# Enforcing Relationships Privacy through Collaborative Access Control in Web-based Social Networks

*(Invited Paper)*

Barbara Carminati and Elena Ferrari
DICOM, Università degli Studi dell'Insubria
Varese, Italy
Email: {barbara.carminati,elena.ferrari}@uninsubria.it

***Abstract*—Web-based Social Networks (WBSNs) are today one of the hugest data source available on the Web and therefore data protection has become an urgent need. This has resulted in the proposals of some access control models for social networks (e.g., [1,4,5,15,16]). Quite all the models proposed so far enforce a *relationship-based* access control, where the granting of a resource depends on the relationships established in the network. An important issue is therefore to devise access control mechanisms able to enforce relationship-based access control by, at the same time, protecting relationships privacy. In this paper, we propose a solution to this problem, which enforces access control through a collaboration of selected nodes in the network. We exploit the ElGamal cryptosystem [11] to preserve relationship privacy when relationship information is used for access control purposes.**

## I. Introduction

The last few years have witnessed the explosion of Web-based Social Networks (WBSNs). Just to give an example, Facebook (http://www.facebook.com) now claims to have more than three hundred million of active users.[1] The opportunities made available by WBSNs in terms of information sharing and knowledge management are terrific in that WBSNs 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. However, the availability of this huge amount of information within a WBSN is both an opportunity and a danger w.r.t. user privacy and confidentiality requirements. Recently, social network users have started to become more and more aware of the risk of the exposure of their personal information and resources trough social networking services, as witnessed by the recent complaints received by Facebook for the use made by some of its services of user personal data [2, 6]. These events have animated several online discussions about privacy in social networkings, and government organizations have started to seriously consider this issue [7, 8, 12, 14].

To partially answer users concerns, some social networks, e.g., Facebook and Videntity (http://videntity.org), have recently started to enforce quite simple protection mechanisms. Additionally, some research proposals have recently appeared [1, 4, 5, 15, 16], able to provide users with more advanced tools to customize and enforce their privacy and confidentiality requirements. One of the common characteristics of the access control models proposed so far is that they enforce *relationship-based access control*. According to this paradigm, the release of a resource to a user is related to the existence of a specific path in the network, modelling the existence of a direct or indirect relationship of a particular type (e.g., friend-of, colleague-of) between the resource requestor and another user in the network, not necessarily the resource owner. Since in some social networks, users can specify how much they trust other users, by assigning them a trust level, some models (e.g., [1, 4]) use also the trust level of the relationships as a further parameter on which access control is based.

Relationship-based access control poses the challenges of protecting relationship information and their trust level during access control enforcement. Indeed, on the one hand, relationship information is needed to decide whether an access request should be granted or not but, on the other, users may have some concern about the release of their personal relationships. It is very common in many application scenarios that a user would like to keep private the fact that he/she has a relationship of a given type with a certain user.

To cope with such requirements, we believe that the naive solution of relying on a trusted reference monitor hosted by the Social Network Management System (SNMS) is not suitable for the WBSN scenario, because it implies to totally delegate to the SNMS the administration of user data/relationships and access control policies and therefore to fully trust it. It is thus necessary to investigate alternative ways of performing access control w.r.t. the traditional centralized one. Therefore in [3] we have proposed a protocol that, starting from the access control model presented in [4], enforces access control through the collaboration of selected nodes in the network. The collaboration is started by the resource owner, on the basis of the access rules regulating the access to the requested resource. Relationships are encoded through certificates and their protection requirements are expressed through a set of *distribution rules*, which basically state who can be aware of the relationship. The aim of the collaboration is to provide the owner with a path, proving that the requestor has the

[1] http://www.facebook.com/press/info.php?statistics.

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339*

relationship required to gain access to the resource. Since each node taking part in the collaboration is aware of the relationships existing among the other nodes taking part in the process, the process is driven by the specified distribution rules: a node is invited to collaborate only if it satisfies the distribution rules of the other nodes taking part in the collaboration. Encryption and signature techniques have been used to avoid trust levels disclosure and forgery, as well as to make a node able to verify the correct enforcement of distribution rules.

However, the protocol proposed in [3] has the following shortcomings. First, it requires the specification and enforcement of distribution rules. Second, the encryption and signature techniques used in [3] are not meant to avoid that a node discloses relationships information to nodes which are not authorized by distribution rules. Rather, they make each node in the path able to verify whether the previous nodes in the path have correctly enforced the distribution rules.

A protocol similar to [3] has been proposed in [10]. The main contribution of [10] w.r.t. [3] is to exploit the multiplicative property of ElGamal encryption [11] to make users participating in the collaborative process (i.e., users with a relationship in the traversed path) able to collaboratively compute an anonymous trust value of the traversed path. However, this collaborative process does not take in account the depth of the traversed path, as such this can not be used as a proof for a relationship-based access control enforcing rules with constraints on the depth and type of the required relationships.

