Collaborative Access Control in On-line Social Networks

(Invited Paper)

Barbara Carminati
DICOM
University of Insubria
Varese – Italy
barbara.carminati@uninsubria.it

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

***Abstract*—Topology-based access control is today a de-facto standard for protecting resources in On-line Social Networks (OSNs) both within the research community and commercial OSNs. According to this paradigm, authorization constraints specify the relationships (and possibly their depth and trust level) that should occur between the requestor and the resource owner to make the first able to access the required resource. In this paper, we show how topology-based access control can be enhanced by exploiting the collaboration among OSN users, which is the essence of any OSN. The need of user collaboration during access control enforcement arises by the fact that, different from traditional settings, in most OSN services users can reference other users in resources (e.g., a user can be tagged to a photo), and therefore it is generally not possible for a user to control the resources published by another user. For this reason, we introduce *collaborative security policies*, that is, access control policies identifying a set of collaborative users that must be involved during access control enforcement. Moreover, we discuss how user collaboration can also be exploited for policy administration and we present an architecture on support of collaborative policy enforcement.**

***Index Terms*—Social Networks, Collaborative Security Policies, Collaborative Access Control**

I. Introduction

Online Social Networks are today one of the key components of our on-line presence. They are more and more used not only for recreational purposes, but also as a way to improve companies business and knowledge sharing. Relevant examples of this last trend are the Communities of Practice (CoPs) that today are more and more widespread used by many institutions and organizations. CoPs are groups of people in organizations that form to share what they know, to learn from one another regarding some aspects of their work and to provide a social context for that work [1]. The market for this kind of solutions is rapidly growing and has attracted the vendor interests (e.g., Microsoft, Tomoye).

The rapid widespread of OSN usage in different scenarios has highlighted the need of tools to protect user privacy and resource confidentiality. This has resulted in many efforts ranging from privacy-preserving tools for mining OSN data, to access control mechanisms [2], just to mention some of them. As far as access control is concerned, almost all the proposals appeared so far enforce *topology-based access control*, according to which access control requirements are expressed in terms of relationship paths existing in the network and their depth. For example, using topology-based access control a user can give access to one of his/her photo only to his/her friends and the friends of his/her friends, or to all his/her direct and indirect colleagues, no matter how distant they are in the network graph. Furthermore, some of the models support a notion of trust/reputation as a further parameter for access control decisions.

However, an important aspect that has been so far not deeply investigated is the collaborative dimension that access control may have in OSNs. The need of user collaboration during access control enforcement arises by the consideration that today OSNs impose to revise the traditional concept of resource ownership. Indeed, between OSN participants and resources there can exist several types of relationships, in addition to simply ownership. Let us consider, for example, a general purpose social network like Facebook. Users "own" a photo, but also can be "tagged" to a photo, they can "post" a comment to a wall, but also they can "reply" to an existing post. Similar examples hold also for CoPs. It is therefore necessary that access control is not only demanded to the resource owner but it involves also the other users that are somehow related to the requested resource. The importance of this, is also witnessed by the recent updates in the Facebook privacy settings. For the first time, Facebook users have the ability to check the contents they are tagged in before they appear on their profile. Moreover, Facebook has significantly expanded its "detagging" tool to help people report abusive posts, request a photo be removed, or block other users. We believe that collaboration is also useful at policy specification time in that, it may be the case that different users, connected to the considered resource, should be involved in the specification of the related access control policies. For these reasons, in this paper we introduce a new class of security policies, called *collaborative security policies* that basically extend topology-based access control by denoting a set of collaborative users. Collaborative users are identified on the basis of their relationships to the considered resource and are those whose feedback should be collected during policy specification or access control enforcement. To make easier the

COLLABORATECOM 2011, October 15-18, Orlando, United States
Copyright © 2012 ICST
DOI 10.4108/icst.collaboratecom.2011.247109

identification of collaborative users, we make use of semantic web technologies. In particular, we assume the existence of a RDF repository which encodes user-to-user and user-to-resource relationships and we show how this repository can be exploited for policy enforcement.

To the best of our knowledge we are not aware of proposals enforcing collaborative access control and policy administration in OSNs. However, some related work exist in the field of collaborative applications. For instance, the work reported in [3] proposes an XACML-based system to decompose a global policy into a set of local policies to be deployed to the collaborating parties. Then, a decision coordinator merges all the locally taken access control decisions to determine the answer to an access request. In contrast, our system is tailored to the OSN environment where it is better not to rely on a central authority to take access control decision. Rather, access control decision are taken by the resource owner. Moreover, [3] does not consider cooperative administration policies. Other proposals [4], [5] exploit role-based access control to provide various features in support of collaborative access control, such as delegation, team-based and task-based access control. However, they are not specifically tailored to OSNs and as such they do not support the kind of policies our system provides. Finally, some access control models have been already proposed for OSNs [2], but none of them support collaborative access control and policy administration.

The remainder of this paper is organized as follows. Section II introduces the reference OSN and access control model we use throughout the paper. Section III provides an overview of the architecture on support of collaborative access control, whereas Section IV introduces collaborative security policies. Policy enforcement is presented in Section V, whereas Section VI discusses the complexity of collaborative policy enforcement. Finally, Section VII concludes the paper.

II. Background on OSNs

In what follows, we briefly introduce how we model an OSN and the reference access control model we use to specify access control policies. More details can be found in [6].

We model an OSN as a directed labeled graph, where nodes correspond to users and arcs denote relationships between users. Given a relationship, the initial node of an arc denotes the user that has established the relationship, whereas the terminal node denotes the user that has accepted that relationship. Labels associated with arcs denote the type of the relationship modelled by the arc itself (e.g., "friend of," "colleague of"). We say that there exists a *direct* relationship of type rt between users A and B, if there is an arc connecting A and B labelled with rt. In contrast, A and B are in an *indirect* relationship of a type rt if there is a path of more than one arc connecting A and B such that all the arcs in the path have label rt. Relationships may be characterized by a trust level, representing how much a given user considers trustworthy another user w.r.t. the relationship modelled by the arc. To model trust levels, we assume that each arc has a further label $t \in [0, 1]$.

