Combining Access Control and Trust Negotiations in an On-line Social Network

(Invited Paper)

Stefano Braghin
DICOM
University of Insubria
Varese – Italy
s.braghin@uninsubria.it

Elena Ferrari
DICOM
University of Insubria
Varese – Italy
elena.ferrari@uninsubria.it

Alberto Trombetta
DICOM
University of Insubria
Varese – Italy
a.trombetta@uninsubria.it

***Abstract*—Protection of On-line Social Networks (OSNs) resources has become a primary need since today OSNs are the hugest repository of personal information on the Web. This has resulted in the definition of some access control models tailored to the protection of OSN resources. One of the key parameter on which access control decisions in OSNs should be based is represented by the trust between OSN users. A well-known approach for the management of trust relationships is represented by trust negotiations [1], [2]. In this paper, we show how access control and trust negotiation can be combined in a framework for the protection of OSN resources. Moreover, we show how the outcome of a trust negotiation can be exploited to dynamically adjust the trust level between OSN users.**

***Index Terms*—social network, access control, trust negotiation, trust management**

I. Introduction

Hundreds of millions of users interact and share their information and resources via *On-line Social Networks* (OSNs, for short). As a result, users store they personal profiles, messages, photos, videos and the like in sites like Facebook, MySpace, Orkut, Flickr and hundreds of other OSNs, which have become the main means of sharing and exchanging (personal) information on the Web. Recently, organization and companies as well have started to use OSNs in order to share information and doing their businesses. The introduction of Semantic Web technologies, such as FOAF (Friend Of A Friend) [3], [4] has simplified – on one side – the automatic sharing of information among different OSNs, while – on the other side – has required a stricter control from the users on the diffusion of their information. In fact, most of the current OSNs allow users to specify the state of their data (i.e., public, private, or readable only by authorized users like direct friends, for example). Along this line, some access control models tailored to OSNs have been proposed (see [5] for a survey). Almost all these proposals express access control/privacy requirements as constraints on OSN users social graph. Such constraints are usually stated in terms of the type, depth and possibly the trust level of the relationships that must hold between two users u and u' because u can access one of u' resources.

A complementary approach for a controlled resource release is represented by trust negotiation [1]. Trust negotiation allows two mutually initially untrusting parties (which do not have any shared, previously agreed access policies to their respective resources) to eventually agree on the disclosure of resources protected by local access policies by acquiring a certain level of trust. This is reached by exchanging – upon an initial resource request – the local access policy for the requested resource, which in turn may ask other resources (protected by other local policies) and so on. The policy exchange is carried on until a resource not protected by any further resource is found. This triggers the exchange of the resources mentioned in the exchanged policies, until the initially requested resource is disclosed. In the case that unprotected resources are not found, the corresponding trust negotiation fails and the initially requested resource is not disclosed. In the past few years, there have been several research efforts that have resulted in frameworks specifying proper negotiation languages and prototypes enforcing trust negotiations in distributed environments [1], [6], [7], [2], [8]. However, none of these proposals consider the OSN scenario. Our aim in this paper is to combine access control and trust negotiation into a framework for the protection of OSN resources to exploit the benefits of both the paradigms. More precisely, aim of the present work is the introduction of trust negotiations within the access control framework for OSNs defined in [9]. In [9], access control policies are expressed as access rules that specify authorized users in terms of attributes like relationship type, depth and trust among users in the social network. The enforcement of such access policies is performed client-side. Thus, a user is guaranteed access to a given information or resource when she/he is able to prove to the information/resource owner that she/he satisfies the corresponding access rules.

More precisely, our contribution is twofold. First, we extend the framework in [9] by allowing a resource to be disclosed by means of a trust negotiation, besides through the satisfaction of a set of access control policies. In such a way we provide a more flexible way according to which access control requirements of a resource owner can be specified and managed. Second, we relate the trust level existing between two OSNs users to the outcomes of the negotiations they have previously carried out. More precisely, we propose a method to dynamically adjust the trust level of a relationship

COLLABORATECOM 2010, October 9-12, Chicago, USA

DOI 10.4108/icst.collaboratecom.2010.18

existing between two users within the OSN on the basis of the results of previous trust negotiations between them. This is achieved by dynamically modify the OSN's graph, both in term of existing relationships and in terms of trust associated with such relationships. In the paper, we also discuss how to take advantages of such relationships in the access control procedure.

The remainder of the paper is organized as follows: Section II discusses related work, whereas in Section III we present a brief overview of OSN's access control models and trust negotiation mechanisms. In Section IV we describe how to extend the access control model in [9] to include resources as requirements and to let it interact with a trust negotiation infrastructure. Section V presents our proposal to take advantage of trust negotiations to dynamically adjust the trust between two OSN users and to speed up the access control process. In Sections VI and VII we present the complexity analysis of the proposed access control system and some experimental results which we performed on such system, respectively. Finally, Section VIII summarizes some conclusions.

II. Related work

To the best of our knowledge, we are not aware of proposals aiming at combining access control and trust negotiation in the framework of OSNs. Indeed, research work focusing on OSN privacy and security is quite recent. As far as privacy is concerned, current work is mainly focusing on protecting private information while performing social network analysis [10]. In contrast, research in the field of access control has resulted in the definition of some access control models and related mechanisms aiming to overcome the restrictions of the protection mechanisms provided by current OSNs (see [5] for a survey). One of the common characteristics of almost all the defined access control models is that access control is *relationship-based*, that is, authorized users are denoted on the basis of constraints on the relationships the requester should have with other network users. Such constraints refer to the depth of the relationship and/or its type. Some models (e.g., [9]) also support constraints on the trust level of a relationship. However, the problem of how to compute the trust of a relationship, which is addressed in the current paper, has not been deeply addressed so far. The only work we are aware of considering the interplay among trust and access control in OSNs is [11], in which a privacy-preserving mechanism has been defined able to help an OSN user to compute the trust values to be assigned to other network nodes. The approach makes use of an audit file, which stores relevant information to estimate OSN user trust. More precisely, each node in the OSN is equipped with an audit file, which reports an anonymized version of all the user decisions with respect to the release of resources and personal data, plus some additional information that makes the other nodes in the network able to evaluate whether the decisions are compliant or not with the specified privacy preferences/access control policies without violating user privacy. In the current paper, we propose a different approach for trust computation, where trust can be estimated based on the result of previous trust negotiations.