To overcome these drawbacks, in this paper, we propose an alternative protocol to perform collaborative access control that does not require the specification and enforcement of policies to protect relationship disclosures. Rather, the protocol makes use of the homomorphic properties of the ElGamal cryptosystem to build an *anonymous path*. An anonymous path is a path where the path participants are not disclosed but that makes, at the same time, the resource owner able to verify that the path matches the access rules associated with the requested resource. A similar data structure is used to make the resource owner able to compute relationship trust in a private way, that is, by not disclosing the trust of each relationship in the path.

The remainder of this paper is organized as follows. Next section introduces some backgrounds on the ElGamal cryptosystem and the access control model proposed by us in [4]. Section III presents our collaborative access control protocol, whereas Section IV presents its security analysis. Finally, Section V concludes the paper and outlines directions for future work.

## II. Background

### A. ElGamal

The ElGamal [11] encryption system is a public key encryption algorithm based on the Diffie-Hellman [9] key agreement. ElGamal encryption can be defined over any cyclic group $\mathbb{G}$. Its security depends upon the difficulty of the Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}$ related to computing discrete logarithms. It is based on the following components:

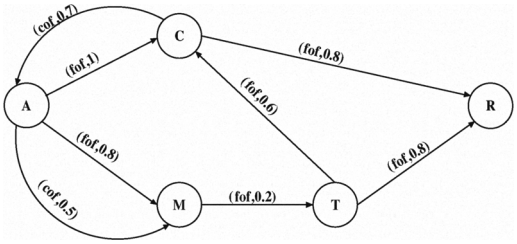


Fig. 1. A portion of a WBSN

- **Configuration.** A cyclic subgroup $\mathbb{G} = \langle g \rangle$ of $\mathbb{Z}_p$ is chosen generated by $g$, with order $q$, where $q|p-1$ ($q$ has to divide $p-1$) for two prime numbers $p$ and $q$. $p$, $q$ and $g$ are public.
- **Key** Choose $sk \in \mathbb{Z}_q^*$ at random, and publish $pk = g^{sk} mod\ p$.
- **Encryption.** Encrypt message $m \in \mathbb{G}$. Take $r \in \mathbb{Z}_q^*$ randomly, compute $R_r = g^r mod\ p$ and $s = m \cdot pk^r mod\ p$. The ciphertext is $E_r(m) = (R, s)$. $r$ is secret.
- **Decryption.** Given the secret key $sk$ and the ciphertext $E_r(m) = (R, s)$, the plain text is given by: $m = \frac{s}{R^{sk}} mod\ p$.

ElGamal encryption system is multiplicatively homomorphic. That is, given $E_{r_1}(m_1) = (g^{r_1}, m_1 \cdot pk^{r_1})$ and $E_{r_2}(m_2) = (g^{r_2}, m_2 \cdot pk^{r_2})$ two encryptions of $m_1$ and $m_2$, we can obtain an encryption of $m_1/m_2$ and $m_1 \times m_2$ as:

$$\frac{E_{r_1}(m_1)}{E_{r_2}(m_2)} = (\frac{g^{r_1}}{g^{r_2}}, \frac{m_1 \cdot pk^{r_1}}{m_2 \cdot pk^{r_2}}) = (g^{r_1 - r_2}, \frac{m_1}{m_2} \cdot pk^{r_1 - r_2})$$

$$E_{r_1}(m_1) \times E_{r_2}(m_2) = (g^{r_1} \cdot g^{r_2}, m_1 \cdot pk^{r_1} \cdot m_2 \cdot pk^{r_2}) = (g^{r_1 + r_2}, m_1 \times m_2 \cdot pk^{r_1 + r_2}).$$

### B. The reference access control model

Before briefly presenting the access control model proposed in [4], we introduce how we model a WBSN. In what follows, we model a WBSN $\mathcal{SN}$ as a tuple $(V_{\mathcal{SN}}, E_{\mathcal{SN}}, RT_{\mathcal{SN}}, T_{\mathcal{SN}}, \phi_{E_{\mathcal{SN}}})$, where $V_{\mathcal{SN}}$ and $E_{\mathcal{SN}}$ are, respectively, the nodes and edges of a graph, $RT_{\mathcal{SN}}$ 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 edge $e \in E_{\mathcal{SN}}$ a relationship type $rt \in RT_{\mathcal{SN}}$ and a trust level $t \in T_{\mathcal{SN}}$.

Given an in/direct relationship of type $rel$ between nodes $v$ and $v'$, the trust level of such relationship denotes how much $v$ trusts $v'$ w.r.t. relationship $rel$. In this paper, for simplicity, the trust level associated with a path is computed by multiply the trust levels associated with all the edges in the path, even if other formulas to compute the trust level can be used as well (e.g., [13]).

The model presented in [4] protects each WBSN resource $rsc$ trough a set of *access rules*, specified by the resource owner. Each access rule is a pair $(rid, AC)$, where $AC$ is

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339*

a set of *access conditions* that need to be all satisfied in order to get access to $rid$. An access condition is a tuple $(v, rt, d_{max}, t_{min})$, 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. The set of access rules protecting the resources of a node are stored and managed locally.

