Trust Negotiation with Hidden Credentials, Hidden Policies, and Policy Cycles [*]

Keith B. Frikken Jiangtao Li Mikhail J. Atallah
Department of Computer Science, Purdue University, West Lafayette, Indiana
{kbf, jtli, mja}@cs.purdue.edu

Abstract

In an open environment such as the Internet, the decision to collaborate with a stranger (e.g., by granting access to a resource) is often based on the characteristics (rather than the identity) of the requester, via digital credentials: Access is granted if Alice's credentials satisfy Bob's access policy. The literature contains many examples where protecting the credentials and the access control policies is useful, and there are numerous protocols that achieve this. In many of these schemes, the server does not learn whether the client obtained access (e.g., to a message, or a service via an e-ticket). A consequence of this property is that the client can use all of her credentials without fear of "probing" attacks by the server, because the server cannot glean information about which credentials the client has (when this property is lacking, the literature uses a framework where the very use of a credential is subject to a policy specific to that credential). The main result of this paper is a protocol for negotiating trust between Alice and Bob without revealing either credentials or policies, when each credential has its own access policy *associated with it (e.g., "a top-secret clearance credential can only be used when the other party is a government employee and has a top-secret clearance"). Our protocol carries out this privacy-preserving trust negotiation between Alice and Bob, while enforcing each credential's policy (thereby protecting sensitive credentials). Note that there can be a deep nesting of dependencies between credential policies, and that there can be (possibly overlapping) policy cycles of these dependencies. Our result is not achieved through the routine use of standard techniques to implement, in this framework, one of the known strategies for trust negotiations (such as the "eager strategy"). Rather, this paper uses novel techniques to implement a non-standard trust negotiation strategy specifically suited to this framework (and in fact unusable outside of this framework, as will become clear). Our work is therefore a substantial extension of the state-of-the-art in privacy-preserving trust negotiations.*

[*]Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

1 Introduction

Whereas in the past access decisions were based on the identity of the entity requesting a resource, in open systems such as the Internet, this approach is ineffective when the resource owner and the requester belong to different security domains that are controlled by different authorities, possibly unknown to each other. One alternative is to use *digital credentials* for satisfying access control policies [3, 12, 21]. Digital credentials are digitally signed assertions about the credential owner by a credential issuer. Each digital credential contains an attribute (or set of attributes) about the owner. The decision to grant access to a resource is based on the attributes in the requester's credentials, such as citizenship, security clearance, employment, group membership, credit status, etc.

A typical scenario for accessing a resource using digital credentials is for the requester, Alice, to send her request to the resource owner, Bob. Bob then responds with the policy that governs access to that resource. If Alice's credentials satisfy Bob's policy, she sends the appropriate credentials to Bob. After Bob receives the credentials and verifies them, he grants Alice access to the resource. Observe that, in this scenario, Alice learns Bob's policy and Bob learns Alice's credentials. Such a strategy is straightforward and efficient, however it is unacceptable if the credentials or the access control policies are considered to be sensitive information.

Clearly, it is advantageous for the requester to protect her credentials, as revealing them may violate her personal privacy. The motivation for hiding the policy is not necessarily protection from an evil adversary. It could simply be the desire to prevent legitimate users from "gaming" the system – e.g., changing their behavior based on their knowledge of the policy (which can render an economically-motivated policy less effective). This is particularly important for policies that are not incentive-compatible in economic terms (e.g., they suffer from perverse incentives in that they reward the wrong kinds of behavior, such as free-loading). Or it could be that the policy is a commercial secret – e.g., Bob has pioneered a novel way of doing business, and knowledge of the policy would compromise Bob's strategy and invite unwelcome imitators. Finally, a process that hides Alice's credentials from Bob is ultimately not only to Alice's advantage but also to Bob's: Bob no longer needs to worry about rogue insiders in his organization illicitly leaking or selling Alice's private information, and may even lower his liability insurance rates as a result. Privacy-preservation is a win-win proposition, one that is appealing even if Alice and Bob are honest and trustworthy entities.

For these and other similar reasons, there has been a substantial amount of recent work [17, 6, 14] on performing this type of attribute-based access control while protecting Alice's credentials and Bob's policies. One assumption of these schemes is that the resource owner does not learn whether the requester obtained access. When this is the case, the requester can use all of her credentials without regard to their sensitivity level, as these schemes do not leak the requester's credentials to the service provider. However, this "resource owner does not learn" property may not hold in practice for the following two reasons:

1. In many scenarios, the server grants access to services rather than messages. Thus, for certain types of services, the server has to know whether the client got access to the services. In fact there are audit and accounting requirements that cause many organizations to require learning whether access took place.

2. Even if the server offers messages rather than services, message requests are not independent in most systems. For example, suppose a client requests message M_1, which contains a hyperlink to message M_2, and that same client subsequently requests M_2 a few minutes later; although the server does not learn for certain that the client successfully obtained M_1, inferences can be made.

For the above reasons, the server could learn whether the client obtained access. This may seem like an insignificant bit of information, but since the server can set his policy to be an arbitrary function, this enables the server to probe the client for sensitive credentials. For example, the server may intentionally set his access control policy to be "only people with top-secret clearance can access the resource". When the client obtains access, the server learns immediately that the client has a top-secret clearance credential.

This *Sensitive CredentiAl Leakage Problem (SCALP)* is not due to any flaw or weakness in the previous protocols (e.g., [17, 6, 14]) but rather exists in any situation where the server can link transactions to the same client and has arbitrary freedom when creating the access control policy. In such situations, the client may understandably have legitimate concerns about using credentials that she deems sensitive – in fact the client may be required to protect certain credentials (e.g., a top-secret clearance credential). This poses a problem for the previous schemes, which require the client's ability to use all of her credential set. Therefore, there is a need for a trust negotiation system that can mitigate these concerns.

In traditional trust negotiation [30, 32, 25, 34, 33, 29] the notion of sensitive credential protection has been well studied (see Section 3). In these schemes, each sensitive credential has an access control policy – a credential is used (or revealed) only when the other party satisfies the policy for that credential. This does not prevent SCALP, but it does allow the user to control the potential leakage of her credentials. The schemes in [14, 17, 6] did not reveal credentials but could not handle policies for credentials (i.e., they dealt with the easier special case where each credential's access control policy was unconditionally "true"). The present paper is the first to combine the techniques for hidden credentials with the notion of policies for sensitive credentials. These credential policies have to be considered sensitive as well, because otherwise the server (or client) can game the system in many ways. For example, if the client knows the access control policies for the server's credentials then she will know the path of least resistance to unlock certain credentials and thus she will be able to probe more easily.

The rest of this paper is organized as follows. We begin with a detailed description of our contributions in Section 2. We review trust negotiation and propose a new definition of trust negotiation that supports policy cycles in Section 3. Next, we formally introduce our approach to trust negotiation in Section 4, and then we review some cryptographic tools in Section 5. We present our protocol for privacy-preserving trust negotiation in Section 6. We give efficiency improvements for our base scheme in section 7. We give a sketch of the proof of security in Section 8. We discuss the related work in Section 9, and we conclude in Section 10.

2 Our Contributions

We introduce a protocol for privacy-preserving trust negotiation, where the client and server each input a set of credentials along with an access control policy for each of their credentials. The protocol determines the set of usable credentials between the client and the server, and then will process the resource or service request based on the client's usable credentials. A credential is *usable* if its access control policy has been satisfied by the other party. Our protocol is complicated by the fact that: (1) the policies for sensitive credentials may themselves be sensitive and therefore cannot be revealed, (2) the client should not learn information about which of her credentials or the server's credentials are usable, and (3) the server should not learn information about which of his credentials or the client's credentials are usable. The rationale for requirement (1) was given in the previous section. Requirements (2) and (3) are because, if the client or server were to learn which of its credentials are usable, then this would reveal more information about the other party's credential set and thus facilitate probing attacks. The technical contributions of this paper include:

1. We develop a new privacy-preserving trust negotiation protocol and several novel cryptographic protocols for carrying it out. One of the challenges is the distinction between having a credential and being able to use that credential (when its access control policy has been satisfied), while requiring that "not having" a credential be indistinguishable from "having but being unable to use" a credential.

2. We propose a *reverse eager trust negotiation strategy* (denoted as RE strategy) that handles arbitrary policy cycles, whereas the existing traditional trust-negotiation strategies (such as the eager strategy [30]) are inherently unable to handle such cycles (even if these strategies were properly implemented in this framework).