Even if not oriented to OSNs, trust negotiation for web-based applications has been an extensively investigated research area in recent years. The idea of trust negotiation was originally introduced by Winsborough et al. [1], which also present different strategies to conduct on-line transactions between strangers.

The research following the work by Winsborough et al. has mainly focused on three issues: i) the definition of languages for expressing resource access policies (e.g., [6], [12], [7], [13]), ii) the design of protocols and strategies for conducting trust negotiations (e.g., [14], [8], [2]), and iii) the development of 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. Winslett and Yu [8] have developed a unified scheme, known as *Unipro*, to model resource protection, which applies to both the actual resources to be protected and policies. A formal framework for trust negotiations has been proposed by Winsborough and Li [15]. Their approach to safe enforcement of policies focuses on a privacy-preserving credential exchange. A formal notion of safety in automated trust negotiation is given, stating when a negotiation is secure against inferences that a party may make against the profile of the other party.

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 caching and trust tickets [2], compliance with privacy policies, and P2P framework [16], and recovery support [17], [18].

III. Background

In this section we will briefly introduce some basic notions which are required in the remainder of the paper.

A. A brief introduction to OSN access control

Similarly to other networks, an OSN SN can be represented as a labeled graph, where each node denotes a user in the network, whereas edges represent the existing relationships between users, and their trust levels. Edge direction denotes which node specified the relationship and the node for which the relationship has been specified, whereas the label associated with each edge denotes the type of the relationship. Figure 1 shows an example of OSN graph.

The number and type of supported relationships depend on the specific OSN and its purposes; our only assumption is that there exists at least one relationship type. We also assume that, if RT denotes the set of supported relationship types, given two nodes $A, B \in SN$, there may exist at most $|RT|$ edges from A to B (from B to A, respectively), all labeled with distinct relationship types. We can now formally define OSNs as follows.

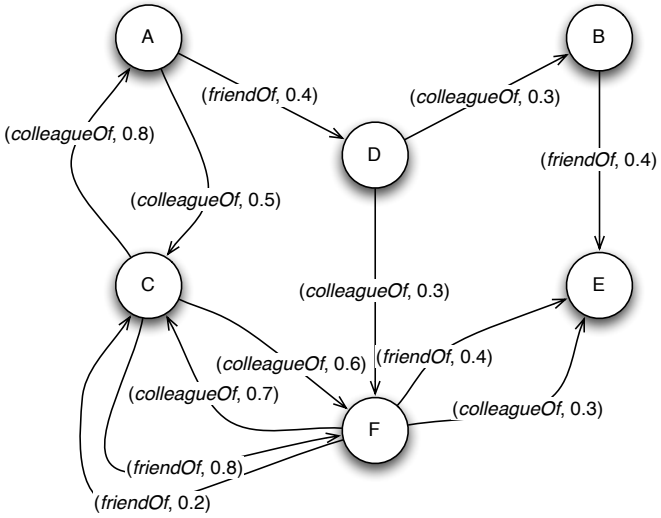

Figure 1. An example of OSN labeled graph

Definition 3.1 (OSN): [9] An OSN SN is a tuple $(V_{SN}, E_{SN}, RT_{SN}, \phi_{SN})$, where RT_{SN} is the set of supported relationship types, V_{SN} and $E_{SN} \subseteq V_{SN} \times V_{SN} \times RT_{SN}$ are, respectively, the nodes and edges of a directed labeled graph $(V_{SN}, E_{SN}, RT_{SN}, \phi_{SN})$, whereas $\phi_{SN} : E_{SN} \rightarrow [0, 1]$ is a function assigning to each edge E_{SN} a trust level t, which is a rational number in the range $[0, 1]$.

An edge $e = vv' \in E_{SN}$ expresses that node v has established a relationship of a given type $rt_e \in RT_{SN}$ with node v'.

In this paper, we assume that OSN resources are protected according to the model presented in [9]. According to this model, access control requirements of OSN users are expressed in terms of *access rules* specified by resource owners. Access rules denote authorized members in terms of the type, maximum depth and minimum trust level of the relationships they must have with other network nodes. Such constraints are expressed as a set of *access conditions* $(v, rt, d_{max}, t_{min})$, where v is the node with which the requesting node must have a relationship of type rt, whereas d_{max} and t_{min} are, respectively, the maximum depth and the minimum trust level that the relationship must have. If $v = *$ and/or $rt = *$, v corresponds to any node in the OSN and/or rt corresponds to any supported relationship type. Whereas if $d_{max} = *$ and/or $t_{min} = *$, there is no constraint concerning the depth and/or trust level, respectively.

Example 3.1: Referring to the OSN in Figure 1, suppose that Alice (A) holds a resource r that she wants to share only with her colleagues or the colleagues of their colleagues no matter of their trust level. She can encode such requirement by the following access condition $(A, colleagueOf, 2, *)$. Moreover, if a resource is protected by the following access rule $\{(A, friendOf, 1, 1)\}$, it means that it can be accessed only by A more trusted direct friends.

In [9], access control is client-based, according to which the requester must provide the resource owner with a proof of being authorized to access the requested resource. As access rules constraint relationships, the proof has to show the existence of a path in the network satisfying the constraints on depth and trust level imposed by the rule. In order to generate valid proofs, it is assumed that a "relationship certificate" is associated with each relationship, containing information on the relationship (i.e., users involved, trust, depth, type), which is signed by both the involved users. A relationship certificate can be seen as a proof that between the involved users there exists a direct relationship of a certain type and with a certain trust level. Proofs of indirect relationships can therefore be generated through a set of certificates confirming the existence of a path of a specified type between them. Certificates are managed by a trusted authority CS which is in charge of path retrieval and delivering to a resource requester.

More details can be found in [9], here we just briefly recall the access control protocol, which is graphically depicted in Figure 2[1], taken from [9], since it is required in the following of the paper. The explanation of the protocol is provided in Figure 3.

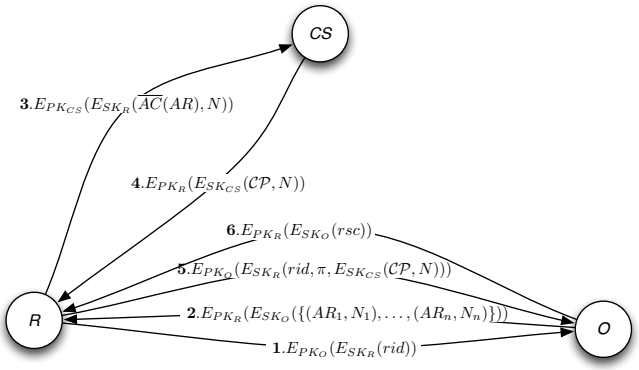

Figure 2. Access control protocol. R is the node requesting a resource with identifier rid, O is the node owning the resource, whereas CS is the certificate server.

