# Private Relationships in Social Networks

Barbara Carminati Elena Ferrari Andrea Perego
DICOM, Università degli Studi dell'Insubria, Varese, Italy
E-mail: {barbara.carminati,elena.ferrari,andrea.perego}@uninsubria.it

## Abstract

*Current social networks implement very simple protection mechanisms, according to which a user can state whether his/her personal data, relationships, and resources should be either public or accessible only by him/herself (or, at most, by users with whom he/she has a direct relationship). This is not enough, in that there is the need of more flexible mechanisms, making a user able to decide which network participants are authorized to access his/her resources and personal information. With this aim, in [2] we have proposed an access control model where authorized users are denoted based on the relationships they participate in. Nonetheless, we believe that this is just a first step towards a more comprehensive privacy framework for social networks. Indeed, besides users' resources and personal data, also users' relationships may convey sensitive information. For this reason, in this paper we focus on relationship protection, by proposing a strategy exploiting cryptographic techniques to enforce a selective dissemination of information concerning relationships across a social network.*

## 1 Introduction

In the last few years, social networks have become one of the most successful services over the Web, exploited by an exponentially increasing number of users [5]. The main aim of Web-based social networks is to make available an information space, where each social network participant can publish and share information—such as personal data, annotations, blogs, and, generically, resources—for a variety of purposes. In some social networks, users can specify how much they trust other users, by assigning them a *trust level*. Information sharing is based on the establishment of relationships of different types among participants (e.g., "colleague of", "friend of"), which are also used to customize social network services. This is the case, for instance, of collaborative rating and filtering, where users' opinions are weighed on the basis of the existing relationships and trust levels. The availability of this information obviously raises privacy and confidentiality issues. In fact, users' personal data and resources are regularly exploited not only by companies, for marketing purposes, but also by governments and institutions for tracking persons' behaviours/opinions, and, in the worst case, even by online predators [1]. Moreover, the adoption of Semantic Web technologies to represent personal information, user relationships, and trust levels [3], have increased information interoperability, thus making simpler to access users' data.

For these reasons, recently some social networks—e.g., Facebook (http://www.facebook.com) and Videntity (http://videntity.org)—have started to enforce quite simple protection mechanisms, according to which users can decide whether their data, relationships, and, generically, resources, should be either public or accessible only by themselves and/or by users with whom they have a direct relationship. However, such mechanisms are not enough, in that they enforce too restrictive protection policies. There is then the need of enforcing more flexible strategies, making a user able to define his/her own rules, denoting the set of network participants authorized to access his/her resources and personal information, even though they are not directly connected through a relationship.

Some preliminary approaches addressing these issues have been recently proposed. In [2] we have described a flexible access control mechanism supporting a selective dissemination of users' resources in social networks. The model presented in [2] denotes the users authorized to access a resource in terms of the relationship type, depth, and trust level that they should have with other users in the network in order to gain access to the requested resource. To the best of our knowledge, this is the first proposal of an access control mechanism for social networks. As far as access control enforcement is concerned, we have adopted the *client-side* approach outlined by Weitzner et al. in [7], where access to resources is granted if the requestor is able to demonstrate that he/she satisfies the requirements (in terms of relationships he/she participates in) stated by access rules. Main benefits of such decentralized enforcement

1-4244-0832-6/07/$20.00 ©2007 IEEE.

mechanism are in terms of scalability and efficiency of access control. A mechanism to protect personal information in social networks by using an anonymity-based approach is described in [6]. The principle is that nodes in the network are anonymous, and thus it is not possible to link a node to a specific user, unless you already know him/her. Such strategy has the advantage of keeping publicly available users' data, but of avoiding that they can be used to track the behaviour of specific persons.

In this paper, we focus on relationship protection, since we believe that this is a fundamental issue that, to the best of our knowledge, has never been addressed before. Relationships in a social network may give rise to some relevant privacy concerns. For instance, a user would like to keep private the fact that he/she has a relationship of a given type with a certain user. In other cases, a user would like to avoid other users know the existence of a relationship of a given type, independently from the user with whom it is established. Suppose, for instance, that a user $u$ participates to a network consisting of company $X$'s consultants. Even though the identity of the other members of the community is not revealed, knowing that user $u$ is involved in such a network discloses by itself private information.

For this purpose, here we complement the model proposed in [2] with a mechanism able to enforce different privacy requirements on social network relationships. This is a major extension in that it requires to first define a format to express privacy requirements on relationships and then to devise mechanisms to avoid leakage of private information concerning relationships. The key point is that, since relationships information is fundamental for access control, we have to devise a method able to protect the privacy of relationships and, at the same time, make such information usable for access control purposes.

In order to address this issue, we specify privacy requirements through a set of *distribution rules*, which basically state the protection requirements to be enforced on a relationship. Cryptographic-based techniques are then used to enforce distribution rules and to avoid privacy breaches that may arise when access control is carried out.

The remainder of this paper is organized as follows. Section 2 briefly summarizes the access control model proposed in [2]. Section 3 illustrates our approach for relationship protection, whereas Section 4 discusses the security issues determined by relationship updates. Finally, Section 5 concludes the paper and outlines future research directions.

## 2 Background

In this section we briefly introduce the access control model and the related mechanism presented in [2].