Each user in the OSN enforces his/her access control requirements[1] on the owned resources, according to the topology-based access control model described in [6]. According to this model, each resource to be shared in the OSN is protected by a set of *access rules*. Each access rule has the form (rsc, AC), where AC is a set of access conditions that need to be all satisfied in order to get access to resource rsc. An access condition is a triple $(v, rt, d_{max}, t_{min})$, where v is the node with whom the requestor must have a relationship of type rt, whereas d_{max} and t_{min} are, respectively, the maximum depth, and minimum trust level that the relationship should have in order to get the access. The depth of a relationship of type rt between two nodes v and $v^{'}$ corresponds to the length of the shortest path between v and $v^{'}$ consisting only of edges labelled with rt. In the current version of our system, we use the TidalTrust algorithm [7] to compute the trust of indirect relationships. However, other algorithms for trust computation can be easily adopted as well.

Example 2.1: Let us assume that a user, say Alice, wishes to protect the resource photo1, so that it can be accessed only by her friends or friends of her friends, but with a minimum trust value of 0.8. To encode these access control requirements, the resource has to be protected by the following access rule: (photo1,$\{$Alice, friend of, 2, 0.8$\}$).

III. Collaborative access control: overall architecture

In this section, we introduce the overall architecture of the framework in support of OSN collaborative access control. We assume that each user is able to locally manage all his/her resources. This is different w.r.t. the current situation of OSNs, where users have to upload all their resources to the SN servers, by delegating to them the full resource management, included access control enforcement. According to this view, we design the OSN as a *decentralized* architecture, where each user is represented as a peer, by which he/she is able to manage resources and relationships.

According to this view, we assume that access rules, defined according to the model introduced in Section II, are enforced directly by the resource owner [6] and not by a central reference monitor. When the owner receives an access request from a given user, he/she verifies whether there exists a relationship between the requestor and him/her that satisfies the constraints specified in the access rules associated with the requested resources. In support of this decentralized access control enforcement, we assume the presence of a trusted entity, called *Social Manager*, who knows the topology of the social network (i.e., the existing relationships and associated trust levels). As depicted in Figure 1, to determine whether a requestor satisfies an access rule associated with the required resource, the owner inquiries the Social Manager looking for a relationship satisfying the rule. This topology-based access control can be enhanced by exploiting the collaboration of OSN users.

[1]Here, we do not consider collaborative access control policies that will be described in Section IV.

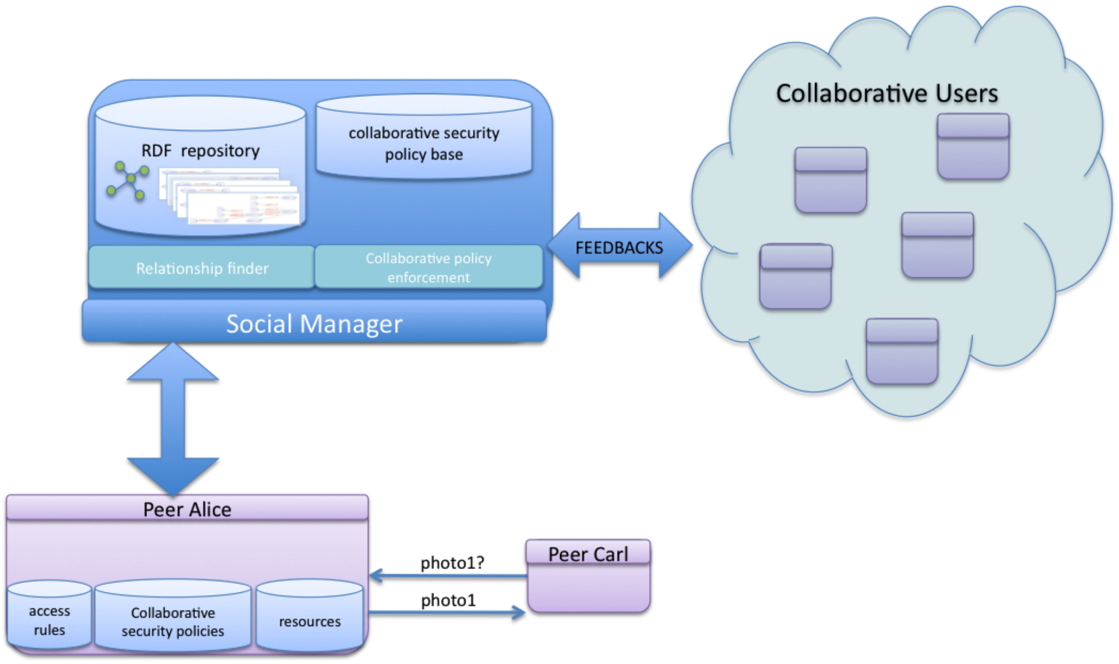


Figure 1. **Collaborative Access Control: Overall Architecture.**

In particular, we see two different roles according to which collaborative users may participate into the access control process. These are briefly summarized in what follows.

Collaborative access request evaluation. The first type of collaboration implies to contact a set of *collaborative users* during access request evaluation to collect their opinions on the resource release. These users can be identified based on the relationships they hold with the requested object (e.g., they are tagged into a photo). This implies that, before releasing a resource according to his/her access rules, the owner asks the approval of a set of collaborative users. As mentioned in the introduction, the need of an approval from collaborative users arises by the consideration that there exist situations where users, that are somehow related to the considered resource, should participate into the resource request evaluation, as the following better example shows.

Example 3.1: Let us assume that Alice owns a photo where Bob is tagged. Moreover, assume that Carl requires that photo. According to "traditional" topology-based access control, Alice retrieves the access rules protecting this photo, and asks to the Social Manager whether Carl has a relationship satisfying the constraints posed by the rules. However, even if the access rules are satisfied by Carl, it would be reasonable asking also for Bob's approval, in that he is on the photo, and he might prefer not to release the photo to Carl. This would imply to see Bob as a collaborative user for this access request, which entails, as a consequence, that also Bob has to inform Alice whether he agrees in releasing the photo to Carl.

Collaborative access rule administration. Collaborative users' feedback could also be relevant during the specification or update of access rules. Indeed, there are scenarios where specific collaborative users should have the right to validate access rules defined on resources owned by other OSN users. Let us consider for instance the following example.