3 Trust Negotiation: Review and Discussion

In trust negotiation [30, 32, 25, 34, 28, 33, 29], the disclosure of a credential s is controlled by an access control policy p_s that specifies the prerequisite conditions that must be satisfied in order for credential s to be disclosed. Typically, the prerequisite conditions are a set of credentials $C \subseteq \mathcal{C}$, where $\mathcal{C}$ is the set of all credentials. As in [30, 32, 25, 34, 33], the policies in this paper are modeled using propositional formulas. Each policy p_s takes the form $s \leftarrow \phi_s(c_1, \ldots, c_k)$ where $c_1, \ldots, c_k \in \mathcal{C}$ and $\phi_s(c_1, \ldots, c_k)$ is a normal formula consisting of literals c_i, the Boolean operators $\vee$ and $\wedge$, and parentheses (if needed). In this paper, s is referred to as the target of p_s, and $\phi_s(c_1, \ldots, c_k)$ is referred to as the policy function of p_s.

Given a set of credentials $\mathcal{C}' \subseteq \mathcal{C}$ and a policy function $\phi_s(c_1, \ldots, c_k)$, we denote $\phi_s(\mathcal{C}')$ as the value of the normal formula $\phi_s(x_1, \ldots, x_k)$ where $x_i = 1$ if and only if $c_i \in \mathcal{C}'$ (otherwise $x_i = 0$). For example, if $\phi_s = (c_1 \wedge c_2) \vee c_3$, then $\phi_s(\{c_1, c_2, c_4\}) = 1$ and $\phi_s(\{c_1, c_4\}) = 0$. Policy p_s is *satisfied* by a set of credentials $\mathcal{C}' \subseteq \mathcal{C}$ if and only if $\phi_s(\mathcal{C}') = 1$. During trust negotiation, one can disclose credential s if $\phi_s(\mathcal{C}') = 1$ where $\mathcal{C}'$ is the set of credentials that she has received from the other party.

A trust negotiation protocol is normally initiated by a client requesting a resource from a server. The negotiation consists of a sequence of credential exchanges. Trust is established if the initially requested resource is granted and all policies for disclosed credentials are satisfied [30, 32]. In this case, the negotiation between the client and server is a *successful* negotiation, and otherwise, it is a *failed* negotiation. We give the formal definition for traditional trust negotiation as follows:

Definition 1 (Traditional Trust Negotiation) *Let $\mathcal{C}_S$ and $\mathcal{P}_S$ ($\mathcal{C}_C$ and $\mathcal{P}_C$) be the sets of credentials and policies possessed by a negotiating server (client). The negotiation is initiated by a request for $s \in \mathcal{C}_S$* [1] *from the client. The goal of trust negotiation is to find a* credential disclosure sequence $(c_1, \ldots, c_n = s)$, *where $c_i \in \mathcal{C}_S \cup \mathcal{C}_C$, and such that for each c_i, $1 \le i \le n$, the policy for c_i is satisfied by the credentials already disclosed, i.e., $\phi_{c_i}(\bigcup_{j<i} c_j) = 1$. If the client and server find a credential disclosure sequence, the negotiation succeeds, otherwise, it fails.*

The sequence of disclosed credentials depends on the decisions of each party; these decisions are referred to as a strategy. A strategy controls which credentials are disclosed, when to disclose them, and when to terminate a negotiation [34]. Several negotiation strategies are proposed in [30, 32, 34]. For example, in the eager strategy [30], two parties take turns disclosing a credential to the other side as soon as the access control policy for that credential is satisfied. For the reader unfamiliar with the eager strategy, we describe it in more detail in Appendix A. Although the cryptographic contributions of this paper will make it possible to implement the eager strategy in the framework considered, we do not pursue this approach because it fails to handle policy cycles. In fact, if there is a policy cycle, the trust negotiation will fail under Definition 1. We now propose a new definition of trust negotiation that supports policy cycles.

Definition 2 (Cycle-Tolerant Trust Negotiation) *Let $\mathcal{C}_S$ and $\mathcal{P}_S$ ($\mathcal{C}_C$ and $\mathcal{P}_C$) be the sets of credentials and policies possessed by a negotiating server (client). The negotiation is initiated by a request for $s \in \mathcal{C}_S$ from the client. The negotiation between the client and server succeeds if there exists usable credential sets $\mathcal{U}_S \subseteq \mathcal{C}_S$ and $\mathcal{U}_C \subseteq \mathcal{C}_C$ for the server and client respectively, such that (1) $s \in \mathcal{U}_S$, (2) $\forall c \in \mathcal{U}_S$, $\phi_c(\mathcal{U}_C) = 1$, and (3) $\forall c \in \mathcal{U}_C$, $\phi_c(\mathcal{U}_S) = 1$. Otherwise, the negotiation fails.*

Note that the above definition allows for many possible $\mathcal{U}_C$,$\mathcal{U}_S$ solution pairs, and does not capture any notion of minimality for such pairs: Some solution pair may be a proper subset of some other pair, and either of them is considered acceptable. This is fine in the framework of this paper, because at the end of the negotiation nothing is revealed

[1] For simplicity, we model service s as a credential. In order to obtain s, the client has to have credentials that satisfy ϕ_s.

about the specific $\mathcal{U}_C, \mathcal{U}_S$ pair, i.e., neither party can distinguish which pair was responsible for access or whether that pair was minimal or not. *It also implies that the trust negotiation strategy we design need not make any particular effort at zeroing in on a particular pair (e.g., a minimal one).*

Example 1 Suppose the client and server have the following policies:

Client	*Server*
$p_{c_1} : c_1 \leftarrow s_2$	$p_s : s \leftarrow c_5 \vee (c_2 \wedge c_4)$
$p_{c_2} : c_2 \leftarrow s_2 \wedge s_3$	$p_{s_1} : s_1 \leftarrow c_6$
$p_{c_3} : c_3 \leftarrow s_6$	$p_{s_2} : s_2 \leftarrow c_1$
$p_{c_4} : c_4 \leftarrow \mathsf{true}$	$p_{s_3} : s_3 \leftarrow c_4$

where s denotes the server's service, $\{s, s_1, s_2, s_3\}$ denote the set of server's credentials, $\{c_1, c_2, c_3, c_4\}$ denotes the set of the client's credentials. Under Definition 1, the negotiation between the client and server would fail as there is a policy cycle between c_1 and s_2, and there exists no credential disclosure sequence ending with s. However, under Definition 2, the negotiation succeeds, as $\mathcal{U}_C = \{c_1, c_2, c_4\}$ and $\mathcal{U}_S = \{s, s_2, s_3\}$ is a solution pair.

Clearly, if the trust negotiation between the client and server can succeed in Definition 1, it will also succeed in Definition 2, but not vice-versa (e.g., see Example 1). In the next section, we describe a reverse eager (RE) strategy that efficiently determines whether the negotiation can succeed (under Definition 2) given $\mathcal{C}_S$, $\mathcal{P}_S$, $\mathcal{C}_C$, and $\mathcal{P}_C$. Then, we will give a privacy-preserving trust negotiation protocol that securely implements the RE strategy *without* revealing $\mathcal{C}_S$ and $\mathcal{P}_S$ to the client and *without* revealing $\mathcal{C}_C$ and $\mathcal{P}_C$ to the server.

4 Our Approach

We begin this section with an intuitive, informal presentation of our approach. The eager strategy for trust negotiations can be thought of as one of "progressively incrementing the usable set": The set of usable credentials is initially set to the unconditionally usable credentials, and each iteration adds to it credentials that have just (in that iteration) become known to be usable. It is, in other words, a conservative approach, whose motto is that *a credential is not usable unless proved otherwise*: The iterative process stops when no more credentials are added to the usable set. This conservatism of the eager approach is also why using that strategy would lead us to deadlock on cycles. Our overall strategy is the opposite, and can be viewed as a "reverse eager" strategy: Initially all credentials are temporarily considered to be usable, and each iteration *decreases* the set of usable credentials (of course the decrease is achieved implicitly, so as not to violate privacy – more on these implementation details is given in the next section). Note that, because of the "optimism" of the RE strategy (in that a credential is tentatively usable, until proven otherwise), cycles no longer cause a problem, because a "self-reinforcing" cycle's credentials will remain usable[2] (whereas it deadlocked in the eager strategy). This RE strategy (the details of which are given later) is made possible by the fact that we carry out the iterative process in a doubly blinded form, so that *neither party learns anything* (not only about the other party's credentials, but also about their use policies for these credentials). The RE strategy and blinded evaluations work hand in hand: The former is useless without the latter, and it should not be used outside of this particular framework.

The rest of this section gives a more precise presentation by first introducing the notation that will be used throughout the rest of the paper, then defining our problem and giving a more detailed overview of our approach.

4.1 Notation and Definitions

Before describing the details of our approach, it is necessary to give a more formal notation than the intuitive terminology of the previous section.

