Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation

Kent E. Seamons
Computer Science Department
Brigham Young University
Provo, UT 84602, USA
seamons@cs.byu.edu

Marianne Winslett, Ting Yu
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{winslett,tingyu}@uiuc.edu

ABSTRACT

Automated trust negotiation is a new approach to establishing trust between strangers through the exchange of digital credentials and the use of mobile access control policies that specify what combinations of credentials a stranger must supply in order to gain access to each local service or credential. In this paper, we show that access control policies can also contain sensitive information that should be protected from inappropriate access by strangers during negotiation. We present and analyze two automated trust negotiation strategies that support protection for access control policies. The first is the relevant credentials set *strategy, which does not directly disclose access control policies and has a fast running time, but may disclose more credentials than strictly necessary. The second strategy is the* all relevant policies *strategy, which freely discloses all relevant access control policies that the other negotiating party has earned access to during negotiation, and offers the possibility of disclosing fewer credentials during negotiation.*

1. Introduction

Automated trust establishment between strangers promises to extend trusted interactions to a much broader range of participants than is possible with traditional security approaches that are based on identity and capabilities. With automated trust establishment between strangers, the number of sensitive business processes that can be accomplished electronically will grow substantially, which in the long run will lead to more efficient markets and reduce the cost of doing business. To accomplish this goal, software to establish trust must be ubiquitous; to reach this point, different approaches for establishing trust must be developed and carefully evaluated. This paper contributes to this effort by showing, for the first time, how automated trust establishment can support access control policies that contain sensitive information that should not be given out to just any stranger.

In open systems such as the Internet, establishing trust between participants engaged in a business transaction is crucial when that transaction involves such things as the exchange of sensitive information or the formulation of contractual obligations. The problem of establishing trust is complicated by the fact that the participants may have no pre-existing relationship and may not share a common security domain. For instance, web clients and servers frequently begin an interaction as complete strangers. Learning the identity of the client will not help the server to determine whether the client should be trusted to access the requested resource, and vice versa. Instead, trust may be established by verifying properties other than identity. For example, a service may need only to ensure that a potential client is an Illinois resident. This can be accomplished through an exchange of *digital credentials* that tell each participant what others have to say about their counterpart. Digital credentials are the on-line analogues of paper credentials that people carry in their wallets.

Today's credentials are digitally signed assertions by a credential issuer about the credential owner. A credential is signed using the issuer's private key and can be verified using the issuer's public key. A credential describes one or more attributes of the owner, using attribute name/value pairs to describe properties of the owner asserted by the issuer. Each credential has a type based on the set of attribute names in the credential. Each credential also contains the public key of the credential owner. The owner can use the corresponding private key to answer challenges or otherwise demonstrate ownership of the credential. The owner can also use the private key to sign another credential, owned by a third entity.

Thus, credentials may be combined into *chains*, where the owner of one credential is the issuer of the next credential in the chain. Credential chains permit one entity to trace a web of trust from a known entity, the issuer of the first credential in the chain, to the submitting entity in which trust needs to be established. Multiple chains can be submitted to demonstrate additional properties of the submitting entity and its relationships with known entities. For example, one credential chain might be used to establish that the server is a member of the Better Business Bureau of the USA, and another chain might be used to demonstrate that the server has agreed to safeguard private information.

While some resources are freely available to all, many require protection from unauthorized access. These "protected" resources should be governed by access control policies that specify the requirements to be satisfied in order to be granted access. Access control policies can be used to protect a wide variety of resources, such as services accessed through URLs, roles in role-based access control systems, and capabilities in capability-based systems. Since credentials themselves, and even access control policies, can contain sensitive information, they can also be viewed as resources whose disclosure will be governed by access control policies. In our work, we are concerned with access control policies that describe what credentials a party must submit to gain access to the protected resource. Trust is established by exchanging credentials and requests for credentials, an iterative process known as *trust negotiation*. Different negotiation strategies determine when and how credentials are disclosed; many strategies will require that access control policies be *mobile*, that is, that the policies themselves, or some distillation of the policies, be sent to the other party in a negotiation, so that the other party can understand the requirements for gaining access to the desired resource. To enable wide-scale electronic deployment of sensitive business processes, it is imperative that trust negotiation be automated.

2. Limitations of existing systems

Figure 1 illustrates a protocol for the flow of policies and credentials during trust negotiation between a client and server [15][17]. In figure 1, the client is a stranger to the server and requests a service from the server without submitting any credentials (step 1). An access control policy governs access to the service and specifies the acceptable credentials for submission. When the server receives a request without those credentials, it responds with the policy governing access to the service (step 2). The client consults the policy to determine if it has credentials that satisfy the policy. If so, the client can repeat the request with the appropriate credentials attached (step 3). If the appropriate credentials are submitted, the server grants access to the service (step 4). Step 2 is the only opportunity for the server to tell the client about the access control requirements the client must satisfy to gain access to the service. At that stage in the negotiation protocol, the server knows nothing about the client and must fully disclose the policy governing access to the service for the negotiation to proceed. Thus, existing systems supporting trust negotiation have limitations when mobile policies contain sensitive information, as illustrated by the examples below.

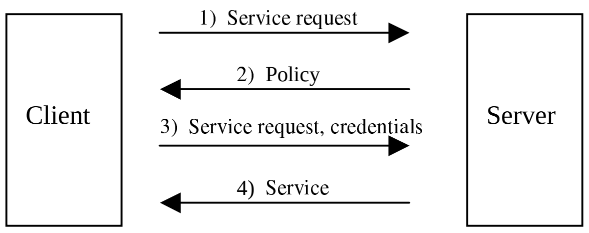

Figure 1. An example trust negotiation protocol without support for sensitive credentials and policies. The client is a stranger to the server and requests a protected service without submitting any credentials.

Example 1. A corporate web server manages information for a collaborative project between the corporation and a secret business partner. The information is accessible only to members of the project team from both companies. To obtain authorized access, team members must submit employee credentials that show that they work in one of the departments associated with the project. Since the access control policy includes information about a business relationship and a secret project, disclosing that policy to a stranger is undesirable.

A solution to this problem is for the server to begin trust negotiation by requesting an employee credential, which the server checks to see if the client works for the corporation or for the business partner, without the client knowing the details of the constraint. If the client is an employee of the appropriate company and department, access is granted. If not, access is denied without the server divulging sensitive information that might disclose the nature of the information the server manages.

Example 2. A corporate web server provides a protected service intended for vice-presidents in Company A and all employees in Company B. Revealing the vice-president constraint to strangers unnecessarily raises interest or concerns regarding the rationale behind the constraint, especially by those who fail to satisfy the constraint. This problem can be solved in the same way as example 1.

Example 3. A web server provides access to sensitive corporate information for the corporation's suppliers. The corporation issues a credential to each supplier organization. Suppliers in turn issue credentials locally to their employees. Suppliers are autonomous, and each supplier has its own approach to issuing credentials. For instance, some suppliers may issue employee credentials at the corporate level, while others will issue site credentials at the corporate level, and employee credentials at the site level. Suppliers will formulate their own policies about which credentials can be used to authenticate their employees. This situation creates the need for supplier-specific access control policies. If the trust negotiation system supports a single access control policy that combines all the supplier-specific policies, outsiders could learn about the security requirements of all the trusted suppliers. In addition, the trusted suppliers would learn about each other's policies, which may not be desirable. The solution is for the server initially to ask for a credential chain that includes the supporting supplier credential. Once the server knows which supplier the client is associated with, the server can release the supplier-specific access control policy information.

Example 4. A web server for a multinational corporation automatically supports benefits enrollment and payroll processing services. The available credentials and associated policies vary dramatically from country to country. To ensure that its policies are scalable and maintainable, and to keep message sizes reasonably small, the server requests a credential indicating the client's country of residence. Then it provides the policy associated with that country only.

These scenarios show that the mere mention of sensitive credentials, and the constraints imposed upon them in an access control policy, can leak sensitive corporate information. Adversaries might make use of that information in attempts to gain illicit access. Even when a policy is disclosed to an entity that eventually proves to be trustworthy, disclosing all the constraints to it may be undesirable. Subdivision of policies also has advantages for performance and policy maintainability.

3. Gradual trust establishment

The examples in the previous section suggest that a solution to the problem of disclosing sensitive policies to strangers is to extend trust negotiation protocols to establish trust gradually, so that a policy referencing a sensitive credential or containing a sensitive constraint is not disclosed to a total stranger. Figure 2 illustrates a protocol that follows this approach. Initially, the client is a stranger to the server and requests access to a service (step 1) without submitting any credentials. The server responds with a policy (step 2) that governs access to the service. This policy includes only information the server is willing to disclose to a stranger. The client determines a combination of credentials that satisfies the server's request and submits them along with another request for the service (step 3). In contrast to the negotiation in figure 1, the client receives a second policy from the server (step 4). This is a sensitive policy that the server was not willing to disclose until it had established sufficient trust in the client using credentials submitted earlier (step 3). The client determines a combination of credentials that satisfies the second policy and repeats the request for service along with those credentials (step 5). Finally, the server determines that it trusts the client enough to provide the service (step 6). Trust could also be established gradually by iterating steps 4 and 5 as many times as necessary.