Example 3.2: Let us consider a CoP, and assume that Alice creates a new document $doc1$ about project X. Moreover, let us assume that project X is lead by Tom. Since $doc1$ content could potentially reveal confidential information about the project, it would be reasonable to ask Tom to collaborate in defining the access rules regulating access to $doc1$.

The system we propose supports both *collaborative access request evaluation* and *collaborative access rule administration*. Both these features require to identify a set of collaborative users, whose cooperation is mandatory during access request evaluation or access rule management. In order to support this collaborative access control, we therefore introduce a further type of policies, called *collaborative security policies*, by which it is possible to identify collaborative users as well as the resources for which their cooperation is required. As shown in Figure 1, collaborative security policies are stored both at the Social Manager and peer side (see Section IV for further details). The idea is to identify collaborative users on the basis of their relationships with the resources. For instance, with reference to Example 3.1, collaboration to an access request evaluation is required to those users that are tagged to the photo, that is, users that are in the tagged relationship with a resource of type photo. In contrast, in the case of document $doc1$ of Example 3.2, collaborative administration of access rules is required to users that are project managers for project X.

To catch relationships between users and resources we exploit semantic web technologies. In particular, we assume that all resources as well as user relationships with resources are described by means of RDF files. These RDF descriptions are stored into a centralized RDF repository managed by the Social Manager (cfr. Figure 1), which encodes also the existing relationships among users (e.g., friends, colleagues, parents, and so on). As such, with a unique RDF repository we model information on the topology of the OSN as well as information on resource descriptions and user/resource relationships. This is slightly against the decentralized view proposed in [6], in that, by accessing information in the RDF repository, the Social Manager is able to infer resource descriptions and some information on user profiles (e.g., Tom is a project manager). This obviously partially breaks user privacy, but not resources confidentiality. Indeed, the Social Manager is able to access information on user relationships and resource descriptions given by the RDF files, but not resource contents, in that these are locally stored in the user machine. Note that, in this paper, we focus on resource confidentiality rather than on user privacy. As such, having access to resource descriptions and not to resource contents is similar to what happens in DBMS access control system, where, even if users could be aware of table definitions they could be not allowed to access the corresponding rows. Protection of user privacy under collaborative access control is a topic we would like to investigate in the future.

IV. Collaborative security policies

To support OSN collaborative access control and policy administration, we need to define new security policies, i.e., collaborative security policies, that will work together with access rules presented in Section II.

In general, a policy language defines policies according to three main components: a *subject specification*, aiming to specify the active entities to which a policy applies (e.g., collaborative users), an *object specification*, to identify the resources to which the policy refers to, and an action specification, specifying the action(s) that subjects can exercise on objects. As such, defining collaborative security policies requires formalizing these three components. Note however that, in case of collaborative security policies, the action that users identified by the subject specification have to perform is implicit. Indeed, the action implied by a collaborative access request evaluation is the evaluation of the access request itself, whereas in case of collaborative access rule administration the implied action is the validation of an access rule. As such, in formalizing collaborative policies we have to focus on the *subject* and *object* specification component only. Regarding the first component, the specification has to identify collaborative users according to their relationships with resources (i.e., users "tagged" to a photo, "manager" of a project). Moreover, the object specification should make a user able to identify resources according to their descriptions (i.e., object of "type" photo, document "about" an object of "type" project). To cope with both these requirements and taking into account that resources are described by means of RDF, we model subject and object specifications as *views* on RDF graphs. More precisely, the subject and the object specification of a collaborative security policy are defined as SPARQL queries [8].

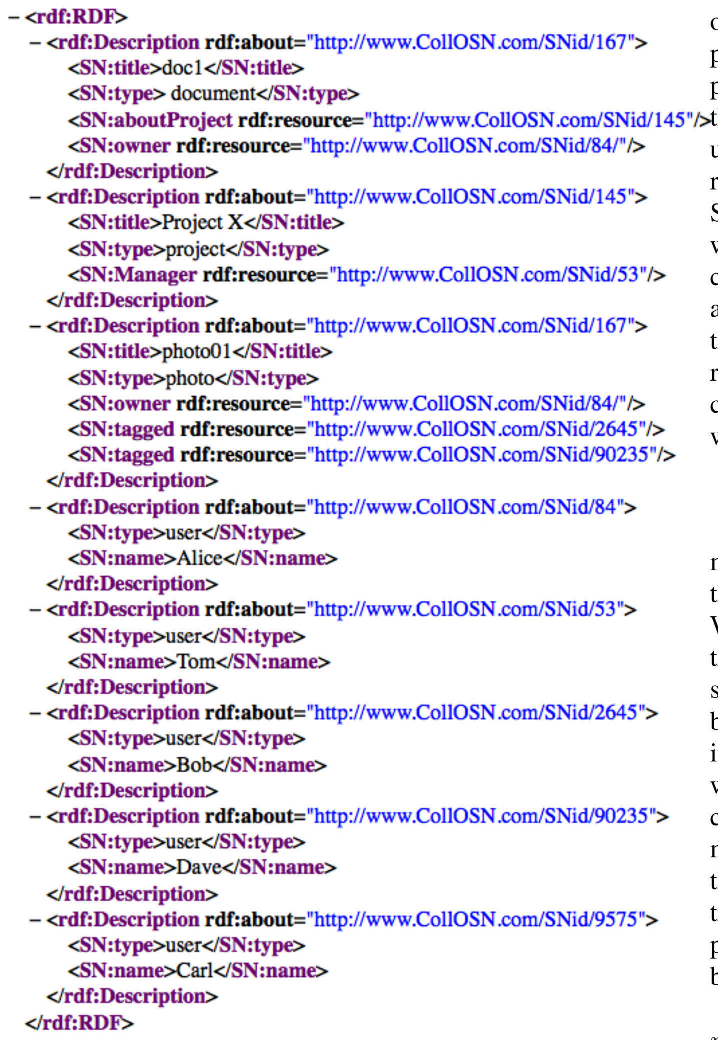
As discussed so far, the underling idea of collaborative access control and policy administration is to involve collaborative users during access request evaluation and/or access rule administration. This basically means to gather feedbacks from collaborative users and to use them to take a decision during these two tasks. However, it could be the case of receiving heterogeneous feedbacks in that, for instance, not all the collaborative users agree on releasing the resource. The way these feedbacks have to be combined to take the final decision greatly depends on the reference scenario. For instance, consider the scenario of Example 3.1, and assume that the photo is tagged to 10 different users, which play the role of collaborative users. Moreover, let us assume that 9 of them agree in releasing the photo whereas one does not. In this scenario, it would be reasonable that the photo is released even if a user disagrees, since the majority agreed on the release decision. Obviously, the scenario described in Example 3.2 for collaborative rule administration may have different stricter requirements. Indeed, in this scenario it could be reasonable that the new rule is validated only if all project leaders, i.e., collaborative users, agree on it.

