Collaborative Automated Trust Negotiation in Peer-to-Peer Systems

Song Ye Fillia Makedon James Ford
Dartmouth Experimental Visualization Laboratory (DEVLAB)
Department of Computer Science, Dartmouth College
{yesong, makedon, jford}@cs.dartmouth.edu

Abstract

The increasing popularity of peer-to-peer systems has promoted the development of new techniques to support various kinds of business transactions. However, users are reluctant to conduct high value transactions over P2P systems due to the inherent untrustworthiness of peers. In this paper, we investigate building trust through automated trust negotiation, which is orthogonal to existing reputation-based approaches. Trust negotiation makes it possible to prove a peer satisfies certain trust requirements imposed by the ongoing business. We also introduce locally trusted third parties, which we demonstrate can effectively promote successful trust negotiations in peer-to-peer systems.

1. Introduction

Traditionally, computer systems have closed and centrally managed security domains. Every entity in such a system has one or more identities/roles associated with it, and its access to a resource is enforced by the system according to the resource access control policy (usually implemented as an access control list, or ACL). Trust is established based on each entity's identity, and thus a remote entity will not be able to interact with the local system and gain access to resources without a locally recognized identity. However, this traditional security model has been challenged by existing distributed applications, especially recently popular P2P systems.

Peer-to-peer (P2P) was originally used to refer to network protocols where all the nodes have the same role and there are no nodes with specific responsibilities to act as the administrators or supervisors of a network. With the evolution of Internet-based applications, the designation of P2P is currently used to identify a class of systems and applications that employ distributed resources to perform some function in a decentralized manner, where every participating node can act as both a client and a server. In many cases each node has different capabilities (processing power, shared files, connection bandwidth, *etc.*). With the wide deployment of variant kinds of computing devices including personal computers, PDAs, cell phones, P2P is increasingly attracting attention in both academic and industry research. P2P systems provide some obvious advantages: improved scalability by avoiding dependency on centralized servers, which are often points of failure of the system; elimination of the need for costly infrastructure by enabling direct communication among peers; and easily facilitation of resource aggregation, for example by harnessing available CPU cycles from a large number of peers[1] to provide massive processing power.

A typical P2P system allows mutually distrustful parties with conflicting interests to join or leave freely. Its open, anonymous and decentralized nature makes it extremely difficult to verify the validity of the resources offered by other peers. Furthermore, *free riders* [5, 13], which only want to use other peers' resources without contributing anything, have greatly compromised the fairness of most P2P systems and discouraged contributing peers from continuing to share resources. In general, peers are only interested in participating in resource sharing, both for providing resources and requesting them, with those peers they trust. These problems have led to the development of reputation systems (*e.g.* [8, 11, 4]) as a means to detect misbehavior and punish malicious peers, and also to encourage good ones.

In a reputation system, a peer makes decisions based on its own experience and other peer's recommendations. Even though reputation systems require some form of persistent node identification, such as a ver-

This work is supported by NSF award 0308229.

1 In this paper, "peer", "party", and "node" are used interchangeably, and all refer to an entity in a P2P system. Sometimes we use the names Alice, Bob, and Carol to designate examples of these entities.

ifiable real-world identity [11, 10], they are not suitable for critical tasks. For example, if a transaction requires Alice to disclose her credit card number to another party whom she does not know, she may not be able to adequately determine the risk involved based solely on the other party's reputation. She would benefit from a proper credential (which is not necessarily a part of each node's identification) issued by some real-world authority showing that the other party is trustworthy. In P2P systems, having only persistent identification and reputations is not enough to support conducting high value transactions. One option for addressing this lack is that peers build trust by exchanging digitally signed certificates.

In this paper we propose a trust negotiation scheme for P2P systems. In our scheme, peers build their trust relationship with each other by collaboratively exchanging their credentials. To the best of our knowledge, this is the first attempt to practically apply trust negotiation in P2P systems. Trust negotiation is orthogonal to existing reputation-based approaches, and thus can be used together with reputation systems to build trust in a P2P system. In Section 2 we introduce automated trust negotiation. Section 3 investigates automated trust negotiation in P2P systems. Section 4 discusses the relationship between reputation systems and trust negotiation. Section 5 gives some experimental results. Finally, Section 6 offers some concluding remarks.

2. Automated Trust Negotiation

Automated trust negotiation (ATN) [20, 23, 18, 14, 17, 16] is an approach to access control and authentication in open and distributed systems. ATN enables open resource access by assigning an access control policy to each resource that could be accessed by strangers. In contrast with the traditional approach, where only identities of the parties that can access a resource are listed, access control policies are used to describe the properties of these parties. Properties typically consist of digital credentials/certificates, which are essentially digital versions of paper credentials that people carry in their wallets today.

