A Flexible Approach to Multisession Trust Negotiations

Anna C. Squicciarini, Elisa Bertino, *Fellow*, *IEEE*, Alberto Trombetta, and Stefano Braghin

Abstract—Trust Negotiation has shown to be a successful, policy-driven approach for automated trust establishment, through the release of digital credentials. Current real applications require new flexible approaches to trust negotiations, especially in light of the widespread use of mobile devices. In this paper, we present a multisession dependable approach to trust negotiations. The proposed framework supports voluntary and unpredicted interruptions, enabling the negotiating parties to complete the negotiation despite temporary unavailability of resources. Our protocols address issues related to validity, temporary loss of data, and extended unavailability of one of the two negotiators. A peer is able to suspend an ongoing negotiation and resume it with another (authenticated) peer. Negotiation portions and intermediate states can be safely and privately passed among peers, to guarantee the stability needed to continue suspended negotiations. We present a detailed analysis showing that our protocols have several key properties, including validity, correctness, and minimality. Also, we show how our negotiation protocol can withstand the most significant attacks. As by our complexity analysis, the introduction of the suspension and recovery procedures, and mobile negotiations does not significantly increase the complexity of ordinary negotiations. Our protocols require a constant number of messages whose size linearly depend on the portion of trust negotiation that has been carried before the suspensions.

Index Terms—Security and management, dependability, trust negotiations, access control.

1 Introduction

Trust negotiation is a mechanism supporting complex, distributed, rule-based access control for sensitive information and resources, through the controlled release of credentials [5], [24], [28]. A trust negotiation is a mutual attribute-based authorization protocol between two entities. Parties are assumed to be strangers which need to establish trust on the fly in order to exchange resources, information, or services.

Current real applications require flexible approaches to trust negotiations, especially in light of the widespread use of mobile devices. Consider, for example, mobile clients negotiating accesses to services hosted on servers' clusters: negotiations may interrupt due to communication channel fault or may be voluntarily suspended, to be resumed under more favorable conditions. Mobile devices need to be able to seamlessly migrate from different physical servers belonging to the same service provider. Also, negotiations may last a considerable time span and the involved parties may not be able to support long negotiations. Existing trust negotiation systems, however, do not currently support any form of suspension or interruption, and do not allow the negotiators to be replaced (or delegated) while the negotiation is ongoing. Interruptions in ongoing trust negotiations can be the result of external, unforeseeable events (e.g., parties' crashes, faulty transmission channels), or decisions by the involved parties. A party may not be able to advance the negotiation for temporary lack of resources. Or the party may not have readily available the credentials required by the counterpart, although eligible to them. For example, users may not have the capabilities or rights of storing certificates such as birth certificates, marriage certificates, and so forth, although entitled to them. Parties may also employ one-time credentials to conduct negotiations. Temporary and one-time credentials allow a party to disclose sensitive information while at the same limiting the possibility for an attacker to steal identity related information. Once such a credential is disclosed, it cannot be reused. Hence, completing a negotiation in which such type of credential is used becomes crucial. Interrupted negotiations however represent not only undesired events, but also vulnerabilities that could facilitate attackers' eavesdropping and other malicious behavior. Unfortunately, there are no approaches addressing such an issue. Trust negotiation research has mostly focused on the assurance of privacy and confidentiality with the goal of guaranteeing that no actual information about a negotiator's properties is disclosed to the counterpart [28], [19], [3]. Typically, these approaches rely on strong cryptographic assumptions, and are seldom applicable in many real-world scenarios, where properties, stated in digital credentials, actually need to be disclosed in clear and not only proved to be true. For example, just proving the possession of a valid credit card is not sufficient to complete a transaction, and actual account information is to be supplied in order to enable charging the amount spent. Additionally, protocols that rely on oblivious credentials or anonymous credentials do not allow parties to follow the progress of the negotiation, since

- *A.C. Squicciarini is with the College of Information Sciences & Technology, Pennsylvania State University, 301D IST Building, University Park, PA 16802. E-mail: acs20@psu.edu.*
- *E. Bertino is with the CERIAS and CS Department, Purdue University, LWSN Bldg, West Lafayette, IN 47907.*
- *A. Trombetta and S. Braghin are with the Computer Science and Communication Department, Insubria University, c/o DICOM, via Mazzini 5, 21100 Varese, Italy. E-mail: {alberto.trombetta, stefano.braghin}@uninsubria.it.*

Manuscript received 15 Mar. 2009; revised 2 Sept. 2010; accepted 22 Mar. 2011; published online 16 May 2011.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2009-03-0040. Digital Object Identifier no. 10.1109/TDSC.2011.31.

1545-5971/12/$31.00 © 2012 IEEE Published by the IEEE Computer Society

information regarding policies satisfaction is hidden for confidentiality purposes [22], [17]. It is thus crucial to extend trust negotiation protocols along several dimensions. First and foremost, the protocols must be able to adapt to context changes and be dependable. A *long lasting* trust negotiation should successfully withstand suspensions and interruptions. Also, given the ubiquitous nature of online peer-to-peer systems, and the increasing number of moving objects involved in online transactions, negotiators must be allowed to switch roles while the negotiation is ongoing, so to guarantee dependability, when contextual conditions, such as availability of resources and peers.

In this paper, we introduce a novel approach to trust negotiations that offers a general solution to those issues by developing major extensions to previous approaches by us and others [4], [37], [30]. The core of our approach is a trust negotiation protocol supported by the Trust-$\mathcal{X}$ system. This protocol, referred to as *multisession* trust negotiation, involves the exchange of digital credentials protected by rule-based disclosure policies (referred to as *disclosure policies*) which make it possible for two (or more) peers to establish mutual trust, so to carry on tasks such as the exchange of sensitive resources or access to a protected service. The main innovative feature of our proposed protocol is that it supports crash recovery and the possibility of completing the negotiation over multiple sessions. To support the execution of multisession negotiations, we extend the original Trust-$\mathcal{X}$ conventional negotiation steps. Savepoints are employed to save the negotiation state, validity checks concerning events which may happen during the negotiation suspension and could possibly invalidate the negotiation steps executed before the suspension. Examples of those events include credential revocation or expiration, or modification of disclosure policies by one of the peers.

An additional novel feature of the proposed framework is that it supports mobile negotiations, that is, negotiations that can be transferred among different peers in different sessions. With mobile we mean that a peer is able to suspend an ongoing negotiation and resume it with a peer different from the peer with which the negotiation started. Under our approach, negotiation portions and intermediate states can be safely and privately be transferred among peers. To support the secure transfer of negotiations, we have defined an authentication protocol, based on a secret-splitting scheme combined with a zero-knowledge proof protocol, to verify the identity of the peer recovering the negotiation and to assure the validity of the exchanged data. Our negotiation protocol also provides a mechanism for recovering from data losses which may occur at one of the involved peers.

In the paper, we present a detailed analysis showing that our protocols have several key properties, including validity, correctness, and minimality. Also, we show how our negotiation protocol can withstand the most significant attacks.

As by our complexity analysis, the introduction of the suspension and recovery procedures, and mobile negotiations does not significantly increase the complexity of ordinary negotiations. Our protocols require a constant number of messages whose sizes linearly depend from the portion of trust negotiation that has been carried before the suspension. We provide some evaluation results, showing the protocols' performance. In summary, the contributions of this paper is an approach supporting multisession and mobile trust negotiations. The approach uses an ontology techniques for replacing corrupted credentials with similar ones and protocols for the privacy-preserving sharing of negotiation tree.

The remainder of the paper is organized as follows: We present relevant cryptographic notions and an overview of Trust-$\mathcal{X}$ in Sections 2 and 3, respectively. Section 4 introduces our approach for sharing a negotiation tree, and supporting the negotiation process among different parties. Such technique is a core component of our approach to multisession negotiations. Section 5 illustrates the proposed negotiation protocol through an example. Sections 6 and 7 present security and performance analysis of the proposed protocol, respectively. Finally Sections 8 and 9 discuss related work and outlines a few conclusions, respectively.

2 Preliminary Notions

In the following sections, we provide the cryptographic background and introduce our adversarial model.

2.1 Cryptographic Protocols

The approach proposed in this paper is based on some well-known cryptographic protocols summarized in what follows.

1. *Shamir's secret sharing scheme.* Secret sharing refers to methods for distributing a secret among a group of participants, each of which is allocated a share of the secret. We adopt the (k, n) threshold scheme by Shamir [29]. Such a scheme splits a secret S into n partial secrets so that k, with $k < n$, partial secrets are required to reconstruct S. The scheme works as follows:

- $(k-1)$ random coefficients $\{a_1, \ldots, a_{k-1}\}$ are chosen.
- A polynomial $f(x) = a_o + a_1 x + a_2 x^2 + \cdots + a_{k-1}x^{k-1}$, with $a_0 = S$, is generated.
- Based on $f(x)$, n shares are constructed. Each share is of the form $(i, f(i))$ where i is the input to the polynomial and $f(i)$ the output. Given any k subset of these pairs, the coefficients of the polynomial can be evaluated using interpolation. The secret, that is, $a_0 = S$ can thus be determined.

2. *Damgård-Fujisaki commitment scheme and auxiliary protocols.* A commitment protocol is a method that allows a party, referred to as the prover, to commit to a value to another party, referred to as the verifier, while keeping hidden and preserving the prover's ability to reveal the committed value later. Here we describe the protocols by Damgård and Fujisaki [12]. The basic steps of the protocols are as follows: The commitment scheme:

- *Setup*. On input s (the security parameter) the outputs are 1) an RSA modulus $n = pq$, such that $p = 2p' + 1$ and $q = 2q' + 1$, where p, p', q, q' are primes; 2) a random $h \in QR_n$, where QR_n denotes the set of quadratic residues modulo n; 3) a random $g \in \langle h \rangle$, where $\langle h \rangle$ denotes the group generated by h. Thus, the public parameters generated by the setup phase are n, g, h.