- We use s to denote the server's service or resource that the client requests. Without loss of generality, we model s as a credential.
- We use $\mathcal{C}_C$ (resp., $\mathcal{C}_S$) to denote the set of the client's (resp., the server's) hidden credentials. We use n_C and n_S to denote the size of $\mathcal{C}_C$ and $\mathcal{C}_S$, respectively. Referring to Example 1, $\mathcal{C}_C = \{c_1, c_2, c_3, c_4\}$ and $n_C = 4$.
- We use $\mathcal{P}_C$ (resp., $\mathcal{P}_S$) to denote the set of the client's (resp., server's) policies.
- We use $R(p_i)$ to denote the set of credentials relevant to (i.e., that appear in) the policy function of the policy p_i. For example, if the policy function for p_i takes the form of $\phi_i(c_1, \ldots, c_k)$, then $R(p_i) = \{c_1, \ldots, c_k\}$.
- We use $R(\mathcal{P}_C)$ (resp. $R(\mathcal{P}_S)$) to denote the union of all the $R(p_i)$'s over all p_i in $\mathcal{P}_C$ (resp. $\mathcal{P}_S$), i.e., $R(\mathcal{P}_C) = \bigcup_{p_i \in \mathcal{P}_C} R(p_i)$. We use m_C and m_S to denote the size of $R(\mathcal{P}_C)$ and $R(\mathcal{P}_S)$, respectively. Referring to Example 1, $R(\mathcal{P}_S) = \{c_1, c_2, c_4, c_5, c_6\}$ and $m_S = 5$.
- We use $\mathcal{U}_C$ (resp., $\mathcal{U}_S$) to denote the set of the client's (resp., the server's) credentials whose policies *are presumed to have been satisfied* (i.e., these are the currently-believed usable credentials); as stated earlier, these sets will decrease from one iteration to another. Initially, $\mathcal{U}_C = \mathcal{C}_C$ and $\mathcal{U}_S = \mathcal{C}_S$, and throughout the iterative process we have $\mathcal{U}_C \subseteq \mathcal{C}_C$ and $\mathcal{U}_S \subseteq \mathcal{C}_S$.

[2]See Section 4.4 for proof

4.2 Problem Definition

The goal of this paper is to develop a solution such that the client and server are able to learn whether trust can be established without either party revealing to the other party anything about their own private credentials and policies (other than, unavoidably, what can be deduced from the computed answer). We formalize the *privacy-preserving trust negotiation* problem as follows.

Problem 1 The server inputs $\mathcal{C}_S$ and $\mathcal{P}_S$ and the client inputs $\mathcal{C}_C$, $\mathcal{P}_C$, and a request for the server's service s. In the end, both the client and server learn whether the client's access to s can be granted based on their credentials and policies, without revealing their sensitive credentials and policies to the other party. In other words, they want to know whether the trust negotiation between the client and server succeeds under Definition 2 without leaking other information, except for n_C, n_S, m_C, and m_S.

Having stated the problem, we will now discuss the information revealed by the protocol. The values n_C and n_S reveal the number of credentials that the client and server respectively have and the values m_C and m_S reveal the size of all policies for all credentials for the client and the server. We do not view this as a problem because the parties can pad their list or their policies with dummy credentials. We now list the security properties required of a solution (a more detailed version is given in Section 8).

1. *Correctness*: If trust can be successfully negotiated, then both the client and server should output true with overwhelming probability if they follow the protocol.
2. *Robustness against malicious adversaries*: If the trust negotiation fails, then both the client and server should output false even if one of the participants is malicious (i.e., behaves arbitrarily) with overwhelming probability.
3. *Privacy-preservation*: The client and server should not learn anything about the other party's private input (credentials and policies) or intermediate results (usable credential sets), other than what can be deduced from the yes/no outcome of the negotiation.

4.3 Overview of Our Approach

As described earlier, our overall strategy for privacy-preserving trust negotiation is the RE strategy. During each round of the RE strategy, a negotiator blindly (i.e., without actually learning the outcome) checks which of their presumed-usable local credentials are in fact not usable (according to whether the policy for it has ceased to be satisfied based on the the new presumed-usable credential set of the other party). After this, the negotiator blindly decreases their own local presumed-usable credential set accordingly. Recall that we use $\mathcal{U}_C$ ($\mathcal{U}_S$) to denote the set of the client's (server's) credentials that are presumed usable, i.e., at a particular stage of the iterative process, for each credential in $\mathcal{U}_C$ ($\mathcal{U}_S$), the corresponding usability policy is currently satisfied (although it may cease to be so in a future iteration). We present the RE strategy in Figure 1.

```
reverse-eager-strategy($\mathcal{C}$, $\mathcal{P}$, $\mathcal{U}_O$)
    $\mathcal{C}$: the local credentials of this party.
    $\mathcal{P}$: the local policies of this party.
    $\mathcal{U}_O$: the credentials used by the other party.
Output:
    $\mathcal{U}$: the local credentials that can be used.
Procedure:
    $\mathcal{U} = \mathcal{C}$;
    For each credential $c \in \mathcal{C}$
        let $c$'s policy be $p_c : c \leftarrow \phi_c$;
        if $\phi_c(\mathcal{U}_O) = 0$, then $\mathcal{U} = \mathcal{U} - \{c\}$;
    return $\mathcal{U}$.
```

Figure 1. Pseudocode for the RE strategy

Our approach to privacy-preserving trust negotiation is to implement the RE strategy in a secure way. We give the high-level description of our protocol in Figure 2. In it, the server first initializes $\mathcal{U}_S$. Then the client and server run a secure version of the RE strategy protocol to update $\mathcal{U}_C$ and $\mathcal{U}_S$ iteratively for n rounds, where $n = \min(n_C, n_S)$ (recall that the trust negotiation using the eager strategy takes at most n rounds). In the end, if $s \in \mathcal{U}_S$ (i.e., s can be used), the negotiation succeeds, otherwise, it fails.

```
privacy-preserving-trust-nego($s$, $\mathcal{C}_C$, $\mathcal{P}_C$, $\mathcal{C}_S$, $\mathcal{P}_S$)
Output:
    true or false
Procedure:
    Initialize $\mathcal{U}_S$;
    For $i = 1, \ldots, \min(n_C, n_S)$
        $\mathcal{U}_C$ = reverse-eager-strategy($\mathcal{C}_C$, $\mathcal{P}_C$, $\mathcal{U}_S$);
        $\mathcal{U}_S$ = reverse-eager-strategy($\mathcal{C}_S$, $\mathcal{P}_S$, $\mathcal{U}_C$);
    If $s \in \mathcal{U}_S$, output true, otherwise, output false.
```

Figure 2. High-level description of privacy-preserving trust negotiation

Clearly, $\mathcal{U}_C$ and $\mathcal{U}_S$ should not be known to either the client or the server. Thus $\mathcal{U}_C$ and $\mathcal{U}_S$ need to be maintained in such a way that the values of $\mathcal{U}_C$ and $\mathcal{U}_S$: (1) are unknown to the client and server and (2) cannot be modified by a malicious client or server. We maintain $\mathcal{U}_C$ in the following split way: For each $c \in \mathcal{C}_C$, the client generates two random numbers $r_c[0]$ and $r_c[1]$, and the server learns one of them, denoted as r_c. If $c \in \mathcal{U}_C$, then $r_c = r_c[1]$, otherwise $r_c = r_c[0]$. The client does not learn which value the server obtains, and so by splitting $\mathcal{U}_C$ in this way, the client does not learn $\mathcal{U}_C$. Furthermore, the server does not learn anything about $\mathcal{U}_C$, as the values he obtains from the client look random to him. We maintain $\mathcal{U}_S$ in an analogous way. Our protocol will keep this form of splitting as an invariant through all its steps. This does not solve all privacy problems of the negotiation, but it will be one of the guiding principles of our protocol.

4.4 Proof of RE Strategy

We now provide a proof of the correctness of the RE strategy for trust negotiations. That is, we prove that at the end of the RE negotiation every unusable credential has been marked as such (the other credentials correctly retain their initial label of "usable"). So not only does RE not produce a minimal usable credential set pair $\mathcal{C}_C, \mathcal{C}_S$, in fact it will produce a maximal pair in the sense that every credential (whether essential or not) is kept usable unless marked otherwise. As stated earlier, this is justified by the indistinguishability to either party of any two solution pairs.

Throughout this section, we use $\mathcal{C}_{X,i}$, $X \in \{C, S\}$, to denote the usable credential set of the client (if $X = C$) or of the server (if $X = S$) after iteration i of the RE negotiation has completed. We use $\mathcal{C}_{X,0}$ to denote the initial (prior to iteration 1) usable credential set (which equals $\mathcal{C}_X$). We use $\bar{X}$ to denote $\{C, S\} - X$.