B. A brief introduction to the Trust-$\mathcal{X}$ framework

Trust-$\mathcal{X}$ is a comprehensive framework for defining and managing on-line trust negotiations [2]. It is based upon a peer-to-peer architecture and a rule-based policy language called $\mathcal{X}$-RNL [19].

A Trust-$\mathcal{X}$ negotiation is an interactive process between two parties – called respectively *Requester* and *Owner* – having the goal to establish mutual trust in order to release a given resource. We assume that the resource description is encoded into a credential, that is, a list of relevant attributes of the resource, along with the corresponding values. We further assume that a resource is protected by a disclosure policy (held by the Owner), which details what conditions are to be satisfied by the Requester before the Owner releases the resource. Typically, the Requester's conditions are encoded

[1]In the figure, $E_k(d)$ denotes the encryption of d with key k.

1) R submits to O an access request for resource rsc, with identifier rid.
2) If the resource is public, access is granted. Otherwise, O returns to R the set of access rules $AR = \{\mathtt{AR}_1, \ldots, \mathtt{AR}_n\}$ regulating the access to rsc. With each access rule $\mathtt{AR}_i \in AR$, $i \in [1, n]$, a distinct nonce value $\mathtt{N}_i$ is associated as a session identifier.
3) R chooses from AR an access rule ar and sends CS the nonce value N associated with ar and the corresponding condition set $AC(\mathtt{AR})$. each $ac \in AC(\mathtt{AR})$ is set to NULL.
4) CS returns R the set $\mathcal{CP}$ of shortest certificate paths, if any, related to the relationship constraints expressed by the access conditions in $AC(\mathtt{AR})$, along with the nonce N associated with AR; otherwise, CS returns a failure message. In the latter case, R goes back to step 3 and chooses another access rule, until CS returns the set $\mathcal{CP}$, if any, or all the access rules have been processed.
5) Based on the certificate paths in $\mathcal{CP}$, R tries to generate a proof π of the existence of a path satisfying ar.
If a proof is not obtained, R goes back to Step 3 and chooses another access rule; otherwise, he/she sends O a message, which contains the resource identifier, the proof π, and the certificate paths obtained from CS. $\mathcal{CP}$ and N are kept encrypted with the private key of CS in order to grant their authenticity.
6) O sends R the requested resource in case the proof π is valid and the nonce value N corresponds to the correct session identifier.

Figure 3. Description of the access control protocol depicted in Figure 2

into predicates about credentials, which are to be disclosed themselves to the Owner, in order to check whether they satisfy the disclosure policy. It may be well the case that such credentials contain sensitive information and, hence, they may be protected by another disclosure policy (held this time by the Requester). Henceforth, a negotiation between Requester and Owner (composed of interleaved, mutual credentials' requests) ensues. The negotiation successfully ends in case both parties agree on a set of credentials that can be unconditionally disclosed.

As such, the negotiation process is divided into three distinct phases:

1) *introductory phase*: the parties identify the resource R to be released;
2) *policy evaluation phase*: the parties iteratively exchange disclosure policies, in order to possibly agree upon a set of credentials to be exchanged for the release of R;
3) *credential exchange phase*: the parties actually exchange the credentials according to the disclosure policies, agreed in the previous phase.

The phases of the negotiation process described above have been extended in several ways to provide different features such as:

- the renegotiation of a request upon a negative reply [20];
- the recovery of a crashed negotiation [17] and
- safely perform a trust negotiation during distinct negotiation sessions [18].

Note that in the present work we use the basic negotiation process. Nevertheless it is trivial to exploit the features provided by the advanced versions of Trust-$\mathcal{X}$ by enabling such features in the framework and by using more advanced constructs in the trust negotiation language.

We now briefly introduce a relevant data structure – the *negotiation tree* – that is used by Trust-$\mathcal{X}$ for carry out trust negotiations. As we will show in Section IV, such data structure plays an important role in the proposed access control protocol.

During the policy evaluation phase, the negotiating parties create a negotiation tree, which is the data structure which tracks the evolution of the negotiation. Moreover, the negotiation tree also indicates all the resources which the negotiating parties have to disclose in order to successfully terminate the trust negotiation.

More formally, given two parties O, the resource owner, and R, the resource requester, such that with each negotiating party a set of resources $\mathcal{R}_k$ and a set of disclosure policies[2] $\mathcal{RP}_k$ with $k \in \{O, R\}$ are associated. Let $r \in \mathcal{R}_O$ be the resource requested by R. A negotiation tree is defined as follows.

Definition 3.2 (Negotiation tree): [2] A *negotiation tree* for a resource $r \in \mathcal{R}_O$ is a multi-edge tree defined as $\langle \mathcal{N}, r, \mathcal{E}, \phi \rangle$ where $\mathcal{N}$ is a set of tuples of the form $\langle n, s, p \rangle$, where n is a resource condition[3], $s \in \{DELIV, UNDELIV, OPEN\}$ is the state of the node, and $p \in \{O, R\}$ is the owner of the node. $\mathcal{E}$ is a set of edges. $e \in \mathcal{E}$ may assume one of the following form:

- $e = (n_1, n_2)$ with $n_1, n_2 \in \mathcal{E}$ and $\exists p \in \mathcal{RP}_O \cup \mathcal{RP}_R$. An edge of such form is called *simple edge* (see Figure 4(a)).
- $e = \{(n, n_1), (n, n_2), \ldots, (n, n_k)\}$ with $n, n_1, \ldots, n_k \in \mathcal{N}$. An edge of such form is called *multi-edge* (see Figure 4(b)).

ϕ is a labeling function which associate a state with each node of the tree.

Informally, the state of node defines whether the resource identified by the node is deliverable or not. Namely a node n is DELIV if there exists a proof of the existence of a set of resources which satisfies at least a disclosure policy associated with n. On other hand, a node is UNDELIV when such set of resources is proved as not existing. Finally, a node is OPEN when it is still unknown whether the node is deliverable or not.