*Example 1:* A simple example of WBSN is depicted in Figure 1. In the figure, the initial node of an edge is the node which established the corresponding relationship. Labels associated with edges denote, respectively, the type and trust level of the corresponding relationship. So for instance, `A(lex)` is friend-of ($fof$) of both `C(arl)` and `M(ark)`. However, `A` trusts `C` more than `M`. `A` and `T(ed)` are indirect friends due to the $fof$ relationships existing between `A` and `M` and `M` and `T`. `C` is also a colleague-of ($cof$) `A`, and the trust assigned to this relationship by `C` is 0.7. Now suppose that `A` wishes to release his resource `res` to his direct and indirect friends of maximum depth four with a trust level of at least 0.5. To specify such requirements he can specify the following access rule `AR = (res,{(A,fof,4,0.5)})`.

In this paper, we constrain the access conditions contained into an access rule by assuming that $v$ can be only equal to the owner of the resource to be protected. As it will be clarified in what follows, this assumption makes the resource owner able to start the collaborative process needed to answer an access request. Additionally, this is not a too strong restriction because it is very common that most of the access control requirements in a $\mathcal{SN}$ are expressed by a resource owner in terms of the relationships it holds with other nodes in the network, rather than in terms of relationships in which it is not involved.

Moreover, in order to adopt the ElGamal cryptosystem, we assume the presence of a trusted party TP, which only implements key management functionalities. Note that the presence of a TP is not equivalent to have a centralized solution, i.e., a central node hosting the access control reference monitor. The encryption keys generated by the TP are only used to generate the anonymized path (see Section III). In contrast, the TP is not be able to access users resources nor their policies.

## III. Collaborative access control

In this section we illustrate our proposal for enforcing access control while preserving WBSN relationship/trust privacy.

### A. Overview

Access control is enforced through a *collaboration* among selected nodes in the $\mathcal{SN}$. The collaboration is needed to prove the owner that the node requesting a resource satisfies the requirements (in terms of relationships it holds and corresponding trust levels and relationship depths) stated by the owner access rules. If the result of the collaboration is the identification of a path with the requirements stated by the owner access rules, then the access is granted. Otherwise, it is denied. The collaborative process is started by the resource owner, on the basis of the access rules regulating the access to the requested resource. The owner contacts its direct neighbours with which it has established a relationship of the type required by one of the access control rules associated with the requested resource. It asks them whether they have a relationship of the required type with the requestor node. If a resource is protected by more than one access rule, the process is iterated till the access can be granted or till all the access rules protecting the resource have been considered. For simplicity, in the following, we assume access rules consisting only of one access condition. The protocols can be easily extended by iterating the described operations for all the access conditions contained into an access rule. Once a node different from the requestor receives a request for collaboration, it propagates the request to those of its neighbours with which it has established a relationship of the type required in order to get access to the resource. This process is iterated until a node having a relationship of the required type with the requestor is reached, or until the request can no longer be propagated along the network.

To verify whether an access can be granted or not the owner must be provided not only with a path of the required type and depth, but it also must know the path trust level. Therefore, when propagating the request for collaboration, a node forwards also the trust level of its relationship. If relationships information and associated trust levels gathered during the collaborative process are sent in clear to all the collaboration participants, then all the nodes taking part to the collaboration are aware of all relationships and corresponding trust values belonging to the traversed path, which obviously breaches relationships privacy. To overcome this problem, we propose a *private collaborative process*, during which information on the traversed path is collected in an anonymized way. This is achieved by means of two data structures, namely, *anonymous relationship path* and *anonymous trust value* (see Section III-B). By exploiting the ElGamal cryptosystem (see Section II), these data structures make the owner able to verify whether the traversed path and its trust level satisfy the access rules associated with the requested resource, without having direct access to relationships information and associated trust levels. Moreover, they prevent each intermediate node taking part to the collaboration to acquire information about path relationships and associated trust levels. Since also the resource owner will have no access on clear-text data, the ElGamal encryption scheme has to be configured by a third party, i.e, the TP entity. In particular, each time a resource owner receives an access request, it contacts the TP that, as a consequence, configures two different pairs of ElGamal keys, say ($sk$,$pk$) and ($sk'$,$pk'$), with relative public parameters $p, q, g$ and $p', q', g'$. As it will be described in Section III-C, these keys will be used to collaboratively generate *anonymous relationship path* and *anonymous trust value*.

In the next sections, we will first present the devised data structures. Then, we describe the access control protocol.

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339*

| | $E_{no}$ | $E_{nd}$ | $E_{rt}$ | $Diff_{no}$ | $Diff_{rt}$ |
|---|---|---|---|---|---|
| $Arel_1$ | $(g^{r1} mod\ p, A \cdot pk^{r1} mod\ p)$ | $(g^{r2} mod\ p, M \cdot pk^{r2} mod\ p)$ | $(g^{r3} mod\ p, fof \cdot pk^{r3} mod\ p)$ | 0 | 0 |
| $Arel_2$ | $(g^{r4} mod\ p, M \cdot pk^{r4} mod\ p)$ | $(g^{r5} mod\ p, T \cdot pk^{r5} mod\ p)$ | $(g^{r6} mod\ p, fof \cdot pk^{r6} mod\ p)$ | $r4 - r2$ | $r6 - r3$ |
| $Arel_3$ | $(g^{r7} mod\ p, T \cdot pk^{r7} mod\ p)$ | $(g^{r8} mod\ p, C \cdot pk^{r8} mod\ p)$ | $(g^{r9} mod\ p, fof \cdot pk^{r9} mod\ p)$ | $r7 - r5$ | $r9 - r6$ |

TABLE I
AN EXAMPLE OF ANONYMOUS RELATIONSHIP PATH

### B. Anonymized data structures

As stated in the previous section, access control enforcement is obtained through a collaboration among nodes in the $\mathcal{SN}$. The collaboration has the aim of identifying a *path* in the $\mathcal{SN}$ satisfying the requirements stated by the access rules specified for the requested resource. The notion of path is formalized as follows.

