Minimal credential disclosure in Trust Negotiations

Federica Paci[1], David Bauer[2], Elisa Bertino[1],Douglas M. Blough[2],
Anna Squicciarini[3]

[1]Cerias and Computer Science Department, Purdue University, West Lafayette,IN

{paci,bertino@cs.purdue.edu}

[2]Georgia Institute of Technology, Atlanta, GA

{gte810u@mail.gatech.edu,doug.blough@ece.gatech.edu}

[3]College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA, USA

{acs20@psu.edu}

Abstract

The secure release of identity attributes is a key enabler for electronic business interactions. In particular, integrity and confidentiality of identity attributes are two key requirements in such context. Users should also have the maximum control possible over the release of their identity attributes and should state under which conditions these attributes can be disclosed. Moreover, users should disclose only the identity attributes that are actually required for the transactions at hand. In this paper we present an approach for the controlled release of identity attributes that addresses such requirements. The approach is based on the integration of trust negotiation and minimal credential disclosure techniques. Trust negotiations support selective and incremental disclosure of identity attributes, while minimal credential disclosure guarantees that only the attributes necessary to complete the on line interactions are disclosed.

1 Introduction

As activities such as shopping, discussion, entertainment and business collaboration are increasingly being conducted in the digital world, identity management systems have became fundamental to underpinning accountability in business relationships, controlling the customization of the user experience, protecting privacy, and adhering to regulatory controls. Digital identity can be defined as the digital representation of the information known about a specific individual or organization. As such, it encompasses not only login names, but many additional information, referred to as *identity attributes*. Managing identity attributes raises a number of challenges. On the one hand, identity attributes need to be shared to speed up and facilitate authentication of users and access control. On the other hand, they need to be protected as they may convey sensitive information about an individual and can be a target of attacks like identity theft. Therefore, the secure release of identity attributes is a key enabler for electronic business interactions. Key requirements for security in such context are integrity and confidentiality. Integrity requires data not to be altered in an unauthorized way. Confidentiality deals with the protection of sensitive identity information from unauthorized disclosure. Identity information should only be accessible by the intended recipients. If an attacker can retrieve other's information, then users lose control on their attributes release and usage. It is therefore essential that mechanisms for confidential release of individuals' attributes be provided. Users should also have the maximum control possible over the release of their identity attributes and should state under which conditions the attributes can be disclosed. Moreover, users should be able to disclose only the identity attributes actually needed for authentication, identity verification, and access control according to the well known *least disclosure principle*. Therefore, there is the need for tools and systems able to guarantee integrity, confidential-

ity and minimal disclosure of identity attributes.

One approach to achieve flexibility and fine grained control over the usage of identity attributes is to adopt automated trust negotiation techniques [2, 18, 19, 20]. Trust negotiation is an approach for establishing trust in open systems, like the Internet. The idea of trust negotiation is to establish trust online between (generally) two negotiating parties through bilateral credential disclosure. Digital credentials are assertions stating one or more identity attributes of a given subject, referred to as the *owner*, certified by trusted third parties called certification authorities (CAs). A key aspect of trust negotiation is that sensitive credentials may be protected by disclosure policies specified by credential owners. Disclosure policies state the conditions under which credentials can be released during a negotiation. Conditions are usually expressed as constraints against the credentials possessed by the interacting parties and the identity attributes encoded in the credentials. Therefore, trust negotiation supports the selective and incremental disclosure of identity attributes but does not guarantee compliance with the least disclosure principle. In conventional trust negotiations, users may exchange credentials that also contain identity attributes that are not actually requested by the counterpart's disclosure policies.

An interesting approach supporting fine-grained disclosure of identity attributes by at the same time simplifying management of these attributes has been proposed by Bauer et al. [1]. Such approach is based on the notion of *micro-claims*, that is, predicates specified in terms of identity attributes. Examples of birth-date-related micro-claims are "Age $\geq$ 18", and "Age $\geq$ 21". The set of all micro- claims derived from the identity attributes of a user is collected in a unique credential. The credential is represented as a Merkle hash tree [15] in which the leaf nodes represent the micro-claims. The use of such Merkle hash tree credentials makes it possible to dynamically specify an arbitrary subset of micro-claims for a given interaction without disclosing the other micro-claims. Such approach also assures integrity of the micro-claims because the credential is released and signed by a certification authority. The major drawback of such approach is that it requires the users to determine which micro-claims to release for a given transaction, and thus it would have to be integrated with a tool able to automatically select, on behalf of the user, the micro-claims to release.

The discussion about the two approaches, namely trust negotiation and minimal credential disclosure, clearly show that their integration is the key to achieve solutions to the problem of identity attribute disclosure addressing all requirements that we have outlined. To date, however, there is no system combining those two approaches. In this paper, we make a step towards the development of such systems and discuss how to extend the Trust-$\mathcal{X}$ negotiation framework [2] with the minimal credential disclosure technique. The resulting system allows users to disclose only the identity attributes necessary to satisfy the counterpart disclosure policies. Rather then sending all the credentials requested by the counterpart policies, clients, on behalf of their users, send only one credential that contains the micro-claims about the identity attributes specified in the counterpart's disclosure policies.