- *Commit*. To commit an $m \in \mathbb{Z}$, the prover randomly chooses $r \in \mathbb{Z}$ and computes $c = g^m h^r \bmod n = \text{commit}(m, r)$.
- *Open*. To open a commitment c, the prover reveals m and r to the verifier which checks whether $c = g^m h^r \bmod n$.

3. *Proofs of knowledge*. In many applications, such as our negotiation protocol, revealing the committed value is not always required. Instead, it is sufficient that the prover proves to the verifier that it knows the committed value. Several such protocols, referred to as *proofs of knowledge* have been proposed. In our approach, we specifically need the following two proofs of knowledge:

- *a committed value is equal to a given value*. Given public parameters n, g, h of the Damgård-Fujisaki commitment scheme, a commitment c and an integer m', the prover proves the knowledge of m and r such that $c = g^m h^r$ and $m = m'$. Following the usual notation, we denote the protocol with

$$PK\{(m, r) : c = g^m h^r \wedge m = m'\}. \quad (1)$$

  We detail a well-known realization of such protocol as from [12]

  - The prover randomly chooses integers $y \in [0, \ldots, TC2^s)$, $t \in [0, \ldots, 2^{B+2s})$, where T is an arbitrarily large integer constant, C is an integer superpolynomially greater than s but smaller than the order of $\langle h \rangle$ and B is an integer is an integer polynomially greater than s. The prover sends $d = g^y h^t$ and y to the verifier.
  - The verifier chooses a random integer $e \in [0, \ldots, C)$ and sends it to the prover.
  - The prover sends to the verifier $v = t + er$. The verifier computes $u = y + em'$ checks whether $g^u h^v = dc^e$.
- *a committed value lies in a given interval*. Given public parameters n, g, h of the Damgård-Fujisaki commitment scheme, a commitment c and integers m' and m'', the prover proves the knowledge of m and r such that $c = g^m h^r$ and $m' \leq m \leq m''$. Following the usual notation, we denote the protocol with

$$PK\{(m, r) : c = g^m h^r \wedge m' \leq m \leq m''\}. \quad (2)$$

  A well-known example of such a protocol is found in [7].

4. *Fiat-Shamir heuristics.* In a noninteractive version of the above proofs of knowledge, the prover, instead of receiving a random challenge from the verifier (step ii) in the above protocol, generates by itself such random value by taking the output of a cryptographic hash function H [14]. The noninteractive version of the above protocol is

a. The prover randomly chooses integers $y \in [0, \ldots, TC2^k)$, $s \in [0, \ldots, 2^{B+2k})$, where T is an arbitrarily large integer constant, C is an integer superpolynomially greater than k but smaller than the order of $\langle h \rangle$ and B is an integer is an integer polynomially greater than k. The prover sends $d = g^y h^s$, $H(d)$ (where H is a cryptographic hash function), $v = s + H(d)r$ and y to the verifier.

b. The verifier computes $u = y + H(d)m'$ checks whether $g^u h^v = dc^e$.

If we assume the (widely accepted) random oracle hypothesis, the modified protocol is a zero-knowledge proof of knowledge, that is, the prover proves to the verified the knowledge of the secret without leaking any information to the verifier [7].

The Damgård-Fujisaki scheme is used for committing the credentials to be disclosed later on, during the execution of the negotiation protocol (see Section 4.2). We adopt this commitment scheme because it fits trust negotiation requirements. First, it imposes fewer constraints on the input than other schemes: it accepts as valid input an arbitrary integer (and not only a group member, as in the case of the Pedersen commitment scheme [26]); second it has efficient knowledge proofs that a committed value is equal to a given value and that a committed value lies in a given interval. This last property is used in our approach to test whether a committed value is less than a given value. The Shamir scheme is instead used to distribute among different peers an encrypted version of the data structure used for storing information about the negotiation. Such shares will be collected by one of the peers in order to resume a suspended negotiation (see Section 4.3).

3 Overview of the Trust-$\mathcal{X}$ Negotiation System

In this section, we first summarize key notions concerning the trust negotiation language and credentials supported by the original Trust-$\mathcal{X}$ system, followed by a summary of the single-session negotiation protocol.

3.1 An Overview of the Trust-$\mathcal{X}$ System

Trust-$\mathcal{X}$ [4], [6], [31] is a comprehensive system for trust negotiation, providing both an XML-based language, referred to as $\mathcal{X}$-TNL, and a suite of negotiation protocols. $\mathcal{X}$-TNL supports the specification of digital credentials and disclosure policies, which is the key information exchanged during negotiations. Digital credentials are assertions describing one or more properties, called attributes, of a given subject, certified by trusted third parties. Notice that our notion of credential is quite general and thus Trust-$\mathcal{X}$ is interoperable with several standards for certificates and security assertions, such as SAML [25]. Disclosure policies regulate the disclosure of a resource by imposing conditions on the credentials the requesting party should possess. We now introduce a simplified version of the rule-based language used by Trust-$\mathcal{X}$ system.

We consider a credential c as a structured object composed of several items corresponding to attributes of the subject to whom the credential belongs, and denote it as a predicate of the form $t(AttrSet)$, where t is a credential type name and $AttrSet = \{att_1, \ldots, att_k\}$ is the set of attributes. Credentials are compliant to a given template, referred to as credential type. Users collect credentials into an $\mathcal{X}$-Profile, which could be remotely stored in case of mobile profiles.

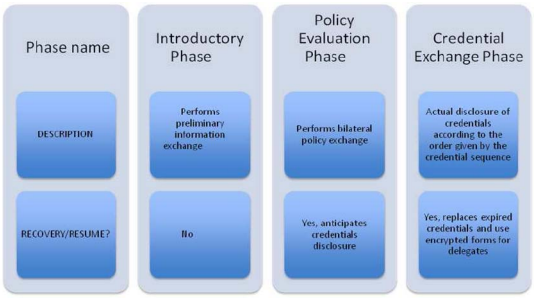

Fig. 1. Negotiation phases.

Terms can be conditionals or not conditionals. Conditional terms in $\mathcal{X}$-TNL have the form $t(CondSet)$, where t is the name of a credential type and $CondSet$ is a set of conditions against attributes in t. A condition is an expression of the form *att pred val*, where *att* is an *attribute name*, *pred* is a (binary) predicate in $\{\geq, \leq, =\}$, and *val* is a value from attribute *att* domain. A term can be *unspecified*, if the credential type name is not dictated. Examples of terms are: $IdCard(ZIPCode = 2342)$ and $X(DateofBirth)$. X denotes any possible credential type which conveys the attribute "date of birth." In what follows, we say that a term t is *satisfied* by a credential c, denoted by t^c, if all conditions in $CondSet$ are verified by c's attributes, and the credential type name matches with the credential name in term t, if specified. A term is *satisfied* if all attributes appearing in $AttSet$ appear in *CondSet*, and the credential name matches the credential name in t. A not conditional term is a term listing a set of attributes to be disclosed, and is of the form $t(AttList)$, where t is again the type and $AttList$ the list of attributes to reveal. For simplicity, we drop the distinction between not conditional and conditional terms and refer to them as terms, unless specified otherwise. We express disclosure policies as a finite set of rules, each of the form: $R \leftarrow \phi(t_1, \ldots, t_k)$. R is the target resource for which the policy is specified, and $t_1, \ldots, t_k$ are terms corresponding to the credentials to be disclosed. Intuitively, a policy is satisfied if the formula $\phi(t_1, \ldots, t_n)$ holds true when the formula's variables are instantiated with credentials' values.

Example 3.1. Examples of policy rules are

R1. LoanRequest $\leftarrow$ X(YearofBirth $= 1987$);
R2. IdCard $\leftarrow$ VeriSign() $\vee$ IdCard(LastName).

3.2 The Trust-$\mathcal{X}$ Protocol

Trust-$\mathcal{X}$ negotiation's main goal is to carry out successful negotiations while protecting credentials and policies. Thus, only the information strictly needed to advance the negotiation is communicated. Trust-$\mathcal{X}$ negotiations can be carried out during one or multiple sessions. The negotiation process can be suspended or interrupted by one of the negotiating parties. Suspension and interruption do not result in any significant leak of information, or in privacy breaches. We discuss the most important steps of the negotiation in the next section followed by a presentation of the core mechanisms implemented to carry out long lasting negotiations.

3.2.1 The Basic Trust-$\mathcal{X}$ Protocols

A Trust-$\mathcal{X}$ negotiation is organized according to four different phases, outlined in Fig. 1. Policies are disclosed first during the policy evaluation phase, and then only the credentials necessary for the negotiation success are disclosed during the credential exchange phase. Various strategies can be used for trust negotiations, which may or may not include all the shown phases. To maximize trust negotiation effectiveness, it is crucial—regardless the specific strategy adopted—to ensure robustness of at least the policy evaluation and credential exchange phase. Both are fundamental to any trust negotiation. The former determines whether or not the negotiation can succeed, while the robustness of latter is crucial since during this phase sensitive credentials may be disclosed. This approach to negotiation has been widely adopted in several other trust negotiation [4], [35], [37], [28] systems. Thus, throughout the paper, we focus on protocols for those two phases.

