Rule-Based Access Control for Social Networks

Barbara Carminati, Elena Ferrari, and Andrea Perego

DICOM, Università degli Studi dell'Insubria, Varese, Italy
{barbara.carminati, elena.ferrari, andrea.perego}@uninsubria.it

Abstract. Web-based social networks (WBSNs) are online communities where participants can establish relationships and share resources across the Web with other users. In recent years, several WBSNs have been adopting Semantic Web technologies, such as FOAF, for representing users' data and relationships, making it possible to enforce information interchange across multiple WBSNs. Despite its advantages in terms of information diffusion, this raised the need of giving content owners more control on the distribution of their resources, which may be accessed by a community far wider than they expected.

In this paper, we present an access control model for WBSNs, where policies are expressed as constraints on the type, depth, and trust level of existing relationships. Relevant features of our model are the use of certificates for granting relationships' authenticity, and the *client-side* enforcement of access control according to a rule-based approach, where a subject requesting to access an object must demonstrate that it has the rights of doing that.

1 Introduction

Web-based social networks (WBSNs) [1] are online communities which allow Web users to publish resources (e.g., blogs) and to establish relationships with other users, possibly of different type ("friend", "colleague", etc.), for purposes which may concern, e.g., entertainment, religion, dating, or business. One of the recent trends in WBSNs is the adoption of Semantic Web technologies, in particular FOAF [2,3], to represent users' personal data and relationships [4]. Thanks to this and to the adoption of decentralized authentication systems such as OpenID [5], it has been made simpler to access and disseminate information across multiple WBSNs. If this has been quite a relevant improvement with respect to the previous situation, it is now necessary that resource owners have more control over information sharing. In fact, differently from 'traditional' social networks, where usually each user knows the others, WBSNs are quite larger, and each node (i.e., user) has direct relationships with only a subgraph of the network. As a consequence, it may be not appropriate to make available any information to all the users of one or more WBSNs. So far, this issue has been addressed by some of the available Social Network Management Systems (SNMSs) by allowing users to state whether a specific information (e.g., personal data and resources) should be public or accessible only by the users with whom the owner of such information has a direct relationship. Such simple access control strategies have the advantage of being straightforward, but, on one hand, they may grant access to non-authorized users, and, on the other hand, they are not flexible enough in denoting authorized users. In fact, they do not take into account the ‘type’ of relationship existing between users and, consequently, it is not possible to state that only, say, my “friends” can access a given information. Moreover, they do not allow to grant access to users who have an indirect relationship with the resource owner (e.g., the “friends of my friends”).

We think that more sophisticated access control mechanisms can be enforced in the current WBSNs, dealing with such issues. Besides relationships, some other information can be used for this purpose. In fact, the graph of a WBSN allows us to exploit the notion of *depth* of a relationship, which corresponds to the length of the shortest path between two nodes. The depth of a relationship may be a useful parameter, which allows us to control the propagation of access rules in the network. Moreover, in some WBSNs, users can specify how much they trust other users, by assigning them a *trust level.* Such information is currently exploited for purposes which encompass the primary objectives of a WBSN, e.g., as a basis for recommender systems, but it can be used as well to denote the subjects authorized to access a resource in terms of their trustworthiness. Note that the notion of trust applies also to users with an *indirect* relationship—i.e., a relationship with depth greater than 1—, and thus we can combine the usage of depth and trust in access policies.

In this paper, we propose a rule-based access control model for WBSNs, which allows the specification of access rules for online resources where authorized subjects are denoted in terms of the relationship type, depth, and trust level existing between users in the network. To the best of our knowledge, this is the first proposal of an access control model for social networks. The different tasks to be carried out to enforce access control are shared among three distinguished actors—namely, the owner of the requested resource, the subject which requested it, and the SNMS. More precisely, we adopt the approach outlined by Tim Berners-Lee & al. in [6], where access control is enforced client-side, since access to resources is granted if the requestor is able to demonstrate that he/she satisfies given requirements. For this purpose, users’ relationships are represented by a specific OWL vocabulary we designed, REL-X [7], whereas access rules are expressed in Notation 3 Logic (N3) [8], and then evaluated by the Cwm reasoner [9] against the existing relationships in order to generate a proof.