A *credential* is a digitally signed assertion by a *credential issuer* about the *credential owner* regarding one or more attributes about the owner. Public-key encryption (also called asymmetric encryption) mechanisms, such as RSA [12], can be used to implement credentials. Public-key encryption involves a pair of keys—a *public key* and a *private key*—associated with an entity. The owner can use them to authenticate its identity electronically or to sign or encrypt data. In a typical application, each public key is published, and the corresponding private key is kept secret. Data encrypted with the public key can be decrypted only with the corresponding private key and *vice versa*. A credential issuer signs a credential using her private key; the public key of this issuer can then be used to verify that the credential is really issued by the issuer. To assure that the claimed owner is really the owner of the credential, her public key is also included in the credential so that anyone can verify that she is the owner of the credential. For these reasons, digitally signed credentials are provably verifiable and unforgeable. It should be noted that while credentials are necessary for trust negotiation, the credential issuers are not required to be directly involved in the trust negotiation.

A reasonable person will not show her credit card to a stranger she meets; similarly, a party should not have to reveal its digital credentials to a stranger in the network unless necessary. A typical trust negotiation is initiated by a party who asks to access a resource belonging to another party. Since each party may have access control policies that the other needs to satisfy, trust is established incrementally through the exchange of digital credentials. For example, if Alice want to buy something in a store using her credit card, before she shows her credit card to the cashier, Bob, Alice may want to have a look at Bob's employee ID. The purpose of trust negotiation is to find a credential disclosure sequence $(C_1, C_2, \ldots, C_k, R)$, where R is the resource to which access was originally requested, such that when a credential C_i is disclosed, its access control policy has been satisfied by credentials disclosed by the other party. The detailed formalized model of trust negotiation will be discussed in Section 3.1.

In [18], two different categories of negotiation strategies are discussed: *eager* and *parsimonious*. An eager negotiation strategy allows flooding-style negotiation, as both sides disclose a credential to the other party as soon as the policy of that credential is satisfied. This guarantees that the negotiation will succeed if a successful negotiation is possible (this property is called *completeness*). No credential disclosure policies need to be disclosed if an eager strategy is used. Although an eager strategy can lead to negotiation success, if this is possible, in the minimum number of rounds, an important disadvantage is that some irrelevant credentials may be disclosed unnecessarily. On the other hand, a parsimonious strategy will not allow credential exchanges until both parties know that a successful negotiation is possible. In this approach, when an incoming request for a unresolved credential is received by a party, it prepares a counter-request according to its disclosure policy and sends it out as a reply. However, the parsimonious strategy is not complete and introduces the difficulty of deciding when the negotiation

should fail and stop.

Another trust negotiation strategy, called the Prudent Negotiation Strategy (PRUNES) is proposed in [20]. PRUNES is complete, which means the trust will be established through negotiation if it is allowed by the two parties' credential policies. PRUNES is based on backtracking, and it ensures that no irrelevant credentials are disclosed. However, because the credentials exchange does not happen until the two negotiation parties successfully find a way to exchange their credentials, an adversary can easily get all the access control policies from the other party without revealing any of its own real credentials and policies.

3. Trust Negotiation in P2P Systems

3.1. Trust Negotiation Model

We consider a P2P resource sharing system composed of n peers: $P_1, P_2, \ldots, P_n$. All peers are identified by their public-private key pairs, which are assumed to be unique. Every peer has some resources that could be accessed by other peers. For simplicity we assume that every peer serves only one unit of resource R, which can be accessed simultaneously by multiple peers. We assume there exist some trusted third parties which can issue credentials to peers showing that those peers have some properties. Unless specified, we consider only two-party resource sharing. In this paper, we *do not* consider the problem of *malicious peers*; that is we assume all peers follow their assigned protocols. Clearly malicious peers pose a significant threat to transactions, but measures to defeat them are beyond the scope of this paper.

