show anything indicating whether K_S possesses $K.r$, it wants to see a proof that K_V possesses $K'.r'$ first. A control edge is always justified.

4.3.3 Messages in the Protocol. As described above, negotiators cooperate through use of the protocol in constructing a shared TTG, *ttg*, a copy of which is maintained by each negotiator. Negotiators alternate transmitting messages, each of which contains a sequence of TTG update operations and a set of credentials to be used in justifying implication edges. On receiving a message m , a negotiator verifies that the update operations it contains are legal before applying the operations to its local copy of the shared TTG. This is done by using the Boolean-valued function $\text{legalUpdate}(m, \text{ttg})$. The following are *legal* TTG update operations:

- Initialize the TTG to contain a given primary TT, specifying a legal initial processing state for this node. (See below.)
- Add a justified edge (not already in the graph) from a TT that is not yet in the graph to one that is, specifying a legal initial processing state for the new node. The new TT is added to the graph as well as the edge.

- Add a justified edge (not already in the graph) from an old node to an old node.
- Mark a node processed. If the sender is the verifier, this marks the node verifier-processed; otherwise, it marks it opponent-processed.

These operations construct a connected graph. The legal initial processing state of a trivial target is fully-processed. An intersection target is initially opponent-processed and an attribute target is initially either opponent- or verifier-processed.

Satisfaction states of trust targets are not transmitted in messages; instead, each negotiation party infers them independently. The satisfaction-state rules presented in the next section ensure that negotiators using the protocol always reach the same conclusions regarding node satisfaction.

4.3.4 Trust Target Satisfaction State Propagation. We now describe how to determine the satisfaction state of targets:

- The initial satisfaction state of an attribute target is unknown. It becomes satisfied when one of its implication children is satisfied. It becomes failed when it is fully processed and either it has no implication child or all of its implication children are failed.
- The initial satisfaction state of an intersection target is unknown. It becomes satisfied when it is fully processed and all of its children are satisfied. It becomes failed when one of its children is failed.
- A trivial target is always satisfied.

The legal update operations do not remove nodes or edges once they have been added, and once a node is fully processed, it remains so thereafter. Consequently, once a target becomes satisfied or failed, it retains that state for the duration of the negotiation.

PROPOSITION 4.1. *If a principal updates a TTG legally and propagates the satisfaction state correctly, then in the TTG, when a target $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$ is satisfied, the credentials associated with the TTG prove that $\text{hasAttr}(K.r, K_S)$. Similarly, when a target $\langle K_V : K_1.r_1 \cap \dots \cap K_n.r_n \stackrel{?}{\leftarrow} K_S \rangle$ is satisfied, the credentials associated with the TTG prove that $\text{hasAttr}(K_1.r_1, K_S) \wedge \dots \wedge \text{hasAttr}(K_n.r_n, K_S)$.*

The proof is found in Appendix C.

Example 6. 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): $\text{Ack}_{\text{Alice}}(\text{StateU.student}) = \text{BBB.member.EPub}$ can prove this by using the following credential:

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

Notice that if an adversary were to ask Alice directly about RegistrarB.student, Alice must not indicate whether she has the attribute unless and until the adversary shows he satisfies Alice's ack policy for StateU.student. In the next

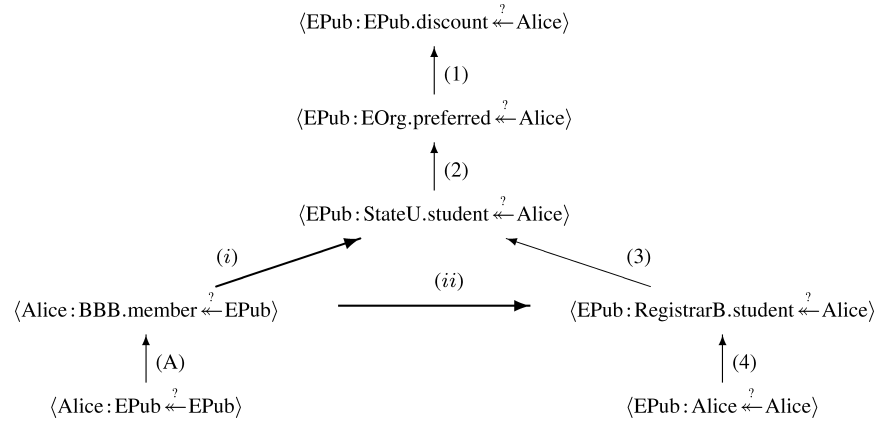


Fig. 3. Trust Target Graph created during negotiation between Alice and EPub. The edges labeled (i) and (ii) represent control edges, corresponding to the dependencies given by Alice’s ack policy (i) and those added by the auxiliary function *init* (ii). Implication edge (A) must be added by EPub and the credential justifying it must be transmitted to Alice before Alice adds implication edges (3) and (4).

section, we introduce an auxiliary function *init* that Alice can use to ensure her ack policy for RegistrarB.student is restrictive enough to ensure this is handled correctly. The TTG constructed during a negotiation between Alice and EPub is shown in Figure 3.

4.4 A Family of TTG Strategies

In this section, we describe a family of negotiation strategies that each use the TTG protocol. In the next section, we prove the safety of this family of strategies.

4.4.1 An Auxiliary Configuration-Initialization Function. We now define an auxiliary function *init*(*G*) that is essential for enforcing ack policies in the context of delegation credentials. It strengthens the ack policies of some attributes so as to ensure that other attributes are adequately protected.

First, we introduce the *delegation credential graph*, which is a directed graph representing the attribute implication relationships documented by credentials in *L*. The node set of the graph is *T* (the set of all attributes) and the edge set is as follows: for each credential $K_0.r_0 \leftarrow K_1.r_1$ in *L*, there is an edge pointing from $K_1.r_1$ to $K_0.r_0$.

The function *init*(*G*) takes $G = \langle K_G, E_G, \text{Ack}_G, \text{AC}_G \rangle$ and returns an extended configuration, given by $\bar{G} = \langle K_G, E_G, \bar{\text{Ack}}_G, \text{AC}_G, \bar{L}_G \rangle$. That is, the function changes the ack policy and collects delegation credentials. We have illustrated the need to strengthen the ack policy above in Example 6. Delegation credentials are issued in a decentralized manner and stored throughout the system. To be used by a negotiator, a delegation credential must first be retrieved. Part of *init*’s job is to retrieve the delegation credentials that may be needed during negotiation. In practice, it is important to avoid the need to collect credentials during negotiation because doing so could create a kind of covert channel: if

the adversary can observe network traffic generated by the negotiator, he may be able to infer sensitive information, for instance, that the negotiator is unfamiliar with an attribute that the adversary has asked about, and so must not have the attribute. This must be avoided if the attribute needs to be protected.