The integration of the two approaches is not trivial. Since the verification that clients have certain properties is based on proving they own certain micro-claims, we had to extend the Trust-$\mathcal{X}$ negotiation language in order to specify disclosure policies that protect the release of single attributes rather than credentials, and that express conditions against these attributes. Moreover, since clients should be able to automatically select the micro-claims that satisfies counterpart disclosure policies, we had to devise different strategies to match the micro-claims with the attributes in the disclosure policies and to select only the micro-claims that satisfy the policies.

The rest of the paper is organized as follows. Section 2 provides an overview of the negotiation language and of the negotiation protocol currently supported by Trust-$\mathcal{X}$. Section 3 presents a short summary about the Merkle hash tree credential structure, and describes how a credential can be built based on micro-claims from different credentials and how these micro-claims can be verified. Section 4 discusses integration issues and then presents the negotiation protocol based on the use of the minimal disclosure techniques. Section 5 discusses the implementation of the integrated approach. Section 6 outlines related work and Section 7 concludes the paper.

2 Trust Negotiation overview

In this section we present Trust-$\mathcal{X}$ [2], a comprehensive XML-based framework for trust negotiations specifically

conceived for peer-to-peer environments. We first give an overview of $\mathcal{X}$-TNL, the negotiation language supported by Trust-$\mathcal{X}$ and then we provide an overview of Trust-$\mathcal{X}$ negotiation protocol.

2.1 Trust-$\mathcal{X}$ language

$\mathcal{X}$-TNL is the XML-based language developed to specify information required to carry on trust negotiations, namely credentials and disclosure policies. $\mathcal{X}$-TNL credentials are the means to convey information about the profile of the parties involved in the negotiation. A credential is a set of properties of a party issued by a CA. All credentials associated with a party are collected into a unique XML document, referred to as $\mathcal{X}$- *Profile*. Trust-$\mathcal{X}$ disclosure policies state the conditions under which a resource or a credential can be released during a negotiation. Conditions are expressed as constraints on the attribute credentials owned by the parties involved in the negotiation. Each party adopts its own Trust-$\mathcal{X}$ set of disclosure policies to regulate release of local information (that is, credentials or policies) and access to services.

Like credentials, disclosure policies are encoded using XML. Regardless of the specific implementation, disclosure policies can be modeled as logic rules. Two building blocks for specifying disclosure policies are *terms* and *R-Terms*. A term is an expression of form $P(C)$ where P is a credential type and C is a (possibly empty) list of conditions on the attributes encoded in credentials of type P. The credential type P can be unspecified (and denoted by a variable), so to express constraints on the counterpart properties without specifying from which types of credential such properties should be obtained from. Such approaches gives the receiver of the policy the flexibility of choosing which credentials to send as a proof of policy satisfiability. $R\text{-}Terms$ are expressions of the form $ResName(attrset)$ where $ResName$ denotes a resource name whereas $attrset$ denotes a set of attributes, specifying relevant characteristics of the resource. Examples of resources are a credential, a file or a Web service.

As such, disclosure policies can assume one of the following forms:
1) $\mathbf{R} \leftarrow \mathcal{T}_\mathbf{1}, \mathcal{T}_\mathbf{2}, ., \mathcal{T}_\mathbf{n}$, n$\geq 1$, where $\mathcal{T}_\mathbf{1}, \mathcal{T}_\mathbf{2}, ., \mathcal{T}_\mathbf{n}$ are terms and $\mathbf{R}$ is an $R\text{-}Term$ identifying the name of the target resource.
2) $\mathbf{R} \leftarrow \mathbf{DELIV}$. A rule of this form is called *delivery rule*, meaning that R can be delivered as is.

A disclosure policy is *satisfied* if the stated credentials are disclosed to the policy sender and the policy conditions (if any) evaluated as true, according to the specific credential content. A delivery rule implies that resource R is ready to be released, and no specific requirement has to be satisfied.

Example 2.1 *The following are examples of disclosure policies:*

- Driving License ← Age>18
- PictureID ← BMV(State = Indiana)

The first policy states that in order to obtain a Driving License credential for the Indiana State the user must prove to be older than 18. The second policy says that in order to release his PictureID, the user wants to see a credential proving that the counterpart is an authorized Bureau of Motor Vehicle branch of the State of Indiana.

2.2 Trust-$\mathcal{X}$ negotiation process

Trust negotiation is the approach to establish a mutual trust relationship between two parties that do not know each other and want to exchange on line resources or services. Trust is established through an exchange of digital credentials. Such credentials are identified during the negotiation process, within which each party decides which credential is willing to disclose to the counterpart and under which conditions.