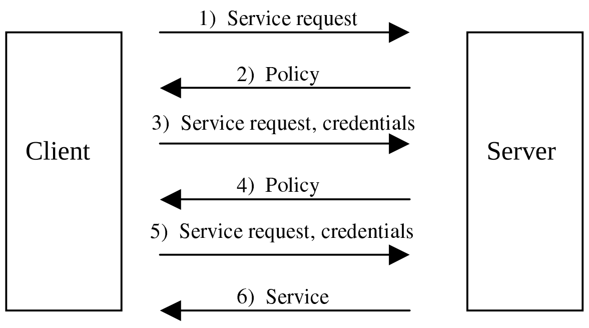
1) Service request
2) Policy
3) Service request, credentials
4) Policy
5) Service request, credentials
6) Service
Client
Server

Figure 2. An example trust negotiation protocol that extends the protocol in figure 1 to support gradual trust establishment.

The protocol in figure 2 allows several possibilities that were missing in figure 1. The policy in step 4 may not be the same for all clients. It could depend on the credentials that were submitted by the client in step 3. Also, constraints, such as the restriction to vice presidents in example 2, can be expressed as policies and evaluated locally using previously acquired credentials, so that the other party never learns the constraints. Although hidden policies can protect the security interests of those who possess them, hidden policies are a potential source of frustration to users, as access control decisions may appear to be non-deterministic when access is denied after the user provides credentials that satisfy a mobile policy. One approach to alleviating this problem is through a provision to inform the user of the presence of hidden policies, assuming such disclosure is considered harmless.

For simplicity, the protocol examples shown thus far show how a server can control access to a protected service, with the service's access control policy being communicated to the client and the client submitting credentials to the server. Current work on trust negotiation includes a generalization of these ideas, where policies and credentials flow in both directions [15]. The remainder of this paper assumes the more general case where both clients and servers possess sensitive resources and sensitive policies that govern access to those resources.

4. Policies and policy graphs

When we step back to formalize the concepts of credentials and policies, we see a need for a more abstract representation of the information contained in credentials and policies, free of implementation details such as encryption protocols and data representation formats. In our work, we assume that the information contained in policies and credentials can be expressed as finite sets of statements in a formal language with a well-defined semantics. Mathematical logic is well-suited to this purpose; for convenience, we will assume that the language allows us to describe the meaning of a policy as the set of all models that satisfy the policy, in the usual logical sense. We say that a set $\boldsymbol{X}$ of statements *satisfies* a policy $\boldsymbol{P}$ if and only if $\boldsymbol{P}$ is true in all models of $\boldsymbol{X}$. To prove that it satisfies a policy $\boldsymbol{P}$, a negotiation participant will submit sets of credentials, whose union forms the set $\boldsymbol{X}$. For convenience, we will often say that a *negotiation participant satisfies a policy* $\boldsymbol{P}$ if the set $\boldsymbol{X}$ of credentials provided so far by the participant satisfies $\boldsymbol{P}$. The empty set of formulas is a policy that is always satisfied.

The layers of access control policies used to guard a resource during gradual trust establishment can be represented as an *access control policy graph* (*policy graph*, for short). In this paper, a protected resource can be a service, a policy, or a credential. A policy graph for a protected resource $\boldsymbol{R}$ is a finite directed acyclic graph with a single source node $\boldsymbol{S}$ and a single sink $\boldsymbol{R}$. (For simplicity, we assume that the name of a node is the name of the resource it represents.) All the nodes except $\boldsymbol{R}$ represent policies that specify the properties that a negotiation participant may be required to demonstrate in order to gain access to $\boldsymbol{R}$. Each sensitive credential and service will have its own separate policy graph. Each policy represented as a node in a policy graph $\boldsymbol{G}$ implicitly also has its own graph---the maximum subgraph of $\boldsymbol{G}$ for which that policy node is the sole sink. Example policy graphs are given in figures 3 and 5; their meaning will be discussed later.

If a negotiating party sends one of its credentials or policies to the other party, we will say that that resource has been *disclosed*. In this paper, to disclose a particular policy node, a party sends the body of the policy associated with the node to the other party, along with the name of the resource whose policy graph contains the node. Alternatively, the party can send some distillation of this information, rather than the exact policy body and resource name. If trust negotiation succeeds and a party is allowed to access the originally requested service $\boldsymbol{R}$, for convenience we will also say that $\boldsymbol{R}$ has been disclosed.

Algorithms for trust negotiation must ensure that every disclosure is *safe*, i.e., that it does not violate the policies put in place to protect the disclosed resource. Under our semantics for policy graphs, it is always safe to disclose the source node of a policy graph. A party can safely disclose a non-source node $\boldsymbol{N}$ in a policy graph if and only if there is a directed path from $\boldsymbol{S}$ to one of $\boldsymbol{N}$'s parents in that graph, such that the other negotiation participant satisfies every policy along the path. (We call such a path an *authorized path* to $\boldsymbol{N}$.) If every disclosure in a negotiation is safe, we say that the negotiation itself is safe. The goal of gradual trust establishment is to find a series of safe disclosures that culminates in the disclosure of $\boldsymbol{R}$.

In this paper, we only briefly mention the issues related to policy languages. The most important point to note is that in general, the semantics of the language(s) used to represent policies must be defined over paths through policy graphs, rather than only over isolated policies. This is because the safety of the disclosure of a resource may depend on the simultaneous satisfaction of many policies in policy graphs, and these policies may need to share references to variables.

The second noteworthy point is that negation must be treated carefully in policy languages. In the context of gradual trust establishment, suppose that a negotiation participant first satisfies policy $\boldsymbol{P_1}$ and then satisfies $\boldsymbol{P_2}$. At that point, the participant might no longer satisfy $\boldsymbol{P_1}$. As a propositional example, perhaps $\boldsymbol{P_1}$ is $\neg\mathbf{p}$ and $\boldsymbol{P_2}$ is $\mathbf{p}$. The participant can satisfy $\boldsymbol{P_1}$ by submitting nothing at all, then satisfy $\boldsymbol{P_2}$ by submitting $\mathbf{p}$, at which point $\boldsymbol{P_1}$ is no longer satisfied. To avoid these situations, in this paper we require that the language be *monotonic*, in the sense that if a set of statements $\boldsymbol{X}$ satisfies $\boldsymbol{P}$, then any superset of $\boldsymbol{X}$ will also satisfy $\boldsymbol{P}$. This restricts, but does not necessarily eliminate, the use of negation in logic-based languages. For example, a policy written in a first-order language could require that a driver's license be presented, and that the state named in the license not be Wisconsin. Suitable subsets of popular logics and their associated semantics have already been identified in previous work, e.g., stable models and the well-founded semantics from the logic programming community [1].

Next we present one potential policy representation language, based on first-order logic without quantifiers or negation, with the following semantics: suppose that a party would like to know if it would be safe to disclose resource $\boldsymbol{P_n}$. In $\boldsymbol{P_n}$'s policy graph, there is an authorized path $\boldsymbol{P_1}, \ldots, \boldsymbol{P_n}$ from the source of the graph to the node representing $\boldsymbol{P_n}$ if and only if the other party has supplied a set $\boldsymbol{W}$ of credentials such that if $\boldsymbol{X_1}, \ldots, \boldsymbol{X_m}$ are the free variables in policies $\boldsymbol{P_1}, \ldots, \boldsymbol{P_{n-1}}$, then $\boldsymbol{W}$ satisfies the formula $\exists \boldsymbol{X_1} \ldots \exists \boldsymbol{X_m} (\boldsymbol{P_1} \wedge \ldots \wedge \boldsymbol{P_{n-1}})$ under the usual first-order semantics.

Example policy graphs for examples 1–4 are shown in figure 3, using the first-order policy language just presented. To save space and simplify presentation, we elide the details of credential chaining and verification of credential ownership, which would be needed in practice.

Although policy graphs are motivated by sensitive policy information, they have other advantages in cases when a policy graph is extremely large, as in example 4. Only the relevant subset of the graph need be disclosed, potentially saving communications and storage resources. For example 4 in figure 3, a client will first be sent the policy asking for an IBM employee credential. If one is supplied, the client is then asked for a passport. (The policy does not compare the names on the two credentials, instead assuming that anyone who can pass the authentication challenges for both credentials is the individual referred to by both credentials.) The server then immediately determines which successor node is satisfied, based on the passport's issuing country, without further communication with the client. The process will then move on to the appropriate subtree. Example 3 in figure 3 is quite similar; here we see how the subtrees join together again.

In example 1 in figure 3, the companies Microsoft and IBM are not mentioned in the source node, instead appearing only in its immediate successors. The employee credential supplied to satisfy the source node will allow the successor nodes to be evaluated without further communication with the client, who will not see the policies in the successor nodes. (If the company names should not be secret, then the source node should also require that the company name be IBM or Microsoft.) Example 2 in the same figure is very similar.

For the remainder of the paper, we switch to a different policy representation language: propositional logic without negation. The semantics of this language will already be familiar to the reader, and its semantics does not need modification for use with policy graphs. This simplicity allows us to focus on the properties of the negotiation strategy rather than on explanations of our handling of variables, negation, graph semantics, and determining whether a credential appears in a formula.