An important component of the Trust-$\mathcal{X}$ negotiation process is the *negotiation tree*, a data structure that keeps track of the negotiation process. The tree is rooted at the requested resource and is initialized when the negotiation starts. The tree is dynamically built and expanded as the negotiation proceeds. Precisely, a negotiation tree NT is a tree in which each node corresponds to a term, and edges correspond to policy rules. A negotiation tree NT is formally modeled as a tuple $T = \langle \mathcal{N}, \mathcal{R}, \mathcal{E} \rangle$, where $\mathcal{N}$ denotes the set of nodes, $\mathcal{R}$ denotes the root of the tree, and $\mathcal{E}$ the set of edges. The order of sibling edges is implied by the policy precondition set of each rule. Furthermore, set $\mathcal{E}$ contains two different kinds of edges: *simple edges* and *multiedges*. A simple edge is used to model policies having only one term on the right side component of the associated rule. By contrast, a multiedge links several simple edges in order to represent policy rules having more than one term on their right side. Nodes belonging to a multiedge are thus considered as a whole during the negotiation. Each node in a negotiation tree can assume two different states: the *deliv* state denoting a delivery resource, that is, a ready-to-be-delivered credential without any further preconditions; and the *open* state, meaning that the credential/resource denoted by the node is not yet ready for delivery delivered. Nodes are appended to the tree according to the policies exchanged and the dependencies among the different terms.

The successful completion of the policy evaluation phase is signaled by a portion of the tree rooted at the requested resource consisting only of nodes with a *deliv* state, that we refer to as *valid view*. When the negotiation tree includes a valid view it is possible to determine a trust sequence for the negotiation. The trust sequence, denoted as $\mathcal{CS}eq$, can be built by traversing the view according to a specified order defined by the labeling function associated with the tree. Finally, the credentials are disclosed by both ends following the order dictated by the view.

A successful negotiation is typically defined as follows:

Definition 3.1. *Let P_1 and P_2 be two negotiation peers, and let $\mathcal{X}$-Profile$_{p1}$, $\mathcal{X}$-Profile$_{p2}$ be the profiles of P_1 and P_2, respectively. A trust negotiation protocol for a resource R is successful if and only if a credential disclosure sequence $(C_1, \ldots, C_n = R)$ is found, where $C_i \in \mathcal{X}$-Profile$_{p1} \cup \mathcal{X}$-Profile$_{p2}$, and such that for each C_i, $1 \leq i \leq n$, the policy $pol_i = C_i \leftarrow \phi(t_1, \ldots, t_n)$ for C_i is satisfied by the credentials already disclosed, i.e., $t_1^{C_{k1}}, \ldots, t_n^{C_{km}}; k1, \ldots, km < i$.*

4 Negotiation Tree Switching

In this section, we provide the core contribution of our approach to multisession. We begin with presenting a high-level discussion of the approach, followed by a detailed presentation of the new Trust-$\mathcal{X}$ algorithms.

4.1 Overview of the Approach

In order to support long lasting negotiations, we introduce two major features to trust negotiation protocols. First, we support multisession negotiations, that is we allow negotiations to be conducted within multiple separate sessions. We depart from the assumption of atomic trust negotiations in order to make the negotiation also suitable for peers with heterogenous capabilities. In the multisession protocol, we do not require both parties to maintain an up-to-date copy of the negotiation state at the time of suspension. Relaxing such assumption does not imply going back to a client-server architecture. Rather, parties are still peers, and therefore able to control the negotiation process; however the task of storing the negotiation data at suspension time can be assigned to one of the two parties. This approach makes trust negotiation protocols suitable for a large variety of environments, in that it may often be the case that one of the negotiating peers is connected via a device with limited connection power or small memory. A second important extension is to allow negotiations to be completed by multiple peers. Essentially, we now allow the negotiations between two peers, say P_1 and P_2, to be suspended and then resumed by different peers. For example, P_2 can be replaced by P_3, provided that the replaced—or *delegated*—peer (e.g., P_3) has the ability to complete the previously started negotiation. We do not expect peers to be replica one of another. We assume that a peer—e.g., P_2—involved in the negotiation maintains a list of delegated peers that are possibly able to carry on the negotiation is suspended and P_2 is not anymore available. Our suspend and resume protocols work with only one delegate at time. That is, we do not consider the case in which a negotiation may be resumed by more than one (possibly conflicting) delegate. As for the choice of the delegate, we do not commit ourselves to some specified solution, since the choice may be heavily driven by the context in which suspension and resumption happen. Hence, the negotiation may result in different outcomes according to the specific policies enforced by selected delegated. This is an acceptable approach since it allows the new peers to conclude the negotiation based on their actual current resources, and to eventually provide the resource or service which originated the negotiation in the first place. Further, it allows the peers to include, in the remaining steps of the negotiation, their preferred policies or conditions that are deemed necessary to the successful completion of the negotiation. To ensure the correctness of this process, and avoid security breaches, we introduce a protocol allowing peers to:

- suspend and resume negotiations with a peer different from the original negotiator;
- share with a different peer in a privacy-preserving way the negotiation tree so far constructed.

A brief description of the main steps of the multisession negotiation (MS, for short) protocol is reported in what follows. We begin with focusing on the case of a negotiation that was suspended during the policy evaluation phase. We assume, without loss of generality, that P_2 is keeping track of the advance of the negotiation (since, e.g., P_1 is a mobile, low-end device).

1. During the execution of the policy exchange phase, P_1 and P_2 exchange credentials as soon as their corresponding nodes in NT are deliverable. Since it may not be the case that all deliv nodes are contained in a valid view (and hence they should not be actually exchanged), the corresponding credentials are sent in an encrypted form, thus preventing information disclosure (see Section 4.2).
2. P_1 and P_2 suspend the negotiation. P_2 processes the NT tree in order to hand it to another peer, which will resume the negotiation, by pruning the delivery nodes from NT.
3. Subsequently, for every term in the pruned negotiation tree, P_2 computes the corresponding commitments (see Section 4.2).
4. P_2 serializes the pruned tree, thus obtaining another tree S. P_2 applies the secret sharing scheme to S, obtaining n, $n \geq 3$, shares; let $S_1, S_2, \ldots, S_n$ be such shares. P_2 sends S_1 to P_1, keeps S_2 for itself, and distributes $S_3, \ldots, S_n$ to a subset $\mathcal{P}'$ of peers in $\mathcal{P}$, where $\mathcal{P}$ corresponds to the set of peers known to P_1 as potential delegates ($\mathcal{P}$ and $\mathcal{P}'$ may coincide). In addition to S_2, P_2 sends the keys for deciphering the encrypted versions of the credentials corresponding to deliv nodes (already exchanged with P_1).
5. Suppose that P_1 has contacted and authenticated peer $P_3 \in \mathcal{P}'$. Upon receiving S_3 from P_1, P_3 reconstructs the node-by-node committed negotiation tree (see Section 4.3). Now P_3 has to build its version of the negotiation tree starting from the committed tree and from its available credentials. For every node in the tree, P_3 (by using a noninteractive zero-knowledge proof of knowledge) checks whether its credentials match the committed terms (see Section 4.4).
6. Having built its version NT'_{open} of the negotiation tree, P_3 sends it to P_1 and they can restart the previously suspended negotiation. In the case of successful termination and in case that the valid view contains credentials sent in the first phase of the negotiation, P_3 sends the corresponding keys to P_1.

Notice that the case of interruptions is essentially an extension of the above described protocol. Unlike suspensions that are voluntarily decided by the negotiating peers, interruptions are unforeseen by the peers and are due to external events, like network and system crashes. To support recovery from interruptions, the steps 2, 3, and 4 above are executed periodically, asynchronously by either peer, with the only difference that at step 2 the negotiation is *not* suspended. Each peer saves a version of the tree and generates a secret for the sharing (step 3). The secret shares are then distributed to $\mathcal{P}'$, that will not use them unless necessary. If neither an interruption nor a suspension occur

for a few negotiation rounds, the peers will periodically update their committed versions of the tree (this can be done incrementally for the new nodes to avoid unnecessary overhead) and create new secrets shares. The receiving peers in $\mathcal{P}'$ will simply replace the old secrets with the new ones. If an interruption occurs, one of the two peers, say P_1 will simply stop sending and receiving messages. When P_1 recovers, the most recent shares sent by P_1 will be used to reconstruct the intermediate state of the negotiation.

Our threat models assumes that a malicious peer participating in the execution of a negotiation may divert from a correct behavior during the negotiation resume phase in several ways. First, a malicious peer may report incorrect information about the saved tree and thus the resume procedure may generate a tree view containing nodes referring to nonexisting terms. Further, the peer may insert fake credentials or terms in a sequence of credentials to be deployed in the negotiation. Finally, malicious peers may collude in order to gather private information contained in the exchanged credentials.

4.2 Negotiation, Suspension, and Nodes Commitment

As outlined in Step 1 above, peers P_1 and P_2 exchange credentials as soon as their corresponding nodes in the negotiation tree NT enter the *deliv* state. This approach is not conventional to trust negotiation strategies that usually postpone credentials disclosure after a successful path is found. The purpose of anticipating the disclosure is twofold: first, it provides partial assurance to the negotiating parties regarding their opponent's commitment to negotiating; second, it reduces the cost of the subsequent phases and removes the already processed nodes from the tree. Notice that, in order to prevent information leakage, what is actually exchanged is an encrypted version of the credentials, computed by P_2 with a (previously agreed upon) symmetric encryption scheme Enc (e.g., AES). Each credential is encrypted with a different key; all these keys are unilaterally chosen by P_2. Upon negotiation suspension (Step 2 above), P_2 prunes from the negotiation tree NT all the deliv nodes, obtaining the tree NT_{open}, which contains open nodes only.

Subsequently, at Step 3, peer P_2 computes for each node in NT_{open} the corresponding commitments using the Damgård-Fujisaki commitment scheme (see Section 2.2). Given a node in NT_{open}, suppose that the associated term has the form $cred[att_1, pred_1, val_1, conn_1, \ldots conn_u, att_u, pred_u, val_u]$, where $conn_i \in \{\vee, \wedge\}$. Then, P_2 computes[1]:

$$\begin{aligned}&\mathrm{commit}(cred)[\mathrm{commit}(att_1)\ pred_1\ \mathrm{commit}(val_1)\\ &\quad conn_1 \ldots conn_{u-1}\mathrm{commit}(att_u)\ pred_u\ \mathrm{commit}(val_u)].\end{aligned}$$

We denote with $\mathrm{commit}(NT_{open})$ the negotiation tree in which every open node has been committed.