The remainder of this paper is organized as follows. Section 2 discusses access control requirements of social networks. Section 3 summarizes the main features of our approach, whereas Sect. 4 illustrates the proposed access control model. Then, Sect. 5 describes the system architecture of the prototype being implemented. Moreover, a running example of how access control is enforced is provided in Sect. 6. Finally, Sect. 7 concludes the paper and outlines future research directions.

2 Access Control in Web-Based Social Networks

In this section, we discuss which are the basic requirements that an access control mechanism for WBSNs should satisfy. However, before discussing access control issues in social networks, we need to briefly introduce what a social network is and how we represent it.

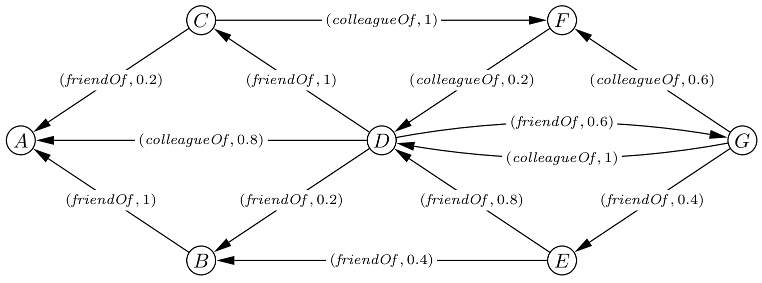

Fig. 1. A simple social network. Labels associated with edges denote, respectively, the type and trust level of the corresponding relationship.

2.1 The Social Network Model

A common way to represent a network is by means of a graph. This representation can be adopted also in the context of social networks, where each node denotes a user in the network, whereas edges represent the existing relationships between users (see Fig. 1 for an example). However, since users of the same community could establish among each others relationships of different nature, the graph representation of a social network must be extended to support the *type* of a relationship. Moreover, we have to take into account that not all relationships are mutual—consider, for instance, the "parent of" relationship. Thus, rather than a simple graph, we have to consider a direct graph, where the direction of the edges denotes, respectively, which user has specified the relationship and the user for whom a relationship has been specified: the former corresponds to the terminal node of the edge, whereas the latter to the initial one. According to this interpretation, we can read the graph in Fig. 1, as follows: A(lice) specified a relationship with both B(ob) and C(arl), denoting them as friends; Bob and Carl specified the same type of relationship with D(avid), and so on.

Relationships can be grouped into two different classes, i.e., *direct* and *indirect* relationships. A direct relationship of type *rt* between user b and user a is represented by a direct edge exiting from the node representing b and entering into the node representing a. As an example, let us consider the social network depicted in Fig. 1, where Bob has a direct relationship of type "friend of" with Alice, F(rank) has a direct relationship of type "colleague of" with David, etc. By contrast, an indirect relationship of type *rt* between user b and user a is represented as a path connecting b to a, consisting of all the edges representing relationships of type *rt*. Examples of indirect relationships are provided in Fig. 1, where E(ve) and Alice are not directly connected, but they are related in that Eve is friend of Bob, and Bob is friend of Alice. Thus, we say that Eve has an indirect relationship with Alice of type "friend of".

Besides their type, relationships are characterized by a *trust level* t (see [1] for an interesting discussion on trust in social networks). Given an in/direct relationship *rel* between user b and user a, the trust level denotes how much user a considers trustworthy user b wrt relationship *rel*. We do not enter here in the details of the formulas used to calculate the trust level, which may vary depending on the social network. However, we need to remark that, regardless of the algorithm used to compute the trust level t

of a relationship of type *rt* between user *b* and user *a*, such calculation needs to take into account the trust levels of all the edges belonging to paths corresponding to all the possible in/direct relationships of type *rt* between *b* and *a*. As will be explained in the next section, this aspect is quite relevant in our architecture.