First of all, we denote a social network $\mathcal{SN}$ by a tuple $(V_{\mathcal{SN}}, E_{\mathcal{SN}}, RT_{\mathcal{SN}}, T_{\mathcal{SN}}, \phi_{E_{\mathcal{SN}}})$, where $V_{\mathcal{SN}}$ and $E_{\mathcal{SN}} \subseteq [V_{\mathcal{SN}}]^2$ are, respectively, the nodes and edges of a digraph $(V_{\mathcal{SN}}, E_{\mathcal{SN}})$, $RT_{\mathcal{SN}}$ is the set of supported relationship types, $T_{\mathcal{SN}}$ is the set of supported trust levels, and $\phi_{E_{\mathcal{SN}}} : E_{\mathcal{SN}} \rightarrow RT_{\mathcal{SN}} \times T_{\mathcal{SN}}$ is a function assigning to each node $e \in E_{\mathcal{SN}}$ a relationship type $rt \in RT_{\mathcal{SN}}$ and a trust level $t \in T_{\mathcal{SN}}$. The number and type of relationships in $RT_{\mathcal{SN}}$ and trust levels in $T_{\mathcal{SN}}$ depend on the specific social network and its purposes; our only assumption is that $|RT_{\mathcal{SN}}| > 0$. Similarly, each social network supports a different range and type of trust levels [4], corresponding either to a set of integers or rational numbers, or to Boolean values. In case a social network does not support trust, this means that all the nodes are equally trustworthy, and thus we assume that each edge is associated with the maximum level of trust. Given an in/direct relationship *rel* between nodes $v$ and $v'$, the trust level of *rel* denotes how much $v$ considers trustworthy $v'$ wrt relationship *rel*. We do not enter here into the details of the formulas used to calculate trust levels [4], 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 nodes $v'$ and $v$, such calculation needs to take into account the trust levels of all the edges belonging to paths corresponding to all the existing in/direct relationships of type $rt$ between $v'$ and $v$. A simple example of social network is depicted in Figure 1. In the figure, the initial node of an edge is the node which established the corresponding relationship. E.g., the edge connecting $A$ to $B$ states that $B$ has a relationship of type $rt$ with $A$.

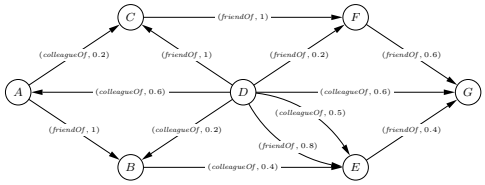


**Figure 1. A portion of a social network. Labels associated with edges denote, respectively, the type and trust level of the corresponding relationship.**

In the model described in [2], access control is enforced through a set of *access rules*. Given a resource *rsc*, owned by a node in $\mathcal{SN}$, an access rule AR is a pair $(rid, AC)$, where *rid* is the identifier (i.e., the URI) of resource *rsc*, and $AC$ is the set of *all* the *access conditions* to be satisfied to access resource *rsc*—i.e., a conjunction of access conditions. An access condition is a tuple $\text{AC} = (v, rt, d\max, t\min)$, where $v \in \mathcal{SN}$ is the node with which the node requesting a given resource must have a direct or indirect relationship,

whereas $rt \in RT_{SN}$, $d$ max, and $t$ min are, respectively, the type, maximum depth, and minimum trust level that the relationship should have. The depth of a relationship of type $rt$ between two nodes $v$ and $v'$ corresponds to the length of the shortest $v$-$v'$ path consisting only of edges labelled with $rt$. Finally, the same resource can be associated with one or more *alternative* access rules: in other words, the set $AR = \{\text{AR}_1, \ldots, \text{AR}_n\}$ of access rules applying to a given resource *rsc*, corresponds to the disjunction $(\text{AR}_1 \vee \cdots \vee \text{AR}_n)$.

**Example 2.1 (Access Rules)** *Consider the social network in Figure 1, and suppose that D(avid) owns a resource rsc, with identifier rid, and that he wishes to make rsc accessible only to his friends, and to the friends of his friends, with the constraint that their trust level must be at least equal to* 0.2*. This policy can be expressed by the following access rule:* $\text{AR}_1 = (rid, \{(D, friendOf, 2, 0.2)\})$*. Suppose now that David decides to grant access also to his colleagues with a minimum trust level equal to* 0.5*. This can be achieved by specifying an additional access rule* $\text{AR}_2 = (rid, \{(D, colleagueOf, 1, 0.5)\})$*. By contrast, if David wishes to grant access to resource rsc only to friends who are also colleagues (with the same constraints on depth and trust level in the examples above), he can specify the following access rule* $\text{AR}_3 = (rid, \{\text{AC}_1, \text{AC}_2\})$*, where* $\text{AC}_1 = (D, friendOf, 2, 0.2)$ *and* $\text{AC}_2 = (D, colleagueOf, 1, 0.5)$*. This access rule ensures that access to rsc will be granted if the requestor satisfies both* $\text{AC}_1$ *and* $\text{AC}_2$*.*

Access control is enforced according to the approach proposed in [7]. When a user (hereafter, the *requestor*) requests a resource to another user (hereafter, the *resource owner*), the former receives from the latter the 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 required. The requestor has to provide the resource owner with a *proof* of the requirements expressed by the received access rules, showing that between them there exists the required relationship, and that this relationship has the required depth and trust level. Therefore, the first issue to be addressed is how a resource owner can be ensured that proofs created by requestors are correct and trustworthy.