To support a flexible strategy of combining user feedbacks, a collaborative security policy states also how many users have at least to give positive feedbacks to grant the access and/or validate an access rule. This is modeled by adding a third component to the policy, called *Mode*. A formalization of collaborative security policies taking in account all the above described requirements is given in what follows.

Definition 4.1: **(Collaborative security policy)**. A collaborative security policy $CollSP$ is defined as a tuple $(SubView, ObjView, Mode, Type)$, where $SubView$ is a SPARQL query identifying collaborative users, $ObjView$ is a SPARQL query identifying the resources to which $CollSP$ applies, $Mode \in \{All, One, Majority\}$ specifies that all, at least one, half plus one of the feedbacks of collaborative users have to be positive in order to enforce the policy. Finally, $Type$ specifies if $CollSP$ is a collaborative access control policy, i.e., $Type = Access$, or a collaborative administration policy, i.e., $Type = Admin$.

Example 4.1: Let us consider the RDF document represented in Figure 2, and assume it represents a simplified portion of the RDF repository modeling resources and users/resources relationships. For simplicity, the document uses only a property vocabulary defined by the Social Manager. However, several other vocabularies (e.g., FOAF) can be exploited. The document provides the RDF description of eight resources (i.e., the <rdf:Description> elements):[2] a resource of type document with title $doc1$ (i.e., a resource with type

[2]Note that, according to the RDF terminology, all described data are referred to as resources.



```
– <rdf:RDF>
  – <rdf:Description rdf:about="http://www.CollOSN.com/SNid/167">
      <SN:title>doc1</SN:title>
      <SN:type> document</SN:type>
      <SN:aboutProject rdf:resource="http://www.CollOSN.com/SNid/145"/>
      <SN:owner rdf:resource="http://www.CollOSN.com/SNid/84/"/>
    </rdf:Description>
  – <rdf:Description rdf:about="http://www.CollOSN.com/SNid/145">
      <SN:title>Project X</SN:title>
      <SN:type>project</SN:type>
      <SN:Manager rdf:resource="http://www.CollOSN.com/SNid/53"/>
    </rdf:Description>
  – <rdf:Description rdf:about="http://www.CollOSN.com/SNid/167">
      <SN:title>photo01</SN:title>
      <SN:type>photo</SN:type>
      <SN:owner rdf:resource="http://www.CollOSN.com/SNid/84/"/>
      <SN:tagged rdf:resource="http://www.CollOSN.com/SNid/2645"/>
      <SN:tagged rdf:resource="http://www.CollOSN.com/SNid/90235"/>
    </rdf:Description>
  – <rdf:Description rdf:about="http://www.CollOSN.com/SNid/84">
      <SN:type>user</SN:type>
      <SN:name>Alice</SN:name>
    </rdf:Description>
  – <rdf:Description rdf:about="http://www.CollOSN.com/SNid/53">
      <SN:type>user</SN:type>
      <SN:name>Tom</SN:name>
    </rdf:Description>
  – <rdf:Description rdf:about="http://www.CollOSN.com/SNid/2645">
      <SN:type>user</SN:type>
      <SN:name>Bob</SN:name>
    </rdf:Description>
  – <rdf:Description rdf:about="http://www.CollOSN.com/SNid/90235">
      <SN:type>user</SN:type>
      <SN:name>Dave</SN:name>
    </rdf:Description>
  – <rdf:Description rdf:about="http://www.CollOSN.com/SNid/9575">
      <SN:type>user</SN:type>
      <SN:name>Carl</SN:name>
    </rdf:Description>
  </rdf:RDF>
```

Figure 2. An example of RDF file.

and title properties set to document and $doc1$, resp.), a project (i.e., a resource with type project) with title *Project X*, a photo (i.e., type=photo) and 5 users, i.e. Alice, Tom, Bob, Dave, and Carl. As described in the RDF document, these resources are connected each other. More precisely, $doc1$ is in relationship with the resource with URI "http://www.CollOSN/SNid/84/", i.e., user Alice, by means of property $owner$. Moreover, it is connected by means of the $aboutProject$ property with "http://www.CollOSN/SNid/145", i.e., resource Project X. This last resource is in turn connected with "http://www.CollOSN.com/SNid/53", that is, user Tom, by means of the $manager$ property. The resource photo is connected to Alice as owner and to Bob and Dave as users tagged to it.

The following example describes some collaborative security policies that can be specified on the RDF description in Figure 2.

Example 4.2: According to Definition 4.1, the components of the collaborative security policy encoding the requirements described in Example 3.2 are represented in the first row of Table I. As required, the collaborate policy applies to all projects, as such the SPARQL query in the ObjView component identifies all resources (i.e., the URIs associated with them) with the type property set to "project". Collaborative users have to be the project managers associated with that resource. These are retrieved by the SPARQL query in the SbjView component, which returns both the resources URI as well as the URI(s) of the corresponding project manager(s). In contrast, the second row of Table I encodes the collaborative access control requirements described in Example 3.1. Here, the SPARQL query in the ObjView identifies URIs of all the resources with type photo, whereas the query in the SbjView component retrieves all the URIs of users tagged to resources with type photo.

V. Collaborative security policy enforcement

In order to describe collaborative security policy enforcement, we have to describe who defines such policies, where these are stored, and, as a consequence, who enforces them. With respect to collaborative administration policies, we think that the most reasonable solution is that they are defined by some sort of SN administrator. Indeed, these policies have to been defined according to some high-level guidelines in place in the SN that should apply to any OSN user. For instance, with respect to Example 3.2, it is reasonable to assume that the collaborative administration policy requiring to ask to project managers before associating a policy to a resource related to their projects is stated by the organization security administrator. Therefore, we assume that collaborative administrative policies are stored at the Social Manager side and evaluated by the Social Manager against the RDF repository.