4.3 Tree Splitting and Sharing

If the negotiation is suspended during the policy evaluation phase, the tree is used for checkpointing. An encrypted version of the tree is generated. Subsequently, P_2 serializes it for improved efficiency. We denote the serialized version of the (pruned and committed) negotiation tree as $S = \mathrm{ser}(\mathrm{commit}(NT_{open}))$. Once serialized, the tree, being now encoded by a (large) string S, is used as input for the Shamir secret sharing scheme: P_2 generates n secret shares, denoted by S_1, S_2, and S_n from S. P_2 then sets $k = 2$ and distributes the share to $P_3, P_4, \ldots, P_n$, all belonging to $\mathcal{P}$ and being trusted by P_2. P_2 will, by default, keep one share for itself, and gives S_1 to P_1. When P_1 recovers and/or restarts, it can select among the peers owning the other share the one to resume the negotiation with. Notice that the notion of trust here is loosely used; the peers with whom P_2 shares the secret are not peers which can immediately access the negotiation status in all its details, but peers who are trusted enough to adhere to the negotiation protocol and complete the suspended process. P_1 therefore will continue the negotiation with any peer among $P_2, \ldots, P_n \in \mathcal{P}$. Suppose that P_3 is selected. Then, the partial secrets of P_1 and P_3 must be combined. Shares are then pooled by P_3. The two shares provide two distinct points $(x; y) = (i; S_i)$ making it possible to computate the coefficients of $f(x)$ by Lagrange interpolation. The secret is recovered by noting that $f(0) = a0 = S$. For increased security, k can be set to three, so that the authentication of peer P_3 is "witnessed" by a third party.

4.4 Tree Recovery

After peer P_3 has recovered the tree $\mathrm{commit}(NT_{open})$, it builds its version of the negotiation tree starting from $\mathrm{commit}(NT_{open})$ and its own credentials. P_3 builds its tree by executing knowledge proofs on the commitments. For every committed node in $\mathrm{commit}(NT_{open})$ and for every credential $cred'[att'opval']$, P_3 performs the following noninteractive zero knowledge proofs. Recall that a committed node has the form

$$\begin{aligned}cc = \mathrm{commit}(cred)[&\mathrm{commit}(att_1)pred_1\mathrm{commit}(val_1)conn_1 \ldots\\ &conn_{u-1}\mathrm{commit}(att_u)pred_u\mathrm{commit}(val_u)].\end{aligned}$$

We denote with c_{P_3} the credential set owned by P_3. Algorithm 1, reported in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2011.31, checks whether, for every committed term cc there is a credential in C_{P_3} with matching credential and attributes names that contains values satisfying the term predicates.

If every zero-knowledge proof succeeds, P_3 has constructed its own negotiation tree NT'_{open} upon which the negotiation with P_1 can be restarted. As a final step (Step 6 in the above outline), P_3 sends the novel negotiation tree to P_1 and they resume the negotiation. In the case in which the trust negotiation succeeds and some encrypted credentials $\mathrm{Enc}(Cred_i, k_i), \ldots, \mathrm{Enc}(Cred_j, k_j)$ (exchanged during the negotiation between P_1 and P_2) belong to the valid view, P_3 sends the corresponding secret keys $k_i, \ldots, k_j$ to P_1, which finally deciphers the encrypted credentials and terminates the negotiation.

4.5 Sharing the Trust Sequence

We now discuss our protocol for credential exchange during a MS negotiation.

1. We do not explicitly indicate the random value r. It is implicit that it changes for every execution of the commitment algorithm.

4.5.1 MS Protocol for Credential Exchange

A suspension may occur during the verification of the set of credentials identified as necessary and sufficient to complete the negotiation. This situation presents some security challenges, because of the sensitive contents exchanged during the credential evaluation phase. While in the case of policy evaluation phase, only uncertified statements are exchanged, here the exchange often involve confidential digitally signed documents. Releasing a credential to a third party for negotiation may not be desirable, even if this third party is considered trustworthy. Additionally, in order for the verification to be meaningful, both parties should be able to verify the credentials against previously exchanged policies. Clearly, this is a not trivial step, given that one of the two peers is, in case of resume, a different one, and may thus have different credentials and policies. Our approach to address these issues is based on carefully preprocessing the sequence generated by the valid view, and on a modified verification protocol. The MS protocol for credentials' exchange is the following:

1. P_1 and P_2 agree to suspend, after a sequence of terms $\mathcal{CS}eq$ is determined and partly exchanged. P_2 encrypts the local credentials corresponding to terms in $\mathcal{CS}eq$ that have not been exchanged yet.
2. P_2 sends to P_1 a single session encryption key k_s (upon agreement on a symmetric encryption scheme Enc) to open the encrypted credentials required by the credential sequence $\mathcal{CS}eq$.
3. P_2 generates an encrypted version of $\mathcal{CS}eq$ as follows: it leaves in clear the terms of P_1, while it replaces its own credentials with encrypted credentials. Then it applies the secret splitting protocol on the modified $\mathcal{CS}eq$, obtaining shares $s_1, \ldots, s_k$.
4. P_2 sends to the possible delegates (i.e., $P_3, \ldots, P_k$) the generated secret shares $s_1, \ldots, s_k$, and leaves.
5. Assume that P_3 is the first one replying to the resume request.[2] P_1 and P_3 can now reconstruct the sequence, and exchange the remaining credentials.
6. During the exchange, P_3 receives P_1's credentials $Cred_i, \ldots, Cred_k$ in order and in clear. Based on the corresponding terms stored in the $\mathcal{CS}eq$, P_3 is able to verify the validity of each credential and whether or not it satisfies the corresponding term. P_3 at its end sends sequentially P_2's encrypted credentials to P_1, which can verify them by using the key k_s obtained by P_2.

This protocol has a number of desirable features. First, it allows P_1 and P_3 to complete the negotiation without requiring that P_2's credentials be disclosed to P_3. Second, it protects from possible changes of the sequence either by P_1 or P_3. Because the secret is split, P_1 cannot decipher with $k_s P_2$'s credentials. Finally, it allows P_3 to verify the credentials from P_1 by using the terms in the sequence. For highly confidential credentials, it is also possible to replace the exchange of the credentials with their committed versions, so that only minimal information is disclosed.

2. The delegate could be selected based on a variety of metrics. We omit this discussion for lack of space.

4.5.2 Replacing Credentials

An issue of the protocol is that it is not able to handle the case in which a credential within the sequence expires during the suspension and the subsequent resume is operated by a different peer. To this extent, we rely on the notion of credential similarity, to identify possible credentials substituting the original ones. We propose a simple protocol to identify similar credentials among the ones possessed by an entity and replace one with another when and if needed.

1. *Credential categorization.* We assume credentials to be categorized based primarily on their semantics. Examples of credentials' categories are "ID," "Work life," and "Financial." Such categories are used to classify the credentials, so as to simplify the identification of the credentials eligible substitutes of an expired or revoked one. Beside a set of predefined categories, we support a "Misc" category that contains the credentials that cannot be associated with existing categories. Each category is associated with a predefined OWL model defining the core attributes a credential is supposed to contain in order to be classified with the category. We rely on an OWL [36] specification, since OWL is a standard language for ontology specification. After the extraction of the credential's OWL model from the XML document, the model is processed by a matching algorithm which evaluates the credential's OWL model against different metrics. We use different metrics in order to identify syntactic and semantic similarities among credentials, such as V-DOC [27] and GMO [18]. The final result of our matching algorithm is the normalized and weighted composition of the results of the comparison of the models. Once the matching algorithm is applied to a credential c, c is associated with the most similar category, referred to as *primary category*, and with a set of other categories, referred to as the *set of secondary categories*. Given a credential c, its primary category is denoted as $Pri(c)$, while the set of secondary categories as $Sec(c)$. The secondary categories are the categories that with respect to c have a similarity value lower than the similarity value of the primary category and higher than a threshold value β. The thresholds α and β are defined in the system but could be refined by the end user to better suit the categories and the credentials contained in its $\mathcal{X}$-Profile.

Example 4.1. Consider the following categories:

1. *Personal Information*. The model of this category has attributes `name`, `surname`, `address`, `date of birth`, and `SSN`;
2. *Work Life*. The model of this category has attributes `name`, `surname`, and `employee number`; and
3. *Financial*. The model of this category has attributes `name`, `surname`, `bank name`, `bank address`, and `bank account`.

A credential representing the passport of a user, called *Passport* and with attributes `name`, `surname`, `address`, `date of birth`, has *Personal Information* as primary category and the other two categories as secondary ones.

More formally:

Definition 4.1 (Related Credentials). *Let c be a credential and $Pri(c)$ be its primary category and $Sec(c)$ be the set of its*

secondary categories. The set of related credentials $\mathcal{S}_c$ for c is defined as

$$\mathcal{S}_c = \{c_i | Pri(c_i) = Pri(c) \vee Pri(c_i) \in Sec(c)\}.$$

2. *Credential substitution.* Using primary and secondary categories, we can identify a substitute for a credential c if c has been revoked or has expired. Note that our notion of similarity is not absolute, but relative to a specific policy which requested the credential in the first place.