It is important to remark that our notion of trust level is slightly different with respect to that defined in [1], where a trust level is associated with users, and not with a specific relationship involving them. Moreover, two types of trust are distinguished, namely, *topical* trust (i.e., how much a user consider trustworthy the opinions of another user about a given topic—e.g., movies) and *absolute* trust (i.e., how much a user consider trustworthy the opinions of another user independently from the topic). Our notion of trust is more similar to the latter, with the difference that we allow users to express different trust levels depending on the type of relationship. As will be mentioned in Sect. 7, we plan to extend our model in order to support also topical trust.

A further relationship's property is its *depth*. The depth of a relationship of type *rt* between user *b* and user *a* corresponds to the length of the shortest among the paths representing all possible in/direct relationships of type *rt* between user *b* and user *a*. For instance, from *G*(reg) to Alice, three paths exist of type "friend of": *GEBA* (length 3), *GEDBA*, and *GEDCA* (both of length 4). Consequently, the depth of the relationship existing between Greg and Alice is equal to 3.

Our notion of relationship can then be represented as a tuple, where its components denote the users participating in it, along with the type, depth, and trust level of the relationship itself. Thus, since a relationship is an entity with a set of attributes (i.e., type, depth, and trust level), we cannot represent it in RDF as a property. Consequently, we cannot adopt either the FOAF vocabulary or its extensions (such as the RELATIONSHIP vocabulary [10] and the Trust Ontology [11]), where relationships, along with their types and trust levels are represented as distinct properties of users. For this reason, we designed an OWL vocabulary, namely, REL-X [7], where a relationship is modelled as an instance of the `relx:Relationship` OWL class, characterized by a set of properties which allow us to associate with a relationship the users participating in it, its type, depth, and trust level.

2.2 Access Control Requirements in WBSNs

Devising an access control model for WBSNs requires to face up with a dynamic environment, where a user in the social network wishes to share his/her data with other users of the same network, on the basis of in/direct relationships existing between them, that may dynamically change. Therefore, a first requirement that we must address is supporting access control on the basis of users' relationships. Let us consider again the social network depicted in Fig. 1, and assume that Alice wishes to share her resources with her direct friends, and some of her indirect friends. More precisely, she wants to grant access to Bob and Carl, since they are direct friends of hers. She wants to allow also Eve to access resources, even if Alice does not know her, because she is a direct friend of Bob. Nonetheless, Alice may be not sure whether Greg is trustworthy or not, since she cannot know how Eve chooses his friends. Thus, Alice could decide to prevent him from accessing her resources. This example points out that the length of the path connecting two nodes is a relevant information for access control, since the longer is a path, the lower is the level of trust existing between its ends. For instance, if Alice states that a given resource r_A can be accessed by "a friend" with depth equal at most to 2 ($1 \leq d \leq 2$), users authorized to access r_A are Bob, Carl ($d = 1$), David, and Eve ($d = 2$). The depth of a relationship is thus a useful parameter, which gives owners more control on the distribution of their resources.

A further relevant aspect, that should be considered in devising an access control model for WBSNs, is the trust level of a relationship. Obviously, the notion of trust gains importance in the context of access control, since it is a further discriminant that can be exploited by users to decide whether a resource can be released or not. Thus, an access control policy for WBSNs should make a user able to state which type of relationship should exist between him/her and the requesting user, along with the maximum depth and the minimum trust level of the relationship.

From the side of access control enforcement, traditionally, this task is performed by an access control mechanism (referred to as *reference monitor*), that is, a trusted software module that intercepts each access request submitted to the system and, on the basis of the specified policies, determines whether the access should be partially or totally authorized, or it should be denied. To adopt this strategy in WBSNs, there is the need of a trusted place where the reference monitor should be safely hosted. A naïve solution is thus to assume that the SNMS hosts the reference monitor. Nonetheless, since the number of participants of a WBSN may be quite big, a centralized reference monitor would be a bottleneck. For this reason, we prefer to investigate an alternative strategy, where the owner him/herself is the only administrator of his/her data. Thus, he/she states the access control policies according to him/her preferences, and for each access request he/she locally determines whether the access should be authorized or not. Obviously, this solution ensures the user a total control over his/her data, but it requires software and hardware resources more powerful than those typically available to social network participants. In order to overcome this drawback, we have decided to take over the view outlined by Tim Berners-Lee & al. in [6], for enforcing access control in the Semantic Web. The main novelty of such approach is that, differently from traditional access control, the task of verifying whether a given user is authorized to access a given object is in charge of the requesting user, who must prove to the resource owner that it satisfies the requirements expressed by access control policies.