This implies that, when a user creates a new resource, say rsc, he/she has to send the corresponding RDF description, RDF_{rsc}, to the Social Manager, together with the access control policy defined for the new resource, say ACP_{rsc}.[3] To determine if ACP_{rsc} has to be approved or not, the Social Manager computes the collaborative users identified by the collaborative administration policies applying to rsc, if any, and asks them a feedback. This process is described by Algorithm 1, which takes as input the set of collaborative administration policies, RDF_{rsc} and ACP_{rsc}, and returns the approval/denial message. For simplicity, Algorithm 1 assumes that there exists at most a unique collaborative administration policy specified for the resource.

As described in the algorithm, the Social Manager first uploads the RDF description of the new resource into the RDF repository. Then, in order to determine the collaborative administration policy to be enforced, Algorithm 1 considers each policy p in the policy base by (1) evaluating the SPARQL query contained into the object specification on the updated RDF repository,[4] and (2) verifying if the URI of the new resource URI_{rsc} is among the returned resource URIs (lines

[3]Here, for simplicity, we assume that the resource owner specifies only one access control policy for each resource.

[4]In the algorithm and throughout the paper we use the dot notation to indicate policy components.

SubjView	ObjView	Mode	Type
PREFIX SN: <http://www.CollOSN.com/rdf/> SELECT ?x ?CollUsers WHERE {?x SN:type "project". ?x SN:Manager ?CollUsers . ?CollUsers SN:name ?Username }	PREFIX SN: <http://www.CollOSN.com/rdf/> SELECT ?obj WHERE { ?obj SN:type "project" }	All	Admin
PREFIX SN: <http://www.CollOSN/rdf/> SELECT ?x ?CollUsers WHERE {?x SN:type "photo". ?x SN:Tagged ?CollUsers . ?CollUsers SN:name ?Username }	PREFIX SN: <http://www.CollOSN/rdf/> SELECT ?obj WHERE { ?obj SN:type "photo" }	Majority	Access

Table I
An example of collaborative security policies

```
input :
        (1) $CollAdminPB$, the policy base of
    collaborative administration policies;
        (2) $REP_{rdf}$, the Social Manager RDF repository;
        (3) $RDF_{rsc}$, the RDF description of the new
    resource $rsc$, where $URI_{rsc}$ denotes its URI;
        (4) $ACP_{rsc}$, the access control policy that $rsc$'s
    owner wishes to apply to $rsc$.
output: Approval or Denial of $ACP_{rsc}$
1  Update $REP_{rdf}$ with $RDF_{rsc}$;
2  Set flag=0, Fb = $\emptyset$;
3  for $p \in CollAdminPB$ do
4      Let $URIset$ be the URIs of those resources that are
       returned by the SPARQL query in $p.ObjView$
       executed on $REP_{rdf}$;
5      if $URI_{rsc} \in URIset$ then
6          flag=1;
7          Let $CollUsers$ be the URIs of users returned by
           the SPARQL query in $p.SbjView$ executed on
           $REP_{rdf}$;
8          for $u \in CollUsers$ do
9              $f_u$=RequestFeedback($u$,$ACP_{rsc}$,$RDF_{rsc}$);
10             if $p.Mode$='One' $\wedge f_u$ is positive then
11                 Return approval of $ACP_{rsc}$;
12                 exit;
13             Let $Fb$=$Fb \bigcup \{f_u\}$;
14         Let $pos \subseteq Fb$ be the set of positive feedbacks
           received from $CollUsers$;
15         if $p.Mode$='All' $\wedge$ $|pos| = |Fb|$ then
16             Return approval of $ACP_{rsc}$;
17             exit;
18         if $p.Mode$= 'Majority' $\wedge |pos| \geqslant (|Fb|/2 + 1)$
           then
19             Return approval of $ACP_{rsc}$;
20             exit;
21 if flag=0 then
22     Return approval of $ACP_{rsc}$;
23 else
24     Return denial of $ACP_{rsc}$ ;
```

4,5). If this is the case, the algorithm computes the set of collaborative users identified by the policy (line 7). These are retrieved by evaluating on the RDF repository the SPARQL query contained into the subject specification. Then, the algorithm requests a feedback to each of the identified collaborative users. This is done by means of function $RequestFeedback()$ that (1) sends the policy to be approved as well as the RDF description of the new resource to each collaborative user u, (2) waits until a feedback is received from u (line 9). Once the user feedback has been gathered, the algorithm verifies if the considered collaborative administration policy requires just a unique positive feedback (i.e., Mode="One"). If this is the case and the feedback is positive, the algorithm returns the approval message (line 11). Otherwise, it collects all collaborative users feedbacks and returns the approval message if (1) all feedbacks are positive and the policy has the mode component set to "All" (line 16) or (2) half plus one of the collected feedbacks are positive and mode is "Majority" (line 18). Finally, the algorithm returns an approval message if no collaborative administration policies apply to the new resource description (i.e., the value of the flag attribute has not been modified, see line 22), or a denial message, if a policy exists but collected feedbacks are not enough (i.e., flag=1). A similar process is performed when users wish to modify access control policies applying to their resources.

Example 5.1: Let us consider again Example 4.1, where Alice creates a new resource, doc1, about Project X. As depicted in Figure 3, once the new resource is uploaded, Alice (i.e., the peer installed at her side) sends the Social Manager the RDF description of the resource together with the access control policy ACP Alice wishes to apply to it. As described in Algorithm 1, the Social Manager updates the RDF repository with the new RDF description (step 3, Figure 3) and evaluates on the updated repository the collaborative administration policies. In particular, under the assumption that the only admin policy is the one described in Table 1, the algorithm verifies whether the URI of doc1 (i.e., http://www.collosn/SNid/145, see Figure 2) is contained among the URIs returned by the SPARQL query contained in the policy ObjView component (see the ObjView column of the first row in Table II). Since this is the case, the Social Manager computes the collaborative users evaluating the SPARQL query in the corresponding

	ObjView Evaluation	SbjView Evaluation	
1	**obj=** http://www.CollOSN.com/SNid/145	**x=** http://www.CollOSN/SNid.com/145	**CollUsers=** http://www.CollOSN.com/SNid/53
2	**obj=** http://www.CollOSN.com/SNid/167	**x=** http://www.CollOSN.com/SNid/167	**CollUsers=** http://www.CollOSN.com/SNid/9035
		x= http://www.CollOSN.com/SNid/167	**CollUsers=** http://www.CollOSN.com/SNid/2645

Table II
Evaluation of the SPARQL queries in the collaborative administration policy of Example 4.2.