Definition 4.2 (Substitute Credential). *Given a credential c and a policy $pol = C_i \leftarrow \phi(t_1, \ldots, t_n)$, the* substitute *for c with respect to p is defined as the credential, denoted by s_c^{pol}, such that:*

1. *$s_c^{pol} \in \mathcal{S}_c$ and*
2. *if $\phi(c, c_1, \ldots, c_n)$ holds true, then $\phi(s_c^{pol}, c_1, \ldots, c_n)$ holds true.*

Notice that not all credentials may be suitable for replacement, in that a different credential type may imply a very different semantic [21]. Consider, for example, a passport versus a Id card. The request of a passport disclosure versus a student id to verify an individual's name may denote the server's need to verify other information, although not explicitly mentioned in the policy, such as citizenship. Or, it may be more trusted than a student id. Hence, we assume that replaceable credentials are indicated at the time of policy specification. For example, the server can choose not to specify the desired credential type, and indicate the desired attributes only, regardless of the credential where they are contained. Alternatively, if the type of the credential is considered essential, it can mark the irreplaceable credential types by adding an additional condition to the term.

Example 4.2. Let

$$\begin{aligned}&\phi(Passport, DrivingLicence)\\ &\quad \equiv X(Name = \text{``}John\text{''}, Surname = \text{``}Doe\text{''})\\ &\quad \wedge DrivingLicence(Name = \text{``}John\text{''}, Surname = \text{``}Doe\text{''},\\ &\qquad LicenseNumber = 1003533, ExpiringYear > 2010)\end{aligned}$$

be a policy to satisfy. If the user originally chose the passport to satisfy the first condition, and the credential Passport is expired, it can be replaced with a credential belonging to the primary category it belongs to. For example, the credential Identity Card belonging to John Doe can be used as a substitute since it belongs to the same category and it can make the policy true. Notice if expired, the Driving License cannot be replaced using Identity Card, since it does not have an attribute indicating the John's license number.

3. *Credential replacement strategies.* Using the notion of replaceable credentials, we identify two different strategies to complete credentials exchange despite possibly expired credentials.

a. P_2 includes within the modified and encrypted $\mathcal{CS}eq$ up to d substitute credentials for each credential $Cred_i$ in $\mathcal{CS}eq$. Substitute credentials are identified according to Definition 4.1. Each encrypted credential $\mathrm{Enc}(Cred_i, k_s)$ will thus be associated with a timestamp defining the time validity period of credential $Cred_i$. After the recovery of the negotiation, the delegated peer P_3 checks whether the credential being sent to P_2 is still valid by verifying if the timestamp has not expired. If this is not the case, P_3 will select a substitute from the list.

b. The resuming peer finds a valid substitute credential. This task is performed by using the credential categories introduced in Section 4.5.2. The suspending peer P_1 adds to each encrypted credential $C = \mathrm{Enc}(Cred_i, k_s)$ in the modified $\mathcal{CS}eq$ the corresponding OWL model M_C. By exploiting M_C, P_3 identifies a substitute credential among its $\mathcal{X}$-Profile following the steps described in Section 4.5.2. Obviously, such steps are performed if and only if the credential $Cred_i$ has expired.

The proposed strategies do not, however, address the problem of revoked credentials. P_1 is the only peer able to discover such case. If a credential has been revoked, P_1 notifies P_3, which will provide a valid substitute credential (using a credential provided by P_2 or using one of its own), if it exists.

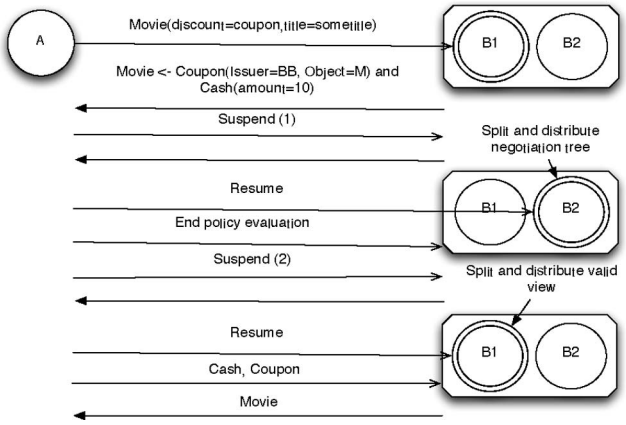

Fig. 2. Running example.

5 Illustrative Example

Suppose that user Alice (A, from now on) would like to buy from BestBuy (B, from now on) a DRM-protected digital movie using a coupon allowing her to obtain a discount on the movie price. We refer to Fig. 2 for a graphical representation of the example.

A connects to B from her PDA, and initiates a negotiation with one of the servers operating for B and identified as in charge of the negotiation for B's. Let this server be denoted by B_1, in Fig. 2 we use the double stroke to highlight the active server of B. In order to provide the required movie, B_1 requests from A the coupon and the amount of e-cash required to buy the movie. This policy is encoded by a rule of the form:

$$\begin{aligned}&Movie(Discount = coupon, title = sometitle)\\ &\quad \leftarrow Coupon(Issuer = BestBuy, Object = Movie)\\ &\quad \wedge Cash(amount = 10).\end{aligned}$$

Before sending her credentials, as required by B_1's policy, A requires some credentials from B_1. In particular, in order to release her coupon, A requires a credential stating that B_1 is

a BestBuy server. Moreover, A requires a ticket which allows her to examine B's bank privacy policies. B_1 replies to the first policy by asking that A presents her BestBuyAccount in order to be authorized to access the credential showing that B_1 is a BestBuy. To disclose the temporary ticket which will let A access its bank policies, B_1 requires to identify a bank account to which to refer the temporary ticket itself. A is not registered at BestBuy so she asks to B_1 for a temporary suspension of the negotiation—Suspend (1) in Fig. 2—since A is connected from a limited device that does not allow multiple transactions at the same time.

Before the suspension of the negotiation B_1 creates the information required to save the intermediate state of the negotiation (see the protocol in Section 4.4) and sends them to every otherserver in B's pool.

When A resumes the negotiation, B_1 is temporarily offline for maintenance; thus another server B_2 (in B's pool) continues the negotiation with A. Since the negotiation was suspended during the exchange of the policies, B_2 checks the committed negotiation tree initially received from B_1 and because B_2 has the same policies referred in the tree, it is able to resume the negotiation with A with no changes. A decides that the credential required in the current round by B_2 are not sensitive and thus does not impose any additional policy on the disclosure of this credential. The decision made by A ends the policy evaluation phase of the negotiation because both A and B_2 are now able to identify a valid sequence of credentials that can lead to the negotiation success. This sequence consists of every credential involved in the negotiation. This means that every credential needs to be exchanged and verified in order to successfully complete the negotiation. Both peers switch to the next phase of a Trust-$\mathcal{X}$ negotiation, that is, the credential exchange phase. A asks again to suspend the negotiation (Suspend (2) in Fig. 2), as she discovers that her bank's credential is expired. B_2 operates the suspension creating the split version of the credentials' list and sends it to the other servers of B's pool along with the involved credential. After A has renewed her credential, she resumes the negotiation. This time, B_1 is available and being the server nearest to A, it is in charge of continuing the negotiation. This time it operates on behalf of B_2 because B_2 is the issuer of the temporary ticket required by A. The exchange of the credential is then completed by B_1 with credentials by B_2, as described in Section 4.5.

6 Security Analysis

In this section, we present a detailed security analysis of the most critical algorithms characterizing the Trust-$\mathcal{X}$ multisession negotiation, along with the important security properties satisfied by our approach.

6.1 Tree Sharing Protocol Analysis

In order to ensure the validity and correctness of multisession negotiations, three properties must be verified by our protocols. Given a negotiation started between P_1 and P_2 and completed on behalf of P_2 by a delegate P_3, the properties are as follows:

- *Completeness*. P_3, upon successful completion of Algorithm 1, obtains a negotiation tree NT'_{open} whose nodes satisfy the corresponding committed terms in $\mathrm{commit}NT_{open}$.
- *Soundness*. P_3, upon successful proof of knowledge for a credential $cred$, can safely omit the names and values contained in $cred$ in the construction of NT'_{open}.
- *Zero-knowledge*. P_3 does not gain any information about the committed values in $\mathrm{commit}(NT_{open})$, upon checking its values against them using Algorithm 1.

Peer P_3 builds its version of the negotiation tree, NT'_{open}, by a node-by-node execution of proofs of knowledge of committed values obtained by P_2 by application of the Damgaard-Fujisaki commitment scheme. Thus, soundness and completeness properties are guaranteed by such commitment scheme, whose security is based on the following well-known cryptographic assumptions (see [12]):

- *Computational Diffie-Hellman (CDH) assumption.* Given a finite cyclic group G, a group generator g and group elements g^a, g^b, there exists no polynomial time algorithm that computes g^{ab}, with nonnegligible probability.
- *Strong RSA assumption.* Given an RSA modulus n and a random value x in $\mathbb{Z}_n^*$, there exists no efficient algorithm that computes $e > 1$ and $y \in \mathbb{Z}_n^*$ (with nonnegligible probability) such that $y^e = x \bmod n$.
- *Random oracle hypothesis.* Informally, a random oracle is a function $H : X \rightarrow Y$ uniformly chosen in the set $\mathcal{H}$ of all function from X to Y. It models a real-world cryptographic hash function, such as SHA-1.

We further note that in our proof Algorithm 1 the repeated execution of the proof of knowledge does not leak information, since all committed values are independent from each other. That is, P_2 uniformly draws each secret values r to be employed in the commitment scheme. Regarding the zero-knowledge property, assuming the random oracle hypothesis, the noninteractive proofs obtained employing the Fiat-Shamir heuristics are zero-knowledge, because the interactive proofs are zero-knowledge [29]. Regarding the security of the resume in the credential exchange phase, described in Section 4.5, we note that P_3 receives an encrypted version $\mathrm{Enc}(Cred_i, k_s)$ of credentials $Cred_i$ for exchange with P_1. However, P_3 is not able to decipher them, since the symmetric key k_s has been sent by P_2 to P_1 only. Another important aspect of our security model is the security of the delegation process. Essentially, two important properties must be guaranteed. First, the delegated party must obtain all and only the data necessary to successfully carry on the negotiation. Second, and equally important, the delegated party must not leak information regarding the peer that originally started the negotiation process. The first property is guaranteed by the results stated by Theorem 6.2. Based on these results, we can guarantee that the secret share disclosed to the delegate, once combined and opened for the secret disclosure, provides all necessary information to advance the negotiation. The nondisclosure property, instead, guarantees that only the information needed to advance the negotiation is provided to the delegate. The zero-knowledge property, introduced earlier, guarantees that the delegates cannot infer any information concerning the credentials' content nor the disclosure policies in that the content of the disclosure policies is hidden.