*Definition 3.1 (Relationship path):* [3] Let $\mathcal{SN} = (V_{\mathcal{SN}}, E_{\mathcal{SN}}, RT_{\mathcal{SN}}, T_{\mathcal{SN}}, \phi_{E_{\mathcal{SN}}})$ be a WBSN. Given a relationship type $rt \in RT_{\mathcal{SN}}$, a relationship path $P$ in $\mathcal{SN}$ of type $rt$ is an ordered list $\langle rel_1, \ldots, rel_k \rangle$, such that for each $rel_j \in P$, $j \in \{1, \ldots, k\}$, (1) $rel_j = (no, nd)$, where $no, nd \in V_{\mathcal{SN}}$, (2) there exists an edge $(no, nd) \in E_{\mathcal{SN}}$ labeled with $rt$, and (3) $rel_j.no = rel_{j-1}.nd$, $\forall j \in \{2, \ldots, k\}$.

Hereafter, given a relationship path $P$, we denote with $P_{depth}$ and $P_{rt}$ the depth and the relationship type of the path, respectively. Moreover, we use the dot notation to refer to specific components within a tuple.

In general, a relationship path and the corresponding relationship type are not enough to prove that a given node satisfies an access condition. Indeed, access conditions pose constraints also on the associated trust value.[2] More precisely, to verify whether a relationship path $P$, built during the collaborative process, satisfies an access condition $AC$, the owner needs to verify the following characteristics of the path: $(a)$ whether all relationships in $P$ have type equal to the one required in $AC$; $(b)$ whether the first node (resp. last node) of the first relationship (resp. last relationship) in $P$ is equal to the owner (resp. the requestor); $(c)$ whether $P$'s depth is less than or equal to the maximum depth required in $AC$; $(d)$ whether the trust value of $P$ is greater than or equal to the minimum trust value required in $AC$. Additionally, it must verify whether the path satisfies Definition 3.1, that is, $(e)$ whether all relationships $rel_j$ in $P$ have the node whose established it equal to the node with which the preceding relationship $rel_{j-1}$ has been established, that is, $rel_j.no = rel_{j-1}.nd$.

Therefore to protect relationships privacy and associated trust levels during the collaborative process, we have devised two separate data structures, namely *anonymous relationship path* and *anonymous trust value*, formally defined in what follows.

*Definition 3.2 (Anonymous Relationship Path):* Let $pk$ and $sk$ be a public and secret key, generated according to the ElGamal cryptosystem, and let $p$,$q$, and $g$ be the corresponding parameters. Let $P$ be a relationship path for a social network $\mathcal{SN}$. The *Anonymous Relationship Path* $Anonymous\_path(P)$, generated for path $P$ is a set containing, for each $rel_j$ in $P$, $j \in \{1, \ldots, P_{depth}\}$, an *anonymous relationship* of the form:

$$Arel_j = \langle E_{no},\ E_{nd},\ E_{rt},\ Diff_{no},\ Diff_{rt} \rangle$$

where:

- if $j \neq 1$, $E_{no}$ is the encryption of $rel_j.no$ using the random number $r\_no_j \in \mathbb{Z}_q^*$, that is, $E_{no} = (R_{r\_no_j}, S\_no_j) = (g^{r\_no_j} mod\ p, rel_j.no \cdot pk^{r\_no_j} mod\ p)$; $E_{no} = rel_j.no$, otherwise;
- if $j \neq P_{depth}$, $E_{nd}$ is the encryption of $rel_j.nd$ using the random number $r\_nd_j \in \mathbb{Z}_q^*$, that is, $E_{nd} = (R_{r\_nd_j}, S\_nd_j) = (g^{r\_nd_j} mod\ p, rel_j.nd \cdot pk^{r\_nd_j} mod\ p)$; $E_{nd} = rel_j.nd$, otherwise;
- $E_{rt}$ is the encryption of the relationship type $rt$ of the edge $(rel_j.no, rel_j.nd) \in E_{\mathcal{SN}}$ using the random number $r\_rt_j \in \mathbb{Z}_q^*$, that is, $E_{rt} = (R_{r\_rt_j}, S\_rt_j) = (g^{r\_rt_j} mod\ p, rel_j.rt \cdot pk^{r\_rt_j} mod\ p)$;
- for $j \in \{2, \ldots, P_{depth}\}$, $Diff_{no}$ is the difference $r\_no_j - r\_nd_{j-1}$; it is 0 otherwise;[3]
- for $j \in \{2, \ldots, P_{depth}\}$, $Diff_{rt}$ is the difference $r\_rt_j - r\_rt_{j-1}$; it is 0 otherwise;

*Example 2:* Let us consider the WBSN in Figure 1, and the relationship path $A \rightarrow M \rightarrow T \rightarrow C$. The corresponding anonymous relationship path is shown in Table I, where each row represents a different anonymous relationship, whereas columns represent its components.