In Trust-$\mathcal{X}$, the negotiation is performed in two main phases: the *policy evaluation phase* and the *credential exchange phase*. The key phase of a Trust-$\mathcal{X}$ negotiation is the *policy evaluation phase*, which consists of a bilateral and ordered policy exchange. The goal is to determine a sequence of credentials, called *trust sequence*, satisfying the disclosure policies of both parties. During each interaction, one of the two parties sends a set of disclosure policies to the other. The receiving party verifies whether its $\mathcal{X}$- *Profile* satisfies the conditions stated by the policies, and whether there are local policies regulating the disclosure of the credentials requested by the policies sent by the other party. If this is the case, the receiving party sends to the other party the disclosure policies protecting the credentials requested by the other party. Otherwise,

the receiver informs the other party that it does not possess the requested credentials. The counterpart then sends an alternative policy, if any, or it halts the process, if no other policies can be found. The interplay goes on until one or more potential trust sequences are determined, that is, whenever both parties determine one or more set of policies that can be satisfied for all the involved resources. To maintain the progress of a negotiation and help detecting a potential trust sequence a tree structure is used. The trust sequence is identified by one tree view, where a view denotes a possible trust sequence that can lead to the negotiation success. The view keeps track of which terms may need to be disclosed to contribute to the success of the negotiation, and of the correct order of certificate exchange. More precisely, a negotiation tree is a labeled tree rooted at the resource that initially started the negotiation. Each node corresponds to a term, whereas edges correspond to policy rules. A negotiation tree is characterized by two different kinds of edges: simple edges and multiedges. A simple edge denotes a policy having only one term on the left side component of the rule. By contrast, a multiedge links several simple edges to represent policy rules having more than one term on their left side component. Nodes belonging to a multiedge are thus considered as a whole during the negotiation.

Example 2.2 *Figure 1 represents an example of negotiation tree. The tree represents the negotiation between a party that requests the release of Driving License and a Bureau of Motor Vehicle (BMV) branch of the State of Indiana. The BMV branch sends to the party two alternative policies for the release of the Driving License. The party has to prove that he is older than 18 or that he has a valid PictureID from the State of Indiana. The party, in order to give proof he is older of 18, wants the BMV branch to prove that is a certified branch of Indiana State Government.*

Once the parties have agreed on a trust sequence, the *credential exchange phase* begins. Each party discloses its credentials, following the order defined in the trust sequence, eventually retrieving those credentials that are not immediately available through credentials chains. Upon receiving a credential, the counterpart verifies the satisfaction of the associated policies, checks for revocation, checks validity dates, and authenticates the ownership (for credentials). The receiving party then replies with an acknowledgment, and asks for the subsequent credential in the sequence, if any. Otherwise, a credential belonging to the subsequent set of credentials in the trust sequence is sent. The process ends with the disclosure of the requested resource or, if any unforeseen event happens, an interruption. If the failure is related to trust, for example a party uses a revoked certificate, the negotiation fails.

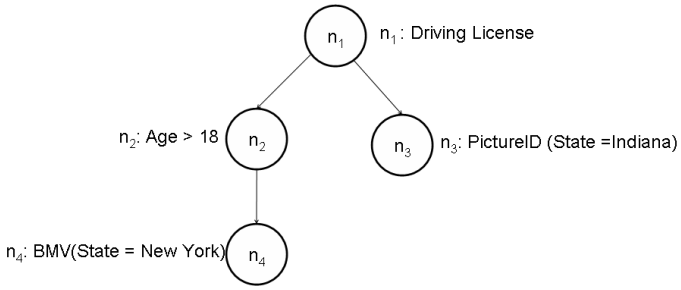

Figure 1: Simple example of negotiation tree

3 Minimal credential disclosure

Users are often required to reveal more information than is necessary when authorizing digital interactions. For example, they might be required to supply their birth date in order to prove that they are at least 18. If such information falls into the wrong hands, there is a serious potential for misuse. Information such as birth date is often used as a secondary identifier by service providers and, therefore, revealing such information unnecessarily can increase the risk of identity fraud. The minimal disclosure credential concept assumes that users' personal information is stored as a set of *micro-claims*. Examples of birth-date-related micro-claims are "Age $\geq$ 18", "Age $\geq$ 21", "Age $\geq$ 25", etc. Similar micro-claims can be constructed for almost any category of personal information. Once information is stored in this manner, minimal disclosure credentials allow users to reveal, in a manner easily verifiable by relying parties, only the minimum set of micro-claims necessary for a particular transaction. This allows users to minimize the amount of their personal information that is revealed to, and possibly maintained by, each different party with which they interact.

The core of a minimal disclosure credential is a Merkle

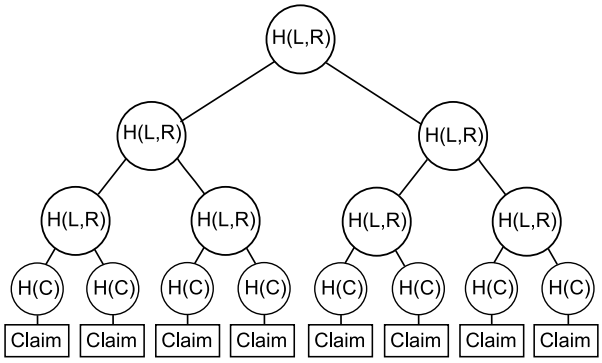

Figure 2: Merkle Hash Tree with Labeled Nodes

hash tree structure, which is similar to the redactable signature scheme of [11]. In the credentials that use this structure, hashes of micro-claims are represented as leaves in a Merkle hash tree, as shown in Figure 2.