3 Overall of the Proposed Approach

As pointed out in Sect. 2.2, we here decided to customize in the context of WBSNs the client-side and decentralized access control outlined in [6]. This solution implies that when a user (hereafter, the *requestor*) requests a resource to another user (hereafter, the *resource owner*), the former receives from the latter a set of *access rules* regulating the release of the requested resource. These rules basically state which type of relationship should exist between the resource owner and the requestor, and the maximum depth and minimum trust level allowed. The requestor has to provide the resource owner with a *proof* showing that between them there exists the required relationship, and that this relationship has the required depth and trust level. Proofs are generated by exploiting a reasoner (see Sect. 5 for details). Thus, the resource owner provided with this proof is able to locally verify whether the resource has to be released or not. More precisely, in our system, access rules are expressed using N3, and evaluated by the Cwm reasoner. The adoption of N3 instead of Semantic Web rule languages, such as RuleML [12] or SWRL [13], is determined by the fact that N3 allows a compact representation of RDF statements, which can be easily delivered via HTTP headers.

Since in a WBSN scenario the access rules are defined against relationships, the requestor has to provide the reasoner with assertions on the existing relationships between him/her and the resource owner. This requires solving the following issue: how can a resource owner be ensured that assertions created by the requestor are correct and trustworthy? For instance, how the resource owner can be ensured that the requestor has not maliciously forged a new relationship? Thus, we need a mechanism which prevents a malicious user from submitting to the reasoner assertions on fake relationships.

To cope with these problems, we propose a solution exploiting the notion of *certificates*. According to this solution, whenever Alice establishes a new relationship with Bob, they both create and sign a certificate stating that between Bob and Alice there exists a direct relationship with a certain trust. By means of certificates, it is possible to verify the correctness of assertions on that relationship. More in general, the correctness of an assertion related to an in/direct relationship between users *a* and *b* can be verified by retrieving the chain of certificates confirming the existence of a path between them. By contrast, proving the assertion on the relationship's trust needs a more complex strategy, since the trust level between two nodes is computed taking into account all the possible paths connecting them [1]. Moreover, the requestor should provide the resource owner with all the corresponding chains of certificates in order to let him/her verify the correctness of the assertion wrt the trust level. Besides its inefficiency, this solution is liable to security attacks, in that the resource owner cannot be sure that the requestor has actually provided all the possible chains of certificates, or he/she has intentionally omitted a chain with a low trust level.

To overcome this drawback, we have investigated a slightly different architecture wrt the one outlined by Tim Berners-Lee & al. in [6]. Indeed, we propose a *semi-decentralized* architecture, according to which a given trusted node in the network, referred to as *central node CN*, is in charge of managing certificates, and of computing the trust levels of relationships. Thus, whenever a requestor receives from the resource owner the access rules regulating the release of his/her resources, the former requests to *CN* the certificate chains proving the existence of a given relationship, as well as its trust level. The certificate chain, if any, is generated by *CN* by consulting the certificate repository, and, before being returned to the requestor, the corresponding trust level is signed by *CN*. Then, the requestor exploits the received certificate chain to generate the assertion for the reasoner. If the reasoner generates a proof, the requestor sends it to the resource owner, together with the certificate chain corresponding to the assertion, and the signed trust level. The main benefit of such semi-decentralized solution is that it fits well with the architecture of current WBSNs. Indeed, in such architectures, users' personal data (i.e., the data provided by users during the registration to a WBSN) and relationships are stored by the SNMS in a central repository, accessible via Web, whereas no information is stored by users on client-side. The central node corresponds then to the SNMS, which usually relies on hardware resources more powerful than those available on client-side. Thus, in the semi-decentralized approach, the central node is in charge of the tasks usually carried out by the SNMS, whereas the other nodes are in charge of enforcing access control for the resources they own. As a last remark, it is important to note that the central node is not in charge of storing users' data, but only relationship certificates. Thanks to this, we can ensure users' data confidentiality wrt the central node.