Figure 4 shows the graphical notation for negotiation trees used in the following of the paper.

[2]A disclosure policy is an expression of the form $rsc \leftarrow c_1, \ldots, c_3$, where rsc is the resource protected by the policy and each c_i is a requirement which has to be satisfied for the disclosure of rsc. The translation from disclosure policies to access rules is trivial

[3]Informally a resource condition is a expression identifying a resource by means of its name – or its identifier – and, eventually, names and values of some of its attributes. We refer to Definitions 4.1, 4.2 and 4.3 for a formal definition.

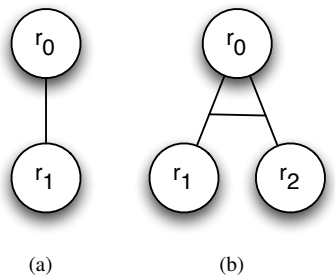

Figure 4. The graphical notation to represent negotiation trees

Definition 3.3 (Valid view): Given a negotiation tree $\mathcal{NT} = \langle \mathcal{N}, r, \mathcal{E}, \phi \rangle$ for a resource $r \in \mathcal{R}_O$, a *valid view* is a subtree of $\mathcal{NT}$ defined as $\langle \mathcal{N}', r, \mathcal{E}', \phi \rangle$ where $\mathcal{N}' \subseteq \mathcal{N}$ and $\forall \langle n, s, p \rangle \in \mathcal{N}'$, $s = DELIV$.

More informally, a valid view is a subtree of the negotiation tree containing the resources exchanged in a successful negotiation.

Example 3.2: Suppose that Bob (B) asks to access a resource r owned by Alice (A) and that such resource is associated with the following disclosure policy:

- $(r, \{(r_1, \varnothing), (r_2, \{(a_2, =, 2)\})\})$, which means that r is released upon the presentation of a resource named r_1 and a resource r_2, the latter with attribute a_2 equal to 2.
- $(r, \{(r_3, \varnothing)\})$, which imposes that the disclosure of the resource r is conditioned by the previous disclosure of r_3.

Suppose that B owns all the resources required by the above described disclosure policies and that, for simplicity, the disclosure policies associated with B's resources are the following:

- $(r_1, \varnothing)$
- $(r_2, \{(r_4, \varnothing), (r_5, \varnothing)\}$
- $(r_2, \{(r_6, \varnothing)\}$
- $(r_3, \{(r_7, \varnothing)\}$

A owns both r_4, protected by $(r_4, \{(r_8, \varnothing)\})$, and r_6, which is deliverable, while she does not own r_5 and r_7.

Before sending to B the disclosure policy for r_4 and r_6, A discovers that the negotiation tree which she is building with B contains a subtree with leaves only labeled as deliverable. Such subtree is highlighted in Figure 5.

Therefore, A communicates to B to switch to the credential exchange phase and sends him r_6. Upon the verification of the correctness of r_6 – which means to verify if r_6 satisfies the disclosure policy which required it – B sends r_2 to A, the resource made deliverable by the disclosure of r_6, and r_1.

Finally, after having verified that r_1 and r_2 satisfy the disclosure policy for r, A is able to deliver B the originally requested resource r.

IV. Extension of the access control rule definition language

In order to introduce trust negotiations in the framework presented in [9], we need to first extend the language to specify

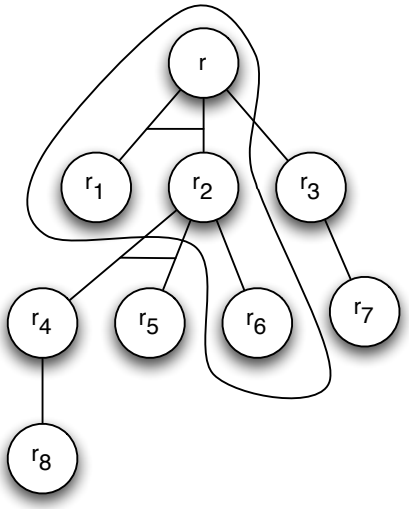

Figure 5. An example of negotiation tree and valid view

access rules. More precisely, we aim at extending the access rule language in order to allow the inclusion of resources – or credentials – as conditions in access rules.

First of all, we need to define the building blocks of a resource/credential condition. We assume the existence of a set $\mathcal{RN}$ of resource names, a set $\mathcal{AN}$ of attribute names and – for every attribute name $a \in \mathcal{AN}$ – a corresponding set $\mathcal{V}_a$ of attribute values.

The formal definition of resource follows.

Definition 4.1 (Resource): A *resource* is a tuple $(R, AttrList)$ where $R \in \mathcal{RN}$ and $AttrList$ is a set of tuples (a, v), where $a \in \mathcal{AN}$ and $v \in \mathcal{V}_a$.

In what follows, given a tuple t, we denote with $comp(t)$ the value of the component $comp$ of tuple t. Therefore, $R(t)$ corresponds to the resource name of the tuple t and $AttrList(t)$ denotes the attribute list of t. We next give precise definitions of attribute and resource conditions.

Definition 4.2 (Attribute condition): An *attribute condition* $\mathcal{AC}$ is a tuple $(a, pred, v)$ where $a \in \mathcal{AN}$, $v \in \mathcal{V}_a$ and $pred$ is one of the binary predicates in $\{<, \leq, =, \geq, >\}$.

Definition 4.3 (Resource condition): A resource condition `RC` is a tuple (rn, AS) where $rn \in RN$ is a resource name and AS is a set, eventually empty, of attribute conditions.

We can now define when a resource satisfies a resource condition.

Definition 4.4 (Resource condition satisfaction): A resource condition rc is satisfied by a resource r if and only if, for each access condition $(a, pred, v) \in AS(rc)$, there exists $(a', v') \in AttrList(r)$ where $a' = a$ and $pred(v', v)$[4]is true.

After having defined resource conditions, we are now able to extend the definition of access rules given in [9] (cfr. Section III) to support both resource and access conditions.

[4] $pred(v', v)$ denotes the predicate $pred$ applied to values v' and v.

Definition 4.5 (Access rule): An access rule AR is a tuple (rid, AC) where rid is the identifier of resource rsc, whereas AC is a set of access and resource conditions. Such set AC expresses the requirements a node must satisfy in order to be allowed to access resource rsc.

Resources associated with an access rule ar with $AC(ar) = \varnothing$ are always accessible. Such resources are fundamental for the successful execution of trust negotiations.