The credential consists of two parts: a public part and a private part. The public part of the credential is a certificate. The certificate holds information about the issuer, the certification chain for the issuer, the type of certificate, the date range over which the certificate is valid, the user's public key, and a signature of the root node of a Merkle hash tree. The certificate should not in general hold any data about the user directly, even data as common as a name. As per standard operation, the certificate will be signed by a certificate authority. The private part of the credential consists of a private key and a Merkle hash tree whereby all of the leaf nodes are attributes of, or micro-claims about, the identity of the user, who is the credential holder. A user who wishes to assert an arbitrary subset of micro-claims supplies the certificate, the values of the asserted micro-claims along with their positions in the hash tree, and whatever additional hash tree intermediate values are necessary to reconstruct the root hash value, which the relying party can then validate against the root hash value in the certificate.

The basic minimal disclosure credential scheme has been extended to allow micro-claims from multiple authorities to be combined within a single credential, while still allowing a user to supply an arbitrary subset of the micro-claims from all authorities [1]. This allows users to have multiple "identity providers" that supply micro-claims only related to their particular subject area, e.g. an employer might maintain employment-related micro-claims, a professional organization might verify membership and specific certifications obtained by the user, a state's Department of Motor Vehicles might maintain address, driver's license category and status, and other miscellaneous items, etc. [1] also presents a detailed performance analysis of minimal disclosure credentials and comparison with other types of anonymous credentials. It is this multiple-authority-capable minimal disclosure credential, which is used in our trust negotiation environment described herein.

4 Trust Negotiation with Minimal Disclosure

In this section we present an enhanced version of Trust-$\mathcal{X}$ negotiation protocol that satisfies the least disclosure principle for credentials. Under such protocol the negotiating parties disclose only the micro-claims about the identity attributes necessary to satisfy counter party's disclosure policies.

4.1 Integration Issues

The integration of minimal credential disclosure techniques in trust negotiations is not trivial and requires to extend both the negotiation language and the protocol.

Trust negotiations are carried out on the basis of disclosure policies that regulate the release of sensitive credentials, while the approach for minimal credential disclosure is based on micro-claims that express conditions on user's identity attributes not related to a particular credential type. As such, $\mathcal{X}$-TNL language must be extended in order to specify disclosure policies that protect the release of attributes rather than credentials. We thus extend the definition of disclosure policies with expressions of the form $\mathcal{A} \leftarrow \mathcal{AT}_1, ..., \mathcal{AT}_n$ where $\mathcal{A}$ is the name of an attribute and terms $\mathcal{AT}_i$ are attribute conditions of the form $a\ op\ expr$. a denotes an attribute name, op is a comparison operator, such as =, $\neq$, $\leq$, $\geq$, $<$, $>$ and $expr$ denotes a constant or a variable name. At the negotiation protocol level challenging issues are how to match the attribute names in the disclosure policies' terms with the attribute names in the micro-claims and how to select the

micro-claims that make true the terms. In fact, the negotiating parties might use different vocabularies and hence the terms in the policies and the micro-claims might have *syntactic* and/or *semantic* differences in attribute names and categorical values. An example of syntactic difference is the use of 'date-of-birth' and 'birthday' to refer to the date of birth of an individual. An example of semantic variation is a case in which the synonym 'job' is used to refer to users' employment attribute. Matching attribute names that have syntactic differences is easy. By contrast, when there are semantic differences, it is necessary to use online dictionaries [22] to derive synonyms or to associate ontologies with attribute names and categorical values and to use ontologies matching algorithms [6]. Once the match between attribute names in the disclosure policies' terms and the ones in the micro-claims is performed, it is important to select the micro-claims that satisfy the terms. Since there might be multiple micro-claims for the same attribute name, different selection strategies can be adopted. A naive strategy is to use string matching to select the micro-claims that satisfy a policy term. This strategy is very restrictive because if there are no micro-claims that textually match a policy term, the term cannot be satisfied even though there are micro-claims that logically match or imply the condition specified in the term. For example, assume that the birthday of a user is 04-15-1978 and hence the user is 30 years old. Suppose that the user has to prove that is age is greater than 18, that is, he has to prove the disclosure policy term "Age $>$ 18". Suppose that the following micro-claims are recorded in his Merkle hash credential: "Age $>$ 16", "Age $>$ 21", "Age $>$ 25", and "Age $<$ 45". It is clear that under the string matching strategy, no one of those micro-claims matches the policy term, even though the micro-claims "Age $>$ 21" and "Age $>$ 25" logically imply the condition "Age $>$ 18" and therefore would "logically" satisfy the policy term.

To address such issue, the most intuitive strategy is the one that we refer to as *tight bind strategy*. Under such strategy, the selected micro-claim is the one that logically implies the term in the disclosure policy and that is closer to the actual value assumed by the attribute from which the micro-claim is derived. Though this strategy is intuitive, it makes it possible for the service provider (or any party receiving the micro-claim) to obtain a tighter bound on the actual value of the attribute. In the scenario described above, where a user has prove to be older than 18, the tight bind strategy would select the micro-claim "Age $>$ 25". If a malicious party knows that the tight bind strategy is adopted, it can reduce the uncertainty about the actual age of the user. An alternative strategy, that we adopt, is to select the micro-claim which, not only logically implies the policy term, but also has the farthest value from the actual value of the attribute. Such strategy, that we refer to as *loose bind strategy*, provides a less tight bound about the actual value of the attributes and thus provides a higher uncertainty about the actual values of the attribute, as compared to the tight bind strategy. According to this strategy, if a user has to prove to be older than 18 in the scenario described above, the loose bind strategy will select the micro-claim "Age $>$ 21".