Specifically, *init* does the following:

1. For each attribute $K.r \in \mathcal{T}$ that is sensitive (i.e., $\text{Ack}_G(K.r)$ is defined), collect all delegation credentials in L that can reach $K.r$ in the delegation credential graph. This enables one to determine all attributes that imply (directly or indirectly) the sensitive attribute $K.r$.
2. Propagate the ack policies so that if $K_1.r_1$ implies $K_0.r_0$, then the ack policy for $K_1.r_1$ is at least as strong as the ack policy for $K_0.r_0$ (in the sense that $\overline{\text{Ack}}_G(K_1.r_1) \models \overline{\text{Ack}}_G(K_0.r_0)$). This defeats the forward positive- and negative-inference attacks described in Section 2 as it ensures that when one reveals whether one has the attribute $K_1.r_1$ or not; the ack policy for $K_0.r_0$ has also be satisfied.

The propagation is achieved as follows. For each attribute $K.r \in \mathcal{T}$, introduce a new attribute name $r_{K.r}$ to be used for representing the ack policy of $K.r$, and define $K_{G.r_{K.r}}$ to be equivalent to the conjunction of the ack policies of all attributes implied by $K.r$. The following are technical details of this process. We assume that the new attribute name $r_{K.r}$ is uniquely determined based on $K.r$. We create one new rule: $K_{G.r_{K.r}} \leftarrow K'_1.r'_1 \cap \dots \cap K'_n.r'_n$ in which $\{K'_1.r'_1, \dots, K'_n.r'_n\} \subseteq \mathcal{T}_D$ enumerates the ack policies of every attribute in \mathcal{T} reachable from $K.r$ in the delegation credential graph, including $K.r$ itself. If $\{K'_1.r'_1, \dots, K'_n.r'_n\}$ is empty, define $\overline{\text{Ack}}_G(K.r) = \text{true}$; otherwise define $\overline{\text{Ack}}_G(K.r) = K_{G.r_{K.r}}$.

3. For each attribute $K.r \in \mathcal{T}$ that is either sensitive or that is possessed by G directly (i.e., $K.r \leftarrow K_G \in E_G$), collect all credentials that are reachable from $K.r$. \overline{L}_G is now defined to be the set of all credentials collected in steps 1 and 3.

Collecting credentials reachable from all attributes possessed by G enables the negotiator to know the set of attributes it satisfies, as will be needed for successful negotiation. Collecting credentials reachable from sensitive attributes is needed to help defeat the backward negative-inference attack described in Section 2. If t_2 is implied by t_1 and t_1 is sensitive, but not held by G , the negotiator needs to be aware of t_2 . When asked if he holds t_2 , the negotiator must not provide the answer until the opponent proves satisfaction of the ack policies for t_1 .

4.4.2 Operations of TTG Negotiation Strategies. We denote strategies in the TTG family by $\text{TTGstrat} = \langle \mathcal{M}_{\text{ttg}}, \mathcal{Q}_{\text{ttg}}, \text{rstart}_{\text{ttg}}, \text{start}_{\text{ttg}}, \text{reply}_{\text{ttg}} \rangle$. \mathcal{M}_{ttg} is the set of messages discussed in the previous section. Each state $q \in \mathcal{Q}_{\text{ttg}}$ is given by a triple consisting of an extended configuration, a TTG, and an opponent principal. The state is initialized by using one of the following two functions:

```

replyttg(⟨ $\bar{G}$ , ttgold,  $K_S$ ⟩, inmsg) =
  if not legalUpdate(inmsg, ttgold), return ⟨failure, -⟩
  ttg = apply(inmsg.ops, ttgold)
  if inmsg is empty and candidates( $\bar{G}$ ,  $K_S$ , ttg) =  $\emptyset$ , return ⟨failure, -⟩ % Cannot make progress
  outmsg.ops = empty list
  while the satisfaction state of root(ttg) is not failed or satisfied do
    candidates = candidates( $\bar{G}$ ,  $K_S$ , ttg)
    o = choice(outmsg, candidates) % Consumes head of stream
    if o = stop, break
    outmsg.ops.append(o)
    ttg = apply(o, ttg)
  if the satisfaction state of root(ttg) is failed, return ⟨failure, -⟩
  if  $K_G$  is the verifier of root(ttg) and the satisfaction state of root(ttg) is satisfied, return ⟨success, -⟩
  outmsg.creds = a set of credentials that justify updates in outmsg.ops
  return ⟨⟨ $\bar{G}$ , ttg,  $K_S$ ⟩, outmsg⟩

```

Fig. 4. The reply_{ttg} function is the core of the TTGstrat family of strategies.

start_{ttg}(G , ρ , K_S) returns the state given by $q_1 = \langle \text{init}(G), ttg, K_S \rangle$ in which ttg consists of a single trust target, $\langle K_G : \text{AC}_G(\rho) \leftarrow K_S \rangle$. It also returns a message m_1 that encodes this TTG initialization step.

rstart_{ttg}(G , K_S) returns the state $\langle \text{init}(G), ttg, K_S \rangle$ in which ttg is empty.

An important optimization is to precompute $\text{init}(G)$, since it is invariant across negotiations.

The key idea underlying the function reply_{ttg}, presented in Figure 4, is as follows. When a negotiator N sees a trust target, which asks N to prove possession of a sensitive attribute $K.r$, N asks the opponent to prove that it satisfies the ack policy for $K.r$. After this is done, N reveals whether it has any type-1 credential proving possession of $K.r$. If some delegation credential says that $K.r$ is implied by $K_1.r_1$, N adds the trust target for $K_1.r_1$ and an implication edge, and then repeats the process for $K_1.r_1$. This way, if N has attribute $K_1.r_1$ and thus has attribute $K.r$, this information is released only after $K_1.r_1$'s ack policy is also satisfied. This defeats the backward positive-inference attack.

The centerpiece of the reply_{ttg} function is a collection of rules for *correct processing*, which define candidate updates that can be performed on the TTG. These updates are legal, and are designed to enforce ack policies. Given an extended configuration \bar{G} , an opponent principal K_S , and a TTG ttg , candidates(\bar{G} , K_S , ttg) return a set of candidate updates. This function is defined in the following sections.

reply_{ttg} is implicitly parameterized with respect to the definition of a choice operation. By defining various choice operations, different strategies in the family are obtained. choice(outmsg, candidates) selects from among candidate updates, candidates, one update to be performed. The choice operation signals the end of a negotiator's turn by returning the flag value Stop. The values the choice operation returns are assumed to satisfy the two following requirements: naturally, choice(outmsg, \emptyset) = Stop; and, to ensure negotiations do not fail needlessly, negotiators must not send empty messages if they can avoid it. Thus when outmsg.size = 0, we assume that choice(outmsg, candidates) is not Stop

unless *candidates* is empty. Given a sequence of one or more update operations *ops* and a TTG *ttg* on which they are legal, another primitive operation we use, $\text{apply}(\text{ops}, \text{ttg})$, returns the TTG that results.