As it will be described in Section III-C, the *Anonymous Relationship Path* data structure is used to verify properties $(a), \ldots, (e)$, and $(e)$. However, it is not enough to check the trust value of the relationship path. To overcome this limitation, we propose another anonymized data structure, i.e., *Anonymous Trust Value*, whose definition is inspired by the work reported in [10].

[2]We recall that, in case of indirect relationships, we compute the corresponding trust value as the multiplication of the trust values associated with each relationship $rel_j$ in the path $P$ modelling the indirect relationship, j $\in \{1, \ldots, P_{depth}\}$.

[3]Note that these and the following differences are needed to verify the correctness of the retrieved anonymous path (see Section III-C for more details).

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339

*Definition 3.3 (Anonymous Trust Value):* Let $pk'$ and $sk'$ be a public and secret key generated according to the ElGamal cryptosystem, and let $p'$,$q'$, and $g'$ be the corresponding parameters. Let $P$ be a relationship path of type $rt$ for a social network $\mathcal{SN}$. Let $rel_j.t$ be the trust value associated with relation $rel_j \in P$. The *Anonymous Trust Value* of $P$ is defined as follows:

$$E_{r_1}(rel_1.t) \times E_{r_2}(rel_2.t) \times \ldots \times E_{P_{depth}-1}(rel_{Pdepth-1}.t) \times E_{P_{depth}}(rel_{_{depth}}.t)$$

where

$$E_{r_j}(rel_j.t) = (R_{r_j}, S\_r_j) = (g^{r_j} mod\ p, rel_j.t \cdot pk^{r_j} mod\ p),$$

with $r_j \in \mathbb{Z}^*_{q'}$, $j \in \{1, \ldots, P_{depth}\}$.

### C. Algorithms

Collaborative access control is enforced by a process initialized and handled by the algorithm in Figure 2. According to this process, upon receiving an access request the owner contacts the TP by sending the access request (step 1). As a consequence, the TP generates two sets, namely $par$, and $par'$, containing the ElGamal parameters and the public keys. These sets, together with just one of the corresponding secret key, are sent back to the owner (steps 2-4). During the collaborative process these two sets are passed to all the participants, since they are used to generate the anonymous relationship path (see Definition 3.2) and the anonymous trust value (see Definition 3.3), respectively. Then, the owner retrieves from its Policy Base the access rules regulating the release of the requested resource (step 5).[4]

Then, in step 7, the owner identifies the set of neighbours with which it has established a relationship of the type $rt$ required by the access condition contained in the considered access rule. It iteratively considers each node in this set (step 9). For each one of these nodes, the algorithm generates two random numbers $r_{nd}$ and $r_{rt}$ from parameters $par$ (step 9.b) and a random number $r_t$ from parameters $par'$ (step 9.c). The first is used to compute the first anonymous relationship, i.e., $Arel_1$ (step 9.e). In contrast, $r_t$ is used to encrypt the trust value of the relationship the owner has with the considered node. The algorithm sends then a message to the considered neighbour node to start the collaboration process. This contains the two sets of parameters $par$, $par'$, the requestor identifier, the needed relationship type $AC.rt$, and the anonymous relationship path and anonymous trust value computed so far. The message is sent by function *SendCollReq()* in step 9.m. Moreover, the message contains the random numbers $r_{nd}$ and $r_{rt}$ used to generate $E_{r_{nd}}(\overline{n})$ and $E_{r_{rt}}(rt)$, that is, the ElGamal encryption of the receiver node and of the requested relationship type. These random numbers are needed by the neighbour node to compute $Arel_2.Diff_{no}$ and $Arel_2.Diff_{rt}$. Once the message has been sent, the algorithm waits for the node reply, which consists of the value 0, if no path can be found, or the number of identified paths, otherwise. In case *SendCollReq()* returns a non zero value, for each identified path a separate message is sent to the owner by the requestor (see procedure *path_builder()* in Figure 3 explained next). More precisely, each message contains the anonymous relationship path and anonymous trust value generated during the collaborative process by the nodes belonging to the path connecting the owner to the requestor. The message also contains the $Trust_r$ variable, storing the sum of random numbers used by each collaborative node to encrypt the trust values (see procedure *path_builder()* in Figure 3). The algorithm elaborates each received message (step 9.n.ii). It first verifies whether the anonymous relationship path corresponds to a valid path (step 9.n.ii.2). This is done by function *check ()* in Figure 4, explained later. If the check succeeds, the algorithm computes the depth of the anonymous path, by counting the number of items in the path, that is, the number of anonymous relationships. Then, by using the private key associated with parameters $par'$ and the $Trust_r$ value, it decrypts the anonymous trust value (step 9.n.ii.3.b). The algorithm checks whether the depth and the decrypted trust value satisfy the constraints stated in the access rule, as well as, whether the first and last node of the anonymous path are the owner and the requestor, respectively (step 9.n.ii.3.e). In this case, the access is granted. Otherwise the process is iterated on the next received message, until there are no more message to be processed. Then, if the access has not been granted, the collaboration is requested to the next node in the set identified in step 7, until the access is granted or all the nodes in the set have been contacted, without finding a suitable path. In this case, the algorithm denies the access and sends the error message received by the *check()* function to the requestor (step 10).