4 The Proposed Access Control Model

In order to enforce access control in WBSN environments, we defined a rule-based model where policies are specified by resource owners, and they denote implicitly the 'profile' of authorized users by means of one or more *access conditions*, i.e. constraints on the type, depth, and trust level of the relationships they have with other users in the network.

In what follows, we denote by V_{SN}, E_{SN}, RT_{SN}, T_{SN} the sets of nodes, edges, relationship types, and trust levels, respectively, of a social network *SN*. The notion of access condition is formally defined as follows.

Definition 1 (Access Condition). *Given a social network SN, an access condition cond against SN is a tuple* $(v, rt, \mathrm{D}\max, t\min)$*, where* $v \in V_{SN} \cup \{*\}$ *is the node with which the requestor must have a relationship,* $rt \in RT_{SN} \cup \{*\}$ *is a relationship type, whereas* $\mathrm{D}\max \in \mathbb{N} \cup \{*\}$ *and* $t\min \in T_{SN} \cup \{*\}$ *are, respectively, the maximum depth and the minimum trust level that the relationship must have. If* $v = *$ *and/or* $rt = *$*, v corresponds to any user in* V_{SN} *and/or rt corresponds to any relationship in* RT_{SN}*, whereas if* $\mathrm{D}\max = *$ *and/or* $t\min = *$*, there is no constraint concerning the depth and/or trust level, respectively.*

Access control requirements of a given object can then be expressed by a set of conditions. More precisely, given an object *obj* owned by v_o, the set of access conditions applying to *obj* are expressed by an *access rule* specified by v_o. Such notion is formally defined as follows.

Definition 2 (Access Rule). *An access rule rul is a tuple* $(oid, cset)$*, where oid is the identifier of object obj, whereas cset is a set of conditions* $\{cond_1, \ldots, cond_n\}$*, expressing the requirements a node must satisfy in order to be allowed to access object obj.*

It is important to note that the conditions in *cset* do not denote a set of alternative requirements, but *all* the requirements to be satisfied. In other words, the semantics of a set of conditions $\{cond_1, \ldots, cond_n\}$ can be expressed as $cond_1 \wedge \cdots \wedge cond_n$. It may be also the case that more than one rule is specified for a given object. For instance, let us suppose that object *obj* is associated with two rules rul, rul'. In such a case, we consider the corresponding two sets of conditions $\{cond_1, \ldots, cond_n\}$ and $\{cond'_1, \ldots, cond'_m\}$ as sets of alternative access control requirements—i.e., $(cond_1 \wedge \cdots \wedge cond_n) \vee (cond'_1 \wedge \cdots \wedge cond'_m)$.

As illustrated in Sect. 3, in general, a requestor is authorized to access an object if he/she provides the resource owner with a proof of the fact that he/she satisfies at least one of the corresponding access rules. To make the task of generating proofs easier, we translate relationships, conditions, and access rules into equivalent logical formulas. In particular, a relationship *rel* of type *rt*, depth $\textsc{d}$, and trust level t, between users b and a, can be expressed by the following assertion:

$$hasSubj(rel,b) \wedge hasObj(rel,a) \wedge hasType(rel,rt) \wedge hasDepth(rel,\textsc{d}) \wedge hasTrust(rel,t)$$

where *hasSubj*, *hasObj*, *hasType*, *hasDepth*, and *hasTrust* are predicates returning "true" if the corresponding relation is satisfied. Similarly, a condition $cond = (v, rt, \textsc{d}\max, t\min)$ can be translated into an equivalent formula p, denoted as follows:

$$hasSubj(?rel,?x) \wedge hasObj(?rel,v) \wedge hasType(?rel,rt) \wedge hasDepth(?rel,?\textsc{d}) \wedge \leq (?\textsc{d},\textsc{d}\max) \wedge hasTrust(?rel,?t) \wedge \geq (?t,t\min)$$