Since proofs demonstrate the existence of specific relationships involving the resource requestor, first of all we certify relationships through the use of *relationship certificates*, which are generated by two nodes establishing a relationship. Whenever a user, say Alice, establishes a new relationship with Bob, they both create and sign a certificate stating that between them there exists a direct relationship of a certain type and with a certain trust. By means of relationship certificates, it is possible to verify the correctness of proofs concerning the corresponding relationship. The correctness of a proof related to an in/direct relationship between Alice and Bob can be verified by retrieving the *chain of certificates* confirming the existence of a path between them. By contrast, verifying the trust level contained into a proof needs a more complex strategy, since the trust level between two nodes is computed taking into account all the possible paths connecting them [4]. Therefore, the requestor must provide the resource owner with all the corresponding chains of certificates in order to let him/her verify the correctness of the trust level certified by the proof. 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 adopted a slightly different architecture wrt the one outlined by Weitzner et al. in [7]. 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 relationship certificates and computing relationships' trust levels. Thus, whenever a requestor receives from the resource owner the access rules regulating the release of one of his/her resources, the former requests to *CN* the certificate chains proving the existence of a given relationship with a given depth, as well as its trust level. The certificate chains, if any, are generated by *CN* by consulting the relationship certificate repository, and, before being returned to the requestor, the corresponding trust level is signed by *CN*. Then, the requestor exploits the received certificate chains to generate the proof.

**Example 2.2 (Access Control Enforcement)** *Consider the social network in Figure 1, and suppose that G(reg) requests to D(avid) the resource with ID rid. Suppose, moreover, that the following access rule applies:* $(rid, \{(D, friendOf, 2, 0.2)\})$*. Two paths of type friendOf and maximum depth equal to* 2 *exist between Greg and David, namely, DEG and DFG, corresponding to the following chains of certificates:* $CC_{DEG} = \{(D, E, friendOf, 0.8), (E, G, friendOf, 0.4)\}$*;* $CC_{DFG} = \{(D, F, friendOf, 0.2), (F, G, friendOf, 0.6)\}$*. Based on such chains, CN computes the trust level of the corresponding relationship. For this purpose, we can sum the product of the trust levels of certificates in the different chains, and then divide the result by the number of chains. Thus we have:* $\frac{1}{2}(0.4 \cdot 0.8 + 0.6 \cdot 0.2) = 0.22$*.*

*After having received from CN the certificate chains and the average trust level, Greg computes the depth of the corresponding relationship, which is equal to the length of the shortest chain. Greg then returns to David the set of certificate chains and the assertion* $(D, G, friendOf, 2, 0.2)$*, stating that between David and Greg there exists a relationship of type friendOf, depth* 2 *and trust level* 0.22*. Since the assertion satisfies the condition* $(D, friendOf, 2, 0.2)$*, access*

*to the resource is granted.*

## 3 Relationship Protection

In the approach proposed in [2], relationships are assumed to be public. Here, we extend such model by enforcing privacy requirements on social network relationships.

Relationships play a crucial role in access control in that relationship certificates are used by a requestor to prove he/she has the credentials to access a given resource (see Section 2). However, relationship certificates may convey information that the nodes involved in the corresponding relationship would like to keep private.

Consider, for instance, Example 2.2. At the beginning, $G$ is aware only of the relationships it participates in—i.e., it knows to be colleague of $D$, and friend of $E$ and $F$. But when $G$ retrieves the chains of certificates from the central node $CN$, it is informed that $D$ is friend of $C$, $E$, and $F$, that $C$ is friend of $F$, and how much they trust each other. In case $C$, $D$, $E$, or $F$ would have liked to keep private all or part of such relationships, the certificates returned by the central node determine an improper dissemination of private information.

For this reason, we propose a mechanism (described in Section 3.1) according to which information about the existing relationships can be accessed only by authorized users. Moreover, the proposed solution protects certificates also from being accessed by the central node $CN$, that therefore it is not required to be trusted.

However, relationship certificates are not the only means by which information about relationships could be improperly disseminated. Also access control should be carefully considered. As an example, consider a node $R$ asking a resource to a node $O$, and suppose, moreover, that $O$ states that in order to get access to the requested resource a node must prove to have a relationship of type $rt$ with it. In such a case, $R$ can infer that $O$ participates in a relationship of type $rt$ with at least a node in the network, otherwise such resource could not be accessed by anyone.

Relationship types may not require to be protected in case they are 'generic', such as "friend", "colleague", etc., but in other scenarios access request answers may lead to an inappropriate distribution of private information. Consider for instance the case of a relationship of type "employee of company $X$": if such relationship type is used in an access rule, all the nodes in $\mathcal{SN}$ may be aware of which is the company that a given user is working for, by simply issuing access requests to that node.

In the following sections, we illustrate the solutions we have devised to address both the above-mentioned privacy issues.

### 3.1 Protecting Relationship Certificates

In order to prevent unauthorized access to relationship certificates, we associate with them one or more *distribution rules*. Similarly to access rules, these rules contain a set of *distribution conditions* determining the characteristics of the users to whom certificate access is granted. The notion of distribution condition is formally defined as follows.

**Definition 3.1 (Distribution Condition)** *A distribution condition* DC *for a relationship certificate* RC *is a tuple* $(v, rt, d\max)$*, where* $v \in V_{\mathcal{SN}}$ *is the node with which there must exist a relationship of type* $rt \in RT_{\mathcal{SN}}$ *and maximum depth* $d\max \in \mathbb{N}$*, in order to gain access to* RC*.*

A set of distribution conditions $DC$ related to a certificate is called *distribution rule*. More precisely, when a node $v$ wishes to establish a relationship with another node $v'$, $v$ negotiates with $v'$ the characteristics of the nodes that are authorized to read the corresponding relationship certificate RC, by specifying a set of distribution conditions $DC$. Then, they generate a *certificate key* $CK$, i.e., a pair $(k_{\text{RC}}, id_{\text{RC}})$, where $k_{\text{RC}}$ is a symmetric key and $id_{\text{RC}}$ is the corresponding key identifier. The resulting distribution rule DR is a pair $(CK, DC)$ consisting of the certificate key $CK$ and the set of conditions $DC$ denoting the nodes authorized to read certificate RC.