4.2 Negotiation protocol with minimal credential disclosure

The adoption of the minimal credential disclosure in Trust-$\mathcal{X}$ requires to extend the profiles of parties to store Merkle hash tree credentials, to modify the strategy according to which a party evaluates the satisfiability of disclosure policies sent by the counterpart, and the strategy according to which the credential exchange phase is executed.

We thus extend users $\mathcal{X}$- *Profile*s to include the credentials according to the Merkle hash tree structure, with all the possible micro-claims about users identity attributes and all the credentials that certify that the users have these identity attributes. For each term $\mathcal{AT}_i$ in the disclosure policy $A \leftarrow \mathcal{AT}_1, ..., \mathcal{AT}_n$ received by the counterpart, a party computes the set $Claim\_Set$ of micro-claims in the Merkle hash tree credential that match the attribute name in $\mathcal{AT}_i$ and that make true $\mathcal{AT}_i$. The match between attributes names in the $\mathcal{AT}_i$ terms and the attributes in the micro-claims is executed by using WordNet [22], an online dictionary containing all the synonyms in the context of English language. If $Claim\_Set$ is empty, the policy cannot be satisfied by the party, and a notification message is sent to the counterpart. Otherwise, the party checks if there are policies protecting the release of the attribute corresponding to term $\mathcal{AT}_i$. If such policies exist, the party sends them to the counterpart. We refer the reader to [2] for the details on how the negotiation tree is built and the trust sequences are computed.

At the end of the policy evaluation phase, when the two parties agree on a trust sequence of terms that must be satisfied, they do *not* exchange and verify one by one the credentials, as typically occurs in Trust-$\mathcal{X}$ negotiations. Rather, they exchange only the public part of the Merkle hash tree credential, called *certificate*, and the information to enable the verification of counterpart disclosure policies. This information includes the micro-claims that match the terms in counterpart disclosure policies, the intermediate node values and path information necessary for the receiving party to verify all the micro-claims. The receiving party verifies that the micro-claims are in the hash tree by computing the hash value of the root and checking if the the root hash in the certificate match the computed one. The receiving party also verifies the signature on the root value in the certificate part of the credential. In order to verify that the counterpart is the holder of the credential, the party also verifies that the counterpart owns the private key which matches the public key claimed by the credential. Algorithms 1 reports the steps of credential exchange phase with minimal credential disclosure.

The algorithm is performed once the two parties have agreed on a trust sequence $Trust\_Seq$ of terms to be satisfied. For each term $\mathcal{AT}_i$ in the *Trust_Seq* that the party must satisfy, the algorithm computes the set *Claim_Set* of micro-claims in the Merkle hash tree credential that match the attribute name in $\mathcal{AT}_i$ and that infer $\mathcal{AT}_i$ (line 2). Then, the method `pick` chooses one of the micro-claims in *Claim_Set*, according to the loose bind strategy (line 3). The method `TreeTraversal` traverses the Merkle hash tree and computes the authentication path *Path* from the leaf representing the chosen claim *SelectedClaim* and selects the set *Nodes* of nodes on the path that enable the verification (line 4). Then, the tuple (*SelectedClaim*, *Path*, *Nodes*) is inserted in the vector *Micro-claims_Info* that is sent to the counterpart (lines 5-7). When the party receives the vector *Micro-claims_Info*$_{CounterPart}$ from the counterpart, the algorithm performs the verification steps. For each tuple (*Claim*$_k$, *Path*, *Nodes*) $\equiv$ *Micro-claims_Info*$_{CounterPart}[k]$ in *Micro-claims_Info*$_{CounterPart}$, `ComputeRootHash` computes the hash value of the root using *Path* and the hash values of the intermediate nodes in *Nodes* (line 10). If the hash root value *Root* returned by `ComputeRootHash` is equal to the value in *Cert*$_{CounterPart}$, *Claim*$_k$ is contained in the tree and as a consequence the corresponding term $\mathcal{AT}_k$ in *Trust_Seq* is verified by the counterpart (lines 11-17). Otherwise, a notification message of negotiation failure is sent to the counterpart. If the counterpart micro-claims verification is successful, the algorithm checks if the signature affixed on the root hash in certificate is valid (line 18). If the signature is valid, the negotiation between the two parties is successful, otherwise the negotiation fails (lines 18-22).