In each of its turns, a negotiator iteratively selects an update operation and performs it locally. These operations are transmitted at the end of the negotiator's turn, along with any credentials needed to justify them. Correctly executing negotiators continue taking turns until either the primary target is satisfied (negotiation success), it is failed (negotiation failure), or neither negotiator can perform a correct update (also negotiation failure). The latter happens if a negotiator does not have a necessary attribute, or if there is a cyclic dependence in the policies of the two negotiators with regard to a necessary attribute. (Such a cyclic dependence manifests itself in the TTG as a cycle involving at least two control edges.)

The choice operation is arbitrary, provided it makes a deterministic selection among candidate updates and it satisfies the two requirements mentioned above. This determinism prevents unintended information flow being encoded by the order in which updates are performed. If desired, choice can be modified to take the current TTG as an input parameter. If a choice operation that uses information about the history of the current negotiation is also desired, an auxiliary component can be added to the state and maintained by the choice operation.

4.4.3 Node Processing State Initialization. When a new node is added to a TTG, its processing state should be initialized as follows:

- A trivial target is fully processed.
- For a attribute target, $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$, if $K.r$ is a dummy attribute, the target is opponent-processed, which means that the opponent cannot process it; otherwise, it is verifierprocessed.
- An intersection target is initially opponent-processed.

4.4.4 Verifier-Side Processing. We now define the candidate update operations a negotiator K_V using configuration \bar{G} can perform on nodes for which K_V is the node's verifier. These rules apply to nodes that are not yet marked verifierprocessed.

1. Processing $T = \langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$

- (a) For each locally available policy statement $K.r \leftarrow K_1.r_1$ in which $K.r$ is a dummy attribute (the negotiator controlling), K_V can add an implication edge $T \leftarrow \langle K_V : K_1.r_1 \stackrel{?}{\leftarrow} K_S \rangle$.
- (b) K_V can mark T as verifierprocessed only after (a) is *done*, meaning that all edges that can be added according to (a) have been added.

2. Processing $T = \langle K_V : K_1.r_1 \cap \dots \cap K_n.r_n \stackrel{?}{\leftarrow} K_S \rangle$

- (a) K_V can add the n intersection edges, $T \leftarrow \langle K_V : K_j.r_j \stackrel{?}{\leftarrow} K_S \rangle$, $1 \leq j \leq n$
- (b) K_V can mark T verifier-processed only after (a) is done.

4.4.5 Opponent-Side Processing. We now define the candidate update operations a negotiator K_S using configuration \overline{G} can perform on nodes for which K_S is the subject. These rules apply to nodes that are not yet marked opponent-processed.

1. Processing $T = \langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$ when $\overline{\text{Ack}}_G(K.r)$ is trivially satisfied (i.e., $K.r$ is not sensitive and does not reach any sensitive attribute in the delegation credential graph)

- (a) If $K.r \leftarrow K_S \in E_G$, K_S can add an implication edge $T \leftarrow \langle K_V : K_S \stackrel{?}{\leftarrow} K_S \rangle$.
- (b) If $K.r \leftarrow K_1.r_1 \in \overline{L}_G$, K_S can add an implication edge $T \leftarrow \langle K_V : K_1.r_1 \stackrel{?}{\leftarrow} K_S \rangle$.
- (c) K_S can mark T as opponent-processed if T is satisfied, or if (a) and (b) are done.

2. Processing $T = \langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$ when $\overline{\text{Ack}}_G(K.r)$ is not trivially satisfied

- (a) K_S can add a control edge $T \leftarrow \langle K_S : e_{Ack} \stackrel{?}{\leftarrow} K_V \rangle$, where $e_{Ack} = \overline{\text{Ack}}_G[K.r]$.
- (b) After (a) is done and $\langle K_S : e_{Ack} \stackrel{?}{\leftarrow} K_V \rangle$ is satisfied, if K_S has the credential $K.r \leftarrow K_S \in E_G$, K_S can add the implication edge $T \leftarrow \langle K_V : K_S \stackrel{?}{\leftarrow} K_S \rangle$.
- (c) After (a) is done and $\langle K_S : e_{Ack} \stackrel{?}{\leftarrow} K_V \rangle$ is satisfied, if K_S has the credential $K.r \leftarrow K_1.r_1 \in \overline{L}_G$, K_S can add the implication edge $T \leftarrow \langle K_V : K_1.r_1 \stackrel{?}{\leftarrow} K_S \rangle$.
- (d) K_S can mark T as opponent-processed if T is satisfied, or all of the above steps are done.

The above processing rules defend against the backward positive- and negative-inference attacks discussed in Section 2. For instance, if K_S should have $K.r \leftarrow K_1.r_1$, and K_V should establish the target, $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$, K_S does not mark this target processed until it has added the implication child, $\langle K_V : K_1.r_1 \stackrel{?}{\leftarrow} K_S \rangle$. K_S will then add a control child to the latter target using K_S 's ack policy for $K_1.r_1$. Consequently, the satisfaction state of $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$ will not become either satisfied or failed until the ack policy for $K_1.r_1$ has been satisfied.

4.5 Safety of TTG Strategies

THEOREM 4.2. *For each choice operation meeting, the requirements discussed in Section 4.4, the induced TTG strategy TTGstrat is credential-combination hiding.*

The proof is found in Appendix D.

5. SAFETY OF ACCESS-CONTROL POLICY ENFORCEMENT

In this paper, we use ack policies, but not AC policies, for protecting credentials and their attribute-information content. Recall that the primary distinction between AC policies and ack policies is that AC policies are defined only for resources the negotiator actually has, while ack policies can be defined for an attribute whether the negotiator has the attribute or not. This means that when a credential's AC policy is used during negotiation, the fact that the

negotiator has the credential may be disclosed. By contrast, ack policies can be used without this information leakage. We emphasize that we believe AC policies cannot be used to safely protect the attribute information contained in credentials according to any notion of safety along the lines discussed in Section 3. However, the use of AC policies for protecting credentials has a longer history [Winsborough et al. 2000] than the use of ack policies [Winsborough and Li 2002b], and may serve a complementary purpose in a system, for instance, if the signed credential is considered more sensitive than its unsigned content.