The semantics of distribution conditions is the same as access conditions, i.e., a distribution condition set $DC(\text{DR}) = \{\text{DC}_1, \ldots, \text{DC}_n\}$ of a distribution rule DR corresponds to the conjunction $(\text{DC}_1 \wedge \cdots \wedge \text{DC}_n)$. In case the distribution of a certificate key should be controlled by alternative conditions, this is obtained by specifying a set $\text{DR}_1 = (CK, DC_1), \ldots, \text{DR}_n = (CK, DC_n)$ of distribution rules containing the same certificate key $CK$ but different condition sets $DC_1, \ldots, DC_n$. Consequently, such set $DR_{CK} = \{\text{DR}_1, \ldots, \text{DR}_n\}$ of distribution rules corresponds to the disjunction $(\text{DR}_1 \vee \cdots \vee \text{DR}_n)$. Thus, the general form of the distribution rules applying to a given certificate can be expressed as the following logical formula in DNF:

$$(\text{DC}_1 \wedge \cdots \wedge \text{DC}_k) \vee \cdots \vee (\text{DC}_m \wedge \cdots \wedge \text{DC}_n)$$

Encrypted relationship certificates are stored into a public certificate directory $\mathcal{CD}$, managed by the central node. More precisely, once the agreement has been reached, $v$ sends to $\mathcal{CD}$ the tuple $(id_{\text{RC}}, E_{k_{\text{RC}}}(\text{RC}))$.[1] The idea is that nodes $v$ and $v'$ distribute the certificate key $CK$ to all and only the nodes satisfying the distribution rules they have specified for certificate RC. For this purpose, any node in $\mathcal{SN}$ is equipped with a certificate key directory $CKD$, storing the set $DR_{CK}$ of distribution rules concerning each certificate key $CK$ it received.

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

**Algorithm 1** Certificate Key Distribution

1: **Input** A node $v \in V_{S\mathcal{N}}$, a set of nodes $V \subseteq V_{S\mathcal{N}}$, and a set of distribution rules $DR \subseteq CKD_v$
2: **for all** $\text{DR} \in DR$ **do**
3:  **for all** $\overline{v} \in V$ **do**
4:  **if** $\forall \text{DC} \in DC(\text{DR})$, $v$ has a relationship of type $rt(\text{DC})$ with $\overline{v}$ **then**
5:  $\overline{d\max} \leftarrow d\max(\text{DC}) - 1$
6:  $\overline{\text{DC}} \leftarrow (\overline{v}, rt, \overline{d\max})$
7:  $\overline{DC} \leftarrow \overline{DC} \cup \{\overline{\text{DC}}\}$
8:  **else**
9:  $\overline{DC} \leftarrow \emptyset$
10:  **break**
11:  **end if**
12:  **if** $|\overline{DC}| > 0$ **then**
13:  **send** $(CK(\text{DR}), \overline{DC})$ **to** $\overline{v}$
14:  **end if**
15:  **end for**
16: **end for**

**Example 3.1 (Distribution Rules)** *Consider the social network in Figure 1, and suppose that D(avid) establishes a relationship rel, which he wishes to disclose only to his friends and to the friends of his friends. This policy can be expressed by the distribution rule* $\text{DR}_1 = (CK, \{(D, friendOf, 2)\})$. *Suppose now that David decides that also his colleagues can read relationship rel. This can be achieved by specifying an additional distribution rule* $\text{DR}_2 = (CK, \{(D, colleagueOf, 1)\})$. *Thus, the certificate key CK will be distributed to all the nodes in* $S\mathcal{N}$ *satisfying either* $\text{AR}_1$ *or* $\text{AR}_2$. *By contrast, in case David wishes to disclose rel only to friends who are also colleagues (with the same constraints on depth in the examples above), he can specify the following distribution rule* $\text{DR}_3 = (CK, \{\text{DC}_1, \text{DC}_2\})$, *where* $\text{DC}_1 = (D, friendOf, 2)$ *and* $\text{DC}_2 = (D, colleagueOf, 1)$.

The distribution of certificate keys is carried out according to a protocol, described by Algorithm 1, involving all the nodes authorized to access the corresponding relationship certificates, which is performed each time a node establishes a new relationship, or when the node receives a new certificate key from one of its neighbours, to verify whether such key must be further distributed. It is important to note that distribution rules are not delivered in their original form, but they are transformed in order to be relative to the receiving node. More precisely, the node components of each distribution condition are substituted with the identifier of the node receiving the rule, whereas the depth components are decreased by a unit. Consequently, as soon as the depth component of at least one of the conditions in the distribution rule is equal to 0, the distribution of the corresponding certificate key is stopped, and then we say that such distribution rule is no longer *active* (see Example 3.2).