Definition 4.6 (Deliverable resource): We define a resource rsc identified by an identifier rid as *deliverable resource* – DELIV for short – if and only if, for each access rule ar with $rid(ar) = rid$, $AC(ar) = \varnothing$.

With the introduction of resource conditions, we need to modify the protocol described in Figure 3. More precisely, if the access rule AR exchanged in Step 2 contains a resource condition rc, the current access control procedure must temporarily pause in order to perform the disclosure of the resource $rn(rc)$. This is achieved by extending the protocol in such a way that it takes advantages of the features described in Section III-B. The procedure is described in Figure 6.

3.0 R chooses from AR an access rule ar.
3.1 For each resource condition $rc \in ar$, R starts a trust negotiation procedure according to the protocol described in Figure 7.
3.2 If one of the trust negotiations does not terminate successfully then R returns to Step 3.0.
3.3 R sends CS the nonce value N associated with ar and the corresponding condition set $AC(ar)$.

Figure 6. Modification to the access control protocol in Figure 2 to support trust negotiations

Let O be the resource owner, R be the resource requester and rc be the resource condition requested by R.

1) O initiates a negotiation tree rooted with $rn(rc)$.
2) O sends R all the access rules ar, where $rn(ar) = rn(rc)$ and add all the access rules to the negotiation tree as leaves of the root. More precisely, each rule is added as an AND-node
3) If there exists a subtree of the negotiation tree consisting only of AND-nodes[a] and with leaves only labeled as $DELIV$ then go to Step 6 else, for each access rule ar received, R sends O the access rule corresponding to the resources in ar. Moreover, R adds the rules to a local copy of the negotiation tree.
4) If there exists a subtree of the negotiation tree consisting only of AND-nodes and with leaves only labeled as $DELIV$, then go to Step 6 else, for each access rule ar received, O sends to R the access rule corresponding to the resources in ar and it updates the negotiation tree adding the rules sent.
5) Go to Step 3
6) The party which found a subtree of the negotiation tree with leaves labeled as $DELIV$ communicates the counterpart to switch to the credential exchange phase using the identified subtree as valid view.
7) The parties iteratively send to the counterpart the resources, one level at a time, beginning from the max level, and according to the owner of the resources.

[a]A node n is defined AND-node if it is part of a multi-edge (see Section III-B).

Figure 7. Trust negotiation protocol

Example 4.1: Suppose that Bob (B) asks to Alice (A) resource rsc, with identifier rid. A defined for rsc the following access rule:

$$(rid, \{(rsc', \{(att_1, =, 5), (att_2, <, 3)\}), (rsc'', \varnothing)\})$$

stating that, for the disclosure of rsc it is required that the requester provides a copy of the resource rsc', having attribute $attr_1$ equal to 5 and attribute $attr_2$ less then 3, and a copy of the resource rsc''.

Also suppose that B owns resource rsc', with identifier rid' and that the corresponding access rule is:

$$(rid', \{(B, friendOf, 4, 2)\})$$

stating that, for the disclosure of rsc' B requires the existence of a path between him and the requester, labeled with the relationship *friendOf* with maximum depth 4 and having trust value greater then or equal to 2. Moreover, the access rule defined by B for resource rsc'' is:

$$(rid'', \{(rsc''', \varnothing)\})$$

For simplicity, suppose that resource rsc''' is deliverable.

Therefore, if A is able to proof the path required by the above access rule, then B will release rsc'. After having verified the validity of resource rsc', finally, A will release the resource rsc to B.

The negotiation tree built during the trust negotiation is presented in Figure 8.

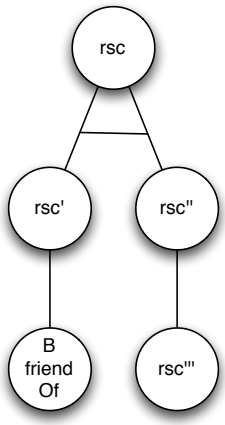

Figure 8. Negotiation tree for Example 4.1

V. Dynamic relationships and trust level adjustment

In a realistic setting, the outcome of a negotiation between two nodes in an OSN may influence the trust level occurring between them. This may occur in a rather natural way: namely, in the case the negotiation is successful, the trust level increases, while, in the opposite case, the trust level is bound to decrease. Our aim in this section is to show how to extend the model described in Section III-A in such a way that negotiation outcomes influence trust levels of the participants in such negotiations. This is achieved by adding some private relationships to the OSN graph, whose trust level is adjusted on the basis of the negotiations performed so far. Such relationships can then be exploited to speed up subsequent access requests.

A. Trust computation

Since the trust relationship between two nodes is asymmetric, the trust level of the first node towards the second and the trust level of the second node towards the first are adjusted in an independent way, by the corresponding nodes. The value of the updated trust level depends on – apart from the positive outcome of a negotiation, of course – the relevance of the resources involved in the negotiation. How a node measures the relevance of its own resources is a far from trivial matter that would deserve a more in-depth investigation than the one presented here. In the following, we limit the presentation on what is strictly needed in the context of the present work.

Definition 5.1 (Relevance of a resource): Given a resource r of a node n protected by an access rule ar, the *relevance* $rel_{ar}(r)$ of resource r given access rule ar is a numeric value computed by aggregating the relevances of the access conditions contained in the access rule ar. The aggregation function used for this purpose can be chosen from $\{sum, max, min\}$.

With respect to a resource protected by a set of access rules, our approach is quite similar: suppose a resource r is protected by a set AR of access rules and that with each access rule $ar_i \in AR$ there is associated a corresponding relevance value $rel_{ar_i}(r)$. Thus, the relevance $rel_{AR}(r)$ of resource r with respect to the access rules set AR is given by aggregating the relevances $rel_{ar_i}(r)$ using as aggregating function a function in $\{sum, avg, max\}$, analogously to what has been done for computing resource's relevance with respect to a single access rule.

In the case that a resource r is not protected by any access rule, the relevance of r is decided by the resource owner by simply assigning it a (possibly normalized) numeric value.

Taking advantage of the concept of relevance of a resource, it is possible to modify the relationships of a user, and the trust levels associated to such relationships, with respect to the resources granted and accessed. Note that such relationships are local to the user, which means they are unknown to the CS.

If a negotiation for a resource r between two users A and B successfully ends, then each user update her/his trust level relative to the other user. Such update depends both on the relevance of the negotiated resource and the previously existing trust level.