Once a node $n$ receives a request for collaboration, it executes procedure *path_builder()*, presented in Figure 3. The aim of this procedure is to elaborate the received message and send the request of collaboration to $n$'s neighbours. The procedure starts by initializing variable $TOT\_FOUND$ to zero (step 2). This variable is used to store the number of identified paths. Procedure *path_builder()* then verifies whether node $n$ is the requestor stated in the message. In this case, the requestor forwards the received message directly to the owner. Note that, similar to other nodes participating to the collaborative process, the requestor is not able to decrypt information in the anonymous relationship path and anonymous trust value, as it is not provided with the corresponding private keys. In case $n$ is not the requestor, the procedure identifies the nodes with which $n$ has a relation of the type $rt$ specified in the received message (step 3.b). If this set is empty, *path_builder()* halts by returning 0 to the sender, since no other path can be found (step 3.c.i-ii). Otherwise, for each neighbour node $\overline{n}$, it generates the random numbers to be used to compute the anonymous relationship ($Arel_j$) and to encrypt the corresponding trust value. *path_builder()* also checks whether $\overline{n}$ is the requestor. If this is the case, a path is found, and, to keep trace of it, the procedure sets variable $flag$ to 1 (step 3.d.vi.1).

[4]For simplicity, the algorithm assumes a single access rule consisting only of one access condition. The algorithm can be easily extended to more access rules, each one consisting of more than one access condition, by simply iterating the steps described in the algorithm.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339

**The collaborative access control protocol**
INPUT: An access request $(req, res)$ submitted to node $own$ by $req$
OUTPUT: $res$, if $req$ satisfies the access control requirements of $own$,
an access denied message, otherwise.
1 $own$ submits to $TP$ the access request $(req, res)$
2 Let $par = (p, q, g, pk)$ be the first set of ElGamal public parameters returned by $TP$
3 Let $par' = (p', q', g', pk')$ be the second set of ElGamal public parameters returned by $TP$
4 Let $sk'$ be the secret key returned by $TP$ corresponding to $par'$
5 $own$ retrieves from its Policy Base the access rule $AR$ associated with resource $res$
6 Let $AC$ be the access condition in $AR$
7 Let $C\_nodes$ be the set of nodes with which $own$ holds a relationship of type $AC.rt$
8 Let $TOT\_error$ be initialized empty
9 **Foreach** $\overline{n} \in C\_nodes$:
a $TOT\_FOUND$:= 0
b Generate random $r_{nd}, r_{rt} \in \mathbb{Z}^*_{par.q}$
c Generate random $r_t \in \mathbb{Z}^*_{par'.q}$
d Generate the ElGamal encryption $E_{r_{nd}}(\overline{n})$ and $E_{r_{rt}}(rt)$
e Let $Arel_1 = \langle own,\ E_{r_{nd}}(\overline{n}),\ E_{r_{rt}}(rt),\ -,\ - \rangle$
f $AnonymousPath := Arel_1$
g Let $t$ be the trust value of the relationship of $own$ with $\overline{n}$
h Generate the ElGamal encryption $E_{r_t}(t)$
i $AnonymousTrust := E_{r_t}(t)$
l $Trust_r := r_t$
m $TOT\_FOUND$ :=$SendCollReq(\{par, par', req, AC.rt, AnonymousPath, AnonymousTrust, r_{nd}, r_{rt}, Trust_r\}, \overline{n})$
n **If** $TOT\_FOUND \neq 0$ :
i Let $TOT\_msg\_rec$ be the set of the $TOT\_FOUND$ messages received from $req$,
where each message contains the $AnonymousPath$ and $AnonymousTrust$ data structures
ii **Foreach** $msg \in TOT\_msg\_rec$:
1 Let $error$ be $NULL$
2 $error$ :=$check(msg)$
3 **If** $error = NULL$
a Let $depth$ be the number of items in $msg.AnonymousPath$
b Let $trust$ be the decryption of $msg.AnonymousTrust$ with $sk'$ and $msg.Trust_r$
c Let $\overline{own}$ be $msg.AnonynousPath.Arel_1.E_{no}$
d Let $\overline{req}$ be $msg.AnonynousPath.Arel_{depth}.E_{nd}$
e **If** $trust \geqslant AC.t_{min}$, $depth \leqslant AC.d_{max}$, $own$=$\overline{own}$ and $req$=$\overline{req}$
i **Return** $(res, msg.AnonymousPath)$
**EndIf**
**Else**
f $TOT\_error$:=$TOT\_error \bigcup error$
**EndIf**
**EndFor**
**EndIf**
**EndFor**
10 **Return** (denied, $TOT\_error$)