We formalize the trust negotiation process using propositional symbols as in [20]. We assume that the information contained in access control policies and credentials can be expressed as finite sets of statements in a language so that all peers agree on the interpretation of a credential or policy. Formally, a credential disclosure policy (a.k.a. access control policy) for a resource R is defined as:

$$\mathcal{P}_R \longleftarrow F_R(C_1, C_2, \ldots, C_k)$$

where $C_1, C_2, \ldots, C_k$ are credentials from the other party, F_R is an expression involving these credentials, the boolean operators $\wedge$ and $\vee$, and parentheses if needed. C_i is satisfied if and only if the other party reveals credential C_i. Resource R can be access by the other party if $F_R(C_1, C_2, \ldots, C_k)$ is evaluated to *true*. For example, suppose Alice wants to buy some beer, R, from a store. The cashier, Bob, asks Alice to show her C_1^{Alice}="Credit Card" and a photo ID proving she is over 21, such as C_2^{Alice}="Driver Licence" or C_3^{Alice}="Passport", which can be written as $\mathcal{P}_R \longleftarrow C_1^{Alice} \wedge (C_2^{Alice} \vee C_3^{Alice})$. It should be noted that in the trust negotiation process, credentials are also treated as a special kind of resources: for a credential C, we have

$$\mathcal{P}_C \longleftarrow F_C(C_1, C_2, \ldots, C_k)$$

Continuing from the above example, if Alice also asks the cashier Bob to show something to prove he is the cashier, such as C_1^{Bob}="Employee ID", before she shows her credit card, we have $\mathcal{P}_{C_1^{Alice}} \longleftarrow C_1^{Bob}$.

A resource R (or credential C) is *unprotected* if its access control policy is always satisfied, that is if[2] $R \longleftarrow true$ or $C \longleftarrow true$. A resource R (or credential C) is *solved* if the corresponding F_R or F_C is satisfied given that the other party discloses some credentials. A solved resource or credential can be revealed to the other party. The *denial policy* $R \longleftarrow false$ means that the party does not have R or simply does not want to disclose it under any circumstances. A party implicitly has a denial policy for each resource/credential it does not possess. In our model, we assume for simplicity that all credential disclosure policies are fixed.

A credential disclosure sequence $\mathcal{S}$ is composed of a series of credentials $C_1, C_2, \ldots, C_{|\mathcal{S}|}$. If $\mathcal{S}$ is applied and satisfies resource R's access control policy, then $\mathcal{S}$ is a *solution* for R. If none of $\mathcal{S}$'s proper subsets is a solution for R, we say $\mathcal{S}$ is a *minimal solution* for R. Obviously, if $\mathcal{S}$ is a minimal solution for R, any $\mathcal{S}' \supseteq \mathcal{S}$ is also a solution for R.

Now we consider a simple example of trust negotiation between two peers P_1 and P_2, where P_1 is the owner of resource R and P_2 is a requestor of this resource. P_1's credentials are denoted by C^{P_1}, and P_2's by C^{P_2}. It should be noted that here we tagged credentials in the access policy with P_1 or P_2 to differentiate credentials and policies belonging to different peers. A notation like $C_1^{P_2}$ is used to indicate clearly that in this two-party scenario P_1 will require C_1 from P_2. The access policy rules of each peer are actually defined for all other peers. Suppose P_1's access policy for R is:

$$\begin{aligned}
R &\longleftarrow C_1^{P_2} \wedge C_2^{P_2} \\
C_1^{P_1} &\longleftarrow C_2^{P_2} \wedge C_3^{P_2} \\
C_2^{P_1} &\longleftarrow C_4^{P_2} \\
C_3^{P_1} &\longleftarrow C_4^{P_2} \wedge C_5^{P_2} \\
C_4^{P_1} &\longleftarrow true \\
C_5^{P_1} &\longleftarrow true
\end{aligned}$$

2 For simplicity, we will often omit $\mathcal{P}$ when $\mathcal{P}_R$ or $\mathcal{P}_C$ appears at the left side of "$\longleftarrow$".

while P_2's access policy is (here we only list the policy for negotiation-related credentials):

$$C_1^{P_2} \longleftarrow C_1^{P_1} \wedge C_2^{P_1}$$
$$C_2^{P_2} \longleftarrow C_2^{P_1} \vee C_3^{P_1}$$
$$C_3^{P_2} \longleftarrow true$$
$$C_4^{P_2} \longleftarrow true$$
$$C_5^{P_2} \longleftarrow true$$

There is at least one credential exchange sequence $\mathcal{S}_1$ leading to a successful negotiation, when both parties are using an eager negotiation strategy and they reveal all their credentials which are allowed by the access control policy (for clarity, we put curly brackets around the credentials disclosed by one peer as a unit):

$$\{C_3^{P_2}, C_4^{P_2}, C_5^{P_2}\}, \{C_2^{P_1}, C_3^{P_1}, C_4^{P_1}, C_5^{P_1}\},$$
$$\{C_2^{P_2}\}, \{C_1^{P_1}\}, \{C_1^{P_2}\}, \{R\}$$