It is straightforward to add AC policies to our formal model of ATN for additional protection of credentials. We now discuss the deficiencies in prior work of the traditional definition of safety for AC policies and present a definition following the spirit of providing meaningful notions of safety.

The existing safety definition of AC policies is inadequate even when not considering the leaking of attribute information. The requirement that “credentials should not be disclosed until AC policies for them are satisfied” is acceptable only for ATN systems of certain kinds, i.e., those that use credentials only by directly transmitting them. It is inadequate for ATN systems which takes advantage of the fact that credentials are structured objects, e.g., by using the signatures to compute messages in a protocol without transmitting the signatures themselves [Li et al. 2003; Holt et al. 2003].

There are two parts of the requirement that are imprecise. First, “credential is disclosed” is undefined. What does it mean? Clearly, sending the exact bit-string of a credential should be viewed as the credential flowing. What if one does not send the exact bit-string, but sends something (presumably derived from the bit-string) that enables everyone to verify that the credential exists? For example, if σ is the signature, then one could send the content (but not the signature) of the credential and $\theta = 2\sigma$; the receiver can recover the signature easily. One may argue that, in this case, the receiver recovers the complete credential, and thus the credential is disclosed. Now consider the case that some value derived from the signature is sent to the opponent, enabling the opponent to verify that the signature exists but not to recover the signature. (Such a value is easily constructed for RSA signatures [Rivest et al. 1978].) Whether this constitutes a disclosure of a credential is not so clear. This becomes even less clear in the case that one uses a zero-knowledge protocol to convince the opponent that one holds the credential, but the opponent cannot use the communication transcript to convince any other party of this. We believe that a suitable notion of AC-policy enforcement should not permit any of these forms of credential disclosure to unauthorized recipients. To capture all such forms of credential disclosure, the precise definition of “a credential is not disclosed” should be “the same communication transcripts can be generated efficiently without having access to the credential.” Note that we do not require such transcripts be generated by negotiators during trust negotiation; we only require that there exists an algorithm that can generate such transcripts efficiently. Since the transcripts can be generated without access to the credential, clearly the credential is not disclosed. This is similar to the notion of simulations and zero-knowledge proofs used in the cryptography literature.

The second part of the requirement that is imprecise is “until AC policies are satisfied.” This is related to the discussion above; how are AC policy satisfied? Does one have to see credential bit-strings, or is it sufficient to be convinced that the credential exists? We argue that a straightforward definition is that “credentials are not disclosed to parties who do not satisfy the corresponding AC policies.” The disclosure of a credential does not violate security so long as the opponent holds the necessary credentials to satisfy the credential’s AC policy.

To summarize, the AC safety requirement should be as follows.

Definition 5.1 (Safety of Access-Control Policies). A negotiation strategy is AC safe if for every configuration G , for every adversary M , and for every feasible attack sequence seq , the response sequence induced from G by seq can be efficiently computed without credentials whose AC policies are not satisfied by M .

The notion of credentials not being disclosed is formalized here by saying that it is not necessary to have access to the credentials to efficiently play the negotiator’s part in the negotiation. Also note that, instead of making requirements on the order of events, we simply require that to receive credentials governed by an AC policy, an opponent must possess credentials satisfying that AC policy.

6. RELATED WORK

Automated trust negotiation was introduced by Winsborough et al. [2000], who presented 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 “parsimonious” 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 implicitly, and in an uncontrolled way, which credentials they hold, by transmitting access-control policy content for them. The length of negotiations in both strategies is, at most, linear in the number credentials the two parties hold. Yu et al. [2000] introduced the quadratic “prunes” strategy, which requires negotiators to explicitly reveal arbitrary attributes with no protection.

Yu et al. [2003] developed families of strategies called disclosure tree protocols that can interoperate in the sense that negotiators can use different strategies within the same family. Seamons et al. [2001] and Yu and Winslett [2003b] studied the problem of protecting contents of policies as well as credentials. These previous works did not address the leaking of sensitive attribute information.

On the aspect of system architecture for trust negotiation, Hess et al. [2002] proposed the Trust Negotiation in TLS (TNT) protocol, which is an extension to the SSL/TLS handshake protocol by adding trust negotiation features. Winslett et al. [2002] introduced the TrustBuilder architecture for trust negotiation systems.

The problem of leaking attribute information was recognized by Seamons et al. [2002] and Winsborough and Li [2002b]. Winsborough and Li [2002a,

2002b] introduced the notion of ack policies to protect this information and studied various inferencing attacks that can be carried out. However, precise notion of safety was not provided in this work.

Yu and Winslett [2003a] have introduced a technique called policy migration that seeks to make it more difficult for the adversary to infer information about a negotiator's attributes based on AC policies. In the versions of credential AC policies disclosed during ATN, the technique moves requirements from policies governing credentials defining sensitive attributes to those of other credentials that are also required by the ATN. This approach obscures the information carried in the ATN about the negotiator's sensitive attributes, but it does not hide it entirely. For instance, by observing multiple negotiations, an adversary can observe that the AC policies presented for a given credential are not always the same and then infer that the negotiator has another credential that the adversary has requested. Moreover, the technique can sometimes cause negotiation to fail when success is possible. For these reasons, it seems clear that policy migration is not an adequate solution to the problem.

The notion of credential-combination hiding is similar to the notion of non-interference [Goguen and Meseguer 1982], which considers a system that has inputs and outputs of different sensitivity levels. A system can be defined as noninterference secure if low-level outputs do not depend upon high-level inputs. The definition for credential-combination-hiding safety says that the behavior the adversary can observe (i.e., low-level outputs) does not depend on credentials proving unacknowledgeable attributes (i.e., high-level inputs). The notion of attribute-combination hiding is similar to the notion of nondeducibility [Sutherland 1986], which requires that low-level outputs be compatible with arbitrary high-level inputs. Our definitions deal with a system that involves communication between the two parties and we want to ensure that one party cannot tell the state of another party. Our notions of indistinguishable configurations are also reminiscent of security definitions for cryptographic protocols.

Inference control has received a lot of attention, particularly in the context of multilevel databases [Staddon 2003], statistical databases [Domingo-Ferrer 2002; Wang et al. 2003] and, to a lesser extent, in deductive databases [Bonatti et al. 1995]. Most of this work focuses on limiting the information that can be deduced from answers to multiple queries. Such schemes require that history information be maintained, allowing multiple interactions with the same party to be correlated, which is a very strong assumption in our context of open systems, an assumption that we do not make. As a result, our approach is quite different.