Fig. 2. The collaborative access control protocol

Then, it updates the anonymous relationship path contained in the received message by appending $Arel_j$. Note that, according to Definition 3.2, in case the neighbour is the requestor, the second component is not encrypted. Moreover, to update the anonymous trust value contained in the received message, in step 3.d.x the procedure multiplies it by the ElGamal encryption of the trust value of the relationship with the neighbour. The received $Trust_r$ value is updated too, by adding to it the random number used to encrypt the trust value (step 3.d.xi). As such, at the end of the collaborative process, this variable will contain the sum of all the random numbers used to generate each ElGamal encryption of a trust value. This sum is needed by the owner to decrypt the anonymous trust value (see step 9.n.ii.3.f) of the algorithm in Figure 2. Once updated the anonymized data structures, the procedure sends to the neighbour $\overline{n}$ the collaboration request. It then waits for the node reply, which consists of the value 0, if no path can be found, or the number of identified paths, otherwise. This number is stored into variable $TOT\_FOUND$ that is updated each time a new neighbour is considered. Once all neighbours have been considered, the procedure sends back to the sender the $TOT\_FOUND$ variable, increased by one in case the procedure has verified that $n$ is connected to the requestor.

The verification that an anonymous path is valid is done by function *check()* illustrated in Figure 4. This function takes as input the message received by the requestor and returns null, in case the anonymous relationship path is valid, a set of error messages, otherwise. More precisely, the procedure has to verify that the anonymous path satisfies properties $(a)$ and $(e)$, in that properties $(b)$, $(c)$ and $(d)$ are verified directly by

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339*

**Procedure** *path_builder*($n$,$msg$)

1 Let $sender$ be the node from which message $msg$ has been received
2 $TOT\_FOUND$ := 0
3 **If** $n$= $msg.req$:
  a Send $msg$ to $own$
**Else**
  b Let $C\_nodes$ be the set of nodes with which $n$ has established a relationship of type $msg.AC.rt$
  c **If** $C\_nodes$= $\emptyset$:
    i Send 0 to $sender$
    ii **Return**
  **EndIf**
  d **Foreach** $\overline{n} \in C\_nodes$:
    i Generate random $r_{no}, r_{nd}, r_{rt} \in \mathbb{Z}^*_{msg.par.q}$
    ii Generate random $r_t \in \mathbb{Z}^*_{msg.par'.q}$
    iii Generate the ElGamal encryptions $E_{r_{no}}(n)$ and $E_{r_{rt}}(rt)$
    iv $Diff_{rt} = r_{rt}$ -$msg.r_{rt}$
    v $Diff_{no} = r_{no}$ -$msg.r_{nd}$
    vi **If** $\overline{n} = req$
      1 flag:=1
      2 $Arel_j$ := $\langle E_{r_{no}}(n),\ req,\ E_{r_{rt}}(rt),\ Diff_{rt},\ Diff_{no}\rangle$
    **Else**
      3 Generate the ElGamal encryption $E_{r_{no}}(\overline{n})$
      4 $Arel_j$ := $\langle E_{r_{no}}(n),\ E_{r_{nd}}(\overline{n}),\ E_{r_{rt}}(rt),\ Diff_{rt},\ Diff_{no}\rangle$
    **EndIf**
    vii $AnonymousPath = msg.AnonymousPath \| Arel_j$
    viii Let $t$ be the trust value of the relationship of type $msg.AC.rt$ of $n$ with $\overline{n}$
    ix Generate the ElGamal encryption $E_{r_t}(t)$
    x $AnonymousTrust$= $msg.AnonymousTrust \times E_{r_t}(t)$
    xi $Trust_r$ := $msg.Trust_r$ + $r_t$
    xii $FOUND$ :=$SendCollReq(\{par, par', req, msg.AC.rt, AnonymousPath, AnonymousTrust, r_{nd}, r_{rt}, Trust_r\}, \overline{n})$
    xiii $TOT\_FOUND$ := $TOT\_FOUND$ + $FOUND$
  **EndFor**
**EndIf**
4 Send ($TOT\_FOUND$ + flag) to $sender$

Fig. 3. Procedure *path_builder()*

**Procedure** *check*($msg$)

1 Let $TOT\_error\_msg$ be initialized empty
2 **If** ($\frac{Arel_2.E_{rt}}{Arel_1.E_{rt}} \neq E_{Arel_2.Diff_{rt}}(1)$)
  a $error\_msg$ :="mismatch between the type of the first and second relationship"
  b $TOT\_error\_msg := TOT\_error\_msg \bigcup error\_msg$
**EndIf**
3 Let $AnonymousPath'$ be a copy of $msg.AnonymousPath$
4 Remove the first item, i.e., $Arel_1$, from $AnonymousPath'$
5 **ForEach** $Arel_j \in AnonymousPath'$
  a **If** ($\frac{Arel_j.E_{no}}{Arel_{j-1}.E_{nd}} \neq E_{Arel_j.Diff_{no}}(1)$)
    i $error\_msg$ :="mismatch between $rel_j.no$ and $rel_{j-1}.nd$"
    ii $TOT\_error\_msg := TOT\_error\_msg \bigcup error\_msg$
  **EndIf**
  b **If** ($\frac{Arel_j.E_{rt}}{Arel_{j-1}.E_{rt}} \neq E_{Arel_j.Diff_{rt}}(1)$)
    i $error\_msg$ :="mismatch between the type of the j-th and (j-1)-th relationship"
    ii $TOT\_error\_msg := TOT\_error\_msg \bigcup error\_msg$
  **EndIf**
**EndFor**
6 **Return** $TOT\_error\_msg$