Another possible sequence $\mathcal{S}_2$ is:

$$\{C_4^{P_2}\}, \{C_2^{P_1}\}, \{C_2^{P_2}, C_3^{P_2}\}, \{C_1^{P_1}\}, \{C_1^{P_2}\}, \{R\}$$

$\mathcal{S}_2$ only reveals necessary credentials, which are much less than the credentials $\mathcal{S}_1$ reveals. Here $\mathcal{S}_2$ is a minimal solution but $\mathcal{S}_1$ is not.

A successful credential exchange sequence is not guaranteed to exist. For example, if we change P_2's policy $C_4^{P_2} \longleftarrow true$ to $C_4^{P_2} \longleftarrow false$, P_2 can no longer get access to R. If we change P_1's policy $C_4^{P_1} \longleftarrow true$ to $C_4^{P_1} \longleftarrow C_4^{P_2}$ and P_1's policy $C_4^{P_2} \longleftarrow true$ to $C_4^{P_2} \longleftarrow C_4^{P_1}$, then P_1 and P_2 again can no longer succeed in trust negotiation either because there is a cyclic credential interdependency: $C_4^{P_1} \longleftarrow C_4^{P_2}$ and $C_4^{P_2} \longleftarrow C_4^{P_1}$, and their credential disclosure policies prevent each of them from disclosing its C_4 first.

3.2. Collaborative Trust Negotiation Among Peers

Although various existing negotiation strategies can guarantee that negotiation parties succeed whenever a successful negotiation is possible, the existence of cyclic interdependent policy rules (as in Section 3.1) can still cause a substantial number of failed trust negotiations in practice. Consider two peers P_1 and P_2 that have cyclic interdependent policy rules preventing them from achieving success in their trust negotiation. An observation is that a third party P_3 trusted by both peers can act as a mediator and disclose their credentials and policy rules to each other when appropriate, thus breaking the cyclic dependency and allowing trust negotiation to succeed. We call this *collaborative trust negotiation.*

A peer that can act as a trusted third party for all other peers would contradict the principles of a P2P system. However, a peer could reasonably act as a trusted third party for a limited number of peers, in much the same way that a reputation-based system relies on peers' knowledge of a limited set of other peers. We call such a peer a *locally trusted third party* (LTTP) for those that trust it. In contrast, a credential issuer, which is a trusted third party for all peers, is called a *globally trust third party* (GTTP)[3]. Here the concepts of LTTP and GTTP are based on credentials: a GTTP is able to issue credentials, while an LTTP is able to cache the disclosed credentials from other parties and reveal certain credentials when requested by their owner (we assume all disclosed credentials can be cached unless specified). Section 4 extends the functionality of LTTPs to allow them to issue credentials.

If two peers P_1 and P_2 successfully conduct one or more trust negotiations and they have exchanged and cached several credentials, they are LTTPs for each other, no matter which peer initiates the resource access request first. A peer P_i's LTTPs are recorded in a table denoted by LT_{P_i}. Each row of LT_{P_i} is indexed by the ID of peer P_j, and the content, $LT_{P_i}[P_j]$, is the set of cached credentials that have been disclosed during any previous trust negotiations. Suppose $\mathcal{S}$ is a successful credential disclosure sequence for P_1 and P_2, $\mathcal{C}_1$ is the set of all credentials P_1 has revealed, and $\mathcal{C}_2$ is the same for P_2; then $\mathcal{S} = \mathcal{C}_1 \cup \mathcal{C}_2$. After the negotiation, we have

$$LT_{P_1}[P_2] = LT_{P_1}[P_2] \cup \mathcal{C}_2$$
$$LT_{P_2}[P_1] = LT_{P_2}[P_1] \cup \mathcal{C}_1$$

Even if all peers in the system trust a GTTP and only a limited number of peers trust an LTTP, the latter is actually more important than a GTTP in some respects. This is because a GTTP is not a peer in the system, and thus can only issue credentials, while an LTTP is a peer, and can interact with those peers that trust it—in the process helping build trust through trust negotiation.