Letting $\mathcal{C}(X)$ denote the correct usable credentials for X, our goal is therefore to prove that, after the last iteration i of the RE negotiation, we have $\mathcal{C}_{X,i} = \mathcal{C}(X)$ and $\mathcal{C}_{\bar{X},i} = \mathcal{C}(\bar{X})$. Note that $\mathcal{C}_{X,i} = f_X(\mathcal{C}_{X,i-1}, \mathcal{C}_{\bar{X},i-1})$ for some monotonic function f_X. (Although in fact $\mathcal{C}_{X_i}$ depends only on $\mathcal{C}_{\bar{X},i-1}$ and not on $\mathcal{C}_{X,i-1}$, it does no harm to give a more general proof, as we do below, for the case when it can depend on both.)

The next lemma proves the intuitive fact that an iteration i cannot cause an unusable credential to become usable.

Lemma 1 *$\mathcal{C}_{X,i} \subseteq \mathcal{C}_{X,i-1}$, for $i = 1, 2, \ldots$.*

Proof: By induction on i. For the basis of the induction, $i = 1$, the claim trivially holds because, prior to iteration 1, all the credentials of each party are in their initial usable set $\mathcal{C}_{X,0}$. We now turn our attention to the inductive step, $i > 1$. Observe that

1. during iteration i, $\mathcal{C}_{X,i}$ is computed based on $\mathcal{C}_{X,i-1}$ and $\mathcal{C}_{\bar{X},i-1}$, i.e., $\mathcal{C}_{X,i} = f_X(\mathcal{C}_{X,i-1}, \mathcal{C}_{\bar{X},i-1})$;
2. during iteration $i - 1$, $\mathcal{C}_{X,i-1}$ is computed based on $\mathcal{C}_{X,i-2}$ and $\mathcal{C}_{\bar{X},i-2}$, i.e., $\mathcal{C}_{X,i-1} = f_X(\mathcal{C}_{X,i-2}, \mathcal{C}_{\bar{X},i-2})$;
3. by the induction hypothesis we have $\mathcal{C}_{X,i-1} \subseteq \mathcal{C}_{X,i-2}$, and $\mathcal{C}_{\bar{X},i-1} \subseteq \mathcal{C}_{\bar{X},i-2}$

The above facts (1), (2), and (3), together with the monotonicity of the function f_X, imply that $\mathcal{C}_{X,i} \subseteq \mathcal{C}_{X,i-1}$. □

A corollary of the above lemma is that, to prove the correctness of RE, it suffices to show that for every credential c of party X, c is unusable if and only if there is some iteration i after which $c \notin \mathcal{C}_{X,i}$. The next lemma proves the "if" part. Recall that $\mathcal{C}(X)$ denote the correct usable credentials for X.

Lemma 2 *For every i, we have $\mathcal{C}(X) \subseteq \mathcal{C}_{X,i}$.*

Proof: By induction on i. The basis, $i = 0$, is trivial because $\mathcal{C}_{X,0} = \mathcal{C}_X$. For the inductive step, $i > 0$, we assume that credential c was removed by iteration i (i.e., that $c \in \mathcal{C}_{X,i-1}$ and $c \notin \mathcal{C}_{X,i}$), and we show that it must then be the case that $c \notin \mathcal{C}(X)$. Observe that

1. $c \notin f_X(\mathcal{C}_{X,i-1}, \mathcal{C}_{\bar{X},i-1})$;
2. by the induction hypothesis, we have $\mathcal{C}(X) \subseteq \mathcal{C}_{X,i-1}$ and $\mathcal{C}(\bar{X}) \subseteq \mathcal{C}_{\bar{X},i-1}$.

The above (1) and (2), together with the monotonicity of f_X, imply that $c \notin f_X(\mathcal{C}(X), \mathcal{C}(\bar{X}))$, i.e., that $c \notin \mathcal{C}(X)$. □

The above lemma proved that every c removed by the RE negotiation deserves to be removed (the "if" part). To complete the proof, we need to prove the "only if" part: That every unusable credential will eventually be marked as such by the RE negotiation. That is, we need to prove that every $c \notin \mathcal{C}(X)$ will, for some i, be removed by iteration i. This is proved in the next lemma.

Lemma 3 *For every $c \notin \mathcal{C}(X)$, there is an iteration i for which $c \in \mathcal{C}_{X,i-1}$ and $c \notin \mathcal{C}_{X,i}$.*

Proof: For every credential c, let the *level* of c be defined as follows:

- If c is unconditionally usable then $level(c) = 1$.
- If the usability policy for c is p_c then $level(c) = 1 + \max\{level(v) : v \in R(p_c)\}$. (Recall that $R(p_c)$ is the set of credentials relevant to policy p_c.)

We claim that a credential $c \notin \mathcal{C}(X)$ is removed after at most $level(c)$ iterations, i.e., that for some $i \leq level(c)$ we have $c \in \mathcal{C}_{X,i-1}$ and $c \notin \mathcal{C}_{X,i}$. This is established by a straightforward induction on $level(c)$, whose details we omit. □

5 Review of Cryptographic Tools and Hidden Credentials System

5.1 Identity-based encryption

The concept of Identity-Base Encryption (IBE) was first proposed by Shamir [26] in 1984, however the first usable IBE systems were discovered only recently [5, 9]. An IBE scheme is specified by following four algorithms:

1. *Setup:* A Private Key Generator (PKG) takes a security parameter k and generates system parameters params and a master secret s. params is public, whereas s is private to PKG.

2. *Extract:* Given any arbitrary $\mathsf{ID} \in \{0,1\}^*$, PKG uses params, s, and ID to compute the corresponding private key d_{ID}.

3. *Encrypt:* It takes params, ID and plaintext M as input and returns ciphertext C.

4. *Decrypt:* It takes params, d_{ID} and ciphertext C as input and returns the corresponding plaintext M.

An IBE scheme enables Bob to encrypt a message using Alice's ID as the public key, and thus avoids obtaining the public key from Alice or a directory. Boneh and Franklin proposed an IBE scheme from the Weil pairing [5]. Their scheme is secure against adaptive chosen ciphertext attacks.

5.2 Homomorphic Encryption

A homomorphic encryption scheme [23, 24, 10, 11] is an encryption scheme in which the plaintexts are taken from a group G, and given the encryptions of two group elements one can efficiently compute a encryption of their sum. Usually this computation involves a modular multiplication of the encryptions, let $G = \mathbb{Z}_M$ where M is a large integer, we have $E(a) \cdot E(b) = E(a + b \bmod M)$. It is easy to see that $E(a)^c = E(c \cdot a \bmod M)$.

Damgård and Jurik [11] recently proposed a homomorphic encryption scheme in which all users can use the same RSA modulus N when generating key pairs. Under the Decisional Composite Residuosity assumption and Decision Diffie-Hellman assumption, the Damgård-Jurik cryptosystem [11] is *semantically secure*. The semantic security property guarantees that an eavesdropper cannot learn any information about a from $E(a)$. More precisely, given two arbitrary message m_0 and m_1, the random variables representing the two homomorphic encryptions $E(m_0)$ and $E(m_1)$ are computationally indistinguishable.

5.3 Scrambled Circuit Evaluation

The scrambled circuit evaluation protocol was developed by Yao [31]. This protocol involves a *generator* and an *evaluator*, in which the evaluator has private input x and the generator has private input y, and they want to jointly compute $f(x, y)$ without revealing their private inputs to the other party.

In the scrambled circuit evaluation protocol, the generator builds a circuit for computing f, constructs a scrambled version of the circuit, and then sends the scrambled circuit to the evaluator for evaluation. In a scrambled circuit, each wire is associated with two random numbers, one corresponds to 0 and the other to 1. Before the evaluation, the evaluator uses oblivious transfer to obtain the random values for the input wires corresponding to each bit of her private input x. During the evaluation, the evaluator learns exactly one random value for each internal wire, yet she doesn't know whether it corresponds to 0 or 1. Finally the evaluator sends the outcome of the evaluation to the generator, who recovers the final result.

The scrambled circuit evaluation protocol is secure against semi-honest adversaries and has been implemented by Malkhi et al. in [22]. Let γ be a security parameter, ρ be the cost of a 1-out-of-2 oblivious transfer, assuming the circuit to compute f is an s-input, t-gate Boolean 2-ary circuit, the cost of the scramble circuit protocol is $O(\rho s + \gamma t)$. When the size of the circuit is linear to the size of the input, the cost of the protocol is $O(\rho s)$.

5.4 Review of hidden credentials system

The hidden credentials system was proposed by Holt et al. [17, 6]. In that system, there is a trusted Credential Authority (CA) who issues credentials for users in the system[3]. Each user in the system is assigned a unique nym, where nym could be either a real name or a pseudonym. A hidden credential is a digitally signed assertion about an attribute of a credential holder by the CA. Roughly speaking, given an IBE scheme, a hidden credential cred for username nym and attribute attr is the private key corresponding to the string $\mathrm{nym}||\mathrm{attr}$.