Algorithm 1: Credential Exchange

Require: $Merkle\_Cred$ Merkle hash tree credential, $Trust\_Seq$: $\mathcal{AT}_1$, ..., $\mathcal{AT}_n$ trust sequence of terms

1: **for** $\mathcal{AT}_i \in Trust\_Seq$ that must be satisfied by the party **do**
2: $Claim\_Set$ = `match`($\mathcal{T}_i$)
3: $SelectedClaim$:= `pick`($Claim\_Set$)
4: {$Path$, $Nodes$}:=`TreeTraversal`($SelectedClaim$)
5: $Micro - claims\_Info$[i]:= (($SelectedClaim$, $Path$, $Nodes$))
6: **end for**
7: `Send`($Micro - claims\_Info, Cert$)
8: `Receive`($Micro - claims\_Info_{CounterPart}$, $Cert_{CounterPart}$)
9: **for** ($Micro - claims\_Info_{CounterPart}[k]$ **do**
10: $Root$ = `ComputeRootHash`($Micro - claims\_Info_{CounterPart}[k]$)
11: **if** $Root \neq Cert_{CounterPart}.Root$ **then**
12: `Send`(Verification Failed)
13: $Failed$:= true
14: `exit for`
15: **end if**
16: **end for**
17: **if** !Failed **then**
18: $Verified$:=`VerifySign`
19: **if** !Verified **then**
20: `Send`(Negotiation Failed)
21: **else**
22: `Send`(Negotiation Succeed)
23: **end if**
24: **end if**

Example 4.1 *Consider a simple negotiation where a party receives the disclosure policy* $DrivingLicense \leftarrow StateofResidence$ = $Indiana$, $Age > 18$ *that requires the party to prove that he is resident in the state of Indiana and that he is older than 18. The party has to demonstrate that he owns a set of micro-claims that satisfy the terms* $StateofResidence$ = $Indiana$ *and* $Age > 18$. *Assume that the party owns the Merkle hash tree credential in Figure 3. The credential contains the micro-claims* $StateofResidence$ = $Indiana$ *and* $Age > 25$

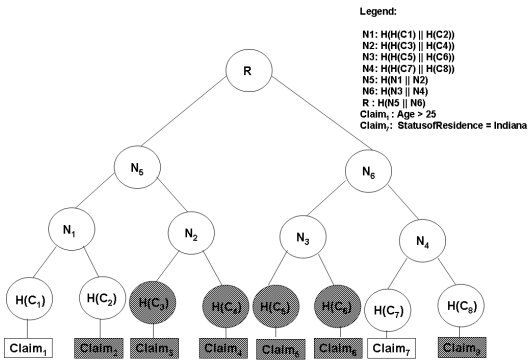

Figure 3: Example of Merkle Hash Tree

Operation	Functionalities
`StartNegotiation`	set-up negotiation parameters
`PolicyEvaluation`	verification of existence of micro-claims satisfying counterpart policies and attributes' disclosure policies retrivial
`CredentialExchange`	verification of counterpart micro-claims and selection of micro-claims to be sent to the counterpart

Table 1: Trust-$\mathcal{X}$ Web service operations

that respectively match the terms $Stateof Residence$ = $Indiana$ and $Age > 18$ in the disclosure policy. The party, sends the certificate and the following information to prove his claims:

- *for the claim $Stateof Residence$ = $Indiana$, the authentication path is $H(C_7)$, $H(C_8)$, N_3, N_4, N_5, N_6, R and the node set is$\{H(C_7)$, $H(C_8)$, N_3, $N_5\}$*
- *for the claim $Age > 25$, the authentication path is $H(C_1)$, $H(C_2)$, N_1, N_2,N_5, N_6,R and the node set is $\{H(C_1)$, $H(C_2)$, N_2, $N_6\}$.*

The counterpart by using the authentication path and the node set computes the hash value of the root R. If the value computed matches the value in the certificate sent by party, the verification succeeds.

4.3 Minimal Disclosure property

In this section we prove that our negotiation protocol guarantees minimal credential disclosure, that is, that only the micro-claims that satisfy the parties disclosure policies are disclosed and no partial information about unrevealed micro-claims is leaked. This property is formalized by the following lemma.

Lemma 4.1 (Minimal Disclosure) *The trust negotiation protocol is minimal with respect to credential disclosure in that a) for each policy $\mathcal{A} \leftarrow \mathcal{AT}_1, ..., \mathcal{AT}_n$ exchanged during the negotiation, the parties disclose only the micro-claims $claim_1$, ..., $claim_n$ in their Merkle tree credential that satisfy terms $\mathcal{AT}_1, ..., \mathcal{AT}_n$ and b) the parties do not disclose information about unrevealed micro-claims.*

Proof Sketch. The proof directly follows from the trust negotiation protocols and on the properties of the micro-claim. According to the negotiation protocol described in Algorithm 1, given a disclosure policy $\mathcal{A} \leftarrow \mathcal{AT}_1, ..., \mathcal{AT}_n$, a party selects, for each term $\mathcal{AT}_i$ to be satisfied, a micro-claim that implies the term $\mathcal{AT}_i$. Moreover, an attacker is not able to guess any information about unrevealed claims as proved in [1].

5 Implementation

Trust-$\mathcal{X}$ system main component is a Web service supporting the operations to carry on a trust negotiation and of a client application that invokes the Web service operations. The Trust-$\mathcal{X}$ Web service has been developed in Java, using the Tomcat Application Server and the Axis Soap Engine. The client application has also been implemented using Java. The disclosure policies driving the negotiation process and the credentials, used to prove that policies can be satisfied, are stored in a MySQL database. Disclosure policies are encoded according to an XML proprietary format, while for credentials both standard X.509 format and an XML proprietary format are supported. The Trust-$\mathcal{X}$ Web service provides three different operations, `StartNegotiation`, `PolicyExchange` and `CredentialExchange` corresponding to the main phases of the negotiation process. Table 1 describes the functions implemented by the

Trust-$\mathcal{X}$ Web service operations.