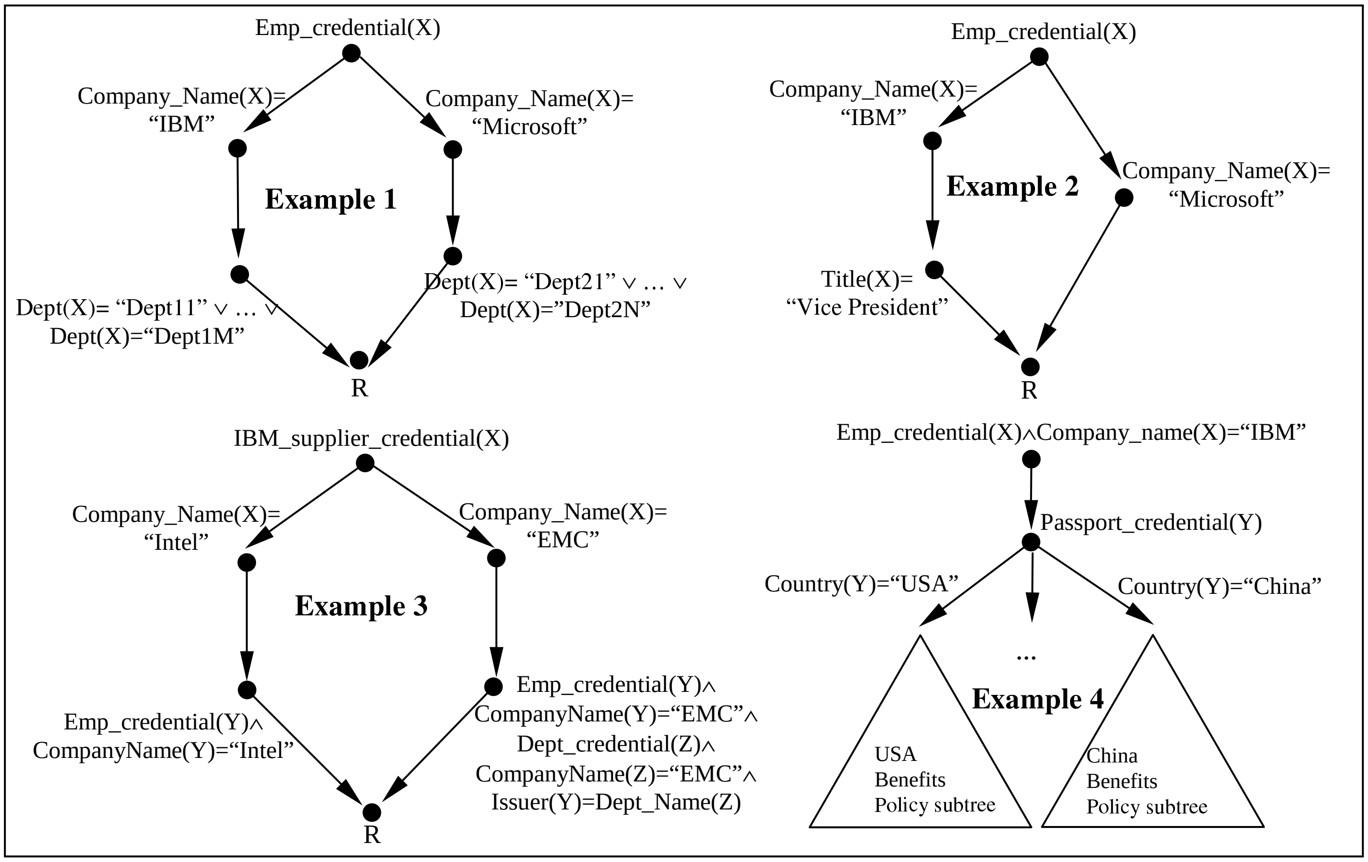
Example 1
Emp_credential(X)
Company_Name(X)= "IBM"
Company_Name(X)= "Microsoft"
Dept(X)= "Dept11" ∨ … ∨ Dept(X)="Dept1M"
Dept(X)= "Dept21" ∨ … ∨ Dept(X)="Dept2N"
R
Example 2
Emp_credential(X)
Company_Name(X)= "IBM"
Company_Name(X)= "Microsoft"
Title(X)= "Vice President"
R
Example 3
IBM_supplier_credential(X)
Company_Name(X)= "Intel"
Company_Name(X)= "EMC"
Emp_credential(Y)∧ CompanyName(Y)="Intel"
Emp_credential(Y)∧ CompanyName(Y)="EMC"∧ Dept_credential(Z)∧ CompanyName(Z)="EMC"∧ Issuer(Y)=Dept_Name(Z)
R
Example 4
Emp_credential(X)∧Company_name(X)="IBM"
Passport_credential(Y)
Country(Y)="USA"
Country(Y)="China"
...
USA Benefits Policy subtree
China Benefits Policy subtree

Figure 3. Policy graphs for examples 1–4, using a subset of first-order logic with a graph-based semantics.

5. Negotiation strategies for sensitive policies

We say that a negotiation strategy is *safe* if all possible negotiations conducted by two parties using the strategy are safe. A strategy is *complete* if whenever there exists a safe sequence of disclosures culminating in the disclosure of ***R***, the strategy will eventually disclose ***R***. While all negotiation strategies share the goal of safely disclosing ***R***, they differ in how they try to construct an authorized path to ***R***. For example, a negotiation participant might choose to send a credential to the other side as soon as that disclosure is safe—an *eager* approach [15]. Eager strategies disclose credentials as soon as possible, trying to negotiate trust as quickly as possible. The primary disadvantage is that some credentials may be disclosed unnecessarily. A more *cautious* alternative is to delay sending the credential until the negotiation stage satisfies certain properties. For example, perhaps the credential will not be sent until the other party has demonstrated a need to see it. At an extreme, a participant might refuse to send any credentials until it has determined that an authorized path exists for the originally requested resource [15]. Another dimension of variation is whether both parties must use the same strategy for a successful negotiation. In this paper, we assume that both parties use the same high-level strategy.

We say that a credential ***C*** is *syntactically relevant* to obtaining access to resource ***R*** if ***C*** appears[1] in the policy of a node ***N*** in ***R***'s policy graph, and an authorized path to ***N*** has already been found. Similarly, ***C*** is *semantically relevant* if an authorized path to ***N*** has been found, ***C*** is a member of some set ***X*** of credentials that satisfy ***N***'s policy, and no proper subset of ***X*** satisfies ***N***'s policy. Further, for either kind of relevance, if ***C*** is relevant to a resource $\boldsymbol{R_1}$, and R_1 is relevant to ***R***, then ***C*** is also relevant to ***R***. Ideally, we would only disclose semantically relevant credentials, so that a policy of the form $\boldsymbol{C_1} \vee (\boldsymbol{C_1} \wedge \boldsymbol{C_2})$ would not tempt us to disclose $\boldsymbol{C_2}$. Depending on the language used to express policies, it can be extremely expensive or effectively impossible to determine which credentials are semantically relevant, or to convert a policy into a form where all syntactically relevant credentials are semantically relevant. Thus in this paper, we consider a credential relevant if and only if it is syntactically relevant.

During negotiations, participants send each other credentials and requests for credentials, often couched as the bodies of policies. In practice, a received credential will be an encrypted object that looks nothing like the description of that credential in the original request for it. In our formal language, however, the same propositional symbol is used both to denote the credential itself and the description of that credential that occurs in the request and in the body of a policy. Thus software would not confuse the two, but a human reader might. To help with this problem, in the pseudocode of our algorithms, we say that the *description* of a credential appears in the body of a policy or in a request for that credential, rather than saying that the credential itself appears in the policy or request.

When there is an authorized path to a resource, we say that the resource is *unlocked*. All unlocked resources can be safely disclosed.

Proposition 1. *Suppose that a party satisfies the policy represented by a policy graph node. Then the party will satisfy that policy for the remainder of the negotiation.* ♦

Proof. Follows from the monotonicity property of the language used to represent credentials and policies. ♦

Corollary 1. *If there is an authorized path to node **P** in a policy graph, then there will be an authorized path to **P** for the remainder of the negotiation.* ♦

Corollary 2. *Once node **P** is unlocked, the credential or policy represented by **P** can safely be disclosed at any point in the remainder of the negotiation.* ♦

Suppose that there is an authorized path to a node ***P*** in a policy graph. If one of ***P***'s children is locked, we say that ***P*** is an *innermost unlocked policy*, and the child is an *outermost unlocked policy*.

Proposition 2. When new credentials are disclosed by the other party in a negotiation, suppose that a node ***N*** in a policy graph becomes unlocked for the first time. Then ***N*** must be a descendant of a node that was an innermost unlocked node immediately before the new disclosures.

Proof. Follows from Proposition 1 and its corollaries, and the definition of an innermost unlocked node. ♦

Proposition 2 tells us that upon receiving new credential disclosures, a stateful negotiation strategy can determine which graph nodes are newly unlocked by simply checking whether any policy at an innermost unlocked node ***P*** is now satisfied. If so, each child, ***Q***, of ***P*** can now be unlocked and tested to see if ***Q***'s policy is now satisfied. If ***Q***'s policy is satisfied, then ***Q***'s children can now be unlocked, and so on. If the sink of the graph becomes unlocked, the whole process can stop. Otherwise, when the process stops, the newly unlocked descendants of ***P*** can be checked to see which, if any, of them are now innermost unlocked nodes.