We now give a simple example of how Alice accesses Bob's resource using the hidden credentials. Suppose Bob's resource M has an access policy which states that M should only be accessed by students. Alice has a student credential cred, i.e., $\mathrm{cred.nym} = \mathrm{Alice}$ and $\mathrm{cred.attr} = \mathrm{student}$. To access M, Alice sends her username Alice to Bob. Bob responds with $I(M, \mathrm{Alice}||\mathrm{student})$, the IBE encryption of M using identity $\mathrm{Alice}||\mathrm{student}$. Alice uses her credential cred to decrypt $I(M, \mathrm{Alice}||\mathrm{student})$ and obtains M. Note

[3]It is possible to support multiple CAs in the hidden credentials system [17]. For simplicity, we assume there is only one CA.

that Bob does not learn whether Alice possesses a student credential from the above interaction.

6 Protocol for Privacy-Preserving Trust Negotiation

6.1 Building Blocks

We now describe two building blocks, one for blinded policy evaluation, the other for equality test for array elements. These building blocks will later be used in the secure RE strategy protocol.

6.1.1 Blinded policy evaluation

The goal of the blinded policy evaluation is for Bob to evaluate Alice's policy without learning her policy. Alice should learn nothing about Bob's input nor the output of the evaluation. We define the input and output for this blinded policy evaluation in Figure 3.

Input: Alice has a private policy function $\phi : \{0,1\}^k \to \{0,1\}$, two random numbers t_0 and t_1, and k pairs of values $\{r_1[0], r_1[1]\}, \ldots, \{r_k[0], r_k[1]\}$. Bob has k values $r_1, \ldots, r_k$ where $r_i \in \{r_i[0], r_i[1]\}$.

Output: Bob learns $t_{\phi(r_1 \overset{?}{=} r_1[1], \ldots, r_k \overset{?}{=} r_k[1])}$. Alice learns nothing.

Figure 3. Input and output of blinded policy evaluation

The protocol for blinded policy evaluation was given in [14], for details see Appendix B. In most cases, it requires a polynomial amount of communication, and works for a family of policy functions.

6.1.2 Equality test for array elements

In an equality test for array elements, Alice has a private array $\langle x_1, \ldots, x_n \rangle$ and Bob has a private array $\langle y_1, \ldots, y_n \rangle$. They want to learn whether there exists an index i such that $x_i = y_i$. The result of the equality test is known to neither Alice nor Bob. We define the input and output for this protocol in Figure 4.

This equality test can be implemented by a scrambled circuit evaluation protocol [31, 22]. The protocol requires $O(\rho^2 n)$ communication and computation, where ρ is the maximum bit-length of each x_i and y_i or the security paramater (whichever is larger). We give an efficiency improvement that reduces that communication and computation requirement to $O(\rho n)$ (that is of independent interest) in Section 7.

Input: Bob has n values $\langle y_1, y_2, \ldots, y_n \rangle$. Alice has n values $\langle x_1, x_2, \ldots, x_n \rangle$ and has two random numbers t_0 and t_1.

Output: Bob learns t_1 if and only if there $\exists\ i \in [1..n]$ such that $x_i = y_i$, and learns t_0 otherwise. Alice learns nothing.

Figure 4. Input and output of equality test for array elements

6.2 Secure RE Strategy Protocol

The goal of the secure RE strategy protocol is to securely implement the RE strategy in Figure 1. We denote the participants of this protocol by Alice and Bob, where Alice is either the client or the server and Bob is the opposite role. In this section, we introduce a protocol to compute secure-reverse-eager-strategy$(\mathcal{C}_A, \mathcal{P}_A, \mathcal{C}_B, \mathcal{U}_B)$ (the items subscripted by A are Alice's values and those subscripted by B are Bob's values), where the output is $\mathcal{U}_A$ in the split-form described earlier. The careful reader may notice a discrepancy between this and the RE strategy defined earlier. Note that in this case $\mathcal{U}_B$ represents an array of Boolean values marking which credentials are usable, whereas in the previous case it represented the actual credentials. A credentials c of Alice's is not usable if Bob's usable credentials do not satisfy Alice's usability policy for c. Figure 5 describes this protocol.

Intuiton of Correctness/Security: In Step 1 of the protocol, Bob will learn $t_i[1]$ if he has credential c_i and he can use it, and otherwise he learns $t_i[0]$. Note that these values were generated by Alice. The first part of this (i.e., Bob has c_i) is captured by the value x; that is, Bob is able to obtain x if and only if he has c_i. Furthermore, if Bob's credential b_j is c_i, then $d_j = x$ in Step 1b. The second part of this (i.e., Bob can use c_i) is captured by the set $\mathcal{U}_B$; that is, Alice will have $r_i^B[1]$ if Bob can use c_i can she will have $r_i^B[0]$ otherwise. Putting these pieces together implies that "b_j equals c_i and Bob can use b_j" if and only if $x + r_j^B[d_j^B] = d_j + r_j^B[1]$. Thus the equality test for arrary elements protocol computes the desired value.

In Step 2 of the protocol Alice and Bob learn their shares of $\mathcal{U}_A$, that is Alice will learn a pair $(r_i^A[0], r_i^A[1])$ and Bob will learn $r_i^A[1]$ if and only if Alice can use credential a_i and he will learn $r_i^A[0]$ otherwise. Note that Alice can use credential a_i only if Bob's usable credential (computed in Step 1) satisfies Alice's policy for a_i. However, this is exactly

Input: Bob inputs: (1) a set of credentials, $\mathcal{C}_B$, which we denote by $b_1, \ldots, b_n$ and (2) his share of $\mathcal{U}_B$, which we denote by ordered pairs $(r_1^B[0], r_1^B[1]), \ldots, (r_n^B[0], r_n^B[1])$. Alice inputs: (1) a set of credentials, $\mathcal{C}_A$, which we denote by $a_1, \ldots, a_m$, (2) a set of policies for these credentials, $\mathcal{P}_A$, which we denote by $p_1, \ldots, p_m$, and (3) her share of $\mathcal{U}_B$, which we denote by $r_1^B[d_1^B], \ldots, r_n^B[d_n^B]$ (note d_i^B is 1 if Bob can use b_i and is 0 otherwise).

Output: Alice learns her share of the updated $\mathcal{U}_A$ which is denoted by ordered pairs $(r_1^A[0], r_1^A[1]), \ldots, (r_m^A[0], r_m^A[1])$. Bob learns his share of the updated $\mathcal{U}_A$ which is denoted by $r_1^A[d_1^A], \ldots, r_m^A[d_m^A]$, where $d_i^A = p_i(\mathcal{U}_B)$.

Protocol Steps:

1. *Determine which credentials in Alice's policies Bob has and can use*: Suppose that the credentials in $R(\mathcal{P}_A)$ are $c_1, \ldots, c_k$. Alice randomly generates k ordered pairs: $(t_1[0], t_1[1]), \ldots, (t_k[0], t_k[1])$. For each credential c_i, Alice and Bob engage in the following steps:
   (a) Alice picks a random number x, and sends $m = I(x, c_i)$ (the IBE encryption of x based on the hidden credential c_i) to Bob.
   (b) Bob decrypts m using each of his hidden credentials, and obtains $d_1, \ldots, d_n$, where $d_i = I^{-1}(m, b_i)$.
   (c) Alice creates a vector $\vec{a}_1 = \langle x + r_1^B[d_1^B], \ldots, x + r_n^B[d_n^B] \rangle$ and Bob creates a vector $\vec{a}_2 = \langle d_1 + r_1^B[1], \ldots, d_n + r_n^B[1] \rangle$. Alice and Bob engage in an equality test protocol for array elements where they each input their own array and Alice inputs $t_i[0]$ and $t_i[1]$. At the end of the protocol, Bob obtains $t_i[x_i]$. Note that x_i is 1 if and only if $c_i \in \mathcal{U}_B$ and Bob has c_i (that is Bob can use the credential and he actually has it) and is 0 otherwise.
2. *Compute $\mathcal{U}_A$:* For each credential a_i, Alice and Bob engage in the following steps:
   (a) Alice randomly generates an ordered pair $(r_i^A[0], r_i^A[1])$.
   (b) Alice and Bob securely evaluate p_i using blinded policy evaluation. Alice inputs $p_i, (r_i^A[0], r_i^A[1]), \{(t_1[0], t_1[1]), \ldots, (t_k[0], t_k[1])\}$ and Bob inputs $\{t_1[x_1], \ldots, t_k[x_k]\}$. At the end of the protocol Bob obtains $r_1^A[d_1^A]$.
3. *Alice and Bob produce $\mathcal{U}_A$:* Alice learns $(r_1^A[0], r_1^A[1]), \ldots, (r_m^A[0], r_m^A[1])$ and Bob learns $r_1^A[d_1^A], \ldots, r_m^A[d_m^A]$

Figure 5. Secure RE **strategy protocol** $\text{secure-reverse-eager-strategy}(\mathcal{C}_A, \mathcal{P}_A, \mathcal{C}_B, \mathcal{U}_B)$

what the blinded policy evaluation in Step 2 does.