Namely, we propose to update the trust level according to the following equation:

$$\phi(e) = \phi'(e) + Relevance(r) \cdot (1 - \phi'(e))$$

where ϕ is the function described in Definition 3.1, e is a the edge representing the relationship whose trust level has to be updated. We denote the value of the function previous to the update with ϕ'. Similarly, if the negotiation fails to successfully end then the trust level is updated to reflect such failure lowering the trust level associated with the relationship. Algorithm 1 presents the procedure to update the trust level of the relationships of the OSN interacting users. Note that, according to our OSN model (cfr. Definition 3.1) the trust level of a relationship between two users may depend on the type of the relation (for instance, I can trust a user more as my friend than as my colleague). Therefore, once two nodes end a negotiation for the first time, two *dynamic edges* are created between the two, of type *disclosedTo* and *receivedFrom*, respectively to which the dynamic computed trust level is assigned.

The variations of the trust levels (a variation for each user) depend – apart from the negotiation outcome – on the current trust levels of the negotiating users, in such a way that the higher the trust levels, the lesser the variations. In other words, if the users trust each other with an high degree, it will take a negotiation involving highly relevant resources to significantly modify the corresponding trust levels.

Data: $(V_{SN}, E_{SN}, RT_{SN}, \phi_{SN})$, the social network graph; $rt_O, rt_R \in RT_{SN}$, the relationship type for dynamic edges, respectively for the resource owner and for the requester
Input: R requester id, O owner id, $out \in \{-1, 1\}$ result of the negotiation, r the resource requested by R

begin
  if *the edge* $e = (O, R, rt)$ *does not exist in* E_{SN} **then**
    Create $e_O = (O, R, rt)$;
    $E_{SN} = E_{SN} \cup \{e_O\}$;
  Update ϕ_{SN} such that $\phi_{SN}(e_O) = \phi'_{SN}(e_O) + (out \cdot \texttt{Relevance}(r) \cdot (1 - \phi'_{SN}(e_O)))$;
  if *the edge* $e_R = (R, O, rt)$ *does not exist in* E_{SN} **then**
    Create $e_R = (R, O, rt)$;
    $E_{SN} = E_{SN} \cup \{e_R\}$;
  Update ϕ_{SN} such that $\phi_{SN}(e_R) = \phi'_{SN}(e_R) + (out \cdot \texttt{Relevance}(r) \cdot (1 - \phi'_{SN}(e_R)))$;

Algorithm 1: Updating of the social network graph's edges and the associated trust levels

We show our approach through the following example.

Example 5.1: Consider the scenario described in Example 4.1 and let us suppose Alice (A) and Bob (B) never

interacted before and that the access control procedure ends successfully. Let us also suppose that the resource r has relevance 0.5 for A and 0.3 for B.

After the successful ending of the access control procedure two new edges are added to the OSN graph. The first one, from A to B, labeled with the relationship type *disclosedTo* and the second one, from B to A, labeled with the relationship type *receivedFrom*. The trust level associated with $e = (A, B, \textit{disclosedTo})$ is computed as follows:

$$\phi(e) = \phi'(e) + (Rel(r) \cdot (1 - \phi'(e)) = 0 + (0.5 \cdot 1) = 0.5$$

On other hand, the trust level associated with $e' = (B, A, \textit{receivedFrom})$ is computed as follows:

$$\phi(e') = \phi'(e') + (Rel(r) \cdot (1 - \phi'(e')) = 0 + (0.3 \cdot 1) = 0.3$$

Consider, instead, a subsequent scenario in which B requests a resource r' to A. Let us suppose that such resource has relevance 0.3 for both users and that the access control protocol fails. In such a case, the trust level associated with the edges e and e' previously introduced is modified as follows:

$$\phi(e) = \phi'(e) - (Rel(r') \cdot (1 - \phi'(e)) = 0.5 - (0.3 \cdot 0.5) = 0.35$$

$$\phi(e') = \phi'(e') + (Rel(r') \cdot (1 - \phi'(e')) = 0.3 - (0.3 \cdot 0.7) = 0.09$$

B. Practical usage of dynamic relationships

The considered access control model does not limit the number of access rules that may be associated with a given resource, and this may result in an increase of the time required to process an access request. In this section, we show how it is possible to take advantage of the relationships introduced as a result of a negotiation to speed up subsequent access requests. This is achieved by creating alternative access rules, which exploit the relationship *disclosedTo* as a shortcut in the access control procedure.

Let us explain how this works by means of an example.

Example 5.2: Suppose that Bob (B) wants to access the resource rsc which belongs to Alice (A). Also suppose that the access rule defined by Alice is:

$$(rid, \{(A, \textit{friendOf}, 3, 0.6), (rid', \{(a = 5)\})\})$$

where rid is the resource identifier of rsc, and rid' is the identifier of resource rsc'. Note that the resource condition $(rid', \{(a = 5)\})$ indicates that, to be satisfied, the resource rsc must have an attribute name a whose value is 5.

Such access rule states that, if there exists a path between Alice and Bob, labeled by the *friendOf* relationship, with maximum depth 3 and with a trust level greater than or equal to 0.6 and if Bob releases to Alice resource rsc', then he can access rsc.

Let us suppose that Bob accesses resource rsc, thus satisfying the access rule. Because of this, a relationship of type *disclosedTo* between Alice and Bob with trust level 0.7 is created. Also suppose Alice defined the following access rule for rsc:

$$(rid, \{(A, \textit{disclosedTo}, 1, 0.5)\})$$

After some time, Bob requires again to access resource rsc. After the request from Bob, Alice verifies the existence of the relationship of type *disclosedTo* with Bob, having a trust level greater than 0.5. Thus, Alice grants access to the resource rsc to Bob without contacting CS and without negotiating for rsc'.

Clearly an important issue when dealing with the introduced dynamic relationships concerns their lifespan. To avoid misuse of the access rules exploiting such relationships, we associate with each dynamic relationship a time limit after which the relationship is removed from the OSN's graph. Such time limit is user-defined, but it directly depends on the trust level of the relationship, in such a way that the *expiration* of the relationship is obtained as

$$exp_e = tl \cdot \phi_{SN}(e)$$

where e is an edge with $rt(e)$ equal to *disclosedTo* or *receivedFrom*, ϕ_{SN} is the function which associates the trust level with each edge, and tl is the time limit defined by the user $v(e)$ (see Section III-A). Note that such time limit is renewed after each interaction between the users which modify the trust level associated with the edge.

