# Automated Trust Negotiation Over the Internet

**Ting Yu*** **Marianne Winslett†** **Kent Seamons‡**

## Abstract

The Internet provides an environment where two parties, who are virtually strangers to each other, can make connections and do business together. Before any actual business starts, a certain level of trust should be established. Each party should make sure that the other one is qualified and can be trusted for the ongoing transaction. Automated trust negotiation is a new approach to establishing trust between strangers through the exchange of digital credentials and the use of access control policies that specify what combinations of credentials a stranger must disclose in order to gain access to each local service or credential. In this paper, we give an overview of automated trust negotiation, including the architecture of automated trust negotiation, trust negotiation protocols and negotiation strategies. Then we briefly discuss various issues in automated trust negotiation, such as strategy interoperability, access control policy protection and privacy protection.

## 1 Introduction

With billions of users on the Internet, most interactions will occur between strangers, i.e., entities that have no pre-existing knowledge about each other and may not share a common security domain. In order for strangers to conduct secure transactions, a sufficient level of mutual trust must be established. For this purpose, it is the *properties* (e.g., employment status, citizenship, group membership) of the participants will be most relevant, instead of their *identities* (e.g., their social security number, fingerprint, institutional tax ID). Traditional security approaches based on identity require a new client to pre-register with the service, in order to obtain a local login, capability, or credential before requesting service; but the same problem arises when the client needs to prove on-line that she is eligible to register with the service. E-commerce needs a more scalable approach that allows automatic on-line pre-registration, or does away entirely with the need for pre-registration. We believe that automated trust establishment is such a solution.

With automated trust establishment, strangers establish trust by exchanging *digital credentials*, the electronic analogues of paper credentials that people carry in their wallets: 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 also contains the public key of the credential owner. Digital credentials can be implemented using, for example, X.509 [5] certificates.

While some resources are freely accessible to all, many require protection from unauthorized access. Access control policies can be used for a wide variety of "protected" resources. Since digital credentials themselves can contain sensitive information, their disclosure will often also be governed by access control policies. For example, suppose that a landscape designer wishes to order plants from Champaign Prairie Nursery (CPN). She fills out an order form on the web, checking an order form box to indicate that she wishes to be exempt from sales tax. Upon receipt of the order, CPN will want to see a valid credit card or her account credential issued by CPN, and a current reseller's license. The designer has no account with CPN, but she does have a digital credit card. She is willing to show her reseller's license to anyone, but she will only show her credit card to members of the Better Business Bureau. Therefore, when protected credentials are involved, a more complex procedure needs to be adopted to establish trust through negotiation.

## 2 Related Work

Credential-based authentication and authorization systems fall into three groups: identity-based,

*University of Illinois, Urbana-Champaign, tingyu@cs.uiuc.edu.

†University of Illinois, Urbana-Champaign, winslett@cs.uiuc.edu.

‡Brigham Young Univeristy, seamons@cs.byu.edu.

property-based, and capability-based. Originally, public key certificates, such as X.509 [5] and PGP [11], simply bound keys to names, and X.509 v.3 certificates later extended this binding to general properties.

Systems have emerged that use property-based credentials to manage trust in decentralized, distributed systems [4, 6]. Johnson et al. [6] use attribute credentials and policy assertions for access control. policy assertions 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 to determine if the use conditions are met. Their work could use automated trust negotiation to protect sensitive credentials.

The Trust Establishment Project at the IBM Haifa Research Laboratory [4] 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. Their work could use our approach to protect sensitive credentials and gradually establish trust.

Bonatti el at. [3] introduced a uniform framework and model to regulate service access and information release over the Internet. Their framework is composed of a language with formal semantics and a policy filtering mechanism. Our work can be integrated with the framework proposed in [3].

The first trust negotiation strategies proposed included a naive strategy that discloses credentials as soon as they are unlocked and discloses no policy information, as well as a strategy that discloses credentials only after each party determines that trust can be established, based on reviewing the other party's policies [9].