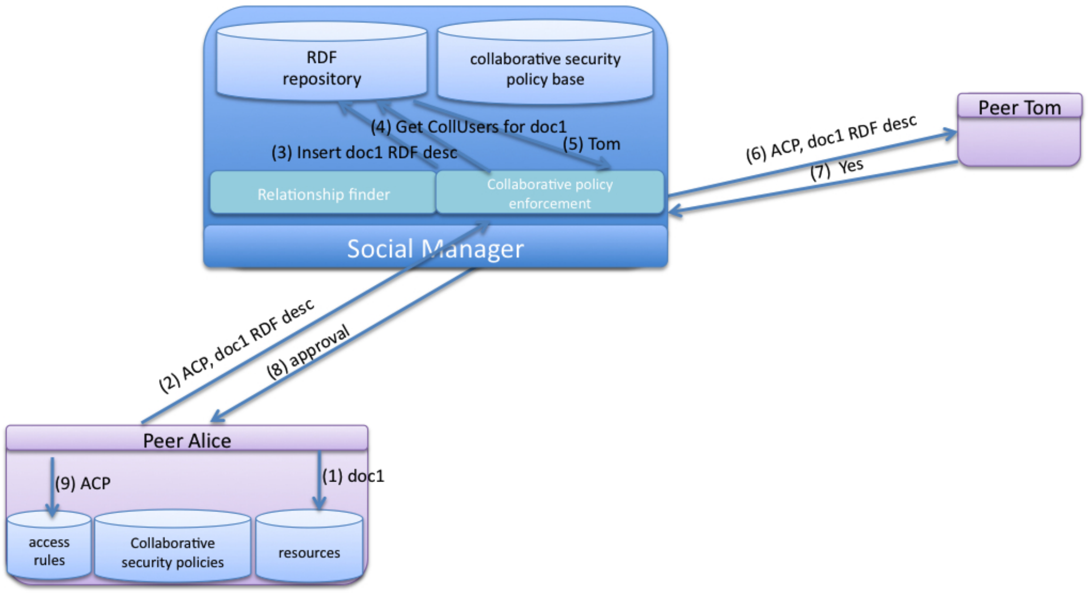


Figure 3. An example of collaborative administration policy enforcement.

SbjView component. The query returns Tom as the unique collaborative user (see the SbjView column of the first row in Table II). Thus, the Social Manager requires Tom feedback (cfr. steps 6,7 in Figure 3), by sending him the RDF description of doc1 as well as the ACP defined by Alice. Assuming a positive feedback, it returns to Alice an approval message. As a result, Alice inserts the new access control policy into her local policy base.

As far as collaborative access control policies are concerned, we see two possible design choices. Indeed, it is reasonable to assume that they are not under the full control of individual users, since SN admin may pose some guidelines also on who has to collaborate during access request evaluation. For example, the OSN admin may specify that in the considered OSN all users that are tagged to a photo have to collaborate in the request evaluation. We refer to these collaborative access control policies as *admin-defined collaborative access control policies*. However, it is also reasonable to assume that in some scenarios the owner of the resource should have the full control of his/her resources and related access control policies. This latter case implies that he/she is the only one that can decide who has to collaborate during access request evaluation, i.e., the only one who should define the so called *user-defined collaborative access control policies*. Admin-defined collaborative access control policies are stored at the Social Manager side, whereas in the user-defined ones are stored at the user side. However, no matter where collaborative access control policies are stored and by whom they are defined, both admin and user-defined access control policies imply a very similar enforcement, in that they both require to query the RDF repository to retrieve collaborative users and ask them a feedback. Therefore, in the following, we describe the enforcement of admin-defined collaborative access control policies. Then, we describe how this has to be modified in order to enforce user-defined collaborative access control policies.

Collaborative access control policy enforcement takes place, if needed, after the enforcement of traditional access control policies introduced in Section II. More precisely, once a user requests a resource, the resource owner verifies whether the requestor is authorized to access it according to his/her access control rules (defined in terms of relationship types, maximum depth and minimum trust). In order to verify these constraints, the owner has to interact with the Social Manager to verify whether the required relationships exist. If the requestor satisfies the owner's rules, collaborative access control policies are enforced. Since we are considering admin-defined collaborative access control policies, these have to be enforced at Social Manager side. As such, the owner first sends to the Social Manager the resource URI so as to make it able to verify if some collaborative access control policies apply to it. This process is similar to collaborative administration policy enforcement described in Algorithm 1. Indeed, as a first step, the Social Manager evaluates the SPARQL queries stored in the ObjView components of admin-defined collaborative access control policies against the RDF repository, and checks whether the received resource URI is among the ones returned by the queries evaluation. Then, if one or more collaborative access control policies apply, for each of them the Social Manager computes the set of collaborative users and contacts them for their feedbacks. If the gathered feedbacks satisfy the constraint specified in the Mode component of the considered admin-defined collaborative access control policy, the policy is satisfied and the access is granted, it is denied, otherwise.

To implement the enforcement of user-defined collaborative access control policies, the users should be able to evaluate SPARQL queries over the RDF repository to identify the triggered collaborative access control policies (i.e., those policies whose ObjView component applies to the RDF description of the required resource) and the associated collaborative users. However, we do not believe assuming an RDF repository and SPARQL queries capabilities at user side is always applicable. As such, in case this solution can not be implemented, we assume that the Social Manager will also enforce user-defined collaborative access control policies. This means that, once a user requests a resource, the resource owner verifies whether he/she is authorized to access it according to his/her access control rules. If the access is granted, it asks the Social Manager to enforce the collaborative access control policies defined by him/her. Therefore, the owner sends to the Social Manager the resource URI as well as his/her collaborative access control policies. The Social manager enforces these policies in the same way as it enforces admin-defined collaborative access control policies.