Example 5.3: Consider, again, the scenario presented in Example 5.2. Supposing the time limit tl defined by Alice is 10 days, the relationship identified by the edge $e = (A, B, disclosedTo)$ expires after $10 \cdot \phi_{SN}(e) = 7$ days.

Another way to use dynamic relationships is to improve the probability that a user can access a required resource. Consider a user B who requires to access a resource rsc. If B has requested the same resource to another user A, in a near past, then she/he would try to request the same resource from the same user A to improve the chances to get it. Hence, the relationship *receivedFrom* acts as a reminder of past interactions with the users of the OSN. Moreover, considering that a resource can, and probably will, be owned by more than a single user in the OSN, requesting such resource from a trusted user would further slightly improve the chances of a successful interaction.

A natural consideration which came out from the presented usage of the relationship types *deliveredTo* and *receivedFrom* is that relationships of such types are, somehow, too general. Indeed, a relationship which refers directly to the negotiated resource would be more significative. However, choosing to keep a fine-grained history of all the transactions between users of the OSN would result in an explosion of private relationships; consider, for example, the number of distinct resource disclosures which took place between two Facebook's users. To address such issue, we propose that the users specify in their preferences a set of relationship types which keep track of the access to specific *key resources*, such as rare or high value ones, or those more frequently accessed. In such a way, it is possible to customize the number of relationships that are inserted in the OSN graph as an outcome of trust negotiations.

VI. Complexity analysis

We now discuss the complexity of the proposed access control protocol.

As explained in [9], Step 4 (see Figure 3) represents the most expensive task of the protocol. More precisely, during such step, given an access rule ar, CS discovers the shortest certificate paths referring to the set of access conditions $AC(ar)$ received by the requester node. This is achieved by exploring the OSN's graph. Such operation requires either $O(V_{SN}+E_{SN})$ or $\Theta(V_{SN}+E_{SN})$ time complexity, depending on the used search technique, for each $ac \in AC(ar)$. However, it is possible to reduce the size of the graph which has to be explored taking advantage of the constraints on the relationship type and depth specified in the access conditions. Hence, in the general case, the time complexity required to evaluate an access condition ac is $O(\sum_{rt \in RT(ac)}(V_{SN_{rt}} + E_{SN_{rt}}))$, where $RT(ac)$ is the set of relationship types specified in the access condition ac and $V_{SN_{rt}}$, $E_{SN_{rt}}$ are, respectively, the sets of nodes and edges of the OSN's graph with relationship type rt. Finally, since an access rule consists of one or more access conditions, evaluating an access rule ar requires $O(\sum_{ac \in AC(ar)} \sum_{rt \in RT(ac)}(V_{SN_{rt}} + E_{SN_{rt}}))$. We refer to [9] for a more detailed discussion about the results here reported.

To analyze the time complexity of the Trust-$\mathcal{X}$ framework we need to analyze the interactions between the negotiating users in each phase composing the negotiation.

During the introductory phase a fixed number of messages are exchanged, depending on which features of the Trust-$\mathcal{X}$ framework are used, therefore the time complexity is constant.

The subsequent phase, the policy evaluation phase, is the most time consuming one, because it is the one in which the policies which compose the negotiation tree are exchanged. The number of messages exchanged is therefore linear with respect to the height of the tree while the size of the message exchanged in each turn i is linear to the number of nodes of the tree at deep i. Hence, given an OSN $(V_{SN}, E_{SN}, RT_{SN}, \phi_{SN})$, the execution of the policy evaluation phase between two nodes $A, B \in V_{SN}$ requires $O(|R_A| + |R_B|)$ messages, where $|R_u|$ is the number of resources owned by node $u \in V_{SN}$.

The credential exchange phase is where the required resources are actually exchanged. As mentioned in Section III-B, the resources which have to be exchanged in order to successfully end the negotiation are the nodes of the selected valid view. Such valid view is a subtree of the negotiation tree constructed during the policy evaluation phase. Hence, the height of the valid view is at most the height of the negotiation tree. Thus, we can state that the credential exchange phase requires at most $O(|R_A| + |R_B|$ messages.

Globally, a trust negotiation executed between two nodes A and B of an OSN requires at most $2 \cdot O(|R_A| + |R_B|)$.

Considering the composition of the two frameworks, it is possible to state that the overall time complexity to evaluate an access rule ar is given by the maximum between the time complexity to evaluate the same access rule purged of resource conditions, and the maximum time complexity to negotiate, in parallel, the previously mentioned resource conditions. Such computational parallelism is ascribable to the fact that the negotiation for each resource condition is independent from the others. Similarly, the proof computed by CS is independent from the negotiations, therefore they can be simultaneously computed.

VII. Experimental Results

The evaluation of the proposed approach has been preliminary done performing several experiments using a Trust-$\mathcal{X}$ prototype. The integration with the access control mechanism described in [9] is currently under development. The prototype has been developed using Java 6.

To run our experiments we used a network of two computers with the following configurations:

- Linux, kernel 2.6.30, CPU 2.20GHz
- Macbook, OS 10.6, CPU 2.53GHz

We took advantage of a MySQL database version 5.1 to store both resources and access rules.

First of all, we evaluated the time required by two nodes to perform a trust negotiation for a given resource with respect to the number of credentials which have to be exchanged.

Figure 9 shows how Trust-$\mathcal{X}$ performances are linear to the number of credentials. More precisely, the performance depends on the structure of the access rules exchanged. For the simplest negotiations, which involve the exchange of two resources, represented by a negotiation of the form $(rid, \{(rid', \varnothing)\})$ with $(rid', \varnothing)$, it is required in average 226 milliseconds, with a lower bound of 188 milliseconds. On the other hand, to negotiate and exchange 50 resources Trust-$\mathcal{X}$ requires 3859 milliseconds.