7. CONCLUSION

Although many ATN schemes have previously been proposed, precise security goals and properties were lacking. In this paper, we have introduced a formal framework for ATN in which we have proposed a precise and intuitive definition of correct enforcement of policies in ATN. We call this safety notion credential-combination hiding and have argued that it captures natural security goals. We have stated two alternative, weaker safety notions that seem somewhat

intuitive, and identified flaws that make them unacceptable. We have formulated the eager strategy using our framework and shown that it meets the requirements set forth in our safety definition, thus supporting our contention that the framework and safety definition are usable. We have presented a family ATN strategies that support a credential system with delegation and shown that these strategies provide credential–combination hiding. This result further supports our contention that credential–combination hiding is a useful definition of safety for ATN.

APPENDIX

A. PROOF OF THEOREM 3.6

THEOREM 3.6. The eager strategy is credential–combination-hiding safe.

PROOF OF THEOREM 3.6. Consider any pair of configurations $G = \langle K, E, \text{Ack}, \text{AC} \rangle$ and $G' = \langle K, E', \text{Ack}, \text{AC} \rangle$ such that $\text{releaseable}(E, \text{UnAcks}(M, G)) = \text{releaseable}(E', \text{UnAcks}(M, G'))$. For any given requester attack sequence, $[K_A, \rho, a_1, a_2, \dots, a_k]$, we show that the reaction sequence it induces given G , $[m_1, m_2, \dots, m_\ell]$, is the same as the response sequence it induces given G' , $[m'_1, m'_2, \dots, m'_\ell]$. Let q_1, q_2, \dots, q_ℓ and $q'_1, q'_2, \dots, q'_\ell$ be the associated states. We use induction on the steps in the eager strategy to show that for each $i \in [1, \ell]$, either $q_i = q'_i \in \{\text{success}, \text{failure}\}$, or $\text{opCreds}_i = \text{opCreds}'_i$ and $\text{locCreds}_i = \text{locCreds}'_i$, in which $q_i = \langle G, \text{opCreds}_i, \text{locCreds}_i, K_O, \rho \rangle$ and $q'_i = \langle G', \text{opCreds}'_i, \text{locCreds}'_i, K_O, \rho \rangle$.

When the attack sequence is a requester attack sequence, the negotiator uses start to begin the negotiation. If $\text{AC}_G(\rho)$ is trivially satisfied, then so is $\text{AC}_{G'}(\rho)$ and both computations return $\langle \text{success}, \text{null} \rangle$, so we are done. So assume otherwise. Referring to the construction of $\langle q_1, m_1 \rangle = \text{start}(G, \rho, K_A)$, clearly $\text{publicCreds} \subseteq \text{releaseable}(E, \text{UnAcks}(M, G))$. By our choice of G and G' , it follows that in the construction using G' , $\text{publicCreds}' = \text{publicCreds}$. (We use primed version of local variable throughout to indicate the values of those variables in the construction using G' and unprimed versions of the variables for the values in the construction using G .) It follows that $\text{locCreds}_0 = \text{locCreds}'_0$. That $\text{opCreds}_0 = \text{opCreds}'_0$ holds is trivial, completing the proof in the base case.

Now we assume $\text{opCreds}_i = \text{opCreds}'_i$ and $\text{locCreds}_i = \text{locCreds}'_i$ for $i \in [1, \ell - 1]$, and show that the induction hypothesis holds for $i + 1$. It is easy to see by inspection of reply that $q_{i+1} = \text{success}$ if and only if $q'_{i+1} = \text{success}$, and the step is shown. Suppose otherwise. Since opCreds_i consists of credentials held by M , it follows that $\text{locCreds}_{i+1} \subseteq \text{releaseable}(E, \text{UnAcks}(M, G))$. Similarly, $\text{locCreds}'_{i+1} \subseteq \text{releaseable}(E', \text{UnAcks}(M, G'))$. Clearly $\text{UnAcks}(M, G) = \text{UnAcks}(M, G')$, so, since $\text{opCreds}_i = \text{opCreds}'_i$ by induction hypothesis, $\text{locCreds}_{i+1} = \text{locCreds}'_{i+1}$. It now follows easily that $\text{opCreds}_{i+1} = \text{opCreds}'_{i+1}$ and $\text{locCreds}_{i+1} = \text{locCreds}'_{i+1}$, as required to complete the induction.

Note that it cannot be that $\ell' > \ell$ because either $\ell = k + 1$ or $q'_\ell \in \{\text{success}, \text{failure}\}$, which terminates the response sequence by definition. Thus the two response sequences are identical, as desired. When the attack sequence is passive, essentially the same proof applies; the base case is simpler and the step is the same. \square

B. PROOF OF THEOREM 3.9

Before we present the proof of this theorem, we note several identities that follow from Definition 3.3.

- (1) $T(\mathbf{E}) \cap U = T(\text{unreleaseable}(\mathbf{E}, U)) \cap U$.

$$\begin{aligned} T(\mathbf{E}) \cap U &= (\cup_{e \in \mathbf{E}} T(e)) \cap U = \cup_{e \in \mathbf{E}} (T(e) \cap U) \\ &= \cup_{e \in \mathbf{E} \wedge T(e) \cap U \neq \emptyset} (T(e) \cap U) \\ &= \cup_{e \in \text{unreleaseable}(\mathbf{E}, U)} (T(e) \cap U) \\ &= (\cup_{e \in \text{unreleaseable}(\mathbf{E}, U)} T(e)) \cap U \\ &= T(\text{unreleaseable}(\mathbf{E}, U)) \cap U \end{aligned}$$
- (2) $T(\text{releaseable}(\mathbf{E}, U)) \cap U = \emptyset$.

$$\begin{aligned} \text{releaseable}(\mathbf{E}, U) \cap U &= (\cup_{e \in \text{releaseable}(\mathbf{E}, U)} T(e)) \cap U \\ &= \cup_{e \in \mathbf{E} \wedge T(e) \cap U = \emptyset} (T(e) \cap U) \\ &= \cup_{e \in \mathbf{E} \wedge T(e) \cap U = \emptyset} \emptyset \\ &= \emptyset \end{aligned}$$
- (3) $\text{releaseable}(\mathbf{E}_1 \cup \mathbf{E}_2, U) = \text{releaseable}(\mathbf{E}_1, U) \cup \text{releaseable}(\mathbf{E}_2, U)$.

$$\begin{aligned} \text{releaseable}(\mathbf{E}_1 \cup \mathbf{E}_2, U) &= \{e \in (\mathbf{E}_1 \cup \mathbf{E}_2) \mid T(e) \cap U \neq \emptyset\} \\ &= \{e \in \mathbf{E}_1 \mid T(e) \cap U \neq \emptyset\} \cup \{e \in \mathbf{E}_2 \mid T(e) \cap U \neq \emptyset\} \\ &= \text{releaseable}(\mathbf{E}_1, U) \cup \text{releaseable}(\mathbf{E}_2, U) \end{aligned}$$
- (4) For all $U' \supseteq U$, $\text{releaseable}(\text{unreleaseable}(\mathbf{E}, U), U') = \emptyset$.