Proof of Correctness/Security: A more detailed proof sketch is given in Section 8.

Cost analysis Steps 1(a)-1(c) are performed k times. Step 1(c) requires $O(n\rho^2)$ (where ρ is a security parameter) communication. Thus Step 1 requires $O(kn\rho^2)$ communication, but this can be reduced to $O(kn\rho)$ if the protocol in Section 7.1 is used for Step 1(c). Assuming that the policies can be computed with circuits that are linear in the number of credentials, Step 2 requires $O(mk\rho)$ communication. Now k is m_A, n is n_B, and m is n_A, and so this protocol requires $O(m_A\rho(n_A + n_B))$ communication (assuming policies can be computed by a circuit of size linear in the number of bits of their inputs).

6.3 Privacy-Preserving Trust Negotiation Protocol

We now "put the pieces together" and give the overall protocol for privacy-preserving trust negotiation. We describe the protocol in Figure 6.

Intuition of Correctness/Security: In Step 1 of the protocol, the server sets its set of usable credentials to all of its credentials (recall that the RE strategy protocol assumes everything is usable initially and that things are removed from this set).

In Step 2 of the protocol, the client and the server take turns updating their usable credential sets based on the other party's usable set. Once a set ceases to change then the usable sets will cease changing and we will have computed the maximal usable credential set. Note that since we are assuming monotonic policies this will take at most $\min\{n_C, n_S\}$ rounds to compute this set.

Finally, as we model the service as a credential s_1, the client will have $r_1^S[1]$ after Step 3 if and only if s_1 is in the $\mathcal{U}_S$.

Proof of Correctness/Security: A more detailed proof sketch is given in Section 8.

Cost analysis Step 2 of the protocol is executed $\min\{n_C, n_S\}$ (call this value n) times. An individual exe-

Input: The client has $\mathcal{C}_C$ and $\mathcal{P}_C$. The server has $\mathcal{C}_S$ (call these credentials $s_1, \ldots, s_{n_S}$) and $\mathcal{P}_S$. Furthermore, s_1 is the service that the client requested.

Output: If the trust negotiation between the client and server can succeed, then both the client and server output true, otherwise, they output false.

Protocol Steps:

1. *Initialize* $\mathcal{U}_S$. For each credential $s_i \in \mathcal{C}_S$, the server picks two random numbers $\{r_i^S[0], r_i^S[1]\}$. The server sends $r_i^S[1]$ to the client. The client calls this value $r_i^S[x_i]$
2. For $i = 1, \ldots, \min(n_C, n_S)$:
   (a) The client and server run the secure RE strategy protocol (Figure 5) to obtain $\mathcal{U}_C = \text{secure-reverse-eager-strategy}(\mathcal{C}_C, \mathcal{P}_C, \mathcal{C}_S, \mathcal{U}_S)$ in split form.
   (b) The server and client run the secure RE protocol (Figure 5) to obtain $\mathcal{U}_S = \text{secure-reverse-eager-strategy}(\mathcal{C}_S, \mathcal{P}_S, \mathcal{C}_C, \mathcal{U}_C)$ in split form.
3. *Output result.* To determine whether $s_1 \in \mathcal{U}_S$, the server sends a hash of $r_1^S[1]$ to the client. The client checks if the hash of $r_1^S[x_1]$ matches this value; if it is a match then the client proves this to the server by sending $r_1^S[x_1]$ to the server (and both parties output true), and if it is not a match the client terminates the protocol (and both parties output false).

Figure 6. Privacy-preserving trust negotiation protocol

cution requires $O(\rho(m_C + m_S)(n_C + n_S))$ communication and thus the protocol requires $O(n\rho(m_C + m_S)(n_C + n_S))$ communication.

7 Efficiency Improvements

7.1 A more efficient equality test for array elements

In this section, we introduce a more efficient protocol for the equality test for array elements. This protocol is related to the protocol proposed by [13] for secure set intersection. Figure 7 introduces this protocol. Note that this protocol requires only $O(n\rho + \rho^2)$ communication (instead of $O(n\rho^2)$ communication). We give the proof sketch of correctness and security in Section 8.

7.2 Reducing the number of rounds

A possible criticism of our protocol for trust negotiation is that it requires $O(\min\{n_C, n_S\})$ rounds. The RE strategy requires this many rounds in the worst case, but in practice it requires much less (it requires rounds proportional to the length of the longest policy chain). Our protocol can be modified to stop as soon as the usable credential sets cease changing. However, this is not recommended as it would leak additional information, and this information allows for additional probing. For example, if the negotiation requires 5 rounds then both parties can deduce that the other party does not satisfy at least 4 of their credentials. Thus, from a privacy standpoint terminating after the usable credential sets cease changing is not a good idea. Another option is to limit the number of rounds to some reasonable constant. This does not have privacy problems, but it could cause the negotiation to succeed where it would not have succeeded under Definition 2 of trust negotiation. However, if there is domain-specific knowledge that bounds the longest credential chain, then this is a viable option.

8 Security Proofs

In this appendix we discuss the security of our protocols. We first define what is meant by security. We then briefly (due to page constraints) sketch components of the proof of security.

8.1 Definition of Security

The security definition we use is similar to the standard model from the secure multi-party computation literature [15, 7]. The security of our protocol is analyzed by comparing what an adversary can do in our protocol against what an adversary can do in an ideal implementation with a trusted oracle. Specifically, we will show our protocol is no worse than this ideal model by showing that for any adversary in our model there is an adversary in the ideal model that is essentially equivalent. Thus if the ideal model is acceptable (in terms of security), then our protocols must also be acceptable.

Defining the ideal model for private trust negotiation is tricky. First, the ideal model has to be defined such that there are no “violations of security” that are achievable in

Input and Output: See Figure 4.

Protocol Steps:

1. Alice and Bob both choose semantically secure homomorphic encryption schemes E_A and E_B that share a modulus M and exchange public parameters.
2. Alice creates a polynomial P that encodes the x values where the constant coefficient is 1 (which can be done since this arithmetic is modular). In other words she finds a polynomial $P(x) = \eta_n x^n + \eta_{n-1}x^{n-1} + \cdots + \eta_1 x + 1$ where $P(x_i) = 0$ for all x_i. She sends to Bob $E_A(\eta_n), \ldots, E_A(\eta_1)$.
3. Bob chooses a value k_B uniformly from $\mathcal{Z}_M^\star$. For each y_i, Bob chooses a value $q_{B,i}$ uniformly from $\mathcal{Z}_M^\star$ and he computes $(E_A(P(y_i)))^{q_{B,i}} E_A(k_B + y_i) = E_A(q_{B,i}P(y_i) + k_B + y_i)$ (call this value $E_A(\alpha_i)$). Bob sends to Alice $E_A(\alpha_1), \ldots, E_A(\alpha_n), E_B(k_B)$.
4. Alice decrypts the values to obtain $\alpha_1, \ldots, \alpha_n$. She then computes $x_1 - \alpha_i, \ldots, x_n - \alpha_n$ She checks for duplicate values, and if there are duplicates she replaces all extra occurrences of a value by a random value. Alice chooses a value k_A uniformly from $\mathcal{Z}_M^\star$. For each of the values $x_i - \alpha_i$ she chooses $q_{A,i}$ uniformly from $\mathcal{Z}_M^\star$ and then she computes $(E_B(k_B)E_B(x_i - \alpha_i))^{q_{A_i}} E_B(k_A)$= $E_B((x_i + k_B - \alpha_i)q_{A,i} + k_A)$ (we will call this value $E_B(\beta_i)$). Alice sends to Bob $E_B(\beta_1), \ldots, E_B(\beta_n)$.
5. Bob decrypts the values to obtain $\beta_1, \ldots, \beta_n$. Bob then creates a polynomial Q that encodes these values where the constant coefficient is 1. In other words Bob finds a polynomial $Q(x) = \gamma_n x^n + \gamma_{n-1}x^{n-1} + \cdots + \gamma_1 x + 1$ where $Q(\beta_i) = 0$ for all β_i. Bob sends to Alice $E_B(\gamma_n), \ldots, E_B(\gamma_1)$.
6. Alice chooses two values k and q_A uniformly from $\mathcal{Z}_M^\star$ and computes $E_B(Q(k_A)q_A + k)$ and sends this value to Bob.
7. Bob decrypts this value to obtain k'. Alice and Bob engage in a scrambled circuit evaluation of an equality circuit where Alice is the generator with input k and she sets the encodings for the output wire to t_0 for the negative encoding and to t_1 for the positive encoding and Bob is the evaluator with input k'.