The algorithm receives as input three parameters (line 1): the node $v$ executing the protocol, a set of nodes $V \subseteq V_{S\mathcal{N}}$, and a set of distribution rules $DR$ included into the certificate key directory $CKD_v$ of $v$. The elements of sets $V$ and $DR$ depend on whether the protocol is executed after $v$ has established a new relationship or when $v$ receives a new certificate key. In the former case, $V$ has only one element, corresponding to the node $\overline{v}$ with which $v$ established the new relationship, whereas $DR$ contains all the *active* distribution rules—i.e., the rules where, for each access condition, the maximum depth component is greater than 0. By contrast, if the protocol is executed when $v$ receives a new certificate key $CK$, $V$ corresponds to the set $V(v)$ of its neighbours, whereas $DR$ contains just the distribution rule $\text{DR} = (CK, DC)$ of key $CK$. Node $v$, then, verifies whether the nodes in $V$ satisfy the set of conditions $DC(\text{DR})$ of each distribution rule $\text{DR} \in DR$ (lines 2-4). This is achieved by retrieving the sets of certificates in the directory $CD_v$ of $v$, concerning relationships between $v$ and each one of its neighbours $\overline{v} \in V$, and then comparing the relationship type components of such certificates with those in the access conditions in $DC(\text{DR})$. Thus, if $\overline{v}$ satisfies DR, $v$ computes the modified conditions to be sent to $\overline{v}$, and builds the corresponding condition set $\overline{DC}$ (lines 5-7); otherwise, $\overline{DC}$ is set to the empty set, and the loop is stopped (lines 9-10). Finally, if the set $\overline{DC}$ of modified distribution conditions is not empty, $v$ sends to $\overline{v}$ a distribution rule $\overline{\text{DR}}$ consisting of the certificate key $CK(\text{DR})$ of the original distribution rule DR, and the set of modified distribution conditions $\overline{DC}$ (lines 12-14).

**Example 3.2 (Certificate Key Distribution)** *Consider the social network in Figure 2(a), where rt and* $rt'$ *stand for friendOf and colleagueOf, respectively. Suppose that D(avid) wishes to establish a new relationship of type friendOf with A(lice). Thus, Alice and David generate a certificate* $\text{RC}=(D, A, friendOf, 0.5)$ *and a certificate key* $CK=(k_{\text{RC}}, id_{\text{RC}})$.

*Suppose they agree on a (single) distribution rule containing the condition* $\text{DC}=(D, friendOf, 2)$, *stating that certificate* RC *should be disclosed only to David's friends, and to the friends of his friends. Then, David sends CK and a transformed version of* DC *to Alice, C(arl), E(ve), and F(rank)—i.e., all David's neighbours with whom he established a relationship of type friendOf. The transformed versions of* DC *are the following: Alice:* $(A, friendOf, 1)$*; Carl:* $(C, friendOf, 1)$*; Eve:* $(E, friendOf, 1)$*; Frank:* $(F, friendOf, 1)$.

*Then, Alice sends CK and further transformed versions of* DC *to Bob, Carl to Frank (who already received CK, and thus he ignores the new distribution rule), whereas both Eve and Frank send them to G(reg). Thus, Bob and Greg will receive the following distribution conditions: Bob:* $(B, friendOf, 0)$*; Greg (from Eve and Frank):* $(G, friendOf, 0)$. *Therefore, Greg receives two copies of the same rule for the same certificate key, of which he will ig-*

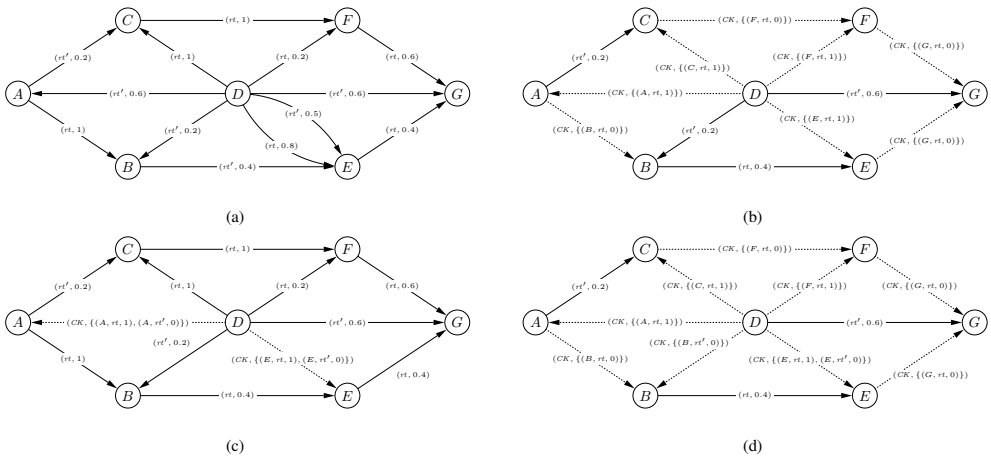


**Figure 2. Figures 2(b)-2(d) depict three examples of certificate key distribution, based on the graph in Figure 2(a), referring to the establishment of a relationship of type $rt = friendOf$ between nodes $D$ and $A$. Continuous edges denote relationships, whereas dotted edges denote certificate keys delivery.**

*nore the later one. Finally, since the maximum depth in the conditions above is* 0*, Bob and Greg do not distribute certificate key CK to their neighbours (see Figure 2(b)).*

*All the nodes involved in the distribution of CK store it in their certificate key directories, along with the distribution condition they received. Thus, whenever Alice or Eve establish a new relationship of type friendOf with a node $v \in \mathcal{SN}$, they will send to v both CK and a modified version of the stored distribution condition, in case v satisfies it.*

*Suppose now that* RC *is associated with an additional distribution condition* $(D, colleagueOf, 1)$*. Therefore, the distribution rule related to* RC *is* $(CK, \{(D, friendOf, 2), (D, colleagueOf, 1)\})$*. According to the semantics of a condition set, the nodes authorized to access certificate* RC *are those participating with David in a relationship of type friendOf, with a depth not greater than* 2*,* and *of type colleagueOf, with a depth not greater than* 1*. Besides Alice, only Eve satisfies such conditions, and thus David sends them the transformed version of the distribution rule* $(CK, \{(D, friendOf, 1), (D, colleagueOf, 0)\})$*. By contrast, Alice and Eve will not distribute the received certificate key to any of their neighbours, since one of the received conditions has a maximum depth equal to* 0 *(see Figure 2(c)).*