$$\begin{aligned} \text{releaseable}(\text{unreleaseable}(\mathbf{E}, U), U') &= \{e \in \{e \in \mathbf{E} \mid T(e) \cap U \neq \emptyset\} \mid T(e) \cap U' = \emptyset\} \\ &= \{e \in \mathbf{E} \mid T(e) \cap U \neq \emptyset \wedge T(e) \cap U' = \emptyset\} \\ &= \emptyset \end{aligned}$$
- (5) For all $U' \supseteq U$, $\text{releaseable}(\text{releaseable}(\mathbf{E}, U), U') = \text{releaseable}(\mathbf{E}, U')$.

$$\begin{aligned} \text{releaseable}(\text{releaseable}(\mathbf{E}, U), U') &= \{e \in \{e \in \mathbf{E} \mid T(e) \cap U = \emptyset\} \mid T(e) \cap U' = \emptyset\} \\ &= \{e \in \mathbf{E} \mid T(e) \cap U = \emptyset \wedge T(e) \cap U' = \emptyset\} \\ &= \{e \in \mathbf{E} \mid T(e) \cap U' = \emptyset\} \\ &= \text{releaseable}(\mathbf{E}, U') \end{aligned}$$

THEOREM 3.9. The relative strength of the safety definitions is as follows:

1. If strat is credential–combination-hiding safe, then it is attribute–combination-hiding safe.
2. If strat is attribute–combination-hiding safe, then it is attribute-hiding safe.

PROOF OF THEOREM 3.9.

Part 1: Given a credential–combination-hiding safe strategy strat, for every configuration $G = \langle K, \mathbf{E}, \text{Ack}, \text{AC} \rangle$, for every subset U of \mathcal{T} , and for every expressible subset U' of U , we can construct a configuration $G' = \langle K, \mathbf{E}', \text{Ack}, \text{AC} \rangle$ as follows. By the assumption that U' is expressible, there exists \mathbf{E}_0 such that $T(\mathbf{E}_0) \cap U = U'$. Let $\mathbf{E}' = \text{unreleaseable}(\mathbf{E}_0, U) \cup \text{releaseable}(\mathbf{E}, U)$.

We now show **(1a)**: \mathbf{E}' induces the desired set of unacknowledgeable attributes, i.e., $T(\mathbf{E}') \cap U = U'$. From Identities 1 and 2, we have the following:

$$\begin{aligned} T(\mathbf{E}') \cap U &= (T(\text{unreleaseable}(\mathbf{E}_0, U)) \cup T(\text{releaseable}(\mathbf{E}, U))) \cap U \\ &= (T(\text{unreleaseable}(\mathbf{E}_0, U)) \cap U) \cup (T(\text{releaseable}(\mathbf{E}, U)) \cap U) \\ &= (T(\mathbf{E}_0) \cap U) \cup \emptyset = U' \end{aligned}$$

We now use credential–combination–hiding safety to show the following **(1b)**: for every M such that $\text{UnAcks}(G, M) \supseteq U$, G and G' are indistinguishable under strat by M . Let U'' be the set of attributes that are unacknowledgeable to M ; we have $U'' \supseteq U$. It is sufficient to show that $\text{releaseable}(E, U'') = \text{releaseable}(E', U'')$, since by the credential–combination–hiding safety property of strat, M cannot distinguish G and G' . This equality follows from Identities 3, 4, and 5 as follows:

$$\begin{aligned} & \text{releaseable}(E', U'') \\ &= \text{releaseable}(\text{unreleaseable}(E_0, U) \cup \text{releaseable}(E, U), U'') \\ &= \text{releaseable}(\text{unreleaseable}(E_0, U), U'') \cup \text{releaseable}(\text{releaseable}(E, U), U'') \\ &= \emptyset \cup \text{releaseable}(E, U'') = \text{releaseable}(E, U'') \end{aligned}$$

Part 2: Given an attribute–combination–hiding safe strategy strat, for every configuration $G = \langle K, E, \text{Ack}, \text{AC} \rangle$, for every attribute t , we need to show that there exists G' that differs from G in t (i.e., $t \in T(E) - T(E')$ or $t \in T(E') - T(E)$) and for every adversary M , if t in $\text{UnAcks}(G, M)$, G' is indistinguishable from G by M . Case one: if $t \in T(E)$, then let $U = \{t\}$ and $U' = \emptyset$. Clearly, U' is an expressible subset to U . By attribute–combination–hiding safety of strat, there exists a configuration $G' = \langle K, E', \text{Ack}, \text{AC} \rangle$ that satisfies the above requirement. Case two: if $t \notin T(E)$, then let $U = \{t\}$ and $U' = \{t\}$. Clearly, U' is an expressible subset of U . (By the setup of the framework, every attribute has at least one credential to prove it.) Again, by attribute–combination–hiding safety of strat, there exists a configuration $G' = \langle K, E', \text{Ack}, \text{AC} \rangle$ that satisfies the above requirement. \square

C. PROOF OF PROPOSITION 4.1

PROPOSITION 4.1. If a principal updates a TTG legally and propagates the satisfaction state correctly, then when a target $\langle K_V : K.r \stackrel{?}{\leftarrow} K_S \rangle$ is satisfied in the TTG, the credentials associated with the TTG prove that $\text{hasAttr}(K.r, K_S)$. Similarly, when a target $\langle K_V : K_1.r_1 \cap \dots \cap K_n.r_n \stackrel{?}{\leftarrow} K_S \rangle$ is satisfied, the credentials associated with the TTG prove that $\text{hasAttr}(K_1.r_1, K_S) \wedge \dots \wedge \text{hasAttr}(K_n.r_n, K_S)$.

PROOF OF PROPOSITION 4.1. We use induction on the order in which nodes are marked satisfied. Initially, only one target exists and it is not satisfied. Consider the i th node to be marked satisfied. If it is a trivial target, the proposition holds trivially. If it is a standard target, one of its implication children is satisfied. If it is an intersection target, all of its intersection children are satisfied. In each case, the result now follows from the induction assumption on the children and, in the implication case, the fact that the edge is justified. \square

D. PROOF OF THEOREM 4.2

THEOREM 4.2. For each choice operation meeting the requirements discussed in Section 4.4, the induced TTG strategy TTGstrat is credential–combination hiding.