# 3 Trust Negotiation

In our approach to automated trust establishment, trust is established incrementally by exchanging credentials and requests for credentials, an iterative process known as *trust negotiation.* While a *trust negotiation protocol* defines the ordering of messages and the type of information messages will contain, a *trust negotiation strategy* controls the exact content of the messages, i.e., which credentials to disclose, when to disclose them, and when to terminate a negotiation. Figure 1 introduces our *TrustBuilder* architecture for trust negotiation. Each participant in the negotiation has an associated security agent (SA) that manages the negotiation. The security agent mediates access to local protected *resources*, i.e., services and credentials. We say a credential or access control policy is *disclosed* if it has been sent to the other party in the negotiation, and that a service is disclosed if the other party is given access to it. Disclosure of protected resources is governed by access control policies. During a negotiation, the security agent uses a local negotiation strategy to determine what local resources to disclose next, and to accept new disclosures from the other party.

The architecture in figure 1 supports a single protocol for establishing trust, and assumes there will be a variety of negotiation strategies that must be supported. All trust negotiation strategies share the goal of building trust through an exchange of digital credentials that leads to obtaining access to a protected resource. Once enough trust has been established that a particular credential can be disclosed to the other party, a local negotiation strategy must determine whether the credential is relevant to the current stage of the negotiation. Different negotiation strategies will use different definitions of relevance, involving tradeoffs between computational cost, the length of the negotiation, and the number of disclosures.

# 4 Access Control Policies

We assume that the information contained in access control policies (*policies*, for short) and credentials can be expressed as finite sets of statements in a formal language with a well-defined semantics. XML or logic programming languages with appropriate semantics may be suitable languages in practice [4, 1].

In this paper we will treat credentials and services as propositional symbols. Each of these resources has exactly one access control policy, of the form $C \leftarrow F_C(C_1, \ldots, C_k)$, where $F_C(C_1, \ldots, C_k)$ is a Boolean expression involving only credentials $C_1, \ldots, C_k$ that the other party may possess, Boolean constants *true* and $false$, the Boolean operators $\vee$ and $\wedge$, and parentheses as needed. $C_i$ is satisfied if and only if the other party has disclosed credential $C_i$. Resource $C$ is *unlocked* if its access control policy is satisfied by the set of credentials disclosed by the other party. A resource is *unprotected* if its policy is always satisfied. The *denial policy* $C \leftarrow false$ means that either the party does not possess $C$, or else will not disclose $C$ under any circumstances. A party implicitly has a denial policy for each credential it does not possess. If the disclosure of a set $S$ of credentials satisfies resource $R$'s policy, then we say $S$ is a *solution set* for $R$. Further, if none of $S$'s proper subsets is a solution set for $R$, we say $S$ is a *minimal solution*

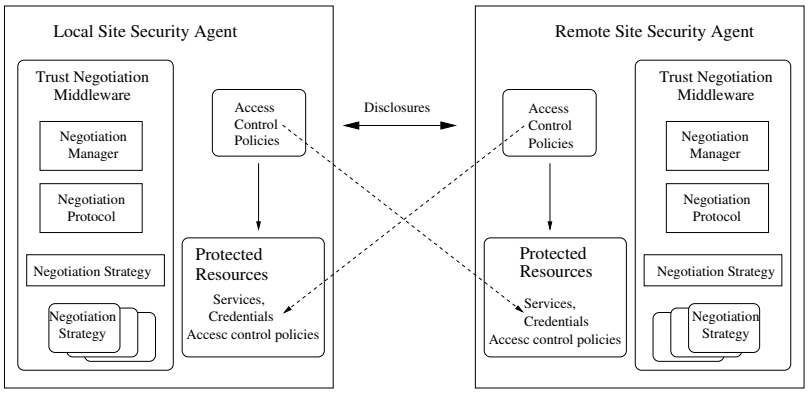


Figure 1: An architecture for automated trust negotiation. A security agent that manages local protected resources and their associated access control policies represents each negotiation participant. A access control policy specifies what resources the other party needs to disclose in order to gain access to a local resource, as indicated by the dotted lines in the figure. Trust negotiation middleware enables negotiation strategy interoperability.

*set* for $R$.