In the following example we better clarify the enforcement with user-defined collaborative access control policies.

Example 5.2: Let us consider again Example 3.1, where Carl asks Alice the access to photo1. As depicted in Figure 4, first Alice verifies wich access control rules apply to this resource. From the local policy base, the peer verifies that in order to access the resource Carl has to be friend of Alice with maximum distance 2 and minimum trust level 0.8. To verify if such relationship exists, the peer inquiries the Social Manager (cfr. step 2 in Figure 4). Suppose that the Social Manager confirms that between Carl and Alice there exists a friendship relationship of distance 2 with trust level 0.9, therefore the access rule is satisfied. To enforce collaborative access control, the peer sends to the Social Manager the RDF description of the photo as well the set of collaborative access control policies defined by Alice (cfr. step 7, Figure 4). The Social Manager performs the collaborative access control enforcement which implies to (1) retrieve the admin-defined collaborative access control policies applying to photo1, if any; (2) compute the collaborative users denoted by user-defined and admin-defined policies, (3) gather their feedbacks, (4) take a decision based on the value of the Mode component in the considered policies. More precisely, considering only the collaborative access control policy in Table 1, the enforcement verifies if photo1's URI (i.e., http://www.collosn/SNid/167, see Figure 2) is contained among the URIs returned by the SPARQL query contained in its ObjView component (see the ObjView column of the second row in Table II). Since this is the case, it computes the collaborative users by evaluating the query in the corresponding SbjView component. This query (see Table II) retrieves Bob and Dave. Thus, the Social Manager requires their feedbacks (cfr. steps 10-13 in Figure 4), by sending them the RDF description of photo1 as well as the requestor name, i.e., Carl. Once their feedbacks have been collected, the Social Manager verifies if the majority is positive, as required by the policy. As all feedbacks are positive, it returns Alice a message authorizing the grant to Carl. As a result, Alice releases photo1 to Carl.

VI. Complexity

In this section, we provide an analysis of the computational cost of collaborative access control and collaborative policy administration. Let us start to consider collaborative access control enforcement, by recalling that this implies the following tasks: (1) evaluation of topology-based access control rules, (2) evaluation of collaborative access rules so as to determine which are the collaborative users to be inquired; (3) the collection of user feedbacks. For all these tasks, we can estimate a cost. More precisely, we refer to [6] for the complexity analysis of topology-based access control enforcement, where it is highlighted that the most expensive computation in the enforcement is required by finding the paths in the OSN satisfying the constraints specified in the access rule. This task is performed in [6] by a Breath-First-Search algorithm, whose time complexity is $O(|E| + |V|)$, where E and V are the set of edges and nodes in the OSN, respectively. However, since the path search does not consider all the OSN edges, rather it is limited only to those with a given relationship type rt (i.e., the type required by the access condition), the complexity of this task can be estimated as $O(|E_{rt}| + |V_{rt}|)$, where E_{rt}, V_{rt} denote the subgraph of an OSN where edges E_{rt} are all labelled with rt type and nodes V_{rt} are all connected at least by an rt edge. As the experiments reported in [6] show, this search time requires about 1 second to explore a subgraph with 6,000 nodes. Note that, assuming an OSN adopting the FOAF relationship dictionary, which defines about 32 different relationship types [9], with relationship types uniformly distributed among the OSN edges, the test on 6,000 nodes is equivalent to a path search on an OSN of 192,000 nodes.

Task (2): the evaluation of collaborative access control policies requires to evaluate two sets of SPARQL queries. The first aims at retrieving the collaborative access control policies applying to the required resource. This implies to evaluate the SPARQL query contained into the ObjView component of each collaborative access control policy to verify whether the URI of the required resource is included. Whereas the second set of queries is just to retrieve the collaborative users

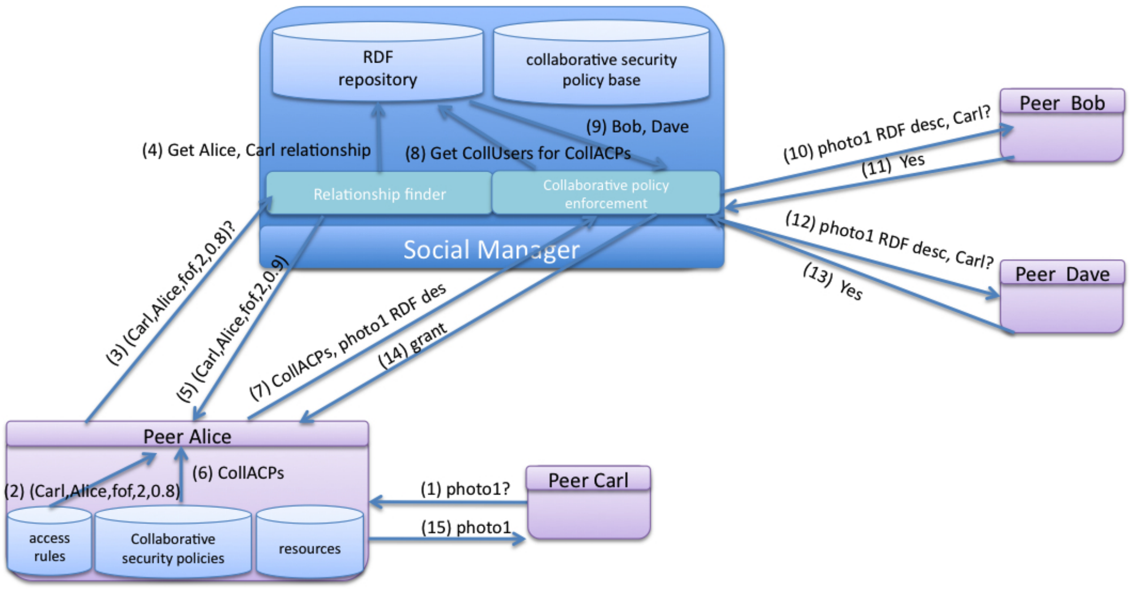

RDF repository
collaborative security policy base
Relationship finder
Collaborative policy enforcement
Social Manager
(4) Get Alice, Carl relationship
(8) Get CollUsers for CollACPs
(9) Bob, Dave
(10) photo1 RDF desc, Carl?
(11) Yes
(12) photo1 RDF desc, Carl?
(13) Yes
Peer Bob
Peer Dave
(3) (Carl,Alice,fof,2,0.8)?
(5) (Carl,Alice,fof,2,0.9)
(7) CollACPs, photo1 RDF des
(14) grant
Peer Alice
(2) (Carl,Alice,fof,2,0.8)
(6) CollACPs
access rules
Collaborative security policies
resources
(1) photo1?
(15) photo1
Peer Carl