Figure 7. Secure Equality Test for Array Elements.

this ideal model; otherwise, there could be "violations of security" in our protocols. Furthermore, the ideal model must be defined in such a way as to allow useful trust negotiation to take place; otherwise it and our protocols will not be useful. This is further complicated by the fact that the RE strategy does not make sense in a non-private setting (as one cannot revoke knowledge from another party). Thus we define a fictitious environment where the parties have "chronic amensia" about the other party's credentials. In such an environment the RE strategy is plausible, and so our ideal model simulates this environment.

We now informally define an ideal model implementation of our scheme. In the ideal model the client sends $\mathcal{C}_C$ and $\mathcal{P}_C$ to the trusted oracle, and the server sends $\mathcal{C}_S$, $\mathcal{P}_S$, and s to the oracle. We model $\mathcal{P}_C$ and $\mathcal{P}_S$ as arbitrary PPT algorithms. These algorithms will simulate the parties' behavior during the RE strategy. Thus these algorithms should be viewed as control algorithms that: (1) define which credentials to use during each round, (2) define the access control policies (which we model as PPT algorithms over the other party's currently usable credentials) for its credentials during each round, and (3) can force the oracle to terminate. We stress that these algorithms cannot do the above operations based upon the state of the negotiation. For example, they cannot force the oracle to terminate when a specific credential becomes unusable. The oracle will simulate the RE strategy using the access control policies defined by each party's control algorithm. At the end of the negotiation the oracle will inform the client and the server whether access is granted. Due to page limitations we do not discuss the above ideal model in more detail.

8.2 Sketch of the Security Proof

We will now sketch part of the proof. As it is too lengthy to include in full detail, we focus only on one specific aspect of the system. We focus on the Secure Reverse Eager strategy protocol (which is the key component of our system). We first show that if Alice is honest, then Bob cannot influence the outcome of the protocols so that he unrightfully keeps one of Alice's credentials usable.

Lemma 4 *In the secure* RE *strategy protocol (Figure 5): If Alice is honest and after the protocol a specific credential a_i (with policy p_i) is in $\mathcal{U}_A$, then Bob has a credential set $\mathcal{C}_B$ such that $p_i(\mathcal{C}_B)$ is true.*

Proof: Because step 2 is done by SCE and Alice is an honest generator, by Lemma 5 all that we must show is that after step 1, Bob learns $t_i[1]$ only when he has credential a_i.

By way of contradiction, suppose Bob does not have credential a_i, and that he learns $t_i[1]$ in Step 1c. By Lemmas 6 and 7, Bob only learns $t_i[1]$ when there is a match in the arrays created by Alice and Bob in Step 1c. If there is a match, then Bob must be able to learn x with a non-negligible probability, but this implies that he can invert the IBE encryption with non-negligible probability, but this contradicts that the IBE encryption scheme is secure. □

Lemma 5 *In scrambled circuit evaluation: If the generator is honest and the evaluator learns at most one encoding for each input wire, then the evaluator learns at most one encoding for the output wire; furthermore this encoding is the correct value.*

Proof: We omit the details of this lemma, but similar lemmas are assumed in the literature □.

Lemma 6 *In the circuit-version of the equality test for array elements: If Alice is honest, Bob learns t_1 only when there is an index i such that $x_i = y_i$.*

Proof: Since Alice is the generator of the circuit and is honest, Bob will input a set of y values and will learn t_1 only when one of his y values matches one of Alice's x values (by Lemma 5). □

Lemma 7 *In the other version (Figure 7) of the equality test for array elements: If Alice is honest, Bob learns t_1 only when there is an index i such that $x_i = y_i$.*

Proof: By way of contradiction, suppose Bob learns t_1 and there is no match in their arrays. In Step 7 of the protocol Bob must know the value k (by Lemma 5). Thus in Step 5 of the protocol, Bob must be able to generate a non-zero polynomial of degree n that has k_A as a root, but this implies he knows k_A with non-negligible probability. This implies that in Step 3, Bob can generate values $\alpha_1, \ldots, \alpha_n$ such that there is an α value that is $x_i + k_B$. This implies Bob knows x_i with non-negligible probability, and this implies that there is a match in the arrays. □

The above only shows one part of the proof. We must also show that if Alice is honest, Bob cannot learn whether he made a specific credential usable (he can force a credential to be unusable, but this has limited impact). Furthermore, we must show that if Bob is honest that Alice does not learn which of her credentials are usable (other than what can be deduced from her policies; i.e., a globally usable credential will definitely be usable). These proofs will be in the full version of the paper. We now show that the protocol is correct, that is if the parties are honest, then the correct usable set is computed.

Proof: In step 1 of the protocol, Bob learns a value $t_i[x_i]$ where x_i is 1 if Bob has credential c_i and can use it. There are 3 cases to consider:

1. *Bob does not have c_i*: In Step 1b of the protocol, Bob will not learn the value x, and thus there will not be a match in Step 1c (with very high probability). Since there is no match in the array, Bob will learn $t_i[0]$, which is correct.
2. *Bob has c_i but cannot use it.* Suppose $b_j = c_i$ and Alice has $r_j^B[0]$. In this case, $d_j = x$, but Bob's vector entry will be $x + r_j^B[1]$ and Alice's will be $x + r_j^B[0]$. Since there is no match in the array, Bob will learn $t_i[0]$, which is correct.
3. *Bob has c_i and can use it.* Suppose $b_j = c_i$ and Alice has $r_j^B[1]$. In this case, $d_j = x$, but Bob's vector entry will be $x + r_j^B[1]$ and Alice's will be $x + r_j^B[1]$. Since there is a match in the array, Bob will learn $t_i[1]$, which is correct.

In step 2 of the protocol, Alice and Bob securely evaluate p_i based upon which credentials are in $\mathcal{U}_B$. If $p_i(\mathcal{U}_B)$ is true, then Bob will learn $r_i^A[1]$ (signifying that Alice can use a_i) and otherwise he will learn $r_i^A[0]$ (signifying that Alice cannot use a_i). □

9 Related Work

Our work is originally motivated from the existing automated trust negotiation literature [4, 30, 25, 34, 33, 29] whose goal is to enable trust establishment between strangers in a decentralized or open environment, such as the Internet. In trust negotiation, each party establishes access control policies to regulate not only the granting of resources, but also the disclosure of credentials (and possibly policies) to others. A negotiation begins when a party requests access to a resource that is protected by an access control policy. The negotiation process consists of a sequence of cautious and iterative exchanges of credentials and possibly access control policies. In successful negotiations, disclosed credentials eventually satisfy the access control polices of the desired resource. A security requirement for trust negotiation is that no credential should be disclosed unless its access control policy has been satisfied. The concept of privacy protection in the previous trust negotiation schemes differs from the one in our scheme. In existing trust negotiation schemes, a resource (e.g., a service, a credential, or a policy) is revealed and delivered to the other party, when the policy for the source has been satisfied. In our framework, neither the credentials nor the policies are revealed to the other party, even when the policies for the resource and the credentials are satisfied. Furthermore, all of the intermediate results of the negotiation remain unknown to each participant. Thus, our scheme offers better privacy protection than the existing schemes.

Recent work on using cryptographic protocols for trust negotiation includes hidden credentials [17, 6, 14], secret handshakes [2], oblivious signature based envelope [20], oblivious attribute certificates [18, 19], and policy-based cryptography [1]. In these protocols, Alice has some private credentials (or attribute values), Bob has a policy (that may or may not be private), Alice and/or Bob want to determines whether Alice's credentials satisfy Bob's policy. While these protocols are useful in general and can be integrated into trust negotiation systems as valuable building blocks, none of the protocols address the SCALP problem, i.e., Alice's credentials are not protected by any of her policies in those protocols. Therefore, our work is substantially different from this other work.

Our problem is closely related to Secure Function Evaluation (SFE) [31, 16, 15]. In SFE, Alice has an input x, Bob has an input y, Alice and Bob want to compute $f(x, y)$, where f is public to both of them. Elegant general constructions have been developed to solve any SFE problems [31, 16, 8]. Our paper uses two-party SFE techniques, however, it is not a routine usage of these techniques because (1) we had to first propose a suitable overall strategy for the negotiation (i.e., what "overall global function" to compute in the first place); and (2) in the standard SFE problems, neither party's inputs are certified, but in our problem, some of the inputs are verified off-line by a third party (recall that Alice and Bob input their credentials issued by the CA instead of directly providing their attributes to the protocol), and verifying the credentials using the general solutions to SFE is expensive.