The following example shows that two peers P_1 and P_2 that are not able to succeed in trust negotiation due to a cyclic dependency in their credential disclosure policies can successfully finish a trust negotiation with the help of P_3, an LTTP. Consider the following example: if we have

3 Although a GTTP is globally trusted, it does not contradict the principles of a P2P system since it is an external, orthogonal service and not a peer in the system.

$$
\begin{array}{lll}
P_1: & R \longleftarrow C_1^{P_2} \wedge C_2^{P_2} & \\
& C_1^{P_1} \longleftarrow C_2^{P_2} \wedge C_3^{P_2} & \\
& C_2^{P_1} \longleftarrow C_4^{P_2} & \\
& C_3^{P_1} \longleftarrow C_4^{P_2} \wedge C_5^{P_2} & \\
& C_4^{P_1} \longleftarrow C_4^{P_2} & \\
& C_5^{P_1} \longleftarrow true &
\end{array}
\qquad
\begin{array}{ll}
P_2: C_1^{P_2} \longleftarrow C_1^{P_1} \wedge C_2^{P_1} & \\
C_2^{P_2} \longleftarrow C_2^{P_1} \vee C_3^{P_1} & \\
C_3^{P_2} \longleftarrow true & \\
C_4^{P_2} \longleftarrow C_4^{P_1} & \\
C_5^{P_2} \longleftarrow true &
\end{array}
$$

P_1 and P_2 can not succeed in their trust negotiation as they have cyclic interdependent policies $C_4^{P_1} \longleftarrow C_4^{P_2}$ and $C_4^{P_2} \longleftarrow C_4^{P_1}$. Neither P_1 nor P_2 wants to disclose its C_4 first. If both P_1 and P_2 have P_3 as an LTTP, and it has $\mathcal{C}_4^{P_1} \in LT_{P_3}[P_1]$ and $\mathcal{C}_4^{P_2} \in LT_{P_3}[P_2]$, P_1 and P_2 can send their related policies and credentials to P_3. P_3 thus acts as a mediator in P_1 and P_2's trust negotiation. After evaluating their policies, P_3 decides it can simultaneously disclose $C_4^{P_2}$ to P_1 and $C_4^{P_1}$ to P_2; thus P_1 and P_2 can successfully continue and finish their trust negotiation.

It should be noted that before two peers P_1 and P_2 ask help from an LTTP, they have to find a common LTTP trusted by both of them. One simple solution is for P_1 to send indexes of LT_{P_1}, which is a set of peer IDs denoted by $I_{LT_{P_1}}$, to P_2 and for P_2 to return a set $I_{LT_{P_1}} \cap I_{LT_{P_2}}$. If $I_{LT_{P_1}} \cap I_{LT_{P_2}} = \varnothing$, P_1 and P_2 cannot succeed in the trust negotiation.

Since the LTTP table may be considered sensitive information by a peer, no peer can be expected to freely disclose it to other peers. To solve this problem, we use convergent encryption [6], which allows sharing without compromising privacy, to find a common set of LTTPs. P_1 encrypts every item $I_i \in I_{LT_{P_1}}$ using $\text{Hash}(I_i)$ as the encryption key, and then sends the encrypted $I_{LT_{P_1}}$ to P_2. P_2 can reveal the content of I_i if and if only it possesses the same I_i in $I_{LT_{P_2}}$.

We have extended two existing negotiation strategies, the eager strategy [18] and PRUNES [20], to support LTTPs. Our enhanced strategies work as follows: first a peer P_i executes the initial strategy. If P_i decides the negotiation is unsuccessful, P_i sends its encrypted LT_{P_i} to the other party P_j. If the two parties find an empty LTTP set, they terminate the negotiation; otherwise they send their disclosed credentials and policies to all the available LTTPs and wait for their replies. Once P_i receives a nonempty reply from an LTTP, it can restart its negotiation strategy since the reply should contain at least a new disclosed credential and/or policy rule belonging to P_j.

3.3. Maintaining LTTP Tables

Although we have assumed that every peer can maintain a table of LTTPs to help in negotiation, there are practical limitations. A peer may not be able to record information for every peer it has negotiated with, especially when peers are running on mobile computing devices with very limited memory and processing power. Therefore we need to choose a proper LTTP table size and, because the size of the LTTP table is fixed, we need an algorithm to choose an old entry to remove from the table when the table is full and a new entry comes.

We consider the following possible algorithms:

- Random: An existing entry is randomly selected and removed from the LTTP table.
- Least Recently Added (LRA): The least recently added entry, namely the oldest one, is removed from the LTTP table. There should be only one such entry.
- Least Recently Used (LRU): The least recently used entry is removed from the LTTP table. Note that this is different from LRA: if an entry is added early on but is frequently used, it will not be removed under LRU (but would be removed under LRA). If multiple least recently used entries are found, the least recently added one is selected.
- Most Recently Added (MRA): The most recently added entry, *i.e.* the newest one, is removed from the LTTP table. There should be only one such entry.
- Most Recently Used (MRU): The most recently used entry is removed from the LTTP table. If multiple most recently used entries are found (as happens when none of the entries has been used since they were added into the table), the most recently added one is selected.