Given sequence $G = (C_1, \ldots, C_n)$ of disclosures of protected resources, if each $C_i$ is unlocked at the time it is disclosed, $1 \leq i \leq n$, then we say $G$ is a *safe disclosure sequence*. The goal of trust negotiation is to find a safe disclosure sequence $G = (C_1, \ldots, C_n = R)$, where $R$ is the resource to which access was originally requested. When this happens, we say that trust negotiation succeeds. Figure 2 shows a safe disclosure sequence for the landscape designer's purchase from CPN discussed earlier. Note that this example, and our paper in general, elide the details of verification of credential contents and authentication of ownership that will be part of any implementation of TrustBuilder.

# 5 TrustBuilder Protocol and Interoperable Strategies

A trust negotiation protocol defines the ordering of messages and the type of information messages will contain. Previous work, which focuses on negotiation strategies, has not explicitly proposed any trust negotiation protocols. Instead, protocols are defined implicitly by the way each negotiation strategy works. This is one reason why no two different previously proposed strategies can interoperate – their underlying protocols are totally different.

We remedy this problem by defining a simple protocol for TrustBuilder. Formally, a *message* in the TrustBuilder protocol is a set $\{R_1, \ldots, R_k\}$ where each $R_i$ is a disclosure of a local credential, a local policy, or a local resource. When a message is the empty set $\emptyset$, we also call it a *failure message*. Further, to guarantee the safety and timely termination of trust negotiation no matter what policies and credentials the parties possess, the TrustBuilder protocol requires the negotiation strategies used with it to enforce the following three conditions throughout negotiations:

1. If a message contains a denial policy disclosure $C \leftarrow false$, then $C$ must appear in a previously disclosed policy.

2. A credential or policy can be disclosed at most once.

3. Every disclosure must be safe.

Before the negotiation starts, the client sends the original resource request message to the server indicating its request to access resource $R$. This request triggers the negotiation, and the server invokes its local security agent with the call TrustBuilder_handle_disclosure_message$(\emptyset, R)$. Then the client and server exchange messages until either the service $R$ is disclosed by the server or one party sends a failure message. The whole negotiation process is shown in figure 3.

Because of the simplicity of TrustBuilder protocol, it encompasses large amount of strategies with various properties. More important, it enables us to study the interoperability between strategies in a systematic way. In the remainder of this paper, unless otherwise noted, we discuss only strategies that can

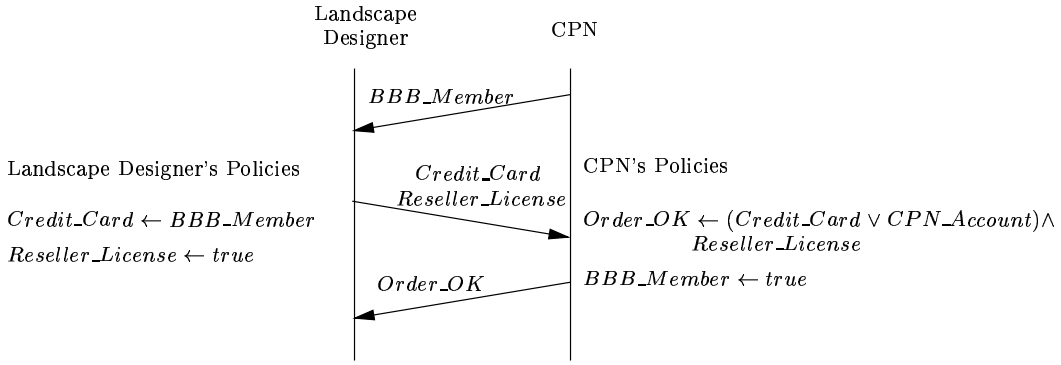


Figure 2: An example of access control policies and a safe disclosure sequence which establishes trust between the server and the client.

be called from the TrustBuilder protocol and satisfy the three conditions above.