6.2 Formal Properties of the Multisession Protocol

We begin by introducing a *correctness* property.

Definition 6.1 (Trust Negotiation Correctness). *A trust negotiation protocol is correct if it successfully completes all and only the negotiations satisfying Definition 3.1.*

In our context, correctness refers to the fact that despite suspension and possible changes to the originally exchanged credentials and/or policies, the success of the negotiation is guaranteed if and only if a set of valid credentials satisfying the policies of both parties is identified.

Validity is informally defined as the property of using only valid credentials (i.e., credentials that have neither been revoked nor expired) to determine success of the negotiation.

Definition 6.2 (Trust Negotiation Validity). *Let $S = (C_1, \ldots, C_k = R)$ be a credential sequence and $pol_1, \ldots, pol_k$ be the policies protecting $C_1, \ldots, C_k$, respectively. A trust negotiation protocol is valid if all the credentials in S are valid.*

Next, we discuss the notion of minimality. Minimality implies that regardless of the number of exchange and intermediate steps in the negotiation, only credentials required for the purpose of the negotiation are disclosed. We begin with presenting the definition of minimal disclosure.

Definition 6.3 (Trust Negotiation Minimal Disclosure). *Let P_1 and P_2 be two peers negotiating for a resource $\mathcal{R}$. Let $\{m_1, \ldots, m_k\}$ be the set of messages exchanged between P_1 and P_2. Let P_3 be P_2's delegate and $\{m_k, \ldots, m_{k+n}\}$ be the second set of messages exchanged by P_1 and P_3, where $m_{k+n} = \mathcal{R}$ or $m_{k+n} = Fail$. A negotiation has the minimal information disclosure property if and only if, given the negotiation tree $NT = \langle \mathcal{E}, R, \mathcal{N} \rangle$, and the related credential sequence $\mathcal{CS}eq$, each m_i in $\{m_1, \ldots, m_k\} \cup \{m_k, \ldots, m_{k+n}\}$ is of one of the following types:*

1. $m_i = ack$ *or* $m_i = deny$
2. $m_i = \{pol_1, \ldots, pol_k\}$ *and*

$$\forall pol_i = R \leftarrow \phi(t_i, \ldots, t_n) \in m_i \exists$$

   a corresponding node $n \in \mathcal{N}$
3. $m_i = \{c_1, \ldots, c_m\}$ *and* $\forall c_i \in \mathcal{CS}eq$
4. $m_i = S$, *that is, a share of a serialized tree*
5. $m_i = \{c_1, \ldots, c_m\}$ *and each* $c_i \in m_i$ *is s.t.* $c_i = s_{c'}^{pol} \wedge c' \in \mathcal{CS}eq$.

Correctness and the following minimality properties are satisfied by the MS protocol described in the paper. The proof for the validity property is straightforward. Validity of the negotiation is guaranteed by the multiple controls about credentials validity that are an integral part of the negotiation process. Our approach prevents the usage of expired credentials by introducing a preprocessing phase that identifies the critical credentials (collected in the $\mathcal{PSC}$ list), subject to expiration, and by carefully replacing credentials as needed, while preserving the correctness and minimality of the process. Formal proofs showing these results are reported in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2011.31, of the paper. The minimality property is analyzed with respect to the negotiation final outcome and the overall negotiation data exchange. First, we consider minimality with respect to the success of negotiation. The definition essentially states that the credentials actually disclosed for completing a successful negotiation, are all and only those required by the policies exchanged, and such that all the policies to be satisfied (belonging to both peers) are satisfied by at least one term.

Definition 6.4 (Trust Negotiation Minimality). *Let $S = (C_1, \ldots, C_k = R)$ be a credential sequence and $pol = \{pol_1, \ldots, pol_k\}$ be the policies protecting $C_1, \ldots, C_k$, respectively. The credential sequence is minimal if:*

1. *for every credential $C_i \in S$ there exists a policy $pol_j = C_i \leftarrow \phi(t_1, \ldots, t_k)$ such that there exists a term $t \in \{t_1, \ldots, t_k\}$ such that t^{C_i} holds, or $C_i = R$;*
2. *for every policy $pol_i \in P$, there exists $C \subseteq S$ such that pol_i^C holds.*

Notice that a credential C used for completing the negotiation may be a substitute of previously committed credentials. By definition, as long as policies are satisfied by one or more credentials, the minimality property is preserved.

The next definition related to minimality is given with respect to the success of the negotiation process itself. This property implies that all credentials and policies exchanged during the negotiation, although they may not be used for the negotiation final sequence of credentials, are required to ensure the negotiation correctness. By nature, the trust negotiation may involve multiple rounds to explore the various alternatives, and thus not leading toward the negotiation success. However, needless rounds are to be prevented, to avoid information leaks and covert channels. Hence, in order to ensure that *minimal* and *correct* negotiations (see Definition 6.1) are supported by our protocols, the trust negotiation minimal disclosure must be satisfied. Trust negotiation minimal disclosure is defined in what follows.

Under the assumption that peers are honest and faithfully follow the negotiation protocol, the following property holds.

Theorem 6.1. *The MS protocol guarantees minimal disclosure.*

Finally, our protocols satisfy the following nondisclosure property. Given a negotiation, a corresponding reawakened negotiation does not disclose any additional information other than the one intended by the negotiation before suspension. We notice that this property holds even in case of replaced negotiations.

Theorem 6.2. *The resume of a MS negotiation satisfies the nondisclosure property.*

6.3 MS Negotiation Resiliency

Our protocol does not directly address issues concerning confidentiality and integrity of the involved parties. However, these can be dealt with using standard cryptographic technologies like encryption and signature schemes, as well as cryptographic hash functions for assuring messages' integrity [32]. Attacks based on messages' reuse like replay attacks can be prevented by adding timestamps to messages.

We will see that our approach is robust enough to tolerate a malicious, active peer that tries to resume a suspended negotiation with fake or nonexisting credentials.

On the other side, using a secret sharing scheme and a commitment scheme prevents the disclosure of confidential information to malicious, colluding peers.

The allowed behavior of the peers involved in the negotiation, will cause the negotiation execution to (correctly) fail because of the lack of the required credentials to be disclosed during the credential exchange phase. Even if the honest peer did not have the complete state of the negotiation handy, because of the assumption of unilateral negotiations, after the suspension phase both the peers will have reconstructed the intermediate state of the negotiation, which provides enough information to detect malicious requests. Finally, in case the negotiation was never suspended and one party had the whole process on its side, we assume that critical information related to the credentials involved and the corresponding policies will be available to it, and sufficient to detect the malicious behavior. In the case of insertion of fake credential or terms in the credential sequence, the negotiation process will fail because of the lack of the corresponding credentials to be disclosed during the credential exchange phase, as indicated by $\mathcal{CS}eq$. Whether the malicious peer is the original peer participating in the negotiation or the delegate, a mismatch of the exchanged credentials with respect to the shared credential will be determined. If the originator tries to alter the content of one of its credentials, the delegate will not find the matching item in the list. If it rejects a decrypted credential belonging to P_2, it may obtain sensitive information and then purposely failing the negotiation. The usage of selective disclosure actually defeats such kind of naive attacks. Finally, we remark that the secret sharing protocol and the tree protocols ensure that any possessor of a share cannot infer any information regarding the negotiation process. Even in case two delegated peers, say P_3 and P_4, collude to gather information regarding the negotiation, they cannot gather any private information. If $S = \mathrm{ser}(\mathrm{commit}(NT_{open}))$ is reconstructed, the only data gathered by the malicious peers are a committed version of some of the tree nodes. In case $S = \mathcal{CS}eq$, the colluding peers can only obtain a sequence of encrypted credentials, the security relies on the strength of the adopted commitment protocol.

7 Complexity Analysis and Experimental Results

In this section, we analyze the complexity of our protocols, in terms of the time and space and in terms of the number of exchanged messages. We consider the case in which one of the two peers may change, during the resumption of a trust negotiation. For the case in which peers do not change, we refer to [30] for results about complexity.

7.1 Policy Exchange Phase Complexity

Unsurprisingly, the nodes commitment, and tree sharing phases (along with the noninteractive proofs associated with the commitments) are the most computing-intensive parts of our approach. We start by analyzing them separately, by evaluating the number of required multiplications.

7.1.1 Nodes Commitment

Suppose that the negotiation tree NT_{open} contains n nodes. Suppose further that the maximum number of predicates contained in a term is p_{max}. Then, the number of commitments to be computed is $n(p_{max}+1)$. Each commitment computation entails two modular exponentiations and each of them can be computed in $O(log(\max\{m,r\}))$ multiplications. Thus, the total number of multiplications needed to commit all the nodes in NT_{open} is $O(n \cdot log(\max\{m,r\}))$. In real-world settings, the value of p_{max} usually does not exceed two or three.

7.1.2 Tree Sharing

The Shamir's secret sharing scheme can be modified in order to yield a very space-efficient version of the shares (see Section 7.2) [20]). The modified Shamir's secret sharing scheme does not add significative computational overhead and very efficient polylog algorithms for polynomial evaluation and interpolation are available [1].