where variable $?x$ represents the requesting user. Finally, according to the semantics of a set of conditions, an access rule $(oid, \{cond_1, \ldots, cond_n\})$ corresponds to a Horn-like clause $(p_1 \wedge \cdots \wedge p_n) \rightarrow canAccess(?x, oid)$, where the predicate *canAccess* states that the user represented by variable $?x$ is authorized to access the object with identifier *oid*. A proof is then obtained by using as arguments the set of existing relationships and access rules. If more than one rule applies to the same object, they are verified one by one, until a valid proof, if any, is obtained.

5 System Architecture

The architecture of the system we are currently implementing consists of two main services: the *central node CN*, corresponding to the SNMS, and a set of *periferal nodes*, corresponding to the nodes in the network. Figure 2 depicts the main components of such services, and an example of interaction between two periferal nodes, where the labelled arrows correspond to the steps of the access control procedure illustrated in Sect. 6. Our system exploits the OpenID framework, and thus users are identified by OpenID addresses. Moreover, we consider as a resource any information accessible via the Web, which is then identified by means of a URI.

The central node is a service run by the SNMS, and it consists of the certificate directory, storing all the certificates generated by the users in the network, and of two main modules: the *certificate manager* and the *certificate retrieval module*. The certificate manager is in charge of receiving the certificates sent by the users in the network, verifying their validity, and performing insert, update, and revoke operations. In our system, users can decide whether the relationships they are establishing (and thus the corresponding certificates) must be public or private. Private relationships can be used only by the users participating in them in order to perform access control. The certificate retrieval module is in charge of replying to requests asking for the chains of certificates existing between two nodes in the network. Such module is also in charge of computing the trust level of a relationship, derived from the information stored into the corresponding chains of certificates.

Periferal nodes are services which must be run by a server machine (either any server in the Internet or the end user's machine itself), and they consist of four main components: the certificate manager, the rule manager, and the access control and access

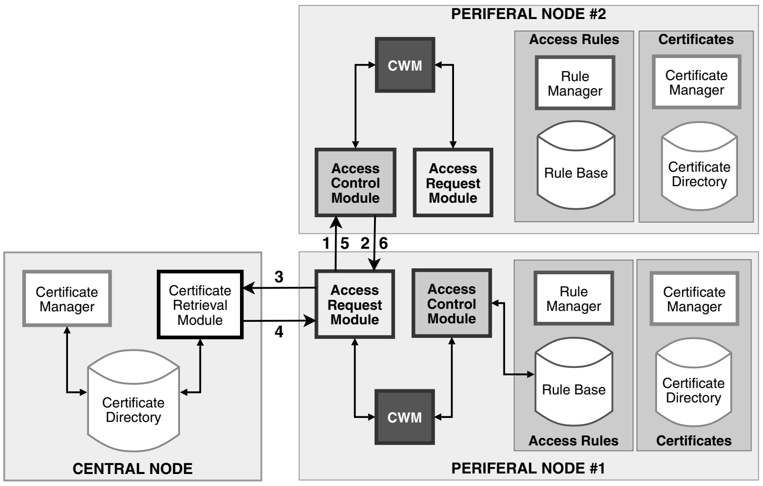

Fig. 2. System architecture

request modules. The certificate manager is in charge of managing and storing the certificates generated by the corresponding user, whereas the access rule manager stores and manages all the access rules specified by the user for the resources he/she owns. The access control module (ACM) replies to the access requests submitted by users in the network. More precisely, the ACM is in charge of a) sending the rules concerning the requested object, b) verifying the corresponding proofs, and c) delivering the requested object. For the first task, the ACM exploits an N3 translator, which transforms the rules stored into the access rule base. The second task is performed by the Cwm reasoner, which verifies the validity of the proof. In addition, the access control module may check the correctness of relationship assertions used as arguments in the proof, by verifying if they match with those obtained from the chains of certificates sent along with the proof by the requesting user. By contrast, the access request module (ARM) is in charge of a) submitting access requests, b) retrieving the chains of certificates from the central node and generating the relationship assertions to be used in a proof, and c) computing the proof itself. As the ACM, for these purposes the ARM exploits the N3 translator and the Cwm reasoner.