In Section 5, we will present the results of using different algorithms and different LTTP table sizes.

4. Reputation Systems and Trust Negotiation

Most of us have encountered reputation management systems in our daily lives. For example, several credit monitoring companies track personal credit histories, which are "reputations" banks take into consideration in lending money. Another good example is eBay [1], which maintains a reputation rating for every seller based on previous buyers' evaluations. However, the above-mentioned centralized systems do not have good flexibility and scalability, which make them unsuitable for large distributed systems like P2P networks. Several trust and reputation systems [8, 4, 19] have been proposed for P2P systems. In these systems, the trustworthiness of a peer is evaluated, recorded, processed and propagated in the system by all the peers

in a distributed fashion. This allows peers that are likely to provide reliable services to be identified, but it also causes those that do not contribute resources very often to be rejected because of their lack of a good reputation.

Trust negotiation can benefit from reputation systems. For example, although reputation is not related to the concept of LTTPs, a peer may be more willing to ask help from an LTTP peer with higher reputation. It is also possible that some credentials are directly mapped to some level of trust. If P_2 shows appropriate credentials to P_1, P_1 may have a higher trust evaluation of P_2. On the other hand, if P_1's trust evaluation of P_2 is higher than some value, P_1 may waive some credentials that it would otherwise request from P_2.

In the previous sections, we have assumed that peers' credentials are issued by some trusted third parties, and that once a credential is disclosed, everyone should believe that the credential is authentic if the issuer (a GTTP)'s signature can be verified. However, we can also allow for credentials issued by peers. Consider three peers, P_1, P_2, and P_3, where P_3 is an LTTP for P_1 and P_2. If P_2 has a "credential", issued by P_3 showing that P_2 is considered trustworthy by P_3, it is reasonable that P_1 should trust P_2 at a certain level. This is effectively the same as P_1 asking P_3 about P_2's trustworthiness; however, with a credential P_1 does not need to communicate with P_3 directly as P_1 can cryptographically verify that P_2's credential is issued by P_3.

A detailed discussion of the combination of trust negotiation and reputation system is beyond the scope of this paper. However, we believe a unified system can provide more flexible and comprehensive support for trust management.

5. Experimental Results

We have conducted several experiments in a simulated P2P environment to justify our ideas about how LTTPs can help peers in trust negotiations. The simulations were run on a dual 2.8GHz Xeon processor machine with 1GB of memory running Red Hat Linux 9. We simulated P2P systems at different scales: 1000, 5000 and 10000 nodes. In all the experiments we have conducted, we have achieved similar results. In this paper we present only the results from a 1000-node P2P system that address the following two questions:

- Are LTTPs helpful in trust negotiations?
- How do different LTTP table sizes and maintenance algorithms affect trust negotiations?

In our simulation, every peer has a unique resource and a randomly chosen number of credentials (from 10 to 20). Every peer has randomly generated access policy rules, which are composed of local and remote credentials, to protect the access to the resource. All peers have same understanding of the credentials. If a peer P_i has a rule: $C_1^{P_i} \longleftarrow C_2^{P_*}$, another peer P_j knows that $C_2^{P_*}$ is mapped to its local credential $C_2^{P_j}$. There are 1 to 3 unprotected local credentials for each peer. Each peer implicitly has a deny rule for any credential that it does not possess. All the peers use the eager negotiation strategy.

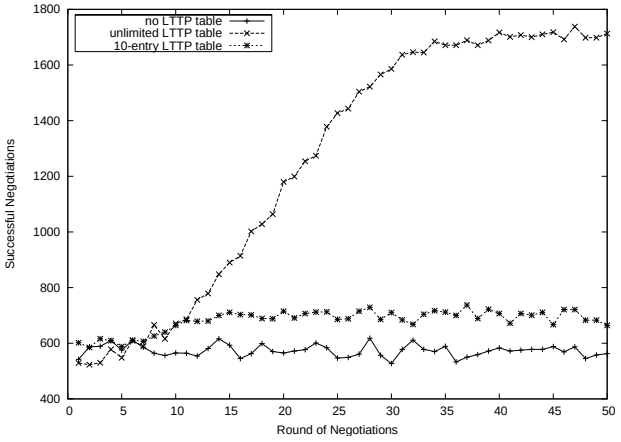

Figure 1. Benefits of using LTTPs

For each trust negotiation, we randomly choose two peers P_i and P_j, $i \neq j$, letting P_i initiates a request to P_j's resource. Trust negotiation between the two peers is conducted using the negotiation strategy described in Section 3.1 and the result of the negotiation is recorded. Each simulation consisted of 10,000 negotiations. We divided these 10,000 negotiations into 50 rounds, and calculated the number of successful negotiations in every round. Each data point in the following figures represents the average of 20 simulation runs with different random seeds.