[1] Note that the definition of "appears" must be tailored to the particular formal language used to represent policies and credentials. For example, in our propositional language, a credential ***C*** appears in a formula if and only if the propositional symbol associated with ***C*** occurs in the formula. In a language based on first-order logic, the term "appears" will need to be carefully defined to include not just exact occurrences of the credential, but also an appropriate notion of unification.

In the pseudocode for our negotiation strategy algorithms, we do not present the details of how each party checks a policy for satisfaction (a language-specific routine) or computes the set of innermost unlocked nodes. A stateful negotiation participant would probably choose to cache the set of innermost unlocked nodes at the end of each round of negotiation.

Proposition 3. *Given a sequence of safe disclosures terminated by the disclosure of **R**, let **C** be the first credential in the sequence. Then **C** must be freely available (i.e., the policies in its policy graph are always satisfied).*

Proof. For ***C*** to be safely disclosed, there must be an authorized path through its policy graph. Since no other credentials have been disclosed, all the policies along that path must be satisfied by the empty set of formulas. In other words, ***C*** must be freely available. ♦

In general, negotiation strategies assume that both participants bargain in good faith. A service provider or client might wish to start its trust negotiation by verifying that its negotiation partner is using a negotiation package that has been certified by an appropriate inspection service, giving confidence that the negotiations will adhere to certain ethical guidelines.

5.1. The relevant credentials set strategy

In the *naive eager* approach, the two participants repeatedly send each other all their credentials for which an authorized path has been found [15]. An advantage of this strategy is that it does not disclose one party's access control policies directly to the other party. However, the naive eager strategy usually results in disclosure of credentials that are actually irrelevant for access to the desired resource. We now introduce a *relevant credentials set* strategy that discloses only credentials that might be considered relevant to the access control decision, while still avoiding direct disclosure of policies.

We say that a credential is *local* to a party if the party possesses or has been asked for that credential; *remote* credentials are those that the other party possesses or has been asked for. We assume that the access policies for a local resource will never ask for credentials from the local party, since the remote party is the one who needs access. For convenience, but without loss of generality, we will go a step further and say that local credentials will not appear in access control policies for local resources.

In the relevant credentials set strategy, the negotiation participants do not send policies to each other; instead, they only tell each other what credentials are syntactically relevant to the authorized paths that they are trying to construct. The two parties are trying to construct authorized paths to the credentials that have previously been requested during the negotiation, and the server is trying to construct an authorized path to the protected service ***R*** that the client originally requested.

More precisely, the parties take turns sending each other messages of the form (*Credentials*, *CredentialRequest*), where *Credentials* is a set containing the sender's relevant credentials that have become unlocked since the last time the sender sent a message, and *CredentialRequest* is a set containing descriptions of all the syntactically relevant credentials that could advance the negotiation. The negotiation terminates with failure whenever one of the participants has no new credentials to disclose or further credentials to request.

During negotiation, each participant maintains four sets of local information constituting the state of the negotiation.[1] The set *DisclosedRemoteCredentials* contains all the credentials disclosed by the other negotiation participant. Set *RequestedLocalCredentials* contains descriptions of the local credentials that have been requested by the other participant during the negotiation. The set *DisclosedLocalCredentials* contains all the local credentials previously disclosed to the other negotiation participant. Set *RequestedRemoteCredentials* contains a description of all the remote credentials requested from the other negotiation participant.

The pseudocode for the relevant credentials set strategy is contained in figure 4. When a client requests a protected service ***R***, the server initiates trust negotiation with the client, to obtain client credentials from the client to authorize access to the service. To do this, the server invokes the *RCS_Message_Handler()* function with the arguments initialized to the empty set. During the negotiation, when a party receives a message (*Credentials*, *CredentialRequest*), it must also invoke the same function.

Figure 5 includes an example of a 9-message trust negotiation using the relevant credentials set strategy. Note that it discloses more credentials than strictly necessary to satisfy the policies.

Theorem 1. *The* relevant credentials set strategy *is safe and complete for monotonic propositional languages.*

Proof. Safety: Under the relevant credentials set strategy, if a credential ***C*** is put into the set *NewlyUnlockedCredentials*, the precondition is that ***C*** is unlocked by the credentials received from the other party. By the definition of "unlocked resource", there is an authorized path to ***C***. Therefore, the disclosure of

[1] To create a stateless version of this negotiation strategy for the server, each message sent between the two parties would need to include the four sets used to record state.

```
RCS_Message_Handler (Credentials, CredentialRequest)
 DisclosedRemoteCredentials = DisclosedRemoteCredentials ∪ Credentials.
 If service R is unlocked by DisclosedRemoteCredentials,      // This check applies only to the server.
   Then exit the negotiation and grant the other party access to R.
 Else   // determine the next outgoing message.
   RequestedLocalCredentials      = RequestedLocalCredentials ∪ CredentialRequest.
   UnlockedRequestedCredentials = the set of all local credentials described in RequestedLocalCredentials
                                  that are unlocked by DisclosedRemoteCredentials.
   NewlyUnlockedCredentials     = the set of unlocked local credentials in UnlockedRequestedCredentials
                                  that are not in DisclosedLocalCredentials.
   RemoteCredentialRequest = ∅.
   LockedLocalCredentials = the set of all local credentials described in RequestedLocalCredentials that are locked.
   For each local resource $R_{local}$ where $R_{local}$ is credential C ∈ LockedLocalCredentials or $R_{local}$ is service R
     Let G be the policy graph for $R_{local}$.
     For each innermost unlocked policy node P in G   // P has an authorized path and is unlocked but not satisfied.
       For each credential $C_{remote}$ that is described in P      // Local policies describe needed remote credentials.
         If $C_{remote}$ ∉ DisclosedRemoteCredentials and $C_{remote}$ ∉ RequestedRemoteCredentials
           // $C_{remote}$ has not been received or requested.
           Then add the description of $C_{remote}$ to RemoteCredentialRequest.
       If NewlyUnlockedCredentials and RemoteCredentialRequest are both empty sets,
         Then send a Failure message to the other party.
       Else
         DisclosedLocalCredentials = DisclosedLocalCredentials ∪ NewlyUnlockedCredentials.
         RequestedRemoteCredentials = RequestedRemoteCredentials ∪ RemoteCredentialRequest.
         Send the message (NewlyUnlockedCredentials, RemoteCredentialRequest) to the other party.
End of RCS_Message_Handler.
```

Figure 4. The pseudo-code for the *relevant credentials set* strategy.

credential ***C*** in the relevant credentials set strategy is safe. Similarly, a description of a remote credential ***C′*** is put into the set *RemoteCredentialRequest* only if ***C′*** appears in a local innermost unlocked policy ***P***, in which case there is an authorized path to ***P***. Therefore, the request for a remote credential ***C′*** is also safe.

Completeness: If there is a safe credential disclosure sequence ***Seq*** terminated by the disclosure of ***R***, we need to prove that when the negotiation stops, ***R*** is disclosed. We prove this by induction on n, the total number of credentials and services possessed by the two parties. By Lemma 1 below, we can assume ***Seq*** does not contain any syntactically irrelevant credentials. When $n = 1$, then ***R*** is the only credential or service possessed by the two parties. If ***R*** is freely available, the negotiation succeeds immediately; otherwise the negotiation fails. Either way, the theorem holds.

Assume when $n = k$, the relevant credentials set strategy is complete. When $n = k+1$, let ***E*** be the set of all policies held by the two parties.

First suppose that the strategy fails without disclosing any credentials. In that case, then when the negotiation stops, no relevant credentials were unlocked. That means there are no freely available relevant credentials. However, this contradicts Proposition 3 and the fact that ***Seq*** is a safe credential disclosure sequence containing only credentials relevant to ***R***. We conclude that the strategy discloses at least one credential.

Suppose credential ***C*** is the first credential to be disclosed during the negotiation. By Proposition 3, ***C*** must be freely available. Let us replace every occurrence of ***C*** in the bodies of all policies by *true*. We call the new set of policies ***E′***. Obviously, ***E*** has a sequence to unlock ***R*** if and only if ***E′*** has one. Since in ***E′*** there are only k credentials and services possessed by the two parties, by the induction hypothesis, at the end of a negotiation using ***E′*** and the relevant credentials set strategy, ***R*** is disclosed. Suppose that ***Seq1*** is the resulting disclosure sequence, and let us watch the strategy as it negotiates using ***E*** and creates a new disclosure sequence ***Seq2***.

Suppose that the first point in which ***Seq1*** and ***Seq2*** differ is the disclosure messages M_1 and M_2, respectively. The difference can only occur because ***C*** has now become relevant for ***Seq2***, while ***C*** will never become relevant for ***Seq1***. Thus these two messages can only differ in that M_2 includes a request for ***C***, and M_1 does not. Further, M_2's successor will include the disclosure of ***C***, while M_1's does not. Other than that, every credential that becomes relevant and is requested in ***Seq1*** will also become relevant and be requested in ***Seq2***, although the request will come one round of messages later if the disclosure of the requested credential depends directly or indirectly on the disclosure of ***C***. Similarly, every credential that is requested and later disclosed in ***Seq1*** will also be requested and disclosed in ***Seq2***, although it will be disclosed one round of messages later in ***Seq2***, if its disclosure directly or indirectly depends on the disclosure of ***C***. We conclude that if ***R*** is disclosed in ***Seq1***, then ***R*** will also be disclosed at the end of ***Seq2***.