It is interesting to note that the ARM is a quite independent component of a periferal node, since it communicates only with the central node and with the ACMs of other periferal nodes. For this reason, we plan to implement periferal nodes in two different applications: the former will be a server application consisting of the ACM, the access rule base, and the certificate directory. By contrast, the latter will be a client application, to be built as a browser extension, implementing the ARM and providing a rule and certificate editor, allowing users to specify access rules and establish relationships without the need of connecting directly to their periferal nodes. Thanks to this, users who do not want to publish resources, or do not have the possibility to run a server application, can use only the browser extension.

6 Running Example

Let us consider the social network in Fig. 1. Suppose that Alice is the owner of an object with identifier `obj1`, and that she wishes to make it available to users who are direct friends of hers, provided that their trust level is not less than 0.5. Moreover, she wishes to grant access also to all her indirect friends, provided that they are also colleagues and that their trust level is not less than 0.5. Such policy can be expressed by the following access rules:

1. $(\mathtt{obj1}, \{(\mathtt{Alice}, \mathtt{friendOf}, 1, 0.5)\})$
2. $(\mathtt{obj1}, \{(\mathtt{Alice}, \mathtt{friendOf}, *, *), (\mathtt{Alice}, \mathtt{colleagueOf}, 1, 0.5)\})$

Let us now suppose that David submits a request to access `obj1`. David must then prove that he satisfies the requirements expressed by at least one of the two access rules applying to `obj1`, which can be transformed into the following Horn-like clauses:

1. $(\mathit{hasSubj}(?r, ?x) \wedge \mathit{hasObj}(?r, \mathtt{Alice}) \wedge \mathit{hasType}(?r, \mathtt{friendOf}) \wedge \mathit{hasDepth}(?r, ?\mathrm{D}) \wedge \leq (?\mathrm{D}, 1) \wedge \mathit{hasTrust}(?r, ?t) \wedge \geq (?t, 0.5)) \rightarrow \mathit{canAccess}(?x, \mathtt{obj1})$
2. $(\mathit{hasSubj}(?r, ?x) \wedge \mathit{hasObj}(?r, \mathtt{Alice}) \wedge \mathit{hasType}(?r, \mathtt{friendOf}) \wedge \mathit{hasSubj}(?r', ?x) \wedge \mathit{hasObj}(?r', \mathtt{Alice}) \wedge \mathit{hasType}(?r', \mathtt{colleagueOf}) \wedge \mathit{hasDepth}(?r', ?\mathrm{D}') \wedge \leq (?\mathrm{D}', 1) \wedge \mathit{hasTrust}(?r', ?t') \wedge \geq (?t', 0.5)) \rightarrow \mathit{canAccess}(?x, \mathtt{obj1})$

According to such rules, the relationships to be proved and evaluated are those existing between David and Alice (see Fig. 1), which are expressed by the following assertions:

A. $\mathit{hasSubj}(\mathtt{r1}, \mathtt{David}) \wedge \mathit{hasObj}(\mathtt{r1}, \mathtt{Alice}) \wedge \mathit{hasType}(\mathtt{r1}, \mathtt{friendOf}) \wedge \mathit{hasDepth}(\mathtt{r1}, 2) \wedge \mathit{hasTrust}(\mathtt{r1}, 0.2)$
B. $\mathit{hasSubj}(\mathtt{r2}, \mathtt{David}) \wedge \mathit{hasObj}(\mathtt{r2}, \mathtt{Alice}) \wedge \mathit{hasType}(\mathtt{r2}, \mathtt{colleagueOf}) \wedge \mathit{hasDepth}(\mathtt{r2}, 1) \wedge \mathit{hasTrust}(\mathtt{r2}, 0.8)$

Note that the trust level in relationship A corresponds to the one actually associated with edge *DA*, whereas the trust level in relationship B is an average of the trust levels associated with the edges in paths *DBA* and *DCA*, computed by using a given algorithm. Relationships A and B alone satisfy neither rule 1 nor rule 2, whereas relationships A and B together satisfy rule 2, since David is both a friend of Alice and a colleague of hers, with trust level equal to 0.8 (and thus ≥ 0.5).