PROOF OF THEOREM 4.2. Given any pair of configurations $G = \langle K, E_G, \text{Ack}, \text{AC} \rangle$ and $G' = \langle K, E_{G'}, \text{Ack}, \text{AC} \rangle$, and any adversary M , by assuming

$\text{releaseable}(E, \text{UnAcks}(M, G)) = \text{releaseable}(E_{G'}, \text{UnAcks}(M, G))$, we show that G and G' are indistinguishable under strat by M .

We will show that the negotiation behaviors obtained by using G and G' are identical. We consider the case in which the attack sequence $[K_A, \rho, a_1, a_2, \dots, a_k]$ is a requester attach sequence and show that the reaction sequence induced by it from G is identical to the reaction sequence induced by it from G' ; the responder attack-sequence case is similar.

Let $\text{init}(G)$ be given by $\overline{G} = \langle K_G, E_G, \overline{\text{Ack}}_G, \text{AC}_G, \overline{L}_G \rangle$ and let $\text{init}(G')$ be given by $\overline{G}' = \langle K_G, E'_{G'}, \overline{\text{Ack}}_{G'}, \text{AC}_G, \overline{L}_{G'} \rangle$. We begin by showing $\overline{\text{Ack}}_{G'}$ is the same as $\overline{\text{Ack}}_G$ and $\overline{L}_G = \overline{L}_{G'}$. By assumption, $\text{Ack}_G(K.r)$ is nontrivial if, and only if, $\text{Ack}_{G'}(K.r)$ is nontrivial, for each attribute $K.r$. So step (1) of init collects the same delegation credentials in both cases. Because Ack_G is identical to $\text{Ack}_{G'}$, the delegation credential graph does not depend on G and the dummy attributes introduced in step 2 are uniquely determined, it follows that $\overline{\text{Ack}}_{G'}$ is the same as $\overline{\text{Ack}}_G$. Thus $\overline{\text{Ack}}_G(K.r)$ is nontrivial if, and only if, $\overline{\text{Ack}}_{G'}(K.r)$ is nontrivial, and consequently, step 3 collects the same delegation credentials in both cases. So $\overline{L}_G = \overline{L}_{G'}$, as desired.

Next we use induction on $i \in [1, \ell]$ to show that the response sequence $[m_1, m_2, \dots, m_\ell]$ generated by using G is identical to the response sequence $[m'_1, m'_2, \dots, m'_\ell]$ generated by using G' . We simultaneously show that except for the configurations they contain, the sequences of states $[q_1, q_2, \dots, q_\ell]$ and $[q'_1, q'_2, \dots, q'_\ell]$ used in the construction of the respective message sequences are also identical.

Base case. Because $\overline{\text{Ack}}_G$ is identical to $\overline{\text{Ack}}_{G'}$, the state q_1 produced by $\text{start}_{\text{ttg}}(G, \rho, K_A)$ is the same as the state q'_1 produced by $\text{start}_{\text{ttg}}(\overline{G}, \rho, K_A)$. It follows that the messages m_1 and m'_1 produced by each of the respective functions are also the same.

Induction step. We must show that for $i \in [2, \ell]$, assuming $m_{i-1} = m'_{i-1}$ and that $q_{i-1} q'_{i-1}$ differ only in their configurations, then $\langle q_i, m_i \rangle = \text{reply}_{\text{ttg}}(q_{i-1}, a_{i-1})$ is the same as $\langle q'_i, m'_i \rangle = \text{reply}_{\text{ttg}}(q'_{i-1}, a_{i-1})$, again, except for the configurations in the states. To do this, we use an inner induction on the iterations of the loop in the definition of $\text{reply}_{\text{ttg}}$ to show that in corresponding iterations, the values of ttg and outmsg obtained by using \overline{G} , are the same as the values obtained by using \overline{G}' , which we denote by ttg' and outmsg' . The base case is straightforward: the outer induction assumption tells us that ttg_{old} and ttg'_{old} are the same, so in the third line of $\text{reply}_{\text{ttg}}$, $\text{apply}(a_{i-1}.\text{ops}, \text{ttg}_{\text{old}})$ is the same as $\text{apply}(a_{i-1}.\text{ops}, \text{ttg}'_{\text{old}})$; the output messages outmsg and outmsg' are both empty lists.

For the inner induction step, we assume that the TTGs and output messages are the same at the top of the loop, and show that they are again the same at the bottom. Because choice is deterministic, it is sufficient to show that $\text{candidates}(\overline{G}, K_A, \text{ttg})$ is the same as $\text{candidates}(\overline{G}', K_A, \text{ttg}')$. This, together with the fact that the TTGs are again the same at the bottom, will show that each of the two negotiations terminate the loop at the same point. This will complete the proof.

We show that $\text{candidates}(\overline{G}, K_A, \text{ttg})$ is the same as $\text{candidates}(\overline{G}', K_A, \text{ttg}')$ by considering each node in the TTGs and each processing rule that might be used

to process it, and by showing that (each instance of) the rule applies when using configuration G if, and only if, it applies when using configuration G' . In the following paragraphs we consider the opponent-side processing rules; cases are labeled according to node-processing rule. The verifier-side processing rules are similar.

Case 1(a) In this case $\overline{\text{Ack}}_G(K.r)$ is trivially satisfied, which means that no member of $\text{UnAcks}(G, M)$ is reachable from $K.r$ in the delegation credential graph. (This follows from step 2 of *init*.) So $K.r \leftarrow K_O \in E_G$ if, and only if, $K.r \leftarrow K_O \in \text{releaseable}(E, \text{UnAcks}(G, M))$. This, in turn, holds if, and only if, $K.r \leftarrow K_O \in \text{releaseable}(E_{G'}, \text{UnAcks}(G, M))$, by the theorem's hypotheses. Finally, $K.r \leftarrow K_O \in \text{releaseable}(E_{G'}, \text{UnAcks}(G, M))$ holds if, and only if, $K.r \leftarrow K_O \in E_{G'}$, again because no member of $\text{UnAcks}(G, M)$ is reachable from $K.r$.

Case 1(b) We have already argued that $\overline{L}_G = \overline{L}_{G'}$, so rule 1(b) applies to ttg if, and only if, it applies to ttg' .

Case 1(c) The satisfaction state of nodes is uniquely determined by the structure of the TTG and the processing state of nodes in it. By induction hypothesis, at the top of the loop ttg and ttg' are identical in structure, including in the processing state of each node. It follows that the satisfaction state of T is the same in the two TTGs. Moreover, since ttg and ttg' are identical, for each node, the same instances of 1(a) and 1(b) can be and have been applied to that node in ttg as in ttg' . (We say a rule “has been applied” if performing the rule does not change the TTG.)

Case 2(a) Because $\overline{\text{Ack}}_G = \overline{\text{Ack}}_{G'}$, the same instances of this rule apply when using G as when using G' .