*Finally, suppose that with certificate* RC *two distribution rules are associated, namely* $(CK, \{(D, friendOf, 2)\})$ *and* $(CK, \{(D, colleagueOf, 1)\})$*. In such a case, the certificate key will be distributed also to Bob, however each authorized neighbour will receive only the distribution rule he/she satisfies. In other words, Alice, Carl, Frank, and Greg will receive only the modified version of* $(CK, \{(D, friendOf, 2)\})$*, Bob the modified versions of* $(CK, \{(D, colleagueOf, 1)\})$ *from David and of* $(CK, \{(D, friendOf, 2)\})$ *from Alice, whereas Eve will receive the modified versions of both from David, since she has with him relationships of both types friendOf and colleagueOf (see Figure 2(d)).*

### 3.2 Protecting Relationships in Access Rules

As we have seen at the beginning of this section, answers to an access request may reveal private information about the relationships existing among social network participants. The reason is that, when a user requests a resource, the resource owner replies with an access rule, which contains, among others, the relationships that the requestor must have with a specific user in order to grant the access. If the owner wants to keep private some types of relationships he/she has with other network nodes, this mechanism could lead to privacy breaches.

To avoid that, we adopt a cryptographic-based approach according to which relationship types are associated with encryption keys that are used to hide relationship information into access control rules. Whenever a node $v$ establishes a relationship of type $rt$ with a node $v'$, besides performing all procedures concerning certificate generation and certificate key distribution illustrated in Section 3.1,

they also agree on a symmetric key $k_{rt}$ and a key identifier $id_{rt}$ to be associated with that specific relationship type, if a key has not yet been assigned to $rt$. Moreover, each node of the social network is equipped with a directory *RKD*, called *relationship key directory*, to store these keys. More precisely, both $v$ and $v'$ store in their directories a tuple $RK_{rt} = (k_{rt}, id_{rt}, rt)$. If, subsequently, nodes $v$ and $v'$ establish new relationships of the same type $rt$ with other nodes, they distribute $RK_{rt}$ also to the new neighbours, so that key $k_{rt}$ can be used also by them when establishing relationships of type $rt$. Therefore, $k_{rt}$ will be associated with relationship type $rt$ for a community (i.e., a subgraph) $SN_{rt} \subseteq \mathcal{SN}$.

To protect relationship information into access control rules, the idea is that access conditions are not maintained in clear into access rules, rather they are encrypted with the keys of the corresponding relationship types. More precisely, suppose that a node $v \in SN_{rt}$ specifies an access rule $\text{AR} = (rid, \{\text{AC}_1, \ldots, \text{AC}_n\})$, and let $rt(\text{AC}_1), \ldots, rt(\text{AC}_n)$ be the relationship types appearing in access conditions $\text{AC}_1, \ldots, \text{AC}_n$, respectively. Each access condition $\text{AC}_i$ is encrypted with the key $k_i$, $i \in [1,n]$, corresponding to the relationship type $rt(\text{AC}_i)$, and then the tuple $(rid, \{(id_1, E_{k_1}(\text{AC}_1)), \ldots, (id_n, E_{k_n}(\text{AC}_n))\})$ is stored into the access rule base of node $v$.

Since all the nodes in $SN_{rt}$ have key $k_{rt}$, they are able to read all the access conditions concerning such relationship type. By contrast, all the other nodes are not able to read the access conditions, and therefore of inferring the existence of a relationship of type $rt$ involving a subset of the network participants.

Suppose now that there exist two communities $SN_{rt}, SN'_{rt} \subset \mathcal{SN}$, such that $SN_{rt} \cap SN'_{rt} = \emptyset$, which use two distinct keys $k_{rt}, k'_{rt}$ for the same type of relationship $rt$. It may then happen that two nodes $v \in SN_{rt}$ and $v' \in SN'_{rt}$ establish a new relationship of type $rt$. In such a case, the two communities are merged, and, consequently, the nodes in $SN_{rt}$ ($SN'_{rt}$) must be authorized to read any access condition concerning relationship $rt$ specified by the nodes in $SN'_{rt}$ ($SN_{rt}$).

In order to address this scenario, we adopt a key agreement and distribution protocol according to which, whenever two nodes $v$ and $v'$ wish to establish a relationship of type $rt$, the corresponding key is determined as follows:

1. If they do not already have a key for such type of relationship, they generate a new key $k_{rt}$; otherwise,

2. If both of them have the same key $k_{rt}$ or if only one of them have a key $k_{rt}$ for such type of relationship, they both use key $k_{rt}$; otherwise,

3. If they have two distinct keys $k_{rt}, k'_{rt}$ for such type of relationship, they exchange keys $k_{rt}, k'_{rt}$, and distribute them to their communities $SN_{rt}$ and $SN'_{rt}$. Then, they use the older for encrypting access conditions. As a result, both the communities $SN_{rt}$ and $SN'_{rt}$ are able to read access conditions concerning relationship type $rt$, specified by one of their nodes.