Lemma 1. Suppose that ***Seq*** is a sequence of safe credential disclosures, terminated by the disclosure of ***R***. Let ***Seq′*** be created by removing from ***Seq*** all disclosures of syntactically irrelevant credentials. Then ***Seq′*** is also a safe disclosure sequence for ***R***.

Proof. Suppose the sequence ***Seq*** of safe credential disclosures is $\boldsymbol{C_1}$, …, $\boldsymbol{C_k}$, ***R***. If there is no syntactically irrelevant credential in the sequence, the lemma holds. Otherwise, let $\boldsymbol{C_j}$ be the last syntactically irrelevant credential appearing in ***Seq***. Therefore, $\boldsymbol{C_j}$ does not appear in the policy graphs of $\boldsymbol{C_{j+1}}$, …, ***R***. Thus, removing $\boldsymbol{C_j}$ from ***Seq*** will not affect the safe disclosure of $\boldsymbol{C_{j+1}}$, …, ***R***. So after removing $\boldsymbol{C_j}$ from ***Seq***, the resulting sequence is still a safe credential disclosure sequence. Repeat the above step until the sequence does not contain any syntactically irrelevant credentials. Therefore the final sequence ***Seq′*** which does not contain any syntactically irrelevant credentials, is also a safe disclosure sequence.♦

For propositional languages, an upper bound on the total number of messages exchanged can be computed by determining the *credential count* of each party, that is, the number of credentials the party possesses plus the number of credentials appearing in the party's policies.

Proposition 3. *In the worst case, the* relevant credentials set *strategy exchanges 2c+2 messages, where c is the smaller of the two credential counts of the*

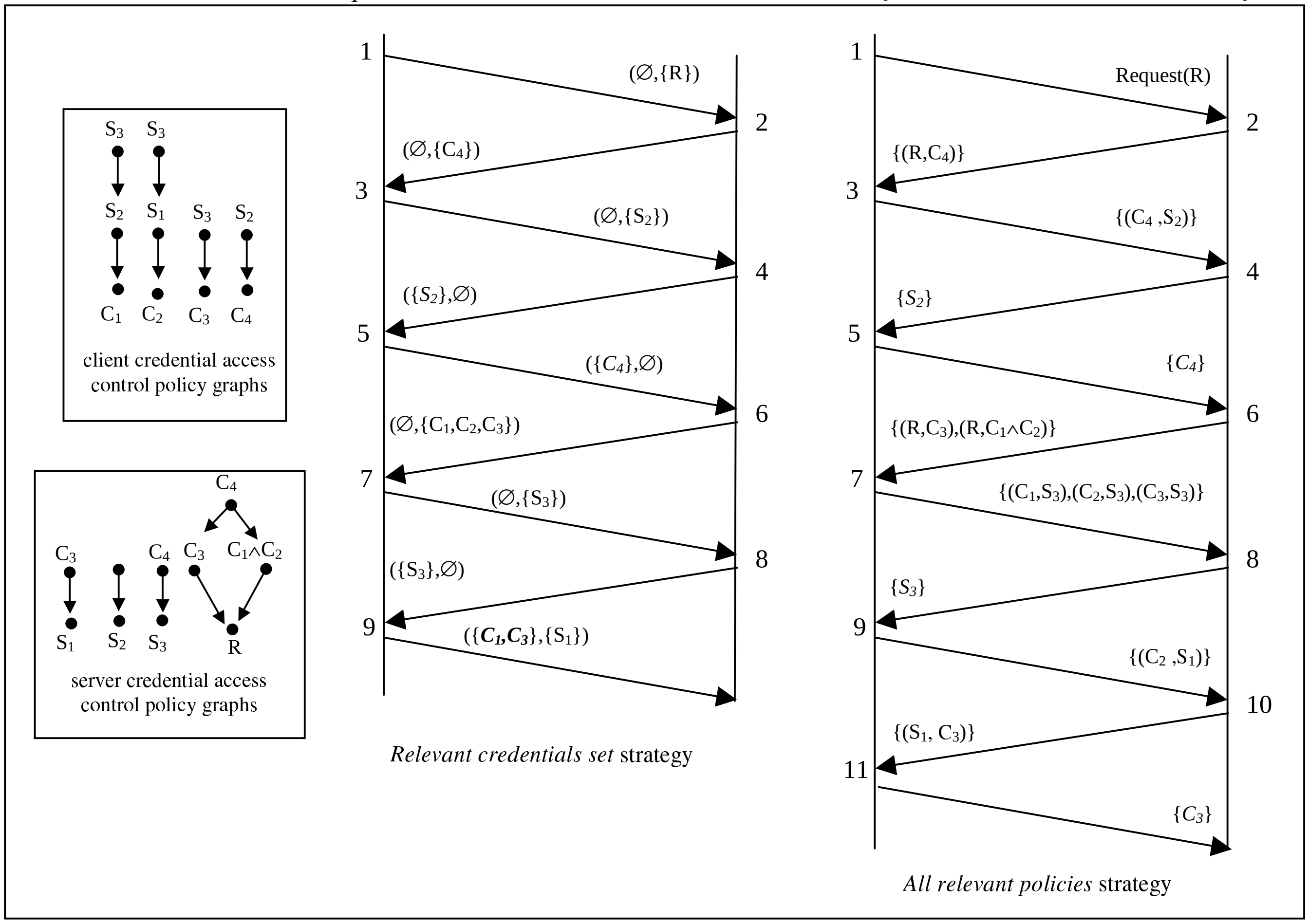

Figure 5. Two example trust negotiations using the *relevant credentials set* strategy and the *all relevant policies* strategy. In this example, the all relevant policies strategy allows the client to access the service with fewer credential disclosures. Credential disclosures are indicated in italics.

negotiating parties. If each credential and credential description has length 1, then the sum of the lengths of all messages sent during negotiation is no more than $c_1 + c_2$,, where c_1 and c_2 are the credential counts of the two parties.

Proof. Suppose the party with the smaller credential count is ***A***. Each time ***A*** sends a message, it either discloses at least one of its credentials in *Credentials* or mentions the description of at least one credential appearing in its policies in *CredentialRequest*. During the negotiation, ***A*** discloses each credential and credential description at most once. Since the two parties of the negotiation take turns sending messages, after at most $2c$ messages, ***A*** will have disclosed all its credentials and descriptions of all the credentials in its policies. After that, if ***A*** receives another message from the other party, it should send a Failure message and the negotiation terminates. Therefore the total number of messages during the negotiation will not exceed $2c+2$.

During the negotiation, a credential's description and its disclosure each appear in at most one message. Thus each credential contributes at most 2 to the sum of lengths of all messages. If a credential is disclosed, then its description must appear in the policy graphs of the other party, so the size of the description of a disclosed credential is included in the other party's credential counts. Each credential appearing in a party's policy graphs but not possessed by the other side can contribute at most 1 to the sum of lengths of all messages. Therefore the sum of lengths of all messages will not be more than $c_1 + c_2$. Here we assume the Failure message's size is 0. ♦

While the relevant credentials set strategy does not directly disclose policies, in the worst case an adversary who satisfies an unlocked policy *P* can learn every disjunct it satisfies in the disjunctive normal form of *P*, by submitting different subsets of its unlocked credentials and seeing which lead to failure. An adversary who cannot satisfy *P* can learn very little.

Proposition 4. *Suppose that during a negotiation where the local party is using the* relevant credentials set *strategy, an adversary has obtained an authorized path to a remote policy graph node representing policy* **P***, and the adversary's set* **U** *of its own credentials does not satisfy* **P***. Let* **C** *be a set containing two or more credentials appearing in* **P** *but not in* **U***. Then the adversary cannot determine whether any particular subset of the union of* **U** *and* **C** *would satisfy* **P***.*

Proof. The relevant credentials set strategy will give the adversary the set ***C*** (more precisely, a superset of ***C***, since other policies may have become newly relevant in the same round of negotiation). Thus in the worst case, the adversary knows that ***P*** is entailed by the union of ***U*** and ***C***. Proposition 4 says that the adversary cannot pin down the form of ***P*** more precisely than that. According to the relevant credentials set strategy, all the adversary can do is to submit a subset of ***U*** and check whether it leads to failure. Since ***U*** cannot satisfy ***P***, the result is always failure. Let ***U*** be the set $\{S_1, \ldots, S_k\}$, and let $C = \{C_1, \ldots, C_t\}$, $t \geq 2$, be those credentials appearing in ***P*** but not in ***U***. To make the adversary's attack even simpler, suppose the adversary knows that ***P*** is one of the following policies: $S_1 \wedge \ldots \wedge S_k \wedge C_1 \wedge \ldots \wedge C_t$ or $(S_1 \wedge \ldots \wedge S_k \wedge C_1) \vee (S_1 \wedge \ldots \wedge S_k \wedge C_2) \vee \ldots \vee (S_1 \wedge \ldots \wedge S_k \wedge C_t)$. Obviously no matter what ***P*** is and what subset of ***U*** the adversary submits, ***P*** is never satisfied. Thus the adversary learns nothing new about ***P***, although the adversary may learn more about the other policies relevant to ***R***.