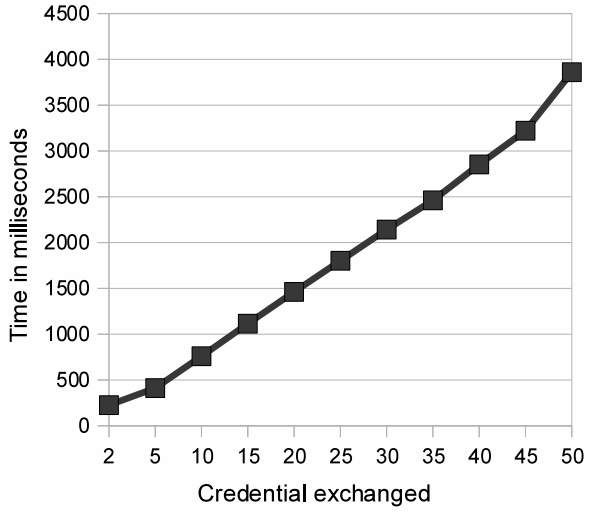

Figure 9. Time required to successfully end a trust negotiation with respect to the number of credentials involved

We also performed a series of tests in order to evaluate the scalability of our prototype with respect to the number of

parallel negotiations. To be able to compare the results, we performed a crescent number of parallel negotiations. Each trust negotiation involves the same resources and, therefore, the same access rules. The results are presented in Figure 10.

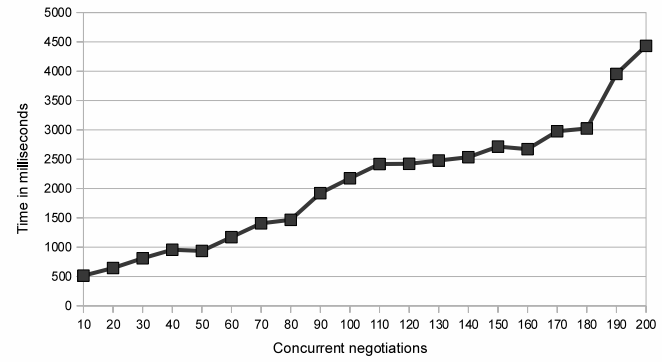

Figure 10. Scalability of the prototype with respect to the number of simultaneous trust negotiations

VIII. Conclusions

We have presented an extension of the framework introduced in [9], aimed at the integration of trust negotiations with an access control mechanism for resources in OSNs. This has been achieved by properly extending the language for expressing access control policies and integrating the relevant features of the Trust-$\mathcal{X}$ framework. Further, a feedback mechanism that takes into account the outcome of a trust negotiation between two nodes to dynamically set their trust level has been presented. Finally, several experiments have been carried out in order to show the feasibility of our approach.

We are currently working at the integration of the Trust-$\mathcal{X}$ prototype within the access control mechanism presented in [9]. Further extensions we plan to work on are related to the automatic setting of the lifetime of dynamic relationships, the automatic identification of key resources and to the development of methods to compute the relevance of a resource.

Acknowledgments

The work reported in this paper is partially funded by the Italian MIUR under the ANONIMO project (PRIN-2007F9437X).

References

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

[2] E. Bertino, E. Ferrari, and A. C. Squicciarini, "Trust-$\mathcal{X}$: A Peer-to-Peer Framework for Trust Establishment," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 7, pp. 827–842, 2004.

[3] D. Brickley and L. Miller, "FOAF vocabulary specification 0.91. Namespace Document," Online: http://xmlns.com/foaf/0.1., Nov 2007.

[4] L. Ding, L. Zhou, T. W. Finin, and A. Joshi, "How the semantic web is being used: An analysis of foaf documents," in *HICSS*. IEEE Computer Society, 2005.

[5] B. Carminati and E. Ferrari, "Privacy-aware Access Control in Social Networks: Issues and Solutions," in *Privacy and Anonymity in Information Management Systems*, J. Nin and J. Herranz, Eds. Springer, to appear.

[6] E. Ferrari, A. C. Squicciarini, and E. Bertino, "X-TNL: An XML Language for Trust Negotiations," *4th IEEE Workshop on Policies for Distributed Systems and Networks, Como, Italy*, June 2003.

[7] W. Nejdl, D. Olmedilla, and M. Winslett, "PeerTrust: Automated Trust Negotiation for Peers on the semantic web," in *Workshop on Secure Data Management in a Connected World (SDM'04)*, Toronto, Canada, Aug. 2004.

[8] T. Yu and M. Winslett, "A unified scheme for resource protection in automated trust negotiation." in *IEEE Symposium on Security and Privacy*, 2003, pp. 110–122.

[9] B. Carminati, E. Ferrari, and A. Perego, "Enforcing access control in web-based social networks," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, 2009.

[10] F. Bonchi and E. Ferrari, Eds., *Privacy-aware Knowledge Discovery: Novel Applications and New Techniques*. Chapman and Hall/CRC Press, 2010.

[11] J. Nin, B. Carminati, E. Ferrari, and V. Torra, "Computing Reputation for Collaborative Private Networks," in *COMPSAC '09: Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference*, 2009, pp. 246–253.

[12] T. Y. K.E. Seamons, M. Winslett, "Protecting privacy during on line trust negotiation," in *2nd Workshop on Privacy Enhancing Technologies, San Francisco, CA*, April 2002.

[13] N. Li and J. C. Mitchell, "Datalog with constraints: A foundation for trust management languages," in *Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages*, Jan. 2003.

[14] K. E. Seamons, M. Winslett, and T. Yu, "Limiting the disclosure of access control policies during automated trust negotiation." in *NDSS*, 2001.

[15] W. H. Winsborough and N. Li, "Safety in automated trust negotiation." in *IEEE Symposium on Security and Privacy*, 2004, pp. 147–160.

[16] E. Bertino, E. Ferrari, and A. C. Squicciarini, "Privacy-Preserving Trust Negotiation." *Proceedings of 4th Privacy Enhancing Technologies Workshop, Toronto, CA*, May 2004.

[17] A. C. Squicciarini, A. Trombetta, and E. Bertino, "Supporting Robust and Secure Interactions in Open Domains through Recovery of Trust Negotiations," in *ICDCS*. IEEE Computer Society, 2007, p. 57.

[18] A. C. Squicciarini, A. Trombetta, E. Bertino, and S. Braghin, "Identity-based long running negotiations," in *Digital Identity Management*, E. Bertino and K. Takahashi, Eds. ACM, 2008, pp. 97–106.

[19] A. C. Squicciarini, F. Paci, E. Bertino, A. Trombetta, and S. Braghin, "Group-based negotiations in p2p systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, 2010.

[20] S. Braghin, I. Nai Fovino, and A. Trombetta, "Advanced trust negotiations in critical infrastructures," *International Journal of Critical Infrastructures*, vol. 6, no. 3, pp. 225–245, 2010.