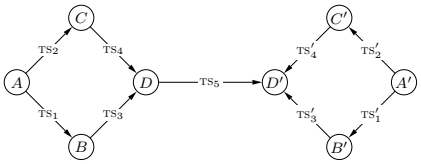


**Figure 3. Two connected subgraphs of a social network, where edges denote the same type of relationship *friendOf*, whereas the labels associated with edges denote the instant TS (TS′) in which the corresponding relationship has been established.**

**Example 3.3 (Relationship Key Distribution)** *Consider the two connected subgraphs $SN, SN'$ of a social network $\mathcal{SN}$ depicted in Figure 3, where $V_{SN} = \{A,B,C,D\}$ and $V_{SN'} = \{A',B',C',D'\}$. We assume that, at a given instant $\text{TS}_0$ ($\text{TS}'_0$), none of the nodes in SN (SN′) participate in a relationship of type friendOf. Then, at instant $\text{TS}_1 > \text{TS}_0$ ($\text{TS}'_1 > \text{TS}'_0$), node A (A′) establishes a relationship of type friendOf with node B (B′). They agree on a relationship key $RK = (k_{friendOf}, id_{friendOf}, friendOf)$ ($RK' = (k'_{friendOf}, id'_{friendOf}, friendOf)$), and they store it into their relationship key directories. When node A (A′), at instant $\text{TS}_2 > \text{TS}_1$ ($\text{TS}'_2 > \text{TS}'_1$), establishes a relationship of type friendOf with node C (C′), since such node does not have a key for relationship friendOf, they use the one formerly agreed by A and B (A′ and B′). The same happens when node B (B′), at instant $\text{TS}_3 > \text{TS}_2$ ($\text{TS}'_3 > \text{TS}'_2$), establishes a relationship of type friendOf with node D (D′). Finally, at instant $\text{TS}_4 > \text{TS}_3$ ($\text{TS}'_4 > \text{TS}'_3$), node C (C′) establishes a relationship of type friendOf with node D (D′). They both have a key for such type of relationship in their relationship key directories, corresponding to the relationship key formerly agreed by A and B (A′ and B′). As a result, we have nodes belonging to two distinct subgraphs which use two distinct relationship keys $RK_{friendOf}$ and $RK'_{friendOf}$. Suppose now that node D, at instant $\text{TS}_5$ (with $\text{TS}_5 > \text{TS}_4$ and $\text{TS}_5 > \text{TS}'_4$), wishes to establish a relationship of type friendOf with node D′. Since they have two distinct keys for the same relationship type, they exchange such keys, and distribute them to SN and SN′, respectively. Nodes D and D′ now have two different keys for the same relationship type. When they need to use*

*them for encrypting access conditions, they will use the older one—i.e., D will use $RK_{friendOf}$ whereas D′ will use $RK'_{friendOf}$.*

Having relationship certificates and access rules not public implies to modify the access control procedure described in Section 2. In particular, one of the main differences is related to the computation of trust. In fact, since relationship certificates are now not accessible by the central node *CN*, we cannot rely on it in order to perform such task. Therefore, when computing trust, we do not consider all the chains of certificates corresponding to the relationship(s) between the requestor and the node(s) specified by the access rule(s), since one or more certificates in such chains may be not accessible to the requestor itself. Rather, we apply a solution according to which trust is computed wrt a single certificate chain satisfying the access rules associated with the requested resource.

In brief, the access control procedure works as follows:

1. when an access rule AR is returned to a requestor, if such node is not able to decrypt one or more of the corresponding access conditions $AC(\text{AR})$, it is aware to be not authorized to access the requested resource, and then it does not carry on the access control procedure; otherwise,
2. the requestor sends to the central node the set of certificate key identifiers stored in its certificate key directory, thus obtaining the corresponding set *C* of encrypted relationship certificates;
3. the requestor decrypts the certificates in *C*, builds the corresponding set *CC* of certificate chains (if any), and computes the depth and trust level of each of them; then,
4. for each condition $\text{AC} \in AC(\text{AR})$, the requestor selects one of the certificate chains in *CC* satisfying AC (if any), and returns both the chain and the proof of AR to the resource owner.

**Example 3.4 (Access Control Enforcement)** *Consider the social network in Figure 1, and suppose that the nodes in the network have the following relationship keys:*

- $(k_0, id_0, friendOf)$*: A, B;*
- $(k_1, id_1, friendOf)$*: C, D, E, F, G;*
- $(k_2, id_2, colleagueOf)$*: A, B, D, E, G.*

*Suppose that D(avid) owns a resource rsc on which the following encrypted access rule applies:* $(rid, \{(id_1, E_{k_1}(D, friendOf, 2, 0.3))\})$*. If A(lice) requests access to such resource, she will not be able to decrypt the access condition, since she does not have key* $k_1$*. Since Alice is not authorized to access resource rsc, rule encryption has two main advantages: on the one hand, the information about the relationship in which David participates is not released to Alice; on the other hand, the access control procedure is not performed uselessly. By contrast, if G(reg), who is authorized to access resource rsc, submits the same access request, he will be able to decrypt the access condition and to perform the access control procedure. This applies also to F(rank), even though he is not authorized to access resource rsc, since his relationship with David has a trust level equal to* 0.2*.*

## 4 Certificate Revocation

In the previous section, we illustrated how information about the existing relationships can be protected, considering a scenario where the number of edges in the graph increases. Nonetheless, in a social network, users are able not only to establish new relationships, but also to revoke existing ones.