Formally, a negotiation strategy is a function. Its input is the current state of the negotiation (i.e., disclosed credentials and policies and local resources and policies) and its output is the next set of disclosures a party should send to the other negotiation participant. Given two strategies $f_A$ and $f_B$, we say they are *compatible* if whenever there exists a safe disclosure sequence for a party $\mathcal{P}_A$ to obtain access to a resource owned by party $\mathcal{P}_B$, the trust negotiation will succeed when $\mathcal{P}_A$ uses $f_A$ and $\mathcal{P}_B$ uses $f_B$. If $f_A = f_B$, then we say that $f_A$ is *self-compatible*. A *strategy family* is a set $\mathcal{F}$ of mutually compatible strategies.

One obvious advantage of strategy families is that a security agent (SA) can choose strategies based on its needs without worrying about interoperability, as long as it negotiates with other SAs that use strategies from the same family. As another advantage under certain conditions, an SA does not need to stick to a fixed strategy during the entire negotiation process. It can adopt different strategies from the family in different phases of the negotiation. For example, during the early phase, since the trust between two parties is very limited, an SA may adopt a cautious strategy for disclosing credentials. When a certain level of trust has been established, in order to accelerate the negotiation, the SA may adopt a less cautious strategy. However, without the closure property, a family may not be large enough for practical use. As an extreme example, given any self-compatible strategy $f$, $\{f\}$ is a strategy family. The closure property guarantees the maximality of a strategy family.

# 6 Disclosure Tree Strategy Family

In this section, we identify a closed strategy family that follows TrustBuilder protocol. Given two compatible strategies $f_1$ and $f_2$, intuitively, when a strategy sends a message $m$, $m$ should contain enough information to guide the other party to carry on the negotiation. Formally, we use the concept *disclosure tree* to describe the progress of a negotiation and to characterize the behavior of different strategies.

**Definition 6.1.** *Suppose $R$ is a resource to which access has been requested. A* disclosure tree *for $R$ is a finite tree satisfying the following conditions:*

1. *The root represents the requested service $R$.*
2. *Except for the root, each node represents a credential. When the context is clear, we refer to a node by the name of the credential it represents.*
3. *The children of a node $C$ form a minimal solution set for $C$.*

*When all the leaves of a disclosure tree $T$ are unprotected credentials, we say $T$ is a* full *disclosure tree.*

The concept of a disclosure tree is of importance because there is a natural mapping from a full disclosure tree to a safe disclosure sequence. Therefore, during the negotiation, theoretically one could determine whether a potential credential or policy disclosure is helpful by examining all the disclosure trees for $R$. Generally, the state of a trust negotiation can be represented by a set of disclosure trees, which is denoted as the current *view* of a trust negotiation. Details on how views are maintained can be found in [10]. Let $\mathcal{P}_A$ be a negotiation party and $T$ be a disclosure tree in the view of a negotiation. If one of the

```
TrustBuilder_handle_disclosure_message (m, R)
 Input: m is the last disclosure message received from the remote party.
      R is the resource to which the client originally requested access.
 TrustBuilder_check_for_termination(m, R). //Stop negotiating, if appropriate.
 TrustBuilder_next_message(m, R).
End of TrustBuilder_handle_disclosure_message.

TrustBuilder_next_message(m, R)
 // First, let the local strategy suggest what the next message should be.
 Let G be the disclosure message sequence so far.
 Let L be the local resources and policies.
 S_m = Local_strategy(G, L, R).
 // S_m contains the candidate messages the local strategy suggests.
 Choose any single message m' from S_m.
 Send m' to the remote party.
 TrustBuilder_check_for_termination(m', R). //Stop negotiating, if appropriate.
End of TrustBuilder_next_message.

TrustBuilder_check_for_termination(m, R)
 If m is the empty set ∅ and this is not the beginning of the negotiation,
      Then negotiations have failed. Stop negotiating and exit.
 If m contains the disclosure of R,
      Then negotiations have succeeded. Stop negotiating and exit.
End of TrustBuilder_check_for_termination.
```

Figure 3: Pseudocode for the TrustBuilder protocol. The negotiation starts by the client sending a service request message to the server. After rounds of disclosure message exchanges, either one party sends a failure message and ends the negotiation or finally the server grants the client access to the requested service.