10 Conclusion

In this paper, we gave an efficient protocol for Alice and Bob to negotiate trust, such that Alice does not learn Bob's credentials and policies, and Bob does not learn Alice's credentials and policies. The only information they learn is whether the trust between them can be established, or in other words, whether Alice is eligible for Bob's service or resource. Our work is a substantial extension of the state-of-the-art in privacy-preserving trust negotiations. The details of our work contain technical results of independent interest, such as the secure protocol for an equality test for array elements.

References

[1] W. Bagga and R. Molva. Policy-based cryptography and applications. In *Proceedings of the 9th International Conference on Financial Cryptography and Data Security*, Feb. 2005.

[2] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H.-C. Wong. Secret handshakes from pairing-based key agreements. In *Proceedings of the IEEE Symposium and Security and Privacy*, pages 180–196, May 2003.

[3] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.

[4] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS-7)*, pages 134–143. ACM Press, Nov. 2000.

[5] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Proceedings of Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[6] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of 11th ACM Conference on Computer and Communications Security*, Oct. 2004.

[7] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[8] R. Canetti, Y. Ishai, R. Kumar, M. K. Reiter, R. Rubinfeld, and R. N. Wright. Selective private function evaluation with applications to private statistics. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 293–304. ACM Press, 2001.

[9] C. Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA International Conference on Cryptography and Coding*, volume 2260, pages 360–363. Springer, Dec. 2001.

[10] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 119–136. Springer, 2001.

[11] I. Damgård and M. Jurik. A length-flexible threshold cryptosystem with applications. In *Proceedings of the 8th Australasian Conference on Information Security and Privacy*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2003.

[12] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. IETF RFC 2693, Sept. 1999.

[13] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology: EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

[14] K. B. Frikken, M. J. Atallah, and J. Li. Hidden access control policies with hidden credentials. In *Proceedings of the 3rd ACM Workshop on Privacy in the Electronic Society*, Oct. 2004.

[15] O. Goldreich. *The Foundations of Cryptography — Volume 2*. Cambridge University Press, May 2004.

[16] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, May 1987.

[17] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society*, Oct. 2003.

[18] J. Li and N. Li. OACerts: Oblivious attribute certificates. In *Proceedings of the 3rd Conference on Applied Cryptography and Network Security (ACNS)*, volume 3531 of *Lecture Notes in Computer Science*. Springer, June 2005.

[19] J. Li and N. Li. Policy-hiding access control in open environment. In *Proceedings of the 24nd ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, July 2005.

[20] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, July 2003.

[21] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, May 2002.

[22] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – secure two-party computation system. In *Proceedings of the 13th USENIX Security Symposium*, pages 287–302. USENIX, 2004.

[23] T. Okamoto, S. Uchiyama, and E. Fujisaki. Epoc: Efficient probabilistic public-key encryption. In *IEEE P1363: Protocols from other families of public-key algorithms*, Nov. 1998.

[24] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology: EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[25] K. E. Seamons, M. Winslett, and T. 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.

[26] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[27] L. G. Valiant. Universal circuits (preliminary report). In *STOC '76: Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 196–203, New York, NY, USA, 1976. ACM Press.

[28] W. H. Winsborough and N. Li. Towards practical automated trust negotiation. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002)*, pages 92–103. IEEE Computer Society Press, June 2002.

[29] W. H. Winsborough and N. Li. Safety in automated trust negotiation. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 147–160, May 2004.

[30] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, pages 88–102. IEEE Press, Jan. 2000.

[31] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, 1986.

[32] T. Yu, X. Ma, and M. Winslett. Prunes: An efficient and complete strategy for trust negotiation over the internet. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS-7)*, pages 210–219. ACM Press, Nov. 2000.

[33] T. Yu and M. Winslett. Unified scheme for resource protection in automated trust negotiation. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 110–122. IEEE Computer Society Press, May 2003.

[34] T. Yu, M. Winslett, and K. E. Seamons. Interoperable strategies in automated trust negotiation. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS-8)*, pages 146–155. ACM Press, Nov. 2001.

A The Eager Strategy

In this appendix we review the eager strategy [30]. Recall that the goal of the eager strategy is to compute a credential disclosure sequence that contains the requested service. In the eager strategy, each negotiator iteratively executes the pseudo-code in Figure 8. The negotiation succeeds if s appears in the output (i.e., $s \in \mathcal{M}$), and it fails if the size of the credential disclosure sequence does not increment after one round of execution (i.e., $\mathcal{M} = \emptyset$). Note that any negotiation using the eager strategy takes at most $\min(n_S, n_C)$ rounds, where n_S and n_C are the sizes of $\mathcal{C}_S$ and $\mathcal{C}_C$, respectively. The following is an example of trust negotiation using the eager strategy.

```
The Eager Strategy($\mathcal{D}, \mathcal{C}, \mathcal{P}, s$)
    $\mathcal{D} = \{c_1, \ldots, c_k\}$: the credential disclosure sequence.
    $\mathcal{C}$: the local credentials of this party.
    $\mathcal{P}$: the local policies of this party.
    $s$: the service to which access was originally requested.
Output:
    $\mathcal{M}$: the set of new released credentials.
Pre-condition:
    $s$ has not been disclosed.
Procedure:
    $\mathcal{M} = \emptyset$;
    For each credential $c \in \mathcal{C}$
        let $c$'s policy be $p_c : c \leftarrow \phi_c$;
        if $\phi_c(\mathcal{D}) = 1$, then $\mathcal{M} = \mathcal{M} \cup \{c\}$;
    $\mathcal{M} = \mathcal{M} - \mathcal{D}$;
    return $\mathcal{M}$.
```

Figure 8. Pseudocode for the Eager Strategy

Example 2 Suppose the client and server have the following policies:

$Client$	$Server$
$p_{c_1} : c_1 \leftarrow s_1$	$p_s : s \leftarrow c_5 \vee (c_2 \wedge c_4)$
$p_{c_2} : c_2 \leftarrow s_2 \wedge s_3$	$p_{s_1} : s_1 \leftarrow c_4$
$p_{c_3} : c_3 \leftarrow s_1 \vee s_2$	$p_{s_2} : s_2 \leftarrow c_1$
$p_{c_4} : c_4 \leftarrow \mathsf{true}$	$p_{s_3} : s_3 \leftarrow \mathsf{true}$

where s denotes the server's service, $\{s, s_1, s_2, s_3\}$ denote the set of server's credentials, $\{c_1, c_2, c_3, c_4\}$ denotes the set of the client's credentials. Using the eager strategy, the client begins by revealing credential c_4, as the policy function for c_4 is `true` (thus it is trivially satisfied). The server then discloses s_3 (which can be revealed freely) and s_1 (which requires the earlier receipt of c_4). The exchange of credentials continues as the final disclosure sequence is $\{c_4, s_1, s_3, c_1, c_3, s_2, c_2, s\}$. Note that all policies for disclosed credentials have been satisfied.

B Protocol for Blinded Policy Evaluation

Figure 9 describes how to achieve blinded policy evaluation, which is a natural extension of Yao's circuit simulation protocol [31].

1. Alice constructs a circuit C that computes her policy (several "useful circuits" are described below) that uses the r_i values as inputs and that has an output wire with two encodings: t_1 for true and t_0 for false. She sends the encodings of the circuit's gates to Bob (note that he already has input encodings).
2. Bob evaluates the circuit and learns the encoding for the output wire.

Figure 9. Blinded Policy Evaluation Protocol

The protocol for Blinded Policy Evaluation uses a circuit to evaluate the policy. This reveals the topology of the circuit to the evaluator (which reveals some information to the evaluator). However, one can build a topology that covers a large class of functions; this can be achieved in several ways including: (1) building a topology that can handle many useful functions, (2) using a universal circuit [27], and (3) using a single n-ary gate for arbitrary functionality (this latter option requires exponential communication). There are several interesting circuit topologies with size linear in the number of inputs to the circuit, including:

1. It is easy to construct an oblivious comparison circuit (i.e., one that can compute $=$, $\neq$, $>$, $<$, $\geq$, and $\leq$ without revealing which comparison is done) with size proportional to the number of bits in the values.

2. A binary tree of oblivious gates (with inputs $a_1, \ldots, a_n$) can be used to compute many useful functions (without revealing which function is being computed) including:

   (a) $\bigwedge_{i=1}^{n} a_i$, $\bigvee_{i=1}^{n} a_i$, $\bigoplus_{i=1}^{n} a_i$, etc.

   (b) For any subset of the values S, $\bigwedge_{i \in S} a_i$,$\bigvee_{i \in S} a_i$, $\bigoplus_{i \in S} a_i$, etc.

   (c) Other functions like: for a subset S_1 of the first half of the values and another subset S_2 of the second half of the values, the function $\bigvee_{i \in S_1} a_i \wedge \bigvee_{i \in S_2} a_i$.