Certificate revocation impacts access control, in that, when access control is performed, it is necessary to verify whether a certificate has been revoked in order to avoid unauthorized accesses to resources. In fact, a node may store all the certificates retrieved from the central node, and then use them to give proofs of access rules. This has the advantage of reducing the number of requests submitted to the central node, but the consequence is that the node owning a resource may accept as proof certificates referring to relationships that do not exist any more. If certificate revocation is not properly managed, we may also have unauthorized distribution of certificate keys. Consider, for instance, the graph in Figure 2(b), and suppose that it depicts the state of the network at a given instant $\text{TS}_0$. Then, at a given instant $\text{TS}_1 > \text{TS}_0$, *D* revokes the certificate to *A*, concerning the relationship of type *rt*, to which the following distribution condition is associated: $(A, rt, 1)$. Since this condition has a maximum depth greater than 0, *A* will distribute the certificate key to its neighbours, when establishing a relationship of type *rt*, even after $\text{TS}_1$, although they are no longer authorized to access the certificate. Relationship keys have a similar problem. In fact, once a node has received them, it can read access rules even though, in a subsequent instant, it will be not authorized to do that, because one or more certificates have been revoked.

In order to take into account these issues, we adopt a strategy according to which, whenever a certificate is revoked, this is notified to the central node *CN*, which removes such certificate from the central certificate directory $\mathcal{CD}$, and stores the corresponding key identifier into a *certificate revocation list* $\mathcal{CRL}$, hosted by the central node itself. Then, *CN* sends a message to all the nodes in the network, informing them that the certificate corresponding to

a given key identifier has been revoked. Consequently, all the nodes remove from their directories the certificate and certificate key, respectively, corresponding to that key identifier. Since such notification may be delayed, an additional check is performed by the nodes in the network before accepting a proof or distributing a certificate key. More precisely, whenever a node receives from another node a certificate chain, it queries the certificate revocation list in order to verify whether one or more certificates in the chain have been revoked. Similarly, before distributing a certificate key, a node verifies whether the corresponding key identifier is in $\mathcal{CRL}$. Note that the strategy illustrated above addresses security issues concerning access control and certificate protection, but it does not apply to relationship keys, which are not directly associated with a certificate, and thus cannot be easily tracked. Indeed, it may happen that a user owning the key corresponding to a given relationship type may exploit it even if it is no longer involved in relationships of that type. Consequently, that user may be aware, by issuing proper access requests, that a member has established a relationship of that type, even if he/she may not know with whom it has been established. We plan to deal with this issue in future work; however, such risk can be limited by periodically re-generating relationship keys.

## 5 Conclusions & Future Work

Web-based social networks store and make publicly available a variety of personal information concerning registered users, which may be exploited for purposes different from the intended ones. In order to address this issue, in this paper we have proposed an approach aimed at enforcing privacy protection on the relationships existing between social network users. We are currently implementing our model into a prototype system making use of the most recent social network technologies. More precisely, the prototype system is built as a set of Web services, communicating through SOAP interfaces, where information concerning relationship certificates and both certificate and relationship keys is stored in RDF format. The central node is built as a social network management system, which supports the basic functionalities of a social network. By contrast, peripheral nodes consist of a client-side and a server-side component. The former is built as an extension for the Firefox browser, and it is in charge of generating and delivering certificates, certificate and relationship keys, submitting access requests, and of carrying out the access control procedure. By contrast, the server-side component is in charge of storing and managing relationship certificates, certificate and relationship keys, and of controlling access to resources.

We plan to extend the work reported in this paper along several directions. First, based on the prototype we are developing, an extensive performance evaluation will be carried out in order to assess the feasibility and scalability of the devised strategies. Then, we will investigate more sophisticated mechanisms for managing updates to users' relationships and distribution rules, with a particular attention to revocation of relationship keys. Finally, we plan to extend our model, on one hand, by taking into account other possible parameters relevant to control information dissemination (such as the number of relationships a node participates in), and, on the other hand, by enforcing both positive and negative access/distribution rules, and by denoting resources to be protected not only by their identifiers, but also on the basis of their characteristics. In fact, in the current version of our model, access/distribution rules can denote only the authorized users, whereas they do not allow the specification of a policy stating that a given resource can be accessed, for instance, by the users who are friends but not colleagues of mine. For the same reason, if we wish to hide a specific portion of a resource, this can be done only by specifying access rules for all the other ones. Finally, content-based access control can be used to further reduce the number of rules to be specified and to simplify their management, based on the principle that resources sharing similar characteristics should have similar protection requirements.

## References


[1] S. B. Barnes. A privacy paradox: Social networking in the United States. *First Monday*, 11(9), Sept. 2006. Available at: `http://www.firstmonday.org/issues/issue11_9/barnes`.

[2] B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In *Proc. of the OTM Workshops 2006*, number 4278 in LNCS, pages 1734–1744. Springer-Verlag, 2006.

[3] L. Ding, L. Zhou, T. Finin, and A. Joshi. How the Semantic Web is being used: An analysis of FOAF documents. In *Proc. of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, page 113.3. IEEE CS, 2005.

[4] J. A. Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, Graduate School of the University of Maryland, College Park, 2005. Available at: `http://trust.mindswap.org/papers/GolbeckDissertation.pdf`.

[5] S. Staab, P. Domingos, P. Mika, J. Golbeck, L. Ding, T. W. Finin, A. Joshi, A. Nowak, and R. R. Vallacher. Social networks applied. *IEEE Intelligent Systems*, 20(1):80–93, 2005.

[6] D.-W. Wang, C.-J. Liau, and T. sheng Hsu. Privacy protection in social network data disclosure based on granular computing. In *Proc. of the 2006 IEEE International Conference on Fuzzy Systems (HICSS'05)*, pages 997–1003. IEEE CS, 2006.

[7] D. J. Weitzner, J. Hendler, T. Berners-Lee, and D. Connolly. Creating a policy-aware Web: Discretionary, rule-based access for the World Wide Web. In E. Ferrari and B. Thuraisingham, editors, *Web & Information Security*, pages 1–31. IDEA Group Publishing, Hershey, PA, 2006.