7.1.3 Commitments Proofs

For each committed value, an equality zero-knowledge proof of knowledge requires P_2 and P_3 to compute each a modular exponentiation (other than those required for computing the commitments themselves, of course). For a zero-knowledge proof that a committed value lies in a given interval, P_2 must compute two additional commitments and five zero-knowledge equality proofs [7]. Hence, the overall time complexity, measured by the number of multiplications needed to execute the commitment, sharing, and zero-knowledge proofs protocols, linearly depends from the number of nodes contained in NT_{open}.

7.2 Communication Complexity

We measure the communication complexity in terms of the number of exchanged messages and in their size. The total number of messages that need to be exchanged is five, assuming that P_2 supports multicast and without taking into account the authentication between P_1 and P_3, as detailed in the following. Once one peer among P_1 and P_2 requests a suspension (first message), P_2 computes the two shares of the committed version of NT_{open} and sends them to P_1 (second message) and $P_3, \ldots, P_n$ (third message, assuming a multicast mechanism for the peers in P_2's pool), respectively. After that, P_1 sends a resume request to P_3 (fourth message) and, in case of its correct authentication, P_1 sends its share to it (fifth message). Once P_3 has correctly built its version of the negotiation tree NT'_{open}, P_3 sends it to P_1 (sixth message). The upper bound on the size of exchanged messages is the size of a share of the negotiation tree. Recall that the space-efficient version of the Shamir's secret sharing scheme operates on a linearized version of the (node-by-node committed) tree NT_{open}, denoted by $\mathrm{ser}(\mathrm{commit}(NT_{open}))$. The size of a committed node linearly depends on the size of security parameter s, typically an integer with a number of bits between 512 to 1,024. Hence, the size of $\mathrm{ser}(\mathrm{commit}(NT_{open}))$ linearly depends on the number of nodes in NT_{open}. Finally, note that the space-efficient version of the Shamir's secret sharing scheme yields two shares whose length is half the length of $\mathrm{ser}(\mathrm{commit}(NT_{open}))$ [20].

7.3 Credential Exchange Phase Complexity

The complexity of the operations performed in the credential exchange phase is significantly smaller than the complexity

of the policy evaluation phase. According to the protocol presented in Section 4.5, during the suspension phase, three messages are exchanged:

- *Message 1*. The request from P_1 to P_2 to suspend the credential exchange;
- *Message 2*. The message from P_2 to P_1 containing the credential sequence and the corresponding encryption keys; and
- *Message 3*. The message containing the credential sequence along with the encrypted credentials belonging to P_2, which are substituted to the proper terms. This message is sent from P_2 to $P_3, \ldots, P_n$ (assuming, again, a multicast mechanism for the peers in P_2's pool).

In addition, one additional message is sent by P_1 to P_3 to resume the negotiation. Hence, in case of correct authentication, the credential exchange phase resumes. The computational complexity of the second message depends on the symmetric encryption scheme chosen. Since standard symmetric encryption schemes (e.g., AES and/or DES) have linear time complexity, we can safely assume that the time complexity of suspension and resume of the credential exchange phase is linear in the number of P_2's credentials contained the credential sequence. The upper bound on the size of exchanged messages coincides with the size of the third message. The reason is that such message contains the credential sequence *and* the encrypted credentials. The number of encrypted credentials linearly depends on the number of P_2's credentials contained in the credential sequence as well as on the strategy adopted on how to deal with the substituted credentials.

7.4 Experimental Results

We have carried out a set of experiments to assess the performances of our approach. In particular, we compared the performance of the currently presented approach with the one previously presented in [30]. The experiments have been executed on two machines having the following features: Linux kernel 2.6.32, Intel(R) Core(TM)2 CPU T5600 @ 1.83 GHz, 2 GB RAM. We evaluated the time required to successfully perform a trust negotiation involving 50 credentials interrupted after 10, 20, 40, and 50 exchanged credentials. Such interruptions occur the policy evaluation phase because, as shown in [30], the time required to restart a negotiation interrupted in the credential exchange phase clearly shows the advantages of the recovery procedure. Moreover, the main runtime overhead introduced in the protocols presented here is caused by the splitting/merging procedure and the computation of the (de)commitments which, being related to the size of the negotiation tree, is constant along the entire credential exchange phase. As reported in Fig. 3, the overhead introduced by the tree splitting, the tree merging and the commitment and the decommitment procedures is negligible. In fact, consider the negotiation interrupted after that 50 policies have been exchanged. The single-side resume of a negotiation, represented in figure by the dashed line, with the above mentioned features, only requires 43 milliseconds more than the resumption of the same negotiation without tree splitting/merging and (de)commitments, represented by the dotted line.

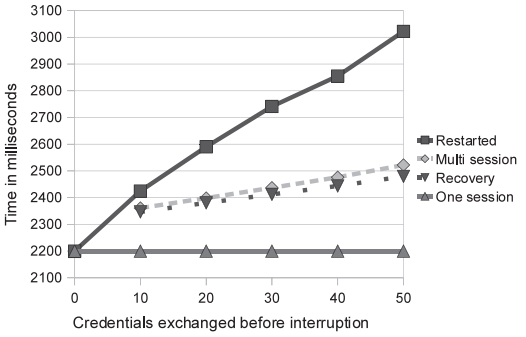

Fig. 3. Negotiation time for a negotiation suspended at different intervals.

On the other hand, another set of experiments shows that the time required to recover the negotiation interrupted during the policy evaluation phase (see Section 3.2.1 and Fig. 1) after having exchanged 50 policies is about less than 500 milliseconds with respect to the time required to restart it from the beginning. We refer also to [30] for further details.

8 Related Works

Approaches for trust negotiation in web-based applications have been extensively investigated in recent years. Trust negotiation was introduced by Winsborough and Li [33], who presented two negotiation strategies for conducting online transactions between strangers. Subsequently, research in trust negotiation has focused on a number of important issues including languages for expressing resource access policies (e.g., [13], [19], [24], [23]), protocols and strategies for conducting trust negotiations (e.g., [28], [37], [4]), and logics for reasoning about the outcomes of these negotiations. Trust negotiation systems have also been investigated with respect to privacy. Work along this direction has focused on the protection of sensitive policies and credentials. Yu and Winslett [37] 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 closer to the notion of access control. A formal framework for trust negotiations has been proposed by Winsborough and Li [34]. Their approach to safe enforcement of policies focuses on a privacy-preserving credential exchange, according to their notion of safety. None of the approaches discussed so far, however, take into account the possibility of suspending or accidentally interrupting a negotiation, nor they allow to delegate the negotiation process. Several privacy-enabled identity management systems have been proposed based on the notion of anonymous credential [10], [11]. Brands developed a form of digital credential in which a user has a single public credential, but that credential is pseudoanonymous, even to the issuer [9]. 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. Camenisch and Herreweghen have proposed and implemented yet another form of digital credential, referred to as Idemix [10]. Idemix is the first system implementing anonymous credentials in a federated identity management system. Idemix provides

mechanisms for efficient multishow credentials and also provides a mechanism for *all or nothing* sharing and PKI-based nontransferability. Anonymous credentials however may not be adequate for trust negotiations' protcols, that often 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. Our approach has some similarities with SecPal [2], a decentralized authorization language. SecPal is a declarative authorization language that supports domain-specific constraints and negation. SecPal represents a potential candidate for supporting the Trust-$\mathcal{X}$ protocol, although it would require several extensions. For example, SecPal does not provide a direct way to address credential-based policies, nor it provides an approach for specifying conditions against credentials. A desirable feature supported by SecPal is delegation, which may be very useful in case mobile negotiators decide to delegate not only the process but also the policies and credentials required to complete the negotiation on behalf of another party. However, SecPal is only a language, and does not have the infrastructure and the framework required to support trust negotiations. Research has also been conducted on logic-based access control languages for single administrative domains that do not require centralized delegation of authority. Many are based on DataLog or DataLog with Constraints [23]. Our approach to the suspend and resume procedures described in the paper is similar to saving and restoring a partial proof graph in a Datalog-based trust management system. When the negotiation is suspended, one of the two negotiating parties save their copies of the partial proof graph along with any variable mappings in effect. Claims that might expire can be denoted along side the proof graph. It would however be necessary to rescan all the provided claims on restart to determine in any of them had been revoked during the suspension period. Although DataLog provides a good foundation for access control in distributed systems, it lacks concrete applicability and does not efficiently model the realistic requirements of peer to peer interactions. Bowers et al. [8] developed an access control framework which uses a particular extension of linear logic—further extended by Garg et al. [15] by means of the introduction of the knowledge of the principals within the model—to deal with consumable credentials. In particular, the atomicity property of their framework is an alternative solution to the issue relative to the consumption of one-time credentials. By means of contract signing protocols their approach avoids the consumption of credentials, and performs a rollback of the transaction when the proof is not successful. However, our approach provides a more flexible solution allowing the parties to perform the verification of the properties over a longer period. Moreover, their approach requires the presence of online "ratifier" to guarantee the correctness of the proof while our approach does not involve external entities. Trust negotiations have been proved to be viable access control solutions for systems through a series of implementations (such as those presented in [24] and [16]) which demonstrate the utility and practicality of these theoretical advances. Implementations of trust negotiation typically provide a means of parsing policies, handling certified attributes, and determining policy satisfaction. PeerTrust [24] is a noteworthy example. Peertrust is a management system that uses a simple policy language based on logic programs language. The underlying trust negotiation protocols are similar to the ones referred in this paper. PeerTrust policies are evaluated by a Datalog-based logic engine that supports delegation of authority, signed rules, expression of complex conditions, and sensitive policies. The prototype system, however, is limited to two party negotiations and does not support recovery or suspension. Finally, this work has its roots in the Trust-$\mathcal{X}$ system. Trust-$\mathcal{X}$ includes an XML-based language for policy and credential specification, and an engine to carry on trust negotiations. Trust-$\mathcal{X}$ unique features include support for efficient negotiations—using cache and trust tickets [4]—compliance with privacy policies, and P2P framework [3] and recovery features [30]. The protocols presented in this paper have some major differences with respect to the previous Trust-$\mathcal{X}$ protocols. The previous protocols relied on Merkle hash tree techniques to save the negotiation state for recovery, while we currently provide a new alternative and more powerful approach for suspension and resume. Furthermore, we 1) did not support credentials' substitution, 2) did not support neither mobile negotiations nor voluntary suspensions. A preliminary approach to voluntary suspension has been sketched in [31].