Figure 1 shows the result when an LTTP is used in the trust negotiation. The number of successful negotiations increases remarkably, even when only a small LTTP table (10 entries) is used. We noted that given unlimited space for LTTP table entries, the maximum number of table entries observed in each simulation was on average 195.3, while the average length of the table was 123.1.

Next we compare the use of different algorithms (Random, MRU, MRA, LRU, LRA) to maintain 20-entry and 40-entry LTTP tables. The results are shown in Figure 2, which shows that the performance of the different algorithms is quite similar. However, Table 1, which summarizes the total number of successful ne-

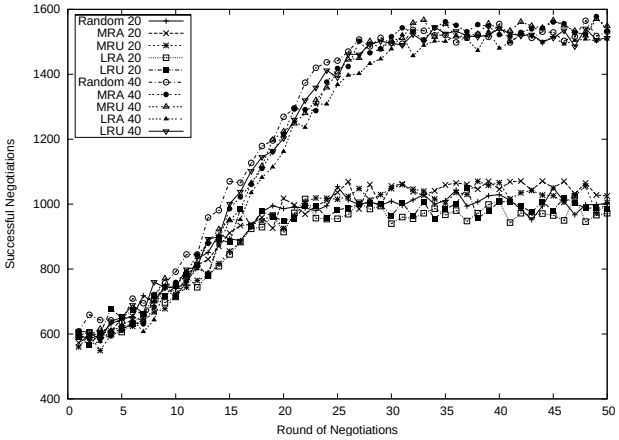

Figure 2. Comparison of different LTTP table maintenance methods

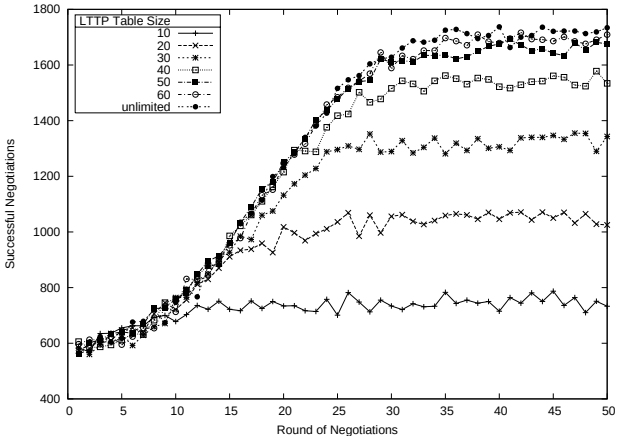

Figure 3. Comparison of LTTP tables with different sizes

LT Size	Random	MRA	LRA
10	707329	714886	696012
20	900080	927127	890735
30	1091527	1107103	1066629
40	1206062	1225697	1198674
50	1267546	1282535	1268283
60	1304435	1297785	1297467

Table 1. Total number of successful negotiations using different LTTP tables and different maintenance algorithms

gotiations in the 20 simulations, does reveal some minor differences between algorithms[4]. Of the three algorithms MRA shows the best performance, with LRA performing the worst, although the difference is less than 5%. However, when LTTP table size is 60, Random surprisingly becomes the best algorithm.

Finally we compare the effects of different LTTP table sizes, as shown in Figure 3. Because of the similar results reported above using different LTTP table maintenance algorithms, we only show the results when MRA was used. Figure 3 shows that an LTTP table with unlimited size behaves almost the same as a table with size 50 or 60. Thus, under the conditions we simulated a peer can use a 40-entry LTTP table and achieve relatively good performance.

It should be noted that the above results are based on the assumptions that each peer making a request randomly selects its target and that every peer has a randomly generated credential disclosure policy. We are currently conducting experiments using a different resource request model, namely a power law model, that has been observed in real P2P systems [13]. We are also working toward a more realistic way to generate credential disclosure policies. For more experimental results and analysis, please refer to [2].