On the other hand, if ***C*** contains only a single credential, the adversary might be able to figure out the exact form of ***P***. For example, suppose the adversary has supplied every credential ever requested but one. When negotiations fail, the adversary will realize that the failure is due to the lack of that single credential, which must appear in one of the party's relevant policies. For example, if failure occurs during the first round of negotiations, then the adversary can think of that single missing credential as the policy associated with the source node of the requested resource's policy graph.[1]♦

We can modify the relevant credentials set strategy to reduce the number of credentials disclosed, on average. To do this, a party may decide to remove one or more credentials from *NewlyUnlockedLocalCred*, reserving the possibility of including them in a later message. If a party has employed this strategy and it receives a *Failure* message, it should then disclose more of its unlocked credentials. The negotiation does not fail until two *Failure* messages in a row are sent.

The disadvantage of this variant is that it will require more messages on average and may sometimes reveal more information about a party's access control policies. For instance, consider the example given in figure 5. After message 9 has been received, C_1 and C_3 are both unlocked. In such a situation, the original relevant credentials set strategy requires the client to disclose both of them, even if the policy to be satisfied is $(C_1 \wedge C_2) \vee C_3$. Under the variant, the client can disclose the credentials one by one. If it discloses C_3 first, then the server will make ***R*** available

[1] Due to the existence of equivalent policy graphs with different topologies, the missing credential might not be logically equivalent to the policy associated with the graph's source node. For example, the source node policy could be the empty set of formulas, and the missing credential could be the policy of one of its children. The distinction is unimportant for us here.

immediately. Then the client knows that C_3 alone is sufficient to gain access to ***R***.

5.2. The *all relevant policies* strategy

As the example in figure 5 shows, the credentials disclosed by the relevant credentials set strategy are relevant but perhaps redundant for gaining access to the desired service ***R***. This is because the parties share only high-level information about their access control policies, even when the policies are unlocked. A more reasoned strategy, with respect to credential disclosure, can be used if the parties do not mind eagerly disclosing relevant unlocked policies.

In the all relevant policies strategy, the parties take turns disclosing policies and credentials to one another. Each negotiation participant must always disclose all innermost unlocked relevant policies. If a participant has no additional policies to disclose, then the participant must disclose enough credentials to unlock at least one more policy. More precisely, the parties take turns sending each other messages of the form (∅, *Policies*) or (*Credentials*, ∅). *Policies* is a set of disclosed policies, i.e., ordered pairs of the form (**C**, $\mathbf{P}_c$), where each **C** describes a credential (or possibly a service in the case of the server) and $\mathbf{P}_C$ is the body of a policy represented by a node in the policy graph for **C**. *Credentials* is a set of the sender's relevant unlocked credentials that, together with the sender's previously disclosed credentials, will satisfy at least one policy the recipient has previously disclosed. The negotiation terminates with failure if the two parties both send empty disclosure messages, one immediately after the other.

During negotiation, each participant maintains four sets of local information constituting the state of the negotiation. The set *DisclosedRemoteCredentials* contains all the credentials disclosed by the other negotiation participant. The set *DisclosedRemotePolicies* contains a set of policy disclosures, each an ordered pair as described above. The bodies of these remote policies contain descriptions of local credentials. The set *DisclosedLocalCredentials* contains all the local credentials previously disclosed to the other negotiation participant. The set *DisclosedLocalPolicies* contains all the local policies (ordered pairs) disclosed to the other negotiation participant. These policies contain descriptions of remote credentials.

The pseudocode for the all relevant policies strategy is contained in figure 6. When a client requests a protected service ***R***, the server initiates trust negotiation with the client to obtain client credentials from the client that authorize access to the service. To do this, the server invokes the *ARP_Message_Handler()* function with the arguments initialized to the empty set. During the negotiation, when a party receives a message, it invokes the same function.

The example negotiation in figure 5 illustrates the main points of the all relevant policies strategy. First, the strategy errs on the side of gathering as much relevant policy information as possible before ever disclosing any credential. For example, once C_3 is unlocked, theoretically the client could disclose C_3 and immediately obtain access to ***R***. However, the client does not know if there are hidden policies that will be disclosed only after it discloses C_3; thus the client does not know that disclosure of C_3 will immediately lead to obtaining the service. Further, from the client's point of view, perhaps C_3 is more sensitive than C_1 and C_2; it might be better not to disclose C_3 and instead to disclose C_1 and C_2, if possible, to gain access to ***R***. Thus once C_3 is unlocked, the client discloses additional policies rather than disclosing C_3 immediately. Only after all innermost unlocked policies are disclosed does the client determine that C_3 is the only credential it has whose disclosure will lead to satisfaction of an additional policy. In general, if the client had other unlocked credentials that satisfied some disclosed policy, it would be free to choose to satisfy that other policy instead of policy C_3 for ***R***. It could also choose to satisfy both policies with a single, larger credential disclosure message. This flexibility allows the client to use whatever intelligence it has at its disposal: it can examine and search potential proof trees using a breadth-first or depth-first approach, look for minimal satisfying sets of credentials, or use whatever heuristics it chooses. At the other extreme, it can use minimal cleverness and simply disclose every unlocked credential it has, in order to satisfy one more disclosed policy.

Theorem 2. *The* all relevant policies *strategy is safe and complete for monotonic propositional languages.*

Proof. Safety: Under the all relevant policies strategy, before a credential ***C*** can be put into the set *DisclosedLocalCredentials*, ***C*** must be unlocked by the credentials received from the other party. By the definition of "unlocked resource", there is an authorized path to ***C***. Therefore, the disclosure of ***C*** in the all relevant policies strategy is safe. Similarly, a policy ***P*** from the policy graph of a local credential ***C*** is put into the set *DisclosedLocalPolicies* only if ***P*** is an innermost unlocked policy in ***C***'s graph, in which case there is an authorized path to ***P***. Therefore, the disclosure of ***P*** is also safe.

Completeness: If there is a safe credential disclosure sequence ***Seq*** terminated by the disclosure of ***R***, we need to prove that when the negotiation stops, ***R*** is disclosed. We prove this by induction on *n*, the total number of credentials and services possessed by the two parties. By Lemma 1, we can assume ***Seq*** does not contain any

```
ARP_Message_Handler (Credentials, Policies)
 DisclosedRemoteCredentials = DisclosedRemoteCredentials ∪ Credentials.
 If service R is unlocked by DisclosedRemoteCredentials,   //  This check applies only to the server.
   Then exit the negotiation and grant the other party access to R.
 Else    // determine the next outgoing message.
   DisclosedRemotePolicies = DisclosedRemotePolicies ∪ Policies.
   PolicyDisclosure  = ∅.      // Create an initially empty set of policy disclosures to send to the other party.
   For each local credential $C_{local}$ that is described in some policy P in Policies   //  Remote policies describe local credentials.
      Add $C_{local}$ to LocalRelevantCredentials.
   For each local resource $R_{local}$ where $R_{local}$ is credential C ∈ LocalRelevantCredentials or $R_{local}$ is service R
      Let G be the policy graph for $R_{local}$.
      If $R_{local}$ is not unlocked by credentials in DisclosedRemoteCredentials,
        Then for each innermost unlocked policy P in G
             If ($R_{local}$, P) is not in DisclosedLocalPolicies, then add ($R_{local}$, P) to PolicyDisclosure.
 If PolicyDisclosure is nonempty,           // We have policy information to disclose.  Do not disclose any credentials.
   Then
     DisclosedLocalPolicies = DisclosedLocalPolicies ∪ PolicyDisclosure.
     Send the message (∅, PolicyDisclosure) to the other party.
 Else          // We must disclose credentials if we can satisfy any remote policy with them.
   UnlockedRelevantCredentials = the set of all local credentials that are not in DisclosedLocalCredentials,
                                are unlocked by DisclosedRemoteCredentials, and are in LocalRelevantCredentials.
   CredentialDisclosure = A subset U of UnlockedRelevantCredentials such that U ∪ DisclosedLocalCredentials
                                will satisfy at least one policy P in DisclosedRemotePolicies
                                that is not satisfied just by the disclosed credentials in DisclosedLocalCredentials.
   If CredentialDisclosure is not empty
      DisclosedLocalCredentials = DisclosedLocalCredentials ∪ CredentialDisclosure.
      Send the message (CredentialDisclosure, ∅) to the other party.
   Else                  // Have negotiations failed?
      If the last message received contained an empty set of disclosures,
         Then send a Failure message to the other party.
      Else
         Send the message (∅, ∅) to the other party.
End of ARP_Message_Handler.
```

Figure 6. The pseudo-code for the *all relevant policies* strategy.

syntactically irrelevant credentials. When $n = 1$, then ***R*** is the only credential or service possessed by the two parties. If ***R*** is freely available, the negotiation succeeds immediately; otherwise it fails. Either way, the theorem holds.

Assume when $n = k$, the all relevant policies strategy is complete. When $n = k+1$, let ***E*** be the set of all policies held by the two parties.