leaf nodes of $T$ represents a credential requested by the other party, then we say $T$ is *evolvable* for $\mathcal{P}_A$ and that leaf node is an evolvable leaf. Evolvable leaves actually give a party guidance on the next message. Based on this idea, we have identified a strategy family called *disclosure tree strategy family* (DTS family) for short. Given a strategy $f$, there are two key requirements for $f$ to be in the DTS family. First, whenever $f$ sends a set of disclosures to the other party, it should make sure that there is at least one evolvable tree for the other party in the view of the current negotiation. Second, if the other party does not cooperate enough, i.e., if there are no evolvable trees for the local party in the current view, then $f$ may choose to terminate the negotiation with failure even though there may possibly be a successful negotiation. The first requirement guarantees the mutual compatibility between strategies in the DTS family while the second requirement guarantees the closeness of the family.

**Theorem 6.1.** *The DTS family is closed.* □

One advantage of DTS family is that as long as both parties uses strategies from the DTS family, they can switch between different practical strategies as often as they like, and trust negotiation will still succeed whenever possible.

# 7 Other Issues in Automated Trust Negotiation

## 7.1 Policy Graphs

Our discussion so far assumes that access control policies are freely accessible, i.e., they can be disclosed to others when requested. However, in complex e-commerce environments, policies may also contain sensitive information. For example, if a policy for online documents of a commercial project requires the requester to present an employee ID from either Microsoft or IBM, then by simply looking at the policy, a stranger may guess with high confidence that the project is probably a cooperation between the two companies, which may be sensitive business information. To protect such sensitive policies, we introduce hierarchies into policies definition and propose the model of policy graphs [7]. A policy graph is a direct acyclic graph with a single source and sink. The sink node represents a protected resource while all the other nodes represent a policy. A policy $P$ can be disclosed only if there is a path from the source to $P$ such that every node along the path, except possibly $P$, has been satisfied by current disclosed credentials. A policy graph can be viewed as a multi-level policy, which maps naturally to the situation where sensitive policies need to be protected.

### 7.2 Privacy Protection

Policy graphs are not only critical to protect sensitive polices. It is also useful in protecting sensitive attributes of credentials. A policy usually contains constraints on attributes a requested credentials. For example, an online shopping mall's policy for cigarette purchasing may require a customer to show his/her driver's license whose age attribute's value is over 18, i.e., $x.type = drivers\_license \wedge x.age > 18$. As we discussed earlier, if a customer has a driver's license credential that satisfies the above policy, then he/she will respond with the policy for the driver's license. The problem is, by getting the response, the other party will know that the customer's age is over 18 even though his/her driver's license has not been disclosed yet. Such information leakage is undesired if the customer considers his/her age information sensitive. One way to remedy this problem is to introduce *dynamic policy* into our trust negotiation architecture [8]. The basic idea is to dynamically transform a policy into a policy graph. After a party receives a policy, before the strategy engine starts to evaluates it, the security agent divides the single policy into several sub-policies and organizes them into a policy graph, such that constraints on sensitive attributes do not appear in the source node. The resulting policy graph, instead of the single policy, is evaluated by the strategy engine. Because of such a dynamic policy transformation, the remote party can not infer any information about sensitive attributes since the response from the local party is only for the source node of the policy graph which has constraints only on insensitive attributes.

## 8 Conclusion

Automated trust negotiation is a new approach to establishing trust between strangers through the exchange of digital credentials. After introducing the architecture for automated trust negotiation, we focus our discussion on interoperable strategies. We propose a simple negotiation protocol based on which a closed strategy family, the DTS family, is identified. In the latter part of the paper, we briefly discuss some other issues raised in our implementation of the automated trust negotiation framework, which include policy graphs and privacy protections.

## References


[1] K. R. Apt, D. S. Warren, and M. Truszczynski (editor). *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. In *Conference on Parallel and Distributed Information Systems*, September 1994.

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

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

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

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

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

[8] K. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting Privacy during On-line Trust Negotiation. In *submitted to Workshop on Privacy Enhancing Technologies*, San Francisco, CA, April 2002.

[9] W. Winsborough, K. Seamons, and V. Jones. Negotiating Disclosure of Sensitive Credentials. In *Second Conference on Security in Communication Networks*, Amalfi, Italy, September 1999.

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

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