Case 2(b) Since ttg and ttg' are identical, for each node, the same instances of 2(a) can be and have been applied to that node in ttg as in ttg' . An argument similar to that used in case 1(c) shows that the satisfaction state of $\langle K_O : e_{\text{Ack}} \stackrel{?}{\leftarrow} K_V \rangle$ is the same in ttg as in ttg' . In the case that the satisfaction state is unknown, rule 2(b) applies neither using G nor using G' . If it is satisfied, it follows by Proposition 4.1 that M has attributes that satisfy $\overline{\text{Ack}}_G(K.r)$, which tells us that no element of $\text{UnAcks}(G, M)$ is reachable from $K.r$ in the delegation credential graph. Thus, as we showed in case 1(a), $K.r \leftarrow K_O \in E_G$ if, and only if, $K.r \leftarrow K_O \in E_{G'}$. So the same instances of rule 2(b) apply using G as using G' .

Case 2(c) Since ttg and ttg' are identical, for each node, the same instances of 2(a) can be and have been applied to that node in ttg as in ttg' . As in case 2(b), the satisfaction state of $\langle K_O : e_{\text{Ack}} \stackrel{?}{\leftarrow} K_V \rangle$ is the same in the two TTGs. We have argued above that $\overline{L}_G = \overline{L}_{G'}$. So the same instances of rule 2(c) apply in ttg using G as apply in ttg' using G' .

Case 2(d) Since the two graphs are identical, the same instances of 2(a), 2(b), and 2(c) apply to ttg as apply to ttg' , each rule instance has been applied to ttg if, and only if, it has been applied to ttg' . So the rule 2(d) can be applied to ttg if, and only if, it can be applied to ttg' . \square

ACKNOWLEDGMENTS

Both authors are supported by NSF ITR grant CCR-0325951 (BYU). We thank the anonymous reviewers for their many helpful suggestions.

REFERENCES

- BLAZE, M., FEIGENBAUM, J., AND LACY, J. 1996. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Piscataway, New Jersey. 164–173.
- BONATTI, P. AND SAMARATI, P. 2000. Regulating service access and information release on the web. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS-7)*. ACM Press, New York. 134–143.
- BONATTI, P., KRAUS, S., AND SUBRAHMANYAN, V. S. 1995. Foundations of secure deductive databases. *Knowledge and Data Engineering* 7, 3, 406–422.
- DOMINGO-FERRER, J., Ed. 2002. *Inference Control in Statistical Databases, From Theory to Practice*. Lecture Notes in Computer Science, vol. 2316. Springer-Verlag, New York.
- GOGUEN, J. AND MESEGUER, J. 1982. Security policies and security models. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Piscataway, New Jersey. 11–20.
- GOLDREICH, O. 2001. *The foundations of cryptography—Vol. 1: Basic tools*. Cambridge University Press, Cambridge.
- HERZBERG, A., MASS, Y., MIHAELI, J., NAOR, D., AND RAVID, Y. 2000. Access control meets public key infrastructure, or: Assigning roles to strangers. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Piscataway, New Jersey. 2–14.
- HESS, A., JACOBSON, J., MILLS, H., WAMSLEY, R., SEAMONS, K. E., AND SMITH, B. 2002. Advanced client/server authentication in TLS. In *Network and Distributed System Security Symposium*. 203–214.
- HOLT, J. E., BRADSHAW, R. W., SEAMONS, K. E., AND ORMAN, H. 2003. Hidden credentials. In *Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society*.
- LI, J., LI, N., AND WINSBOROUGH, W. H. 2005. Automated trust negotiation using cryptographic credentials. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*. ACM Press, New York. 46–57.
- LI, N., MITCHELL, J. C., AND WINSBOROUGH, W. H. 2002. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Piscataway, New Jersey. 114–130.
- LI, N., DU, W., AND BONEH, D. 2003a. Oblivious signature-based envelope. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*. ACM Press, New York.
- LI, N., GROSOF, B. N., AND FEIGENBAUM, J. 2003b. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security (TISSEC)* 6, 1 (Feb.), 128–171.
- LI, N., WINSBOROUGH, W. H., AND MITCHELL, J. C. 2003c. Distributed credential chain discovery in trust management. *Journal of Computer Security* 11, 1 (Feb.), 35–86.
- RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. M. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 120–126.
- SEAMONS, K. E., WINSLETT, M., AND YU, T. 2001. Limiting the disclosure of access control policies during automated trust negotiation. In *Proceedings of the Symposium on Network and Distributed System Security (NDS'01)*.
- SEAMONS, K. E., WINSLETT, M., YU, T., YU, L., AND JARVIS, R. 2002. Protecting privacy during on-line trust negotiation. In *2nd Workshop on Privacy Enhancing Technologies*. Springer-Verlag, New York.
- STADDON, J. 2003. Dynamic inference control. In *Proceedings of the 8th ACM SIGMOD Workshop on Research issues in data mining and knowledge discovery*. ACM Press, New York. 94–100.
- SUTHERLAND, D. 1986. A model of information. In *Proceedings of the 9th National Computer Security Conference*. 175–183.

- WANG, L., WIJESEKERA, D., AND JAJODIA, S. 2003. Cardinality-based inference control in data cubes. *Journal of Computer Security* 12, 5 (Sept. 2004), 655–692.
- WINSBOROUGH, W. H. AND LI, N. 2002a. Protecting sensitive attributes in automated trust negotiation. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*. ACM Press, New York, 41–51.
- WINSBOROUGH, W. H. AND LI, N. 2002b. Towards practical automated trust negotiation. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002)*. IEEE Computer Society Press, Piscataway, New Jersey. 92–103.
- WINSBOROUGH, W. H., SEAMONS, K. E., AND JONES, V. E. 2000. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*. Vol. I. IEEE Press, Piscataway, New Jersey. 88–102.
- WINSLETT, M., YU, T., SEAMONS, K. E., HESS, A., JACOBSON, J., JARVIS, R., SMITH, B., AND YU, L. 2002. Negotiating trust on the web. *IEEE Internet Computing* 6, 6 (Nov./Dec.), 30–37.
- YU, T. AND WINSLETT, M. 2003a. Policy migration for sensitive credentials in trust negotiation. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*. ACM Press, New York. 9–20.
- YU, T. AND WINSLETT, M. 2003b. Unified scheme for resource protection in automated trust negotiation. In *Proceedings of IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Piscataway, New Jersey. 110–122.
- YU, T., MA, X., AND WINSLETT, M. 2000. 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)*. ACM Press, 210–219.
- YU, T., WINSLETT, M., AND SEAMONS, K. E. 2003. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security (TISSEC)* 6, 1 (Feb.), 1–42.

Received May 2004; revised July 2005; accepted April 2006