First suppose that the negotiation fails without disclosing any credentials. In that case, the negotiation participants send only policy disclosure messages until the negotiation stops. Further, once the two parties ran out of policies to disclose, they did not disclose any credentials. But by Proposition 3, in any sequence of safe disclosures terminated by **R**, the first disclosed credential is freely available. Further, by Lemma 1, there must be a safe sequence terminated by **R** in which the first disclosed credential is relevant. That means no party's freely available credentials can be used to satisfy any relevant credentials' innermost unlocked policies, and the theorem holds.

Otherwise, suppose credential ***C*** is the first credential to be disclosed during the negotiation. By Proposition 3, ***C*** must be freely available. Let us replace every occurrence of ***C*** in the bodies of all policies by *true*. We call the new set of policies ***E′***. Obviously, ***E*** has a sequence to unlock ***R*** if and only if ***E′*** has one. Since in ***E′*** there are only k resources possessed by the two parties, by the induction hypothesis, at the end of a negotiation using ***E′*** and the all relevant policies strategy, ***R*** is disclosed. Suppose that ***Seq1*** is the resulting disclosure sequence, and let us watch the strategy as it negotiates using ***E*** and creates a new disclosure sequence ***Seq2***. We assume that the details not specified in the pseudocode are handled in the same manner during the runs for ***Seq1*** and ***Seq2***.

Suppose that the first point in which ***Seq1*** and ***Seq2*** differ is the messages M_1 and M_2, respectively. The difference can only occur because an innermost unlocked policy in which ***C*** appears has become relevant and been disclosed in $\boldsymbol{M_2}$, while that same innermost unlocked policy ***P*** (but with ***C*** replaced by *true*) has become relevant and either been disclosed in $\boldsymbol{M_2}$, or if ***P*** is already satisfied due to the replacement of ***C*** by *true*, not disclosed at all (in that case either a policy node that is one of ***P***'s descendants will be disclosed, or if all unlocked innermost policies from all relevant policy graphs have already been disclosed, then a credential disclosure message containing the credential at the graph's sink will be sent). If the immediately following messages disclose only policies, those messages will be identical in ***Seq1*** and ***Seq2***. The next difference will not come until the next credential disclosure message. In that message, the all relevant policies strategy can choose to disclose ***C*** in ***Seq2***, while ***C*** will never be disclosed in ***Seq1***. We assume that the strategy does disclose ***C*** at this point (the pseudocode does not specify that level of detail).

After ***C*** has been disclosed in ***Seq2***, every policy that is subsequently disclosed in ***Seq1*** will also be disclosed in ***Seq2***, although the disclosure will be delayed to a later round of policy disclosure messages if the policy's disclosure depends directly or indirectly on ***C***. Even the descendants of ***P*** that are disclosed in ***Seq1*** will eventually be disclosed in ***Seq2***. Similarly, every credential that is disclosed in ***Seq1*** will also be disclosed in ***Seq2***, although it will be disclosed one round of credential disclosures later in ***Seq2***, if its disclosure directly or indirectly depends on the disclosure of ***C***. We conclude that if ***R*** is disclosed in ***Seq1***, then ***R*** will also be disclosed at the end of ***Seq2***. ♦

Proposition 5. *For monotonic propositional languages, the* all relevant policies *strategy requires $2c + p + 2$ messages in the worst case, where c is the total number of credentials possessed by the two parties and p is the total number of policies at both parties. If each credential and description has length 1, then the sum of the lengths of all messages sent during negotiation is bounded by $c + p_1$, where p_1 is the total size of all policies possessed by the two parties.*

Proof. The messages exchanged during a negotiation using the all relevant policies strategy can be divided into four categories:

1. Messages containing only policies: since a policy is disclosed at most once during the negotiation, the number of such messages will not exceed *p*.
2. Messages containing only credentials: since a credential is disclosed at most once during the negotiation, the number of such messages will not exceed *c*.
3. Empty messages in the form (∅, ∅): according to the all relevant policies strategy, if an empty message does not cause the other party to have nothing to disclose, then the other party will disclose some credentials in the next message instead of disclosing a policy. Therefore, including the last empty message which may cause negotiation failure, there are no more than $c+1$ empty messages.
4. Failure message: there is at most one failure message during the negotiation.

Therefore, the total number of messages will not exceed $2c+p+2$.

In the worst case, all the credentials and policies are disclosed. Since each credential or policy is disclosed at most once, the total length of all messages sent during the negotiation will be no more than $c+p_1$. Here we assume the size of empty messages and Failure messages is 0. ♦

Theorem 3. *Suppose that we remove the line of code in the relevant credentials set strategy that halts negotiations once **R** is unlocked. Then the set of credentials disclosed by the* all relevant policies *strategy is always a subset of the set of credentials disclosed by the* relevant credentials set *strategy.* ♦

Proof. We call the new strategy RCS2. The proposition holds in this special case because the two strategies visit the exact same policy nodes. The proof proceeds by induction on *n*, the number of credentials and services owned by the two parties. If $n = 1$, then ***R*** is the only resource and the negotiation will succeed or fail immediately, so the theorem holds.

Assume that the theorem holds for $n = k$, and now consider the case where $n = k + 1$. Suppose that ***C*** is the first credential disclosed by the all relevant policies strategy (ARP). Then by Proposition 3, ***C*** must be freely available. Replace all occurrences of ***C*** by the special prepositional constant *true* in the bodies of all the policies of the two parties, and the result is a set of policy graphs that no longer include a policy graph for ***C***. The revised graphs will have a safe disclosure sequence for ***R*** if and only if the original policies did. By the induction hypothesis, we can run ARP and RCS2 on the revised policies, and all credential disclosures made by ARP are also made by RCS2. As in the proofs of Theorems 1 and 2, we can turn the disclosure sequences for the revised policy bodies into disclosure sequences for the original bodies. It remains to show that ***C*** will occur in the revised disclosure sequence for RCS2 and the original policy bodies.

Because no credential disclosures are made in ARP before ***C*** is disclosed, ***C*** must appear in a freely available unsatisfied policy $\boldsymbol{P_1}$ for a credential $\boldsymbol{C_1}$ that is mentioned in a freely available unsatisfied policy $\mathbf{P}_2$ for a credential $\boldsymbol{C_2}$ that—and so on, to a credential $\boldsymbol{C_n}$ that is mentioned in a freely available unsatisfied policy $\boldsymbol{P_n}$ for resource ***R***.

Because $\boldsymbol{P_n}$ is freely available, unsatisfied, and immediately relevant to gaining access to $\boldsymbol{R}$, RCS2 will request all the credentials in $\boldsymbol{P_n}$ in the first message of negotiations. Having received a request for all the credentials in $\boldsymbol{P_n}$, the RCS2 strategy will request all the credentials in $\boldsymbol{P_{n-1}}$ in the next message, even if it also discloses enough credentials in the same message to satisfy $\boldsymbol{P_n}$. And so on: after the *n*th message, RCS2 will have requested all the credentials that appear in $\boldsymbol{P_1}$, including $\boldsymbol{C}$. RCS2 discloses every unlocked requested credential, and $\boldsymbol{C}$ is freely available, so RCS2 will disclose $\boldsymbol{C}$ in the next message.

The theorem does not hold for the original relevant credentials set strategy (RCS) because RCS finds the shortest path through the policy graphs that leads to disclosure of **R**. For example, suppose that the sole unlocked and unsatisfied policy remaining in **R**'s graph is $\mathbf{C_1} \vee \mathbf{C_2}$. Suppose that $\mathbf{C_1}$ is already unlocked, but $\mathbf{C_2}$ is locked and has not been mentioned previously. RCS will disclose $\mathbf{C_1}$ immediately, thus ending negotiations before the graph for $\mathbf{C_2}$ can be explored. RCS2 will disclose $\mathbf{C_1}$ and go on to explore the graph for $\mathbf{C_2}$. ARP will explore the graph for $\mathbf{C_2}$ before disclosing anything, and may eventually decide to disclose only one of $\mathbf{C_1}$ and $\mathbf{C_2}$. ♦

In general, the all relevant policies strategy may actually disclose more credentials than the relevant credentials set strategy. This is because the all relevant policies strategy can look at all the unlocked policies, and choose which one to satisfy first, e.g., which branch to take out of a node. If that branch eventually leads to additional policies which cannot be satisfied, then the choice was an unlucky one and the disclosures required to travel in that direction may have been completely wasted, as the negotiating party will have to backtrack and follow a different branch out of one of the previously satisfied policy nodes. The possibility of hidden policies means that a negotiating party can never be entirely sure of the wisest course of action, until after the fact.

We could lessen this problem by having the all relevant policies strategy tell the other party whether additional locked policies apply to a credential. Such information could be very helpful to the other party in deciding exactly which subset of *Candidates* to disclose. For example, a party might choose to satisfy those policies that would give immediate access to new credentials or the service, rather than just giving access to another layer of policy.

6. Technology transfer

We have implemented the relevant credentials set strategy and the all relevant policies strategy in Java. We plan to integrate these strategies into a system based on a trust negotiation framework we built earlier at IBM in collaboration with Dr. William Winsborough [16]. The framework currently does not support sensitive policy information. It also does not exchange policy information during trust negotiation; instead it utilizes a naive strategy of disclosing all unlocked credentials at each stage of the negotiation. The components we have developed can serve as a basis for extending the framework to manage sensitive policy information during trust negotiation.