9 Conclusion

In this paper, we have presented a multisession dependable approach to trust negotiations. The proposed framework supports voluntary interruptions, enabling the negotiating parties to complete the negotiation despite temporary unavailability of resources. In designing the protocols, we have carefully considered all possible issues related to validity, temporary loss of data, and extended unavailability of one of the two negotiators. To this extent, we introduced protocols for mobile negotiations. Using Trust-$\mathcal{X}$, a peer is able to suspend an ongoing negotiation and resume it with another (authenticated) peer. The protocols presented in this work can be applied to any trust negotiation system that adopts a two phase negotiation protocol. It is part of our future work to further investigate how to migrate the proposed protocols to any general trust negotiation infrastructure.

Acknowledgments

The work reported in this paper has been partially supported by the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.
[2] M.Y. Becker, C. Fournet, and A.D. Gordon, "Design and Semantics of a Decentralized Authorization Language," *Proc. IEEE 20th Computer Security Foundations Symp. (CSF),* pp. 3-15, 2007.
[3] E. Bertino, E. Ferrari, and A.C. Squicciarini, "Privacy-Preserving Trust Negotiation," *Proc. Fourth Privacy Enhancing Technologies Workshop,* May 2004.
[4] E. Bertino, E. Ferrari, and A.C. Squicciarini, "Trust-$\mathcal{X}$: A Peer-to-Peer Framework for Trust Establishment," *IEEE Trans. Knowledge Data Eng.,* vol. 16, no. 7, pp. 827-842, July 2004.
[5] E. Bertino, E. Ferrari, and A.C. Squicciarini, "Trust Negotiations: Concepts, Systems and Languages," *Computing in Science Eng.,* vol. 6, no. 4, pp. 27-34, 2004.

[6] E. Bertino, I. Ray, A.C. Squicciarini, and E. Ferrari, "Anonymity Preserving Techniques in Trust Negotiations," *To appear in Proc. Fifth Privacy Enhancing Technologies Workshop,* 2005.
[7] F. Boudot, "Efficient Proofs that a Committed Number Lies in an Interval," *Proc. EUROCRYPT,* pp. 431-444, 2000.
[8] K.D. Bowers, L. Bauer, D. Garg, F. Pfenning, and M.K. Reiter, "Consumable Credentials in Linear-Logic-Based Access-Control Systems," *Proc. Network and Distributed System Security Symp. (NDSS),* 2007.
[9] S.A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, 2000.
[10] J. Camenisch and E.V. Herreweghen, "Design and Implementation of the *Demix* Anonymous Credential System," *Proc. ACM Conf. Computer and Comm. Security,* pp. 21-30, 2002.
[11] D. Chaum, "Showing Credentials without Identification Transferring Signatures between Unconditionally Unlinkable Pseudonyms," *Proc. Int'l Conf. Cryptology on Advances in Cryptology (AUSCRYPT),* pp. 246-264, 1990.
[12] I. Damgård and E. Fujisaki, "A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order," *ASIACRYPT '02: Proc. Eighth Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology,* pp. 125-142, 2002.
[13] E. Ferrari, A. Squicciarini, and E. Bertino, "*X*-tnl: An Xml Language for Trust Negotiations," *Proc. IEEE Fourth Workshop Policies for Distributed Systems and Networks,* June 2003.
[14] A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems," *Proc. Int'l Cryptology Conf. (CRYPTO '86),* pp. 186-194, 1986.
[15] D. Garg, L. Bauer, K.D. Bowers, F. Pfenning, and M.K. Reiter, "A Linear Logic of Authorization and Knowledge," *Proc. European Symp. Research in Computer Security (ESORICS),* pp. 297-312, 2006.
[16] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K.E. Seamons, and B. Smith, "Advanced Client/Server Authentication in TLS," *Proc. Network and Distributed System Security Symp. (NDSS),* 2002.
[17] J.E. Holt, R.W. Bradshaw, K.E. Seamons, and H. Orman, "Hidden Credentials," *WPES '03: Proc. ACM Workshop Privacy in the Electronic Soc.,* pp. 1-8, 2003.
[18] W. Hu, N. Jian, Y. Qu, and Y. Wang, "Gmo: A Graph Matching for Ontologies," *Proc. Workshop Integrating Ontologies,* 2005.
[19] T. Yu, K.E. Seamons, and M. Winslett, "Protecting Privacy During on Line Trust Negotiation," *Proc. Second Int'l Conf. Privacy Enhancing Technologies,* Apr. 2002.
[20] H. Krawczyk, "Secret Sharing Made Short," *CRYPTO '93: Proc. 13th Ann. Int'l Cryptology Conf. Advances in Cryptology,* pp. 136-146, 1993.
[21] A.J. Lee and M. Winslett, "Enforcing Safety and Consistency Constraints in Policy-Based Authorization Systems," *ACM Trans. Information and System Security,* vol. 12, no. 8, pp. 1-33, Dec. 2008.
[22] J. Li and N. Li, "Oacerts: Oblivious Attribute Certificates," *IEEE Trans. Dependable and Secure Computing,* vol. 3, no. 4, pp. 340-352, Oct.-Dec. 2006.
[23] N. Li and J.C. Mitchell, "Datalog with Constraints: A Foundation for Trust Management Languages," *Proc. Fifth Int'l Symp. Practical Aspects of Declarative Languages,* Jan. 2003.
[24] W. Nejdl, D. Olmedilla, and M. Winslett, "PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web," *Proc. Workshop Secure Data Management in a Connected World (SDM '04),* Aug. 2004.
[25] Organization for the Advancement of Structured Information Standards (OASIS) "Security Assertions Markup Language (SAML), Version 2.0," http://wiki.oasis-open.org/security, 2005.
[26] T.P. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," *CRYPTO '91: Proc. 11th Ann. Int'l Cryptology Conf. Advances in Cryptology,* pp. 129-140, 1991.
[27] Y. Qu, W. Hu, and G. Cheng, "Constructing Virtual Documents for Ontology Matching," *Proc. 15th Int'l Conf. World Wide Web (WWW),* pp. 23-31, 2006.
[28] K.E. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation," *Proc. Network and Distributed System Security Symp. (NDSS),* 2001.
[29] A. Shamir, "How to Share a Secret," *Comm. ACM,* vol. 22, no. 11, pp. 612-613, 1979.
[30] A.C. Squicciarini, A. Trombetta, and E. Bertino, "Supporting Robust and Secure Interactions in Open Domains through Recovery of Trust Negotiations," *Proc. 27th Int'l Conf. Distributed Computing Systems (ICDCS),* p. 57, 2007.
[31] A.C. Squicciarini, A. Trombetta, E. Bertino, and S. Braghin, "Identity-Based Long Running Negotiations," *Proc. Fourth ACM Workshop Digital Identity Management,* pp. 97-106, 2008.
[32] W. Stallings, *Cryptography and Network Security.* second ed. Prentice Hall, 1998.
[33] W.H. Winsborough and N. Li, "Towards Practical Automated Trust Negotiation," *Proc. Third Int'l Workshop Policies for Distributed Systems and Networks (Policy '02),* pp. 92-103, June 2002.
[34] W.H. Winsborough and N. Li, "Safety in Automated Trust Negotiation," *Proc. IEEE Symp. Security and Privacy,* pp. 147-160, 2004.
[35] W.H. Winsborough and N. Li, "Safety in Automated Trust Negotiation," *ACM Trans. Information and System Security,* vol. 9, no. 3, pp. 352-390, 2006.
[36] World Wide Web Consortium "OWL Web Ontology Language," http://www.w3.org/TR/owl-ref/, 2004.
[37] T. Yu and M. Winslett, "A Unified Scheme for Resource Protection in Automated Trust Negotiation," *Proc. IEEE Symp. Security and Privacy,* pp. 110-122, 2003.

Anna C. Squicciarini received the PhD degree in computer science from the University of Milan in 2006. She is currently an assistant professor of Information and Sciences and Technology at the Pennsylvania State University. Previously, she was a postdoctoral research associated at Purdue University. Her research interests lie at the intersection of the computer security, privacy, and distributed systems fields. She is particularly interested in access control approaches for web applications.

Elisa Bertino is a professor of Computer Science at Purdue University and serves as research director of the Center for Education and Research in Information Assurance and Security (CERIAS). Her main research interests include security, privacy, digital identity management systems, database systems, distributed systems, multimedia systems. In those areas, She has published more than 400 papers in all major refereed journals, and in proceedings of international conferences and symposia. She received the 2002 IEEE Computer Society Technical Achievement Award for "For outstanding contributions to database systems and database security and advanced data management systems" and the 2005 IEEE Computer Society Tsutomu Kanai Award for pioneering and innovative research contributions to secure distributed systems. She is currently serving on the Board of Governors for the IEEE Computer Science Society. She is a fellow of the IEEE and a fellow of the ACM and has been named a Golden Core Member for her service to the IEEE Computer Society.

Alberto Trombetta received the PhD degree in computer science from the University of Turin, Italy. He is an assistant professor at the Department of Computer Science and Communication of Insubria University at Varese, Italy. His main research interests are in security and privacy issues of large, data-intensive distributed applications.

Stefano Braghin received the PhD degree in computer science in 2011, from the University of Insubria at Varese, Italy. He is currently a research associate at the University of Insubria, Italy. His main research interests are related to trust management and privacy in distributed systems.