The client application is equipped with a GUI, by means of which a user specifies the parameters of the negotiation. The user can also monitor the negotiation process, checking the exchanged disclosure policies and credentials.

In order to integrate minimal credential disclosure in Trust-$\mathcal{X}$ system, we first added to the current credential formats supported by Trust-$\mathcal{X}$, the Merkle hash tree credential format. As we have described in Section 3, a Merkle hash tree credential is composed of a public part, the certificate, and a private part containing the Merkle hash tree and the user private key. The Merkle hash tree credential is implemented by the Java Class `ExtendedCredential.java`. The constructor method of this class has as input parameters the Merkle hash tree of micro-claims and the reference to the keystore where the public and private key certificates of the credential owner are stored. The Merkle hash tree of claims is implemented by the Java class `VerificationTree` that provides both the methods to create the Merkle hash tree from a set of micro-claims and to perform the verification, while X.509 certificates are used to encode the certificate and the private key of the user. The Merkle hash tree generated by the class `Verification Tree` is stored in the MySQL database along with the reference to the certificate and private key certificate stored in the kestore. We have modified the implementation of `PolicyExchange` operation in order to support the parsing of disclosure policies of the form $A \leftarrow \mathcal{AT}_1, ..., \mathcal{AT}_n$ protecting the disclosure of attributes rather than credentials and to check the existence of micro-claims in Merkle hash tree that verify the terms in the disclosure policies. Finally, we have extended the implementation of `CredentialExchange` to support the verification of Merkle hash tree micro-claims. Since different credential formats are supported by Trust-$\mathcal{X}$, the verification that must be performed by `CredentialExchange` is ruled by the type of credential adopted during the negotiation process. When Merkle hash tree credentials are used, `CredentialExchange` executes the verification methods implemented by `VerificationTree` class. It is important to note than when minimal credential disclosure is adopted, `CredentialExchange` is invoked only one time by each party because the parties exchange only one credential with the information to enable the verification of all the micro-claims satisfying the counterpart disclosure policies. Therefore high efficiency is achieved.

6 Related Work

Trust negotiation for web-based applications has been recognized as an interesting and challenging research area to explore, and it has been extensively investigated in recent years. As a result, a variety of techniques and prototypes have been developed [8, 19, 20].

Work related to privacy in the context of trust negotiation systems has focused on the protection of sensitive policies and credentials. In particular, Winslett et al. [23] have developed a unified scheme, known as *Unipro*, to model resource protection, which applies to both the actual resources to be protected and to the policies. Those approaches, however, are based on a notion of protection which is closer to the notion of access control and as such they are not able to support anonymization. Our work, instead, has the goal of providing stronger privacy to individuals through the use of minimal credential disclosure techniques. A formal framework for trust negotiations has been proposed by Winsborough and Li [21]. They provide an approach for safe enforcement of policies that focus on a privacy-preserving credential exchange. A formal notion of safety in automated trust negotiation is given, that states when a negotiation is secure against inferences that a party may make against the profile of the other party.

Two significant approaches dealing with selective disclosure of attributes are by Holt et. al. [10] and [13]. Holt's work focuses on hidden credential features, and on how to improve performance of hidden credentials constructed from identity-based crypto systems which satisfies credential indistinguishability. The notion of credential indistinguishability, adopted in such an approach, is very different from ours. The key idea by Holt et al. is to make credentials indistinguishable to any recipient which does not possess either of the credentials corresponding to P and $P0$, where P and $P0$ are elements of the set of possible single-credential policies. Li et al. proposed a scheme called Oblivious Signature Based Envelope (OSBE) [13]. OSBEs are similar to Hidden Credentials in that the ability to read a message is contingent

on having been issued the required secret. In [14], the authors propose a scheme that allows a credential holder to access to a service depending on whether his attribute values satisfy the service provider's access policy, without disclosing the actual attribute values.

Several privacy-enabled identity management systems have been based on the notion of anonymous credential [4, 5]. In anonymous credential systems, organizations know the users only by pseudonyms. Different pseudonyms of the same user cannot be linked. Yet, an organization can issue a credential to a pseudonym, and the corresponding user can prove possession of this credential to another organization (who knows her by a different pseudonym), without revealing anything more than the fact that she owns such a credential.

Stefan Brands developed a form of digital credential in which a user has a single public credential, but that credential is pseudo-anonymous, even to the issuer [3]. The credential holds attributes that the user can selectively prove to a service provider. Repeated showings of the same credential are linkable, both if shown to the same or different service providers. However, since the credential is issued blind by the identity provider, the effect is that a user has one global pseudonym. The credential can be reissued easily, allowing the user to change the global pseudonym, as permitted by the identity provider. Credentica's U-Prove Software Development Kit is based on Brands' work [3]. In [1], it is shown that verifying minimal-disclosure credentials of the type we use herein is approximately 3 orders of magnitude faster than verifying Brands' credentials. This is due to the fact that the minimal-disclosure credentials require mainly hash computations and only one public-key operation to verify the signature on the certificate, whereas Brands' credential requires a large number of exponentiation operations.