The system components in charge of carrying out the tasks concerning access control enforcement are shown in Fig. 2. Let us suppose that periferal node #1 corresponds to David, whereas periferal node #2 to Alice. David's ARM sends the access request for `obj1` (step 1), and Alice's ACM replies with the set of access rules to be satisfied (step 2). David's ARM then asks to the central node the chain of certificates concerning the existing relationship between him and Alice, and the corresponding trust level (steps 3-4); after having processed them, David's ARM obtains a set of assertions which are delivered to Cwm, along with the access rules sent by Alice's ACM. If a proof is returned, David's ARM sends it to Alice's ACM along with the corresponding certificate chain (step 5). After having verified the validity of the proof, Alice's ACM returns the requested object (step 6).

7 Conclusions and Future Work

In this paper, we presented an access control model for WBSNs, where policies are specified in terms of constraints on the type, depth, and trust level of relationships existing between users. Relevant features of our model are the use of certificates for granting relationships' authenticity, and the client-side enforcement of access control according to a rule-based approach, where a subject requesting to access an object must demonstrate that it has the rights of doing that by means of a proof. We have also proposed a decentralized system architecture on support of access control enforcement, based on the interaction of two agents: the central node of the network, which stores and manages certificates specified by users, and a set of periferal nodes, in charge of storing access rules and performing access control. Future work includes two main research directions, namely, the support for topical trust and the usage of access rules also for certificate protection.

References

1. Golbeck, J.A.: Computing and Applying Trust in Web-based Social Networks. PhD thesis, Graduate School of the University of Maryland, College Park (2005) http://trust.mindswap.org/papers/GolbeckDissertation.pdf.
2. Brickley, D., Miller, L.: FOAF vocabulary specification. RDF Vocabulary Specification (2005) http://xmlns.com/foaf/0.1.
3. Ding, L., Zhou, L., Finin, T.W., Joshi, A.: How the Semantic Web is being used: An analysis of FOAF documents. In: HICSS 2005 Proc. (2005)
4. Finin, T.W., Ding, L., Zhou, L., Joshi, A.: Social networking on the Semantic Web. The Learning Organization **12**(5) (2005) 418–435
5. Fitzpatrick, B.: OpenID 1.1. Technical Specification, OpenID (2005) http://www.openid.net/specs.bml.
6. Weitzner, D.J., Hendler, J., Berners-Lee, T., Connolly, D.: Creating a policy-aware Web: Discretionary, rule-based access for the World Wide Web. In Ferrari, E., Thuraisingham, B., eds.: Web & Information Security. IDEA Group (2006) 1–31
7. Carminati, B., Ferrari, E., Perego, A.: The REL-X vocabulary. OWL Vocabulary (2006) http://www.dicom.uninsubria.it/~andrea.perego/vocs/relx.owl.
8. Berners-Lee, T.: Notation 3 logic: An RDF language for the Semantic Web. W3C Draft, W3C (2005) http://www.w3.org/DesignIssues/N3Logic.
9. Berners-Lee, T.: Cwm – A general purpose data processor for the Semantic Web. Project Web site, W3C (2006) http://www.w3.org/2000/10/swap/doc/cwm.html.
10. Davis, I., Vitiello Jr, E.: RELATIONSHIP: A vocabulary for describing relationships between people. RDF Vocabulary Specification (2005) http://purl.org/vocab/relationship.
11. Golbeck, J.A.: The trust ontology. OWL Vocabulary (2006) http://trust.mindswap.org/ont/trust.owl.
12. REI: The rule markup initiative. Project Web site (2006) http://www.ruleml.org.
13. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web rule language combining OWL and RuleML. W3C Member Submission, W3C (2004) http://www.w3.org/Submission/SWRL.