The trust negotiation framework is written in Java. The framework's architecture includes a security agent that manages credentials and access control policies that govern disclosure of those credentials. The framework utilizes the Trust Establishment Package that was developed at IBM's Haifa Research Center [9], available at IBM's alphaWorks web site (see http://www.alphaworks.ibm.com/).

The trust negotiation framework was integrated into a web-based demonstration prototype. The prototype architecture is built around a standard web browser and web server, with security agents managing credentials and access control policies for the client and server. The various system components communicate via HTTP. Browser requests are routed through a client security agent that manages trust negotiations on behalf of the client. A server security agent negotiates trust with clients on behalf of the web server before granting access to web resources.

In the future, information regarding the integration of this work into a system that extends the trust negotiation framework can be obtained at either http://drl.cs.uiuc.edu/security/ or http://www.cs.byu.edu/~seamons/.

7. Related work

Credential-based authentication and authorization systems fall into three groups: identity-based, property-based, and capability-based. Originally, public key certificates, such as X.509 [18] and PGP [19], simply bound keys to names, and X.509 v. 3 certificates later extended this binding to general properties (attributes). Such certificates form the foundation of identity-based systems, which authenticate an entity's identity or name and use it as the basis for authorization. Identity is not a useful basis for our aim of establishing trust among strangers. Bina et al. [2] introduced our digital credentials to allow the binding of arbitrary attributes and support trust negotiation between strangers. Systems have emerged that use these attribute-describing credentials to manage trust in decentralized, distributed systems [9][15][17]. This paper extends our earlier work by providing support to gradually establish trust before allowing access to a credential, policy, or service, as well as offering new strategies for gradual and immediate trust establishment.

Johnston et al. [11] use attribute certificates (property-based credentials) and use-condition certificates (policy assertions) for access control. Use-condition certificates enable multiple, distributed stakeholders to share control over access to resources. In their architecture, the policy evaluation engine retrieves the certificates associated with a user in order to determine if all the use conditions are met. Their work could be extended using our approach to protect sensitive certificates.

The Trust Establishment Project at the IBM Haifa Research Laboratory [9] has developed a system for establishing trust between strangers according to policies that specify constraints on the contents of public-key certificates. Servers can use a collector to gather supporting credentials from issuer sites. Each credential contains a reference to the site associated with the issuer. That site serves as the starting point for a collector-controlled search for relevant supporting credentials. Security agents in our work could adopt the collector feature, and we could use their policy definition language. Their work could be extended using our approach to protect sensitive credentials and gradually establish trust.

Capability-based systems manage delegation of authority for a particular application. Capability-based systems are not designed to establish trust between strangers, since clients are assumed to possess credentials that represent authorization of specific actions with the application server. In the capability-based KeyNote system of Blaze et al. [3][4], a credential describes the conditions under which one principal authorizes actions requested by other principals. KeyNote policies delegate authority on behalf of the associated application to otherwise untrusted parties. KeyNote credentials express delegation in terms of actions that are relevant to a given application. KeyNote policies do not interpret the meaning of credentials for the application. This is unlike policies designed for use with attribute-describing credentials, which typically derive roles from credential attributes. The IETF Simple Public Key Infrastructure [14] uses a similar approach to that of KeyNote by embedding authorizations directly in certificates.

The P3P standard [12] defined by W3C focuses on negotiating the disclosure of a user's sensitive private information based on the privacy practices of the server. Trust negotiation is generalized to base disclosure on any server property of interest to the client that can be represented in a credential. The work on trust negotiation focuses on certified properties of the credential holder while P3P is based on data submitted by the client that are claims the client makes about itself. Support for both kinds of information in trust negotiation is warranted.

SSL [8], the predominant credential-exchange mechanism in use on the web today, and its successor TLS [6][7], support credential exchange during client and server authentication. The protocol is suited for identity-based credentials and would need extension to make it adaptable to property-based credentials. Needed additions include protection for sensitive server credentials and a way for the client to explain its policies to the server.

Islam et al. [10] show how to control downloaded executable content using policy graphs. Their definition of policy graphs is different from ours, and their information that is akin to policies is not mobile, thus has no access control mechanism. Their system assumes that all the appropriate credentials accompany downloaded content. Their work could be extended using our approach to mobile policies and negotiation.

8. Conclusions and future work

This paper presented two trust negotiation strategies that support limited disclosure of credentials and access control policies during automated trust negotiation. The *relevant credentials set* strategy eagerly discloses credentials, but is more cautious in disclosing policy information. The *all relevant policies* strategy eagerly discloses policies, but allows each party to be more careful about which credentials to disclose. Both strategies are safe and complete and have low communication costs.

Previous work in trust negotiation assumed that participants were using the exact same negotiation strategy. The all relevant policies strategy provides flexibility in how each participant selects the policy it will satisfy first. Further work in negotiation strategies is needed to explore the limits on when and how negotiation participants can use different negotiation strategies.

Another area for potential future exploration is tighter bounds on the meaning of the term "relevant". For example, the all relevant policies strategy allows a party to send credentials to satisfy a policy node ***P***, even if the credential to whose graph ***P*** belongs has already been disclosed. More generally, the two strategies presented in this paper do not include any mechanism for deciding or declaring that certain policies or credentials are no longer relevant. Even the definition of "innermost unlocked policy" used in this paper may cause a party to spend time checking the satisfaction of a policy ***P*** that is no longer relevant for finding an authorized path to the sink of a credential's policy graph. For example, ***P*** may be irrelevant due to the presence of other authorized paths that already go deeper into the graph than ***P***.

Another potential area for future exploration is the use of policy languages that violate our assumptions regarding monotonicity and/or satisfiability. The most important of these are languages that can express constraints on the times of day when access is permitted. In such a language, a party might satisfy a policy now, but fail to satisfy the policy later on.

Finally, the strategies presented in this paper do not guarantee disclosure of a minimal set of credentials, under any reasonable definition of "minimal". Such a guarantee is impossible in the general case, due to the presence of hidden policies. Even for policy graphs containing just two nodes, we can show that it is an NP-complete problem to minimize the size of the set of disclosed credentials for a negotiation.

9. Acknowledgements

This research was supported by DARPA through AFRL contract number F30602-98-C-0222 and by DARPA through AFRL contract number F30602-97-C-0336 to NAI Labs. The authors would like to thank William Winsborough, Vicki Jones, and Sushil Jajodia for their helpful discussions on trust negotiation. The authors also thank the reviewers for their extremely detailed feedback, which directly helped improve the quality of the paper.

10. References

[1] Krzysztof R. Apt, David S. Warren, and Mirek Truszczynski (editors), *The Logic Programming Paradigm: A 25-Year Perspective*, Springer-Verlag, 1999.

[2] E. Bina, V. Jones, R. McCool and M. Winslett, "Secure Access to Data Over the Internet," *ACM/IEEE International Conference on Parallel and Distributed Information Systems*, Austin, Texas, September 1994.

[3] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The KeyNote Trust Management System Version 2," Internet Draft RFC 2704, September 1999.

[4] M. Blaze, J. Feigenbaum, and A. D. Keromytis, "KeyNote: Trust Management for Public-Key Infrastructures," Security Protocols, 6th International Workshop, Cambridge UK, 1998.

[5] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," *IEEE Symposium on Security and Privacy*, Oakland, May 1996.

[6] T. Dierks, C. Allen, "The TLS Protocol Version 1.0," draft-ietf-tls-protocol-06.txt, Nov. 12, 1998.

[7] S. Farrell, "TLS Extensions for Attribute Certificate Based Authorization," draft-ietf-tls-attr-cert-01.txt, Aug. 20, 1998.

[8] A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol," Netscape Communications Corp., Nov. 18, 1996.

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

[10] N. Islam, R. Anand, T. Jaeger, and J. R. Rao. "A Flexible Security System for Using Internet Content." *IEEE Software*, Vol. 14, No. 5, September - October 1997.

[11] W. Johnston, S. Mudumbai, and M. Thompson, "Authorization and Attribute Certificates for Widely Distributed Access Control," *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1998.

[12] "Platform for Privacy Preferences (P3P) Specification," W3C, http://www.w3.org/TR/WD-P3P/Overview.html.

[13] B. Schneier, Applied Cryptography, John Wiley and Sons, Inc., second edition, 1996.

[14] Simple Public Key Infrastructure (SPKI), http://www.ietf.org/html.charters/spki-charter.html.

[15] W. Winsborough, K. Seamons, and V. Jones, "Automated Trust Negotiation," *DARPA Information Survivability Conference and Exposition*, Hilton Head, January 2000.

[16] W. Winsborough, K. Seamons, and V. Jones, "Automated Trust Negotiation," submitted for journal publication, April 2000, currently available at http://www.csc.ncsu.edu/faculty/vej/atn.ps.

[17] M. Winslett, N. Ching, V. Jones, and I. Slepchin, "Using Digital Credentials on the World-Wide Web," *Journal of Computer Security*, **5**, 1997, 255-267.

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

[19] P. Zimmerman, *PGP User's Guide*, MIT Press, 1994.