Camenisch et al., have proposed and implemented yet another form of digital credential, or more precisely, yet more forms [4]. While they describe a system for implementing a Chaum-like pseudonym system, their system is much more flexible, and can be used without pseudonyms. While having significantly better anonymity properties, the algorithms are also slower than Brands' credentials. IBM's idemix system is based on Camenisch, et al.'s work [4]. Idemix is the first system implementing anonymous credentials in a federated identity management system. Idemix provides mechanisms for efficient multi-show credentials and a flexible scheme for issuing and revoking anonymous credentials. It also provides a mechanism for *all or nothing* sharing and PKI-based non-transferability. Anonymous credentials however may not be adequate for several real world e-commerce applications and web services that require disclosure of various attributes. In our approach, we do not require the user identity to be hidden, even if we protect his/her attributes.

7 Conclusions

In this paper we have presented an approach for the controlled release of identity attributes based on the integration of trust negotiation and minimal credential disclosure techniques. The minimal credential disclosure approach guarantees that during the negotiation process a party discloses only the attributes necessary to complete the interactions with other parties.

As part of future work, we plan to investigate how to leverage the approach based on the verification of micro-claims in the context of federations. The idea is that a service provider, which has successfully verified a set of counterpart's micro-claims during a negotiation process, can release a signed assertion listing the counterpart's micro-claims and specifying how the micro-claims have been obtained and verified. The counterpart can use such assertion during another negotiation or to authenticate itself in order to access a service or a resource in the federation. We also plan to investigate privacy-preserving strategies for the generation and selection of micro-claims. Micro-claims could be generated for example based on anonymization criteria, like k-anonymity.

References

[1] D. Bauer, D. Blough, Minimal Information Disclosure with Efficiently Verifiable Credentials, submitted to DIM 2008.

[2] E. Bertino, E. Ferrari, A.C Squicciarini, Trust-$\mathcal{X}$- A Peer to Peer Framework for Trust Establisment, *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 16 N. 7, pp. 827-842, July 2004.

[3] S. Brands, Credentica - u-prove sdk, 2007.

[4] J. Camenisch, E. V. Herreweghen, Design and implementation of the idemix anonymous credential system, *In Proceedings of the 9th ACM Conference on Computer and Communications Security*, Washington, DC, USA, November 2002.

[5] D. Chaum, Security without identification: transaction systems to make big brother obsolete, *Commununications of ACM*, 28(10), pp. 1030–1044, 1985.

[6] N. Choi, I.-Y. Song, and H. Han, A survey on ontology mapping, *SIGMOD Rec.* 35(3), pp. 34-41, 2006.

[7] R.D. Jarvis, Selective Disclosure of Credential Content during Trust Negotiation, Master of Science Thesis, Brigham Young University, Provo, Utah, April 2003.

[8] A. Herzberg, J. Mihaeli, et. al., Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers, *In Proceedings IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.

[9] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. E. Seamons, B. Smith, Advanced Client/Server Authentication in TLS, *In Proceedings Network and Distributed System Security Symposium*, San Diego, CA, February 2002.

[10] J. Holt, R. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials, *2nd ACM Workshop on Privacy in the Electronic Society*, Washington, DC, October 2003.

[11] R. Johnson, D. Molnar, D. X. Song, D. Wagner, Homomorphic signature schemes, Topics in Cryptology – CTRSA 2002, Vol. 2271, pp. 244–262, Springer-Verlag, 2002.

[12] A. J. Lee, M. Winslett, J. Basney, V. Welch, Traust: a trust negotiation-based authorization service for open systems, *In Proceedings of the eleventh ACM symposium on Access control models and technologies*, pp. 39–48, Lake Tahoe, California, USA, 2006.

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

[14] J. Li and N. Li. Oacerts: Oblivious attribute certificates. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pp. 301–317, 2005.

[15] R. Merkle, A Digital Signature Based on a Conventional Encryption Function, *In Proceedings of Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, pp. 369–378, 1987.

[16] P. Persiano, I. Visconti, User Privacy Issues Regarding Certificates and the TLS Protocol, *In Proceedings of the ACM Conference on Computer and Communication Security*, Athens, Greece, 2000.

[17] K. E. Seamons, M. Winslett and T. Yu, Limiting the disclosure of Access Control Policies during Automated Trust Negotiation, *In Proceedings of Network and Distributed System Security Simposium*, San Diego, CA, February 2001.

[18] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jarvis, B. Smith, L. Yu, Negotiating Trust on The Web, *IEEE Internet Computing*, 6(6), pp. 30-37, November 2002.

[19] W. H. Winsborough, K. E. Seamons, V. Jones, Automated Trust Negotiation, *DARPA Information Survivability Conference and Exposition*, Volume I, pp. 88–102, IEEE Press, January 2000.

[20] W. H. Winsborough, N. Li, Protecting sensitive attributes in automated trust negotiation, *ACM Workshop on Privacy in the Electronic Society*, November 2002.

[21] W. H. Winsborough and N. Li. Safety in Automated Trust Negotiation. *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

[22] WordNet, http://wordnet.princeton.edu/.

[23] T. Yu, M. Winslett, A Unified Scheme for Resource Protection in Automated Trust Negotiation, *In Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003.

[24] T. Yu, M. Winslett, K. E. Seamons, Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation, *ACM Transactions on Information and System Security*,6(1), February 2003.