Figure 4. An example of collaborative access control enforcement with user-defined collaborative policies.

specified by the policies identified in the previous step, and therefore it requires to evaluate the SPARQL queries contained in their SubjView components. Let Spa be the time required to evaluate a SPARQL query, and $Coll\_PB$ be the set of collaborative access control policies, task (2) complexity is $O(Coll\_PB \times Spa)$.[5] As such, the time of task (2) is strictly bound to the complexity of SPARQL query execution. This has been deeply investigated in the last fews years and several optimization techniques have also proposed (see for instance [10]). Moreover, we expect that queries in the ObjView and SubjView components of a policy will have a low complexity as they will mainly pose conditions on resources descriptions (e.g., the type property in the examples above) and on its directly connected resources (e.g., persons "tagged").

The final task in the collaborative access control process (i.e., task (3)) implies the gathering of feedbacks from collaborative users. It is important to note that in the collaborative process described in this paper, we assume that feedbacks are explicitly given by collaborative users. As such this task implies a time based on users reaction, which might greatly vary. Since the feedbacks are required in parallel to all users, we can estimate this time as the maximum delay with which users return feedbacks. We denote this time as $MaxDelay$. Therefore, the time required for collaborative access requests evaluation is $O(|E_{rt}|+|V_{rt}|+Coll\_PB\times Spa+MaxDelay)$. A similar analysis can be conducted for collaborative policy administration, where the main tasks are: (1) evaluation of collaborative administration policies, so as to determine which are the collaborative users to be inquired; (2) the collection of user feedbacks. Since, as discussed in Section V, enforcement of collaborative administration policies is similar to collaborative access control enforcement, we can conclude that time complexity is $O(Coll\_PB \times Spa + MaxDelay)$. As this complexity analysis highlights an important issue we need to address is the reduction of time needed to gather the feedbacks. To cope with this requirement we are investigating a mechanism in support of automatic generation of user feedbacks based on user additional access rules and preferences.

[5]Note moreover, that the complexity of this task can be reduced by adopting indexes associating collaborative policies to resources without the need to re-run task (2) for each access request.

VII. Conclusions

In this paper, we have shown how OSN topology-based access control can be enhanced by user collaboration. Collaboration can take place both during access control enforcement and policy specification and is regulated by proper collaborative security policies, which denote the set of users to be involved in the collaboration. Semantic web technologies are employed to support a rich way of denoting collaborative users based on the relationships they have with the considered resources.

Currently, we are implementing a prototype to test the performance of our system for different scenarios and OSNs. We also plan to investigate techniques to automate feedback generation and different ways of combining feedbacks as well as user-defined and admin-defined collaborative access control policies. Moreover, we plan to extend our system to enforce protection of user privacy.

References

[1] C. M. Hoadley, "Using technology to transform communities of practice into knowledge-building communities," *SIGGROUP Bull*, vol. 25, pp. 31–40, 2005.

[2] B. Carminati and E. Ferrari, "Privacy-aware access control in social networks: Issues and solutions," in *Privacy and Anonymity in Information Management Systems*, ser. Advanced Information and Knowledge Processing, J. Nin and J. Herranz, Eds. Springer London, 2010, vol. 0, pp. 181–195.

[3] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo, “Policy decomposition for collaborative access control,” in *Proceedings of the 13th ACM symposium on Access control models and technologies*, ser. SACMAT ’08. New York, NY, USA: ACM, 2008, pp. 103–112. [Online]. Available: http://doi.acm.org/10.1145/1377836.1377853

[4] E. Cohen, R. K. Thomas, W. Winsborough, and D. Shands, “Models for coalition-based access control (cbac),” in *Proceedings of the seventh ACM symposium on Access control models and technologies*, ser. SACMAT ’02. New York, NY, USA: ACM, 2002, pp. 97–106. [Online]. Available: http://doi.acm.org/10.1145/507711.507727

[5] J. Jin and G.-J. Ahn, “Role-based access management for ad-hoc collaborative sharing,” in *Proceedings of the eleventh ACM symposium on Access control models and technologies*, ser. SACMAT ’06. New York, NY, USA: ACM, 2006, pp. 200–209. [Online]. Available: http://doi.acm.org/10.1145/1133058.1133086

[6] B. Carminati, E. Ferrari, and A. Perego, “Enforcing access control in web-based social networks,” *ACM Trans. Inf. Syst. Secur.*, vol. 13, pp. 6:1–6:38, November 2009. [Online]. Available: http://doi.acm.org/10.1145/1609956.1609962

[7] J. A. Golbeck, “Computing and applying trust in web-based social networks,” Ph.D. dissertation, College Park, MD, USA, 2005, aAI3178583.

[8] E. Prud’hommeaux and A. Seaborne, “Sparql query language for rdf (working draft),” W3C, Tech. Rep., March 2007. [Online]. Available: http://www.w3.org/TR/2007/WD-rdf-sparql-query-20070326/

[9] D. Brickley and L. Miller, “FOAF vocabulary specification,” RDF Vocabulary Specification, Jul. 2005, available at: http://xmlns.com/foaf/0.1. [Online]. Available: http://xmlns.com/foaf/0.1

[10] M. Schmidt, M. Meier, and G. Lausen, “Foundations of sparql query optimization,” in *Proceedings of the 13th International Conference on Database Theory*, ser. ICDT ’10. New York, NY, USA: ACM, 2010, pp. 4–33. [Online]. Available: http://doi.acm.org/10.1145/1804669.1804675