4 The data for MRU and LRU are not shown since they are almost the same as for MRA and LRA, respectively.

6. Related Work

Many reputation systems [8, 4, 19, 7, 3] have been proposed to manage trust in distributed systems such as P2P. In these systems, peers build trust with each other by conducting resource sharing transactions and exchanging their trust evaluations about others. It has been widely accepted that reputation systems require some form of persistent node identification, such as a verifiable real-world identity [11, 10].

Automated trust negotiation (ATN) [20, 18, 14] was proposed as an approach to support access control and authentication in open and distributed systems through exchange of digitally signed certificates. Several vulnerability-related issues have been investigated, such as protecting sensitive policies [15, 22] and sensitive attributes [15, 17, 16, 21]. Recently Li *et al.* [9] proposed a mutual signature verification scheme called the Oblivious Envelope-Based System (OEBS) to solve the problem of cyclic policy interdependency in trust negotiation. However, this solution requires that the two parties' credentials are issued by the same certification authority; collaborative trust negotiation proposed in this paper does not have this constraint.

7. Conclusion

In this paper we present a collaborative trust negotiation scheme, which is an initial step to applying trust negotiation in P2P systems. In our scheme, peers build trust with each other through the exchange of digital credentials. We introduce the concept of *locally trusted third parties*(LTTPs) to enable collaborative trust negotiation and solve the problem of cyclic credential disclosure policy interdependencies. Our experimental simulation results show that LTTPs substantially improve the number of successful trust negotiations, even with a limited size LTTP table. We believe the combination of trust negotiation and reputation systems will lead to a promising solution to trust management.

References

[1] http://www.ebay.com.

[2] Trust negotiation in peer-to-peer systems. Technical Report (in progress), 2004, available at http://scens.cs.dartmouth.edu.

[3] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the Hawaii International Conference on System Sciences*, 2000.

[4] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 310–317. ACM Press, 2001.

[5] E. Adar and B.A. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.

[6] William J. Bolosky, John R. Douceur, David Ely, and Marvin Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In *Proceedings of the 2000 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 34–43. ACM Press, 2000.

[7] Fabrizio Cornelli, Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Choosing reputable servents in a P2P network. In *Proceedings of the eleventh international conference on World Wide Web*, pages 376–386. ACM Press, 2002.

[8] Ernesto Damiani, De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–216. ACM Press, 2002.

[9] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing*, pages 182–189. ACM Press, 2003.

[10] A. Ganesh A. Rowstron M. Castro, P. Druschel and D. S. Wallach. Security for structured peer-to-peer overlay networks. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02)*, pages 172–179, 2002.

[11] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of ACM*, 43(12):45–48, 2000.

[12] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[13] S. Saroiu, G.P.Krishna, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of the SPIE Conference on Multimedia Computing and Networking*, pages 156–170, 2002.

[14] Kent E. Seamons, Marianne Winslett, and Ting Yu. Limiting the disclosure of access control policies during automated trust negotiation. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS'01)*, February 2001.

[15] Kent E. Seamons, Marianne Winslett, Ting Yu, L. Yu, and R. Jarvis. Protecting privacy during on-line trust negotiation. In *Proceeding of the 2nd Workshop on Privacy Enhancing Technologies*, April 2002.

[16] William H. Winsborough and Ninghui Li. Protecting sensitive attributes in automated trust negotiation. In *Proceeding of the ACM Workshop on Privacy in the Electronic Society*, pages 41–51. ACM Press, 2002.

[17] William H. Winsborough and Ninghui Li. Towards practical automated trust negotiation. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, pages 92–103, June 2002.

[18] William H. Winsborough, Kent E. Seamons, and Vicki E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition, Volume 1*, pages 88–102. IEEE Press, January 2000.

[19] Bin Yu and Munindar P. Singh. A social mechanism of reputation management in electronic communities. In *Proceedings of Fourth International Workshop on Cooperative Information Agents*, 2000.

[20] Ting Yu, Xiaosong Ma, and Marianne Winslett. PRUNES: an efficient and complete strategy for automated trust negotiation over the internet. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 210–219. ACM Press, 2000.

[21] Ting Yu and Marianne Winslett. Policy migration for sensitive credentials in trust negotiation. In *Proceeding of the 2nd Workshop on Privacy in the Electronic Society*, November 2003.

[22] Ting Yu and Marianne Winslett. A unified scheme for resource protection in automated trust negotiation. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 110–122. IEEE Computer Society, 2003.

[23] Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information System Security*, 6(1):1–42, 2003.