Fig. 4. Procedure *check()*

the algorithm in Figure 2 (see steps 9.n.ii.3.a-e). We recall that these properties require to verify: $(a)$ whether all relationships in the path have type equal to the one required in $AC$; $(e)$ whether all relationships $rel_j$ in the path have the node whose established it equal to the node with which the preceding relationship $rel_{j-1}$ has been established, that is, $rel_j.no = rel_{j-1}.nd$. To check property $(a)$ the procedure starts to verify whether the encryption of $rel_1$ type (i.e., $Arel_1.E_{rt}$) is equal to $rel_2$ type (i.e., $Arel_2.E_{rt}$). This is performed according to the ElGamal cryptosystem in step 2. If this check succeeds, it means that the second relationship in the path has the correct relationship type. To check the remaining relationships, the procedure iterates the above check for each anonymous relationship (see the cycle in step 5). Here, given the input path $P$, it verifies whether the encryption of $rel_j$ type (i.e., $Arel_j.E_{rt}$) is equal to $rel_{j-1}$ type (i.e., $Arel_{j-1}.E_{rt}$), for each $j = \{2, \ldots, P_{depth}\}$. Note that all these checks are possible since, by Definition 3.2, the anonymous relationship path data structure contains the differences $r\_rt_j - r\_rt_{j-1}$, for each $j = \{2, \ldots, P_{depth}\}$. If all these checks succeed, the owner is ensured that all relationship types in the path are equal to the one required by $AC$, in that they are all equal to the one of $rel_1$, which has been generated by the owner itself,

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339*

otherwise an error message is stored in the $error$ variable. Property $(e)$ is checked in a similar way. Indeed, in the for cycle the procedure verifies whether the encryption of $rel_j.no$ (i.e., $Arel_j.E_{no}$) is equal to node $rel_{j-1}.nd$ (i.e., $Arel_j.E_{nd}$), for each $j = \{2, \dots, P_{depth}\}$. Procedure ends sending back to the owner the $error$ variable, which is NULL in case no error occurs.

## IV. Security Analysis

In this section, we discuss the robustness of our system against possible attacks. As adversary model, we assume that the adversary is a node in the $\mathcal{SN}$ which can collude with other network nodes to attack the system. We assume a semi-honest model, that is, that each node in the network adheres to the protocol but tries to learn extra information from the exchanged messages.

Under this adversary model, the main attacks that a node can perform are: *(1)* learn the trust level of previous nodes in the path, and *(2)* learn the relationships established in the network, from the data structures received during the execution of the access control protocol.

Let us consider these two attacks in details. Information on the trust level of a path is received by the owner through the anonymous trust value data structure (cfr. Definition 3.2). By using the information in this data structure, an intermediate node is not able to decrypt it, because it does not know the corresponding secret key. However, an intermediate node could be able to retrieve the random number $r_t$ used to encrypt the trust value from the $Trust_r$ component of the received message. Moreover, the second node in the traversed path receives $r_t$ directly by the owner (see step 9.m of the algorithm in Figure 2). By using $r_t$, an intermediate node could infer the real trust value. To avoid this inference, the owner could send to its direct neighbours a distorted random number. More precisely, the algorithm has to be modified thus to add a noise $X$ to $r_t$ before the owner sends it to a neighbour. This noise has to be subtracted to $Trust_r$ by the resource owner before decrypting the anonymous trust value. As a consequence, any intermediate node is not able to infer the real random number $r_t$. Note that, the same solution can be adopted also for avoiding inferences on the other random numbers (i.e., $r_{rt}$, $r_{nd}$).

In contrast, the owner is able to know the product of the trust levels associated with the edges in the path, since it knows the corresponding secret key. However, by knowing this value, the owner is not able to know which are the nodes belonging to the path, nor the trust level of any single arc in the path, except in one case explained at the end of this section.

Let us consider the attacks that can be performed against user relationships by inspecting the anonymous path data structure. By Definition 3.2, information on the relationship types and the involved nodes are encrypted using the ElGamal cryptosystem. Since the corresponding secret key is not known by any node in the network, information on the relationships existing in the network cannot be inferred. However, an inference is still possible even if relationship information are encrypted, as the following example shows.

*Example 3:* Consider the WBSN in Figure 1, suppose that A is the owner of a resource requested by R and that, in order to verify whether the access can be granted, A sends the request of collaboration to M, that then forwards it to T. The request of collaboration received by T from M contains the required relationship type (fof in this case). Moreover, by inspecting the received anonymized trust path, T verifies that it contains only two components. T may also infer that the first component refers to a relationship established by A, i.e., the resource owner. Therefore, it can infer that a relationship of type fof exists between A and M.

This inference can be avoided by simply let the owner insert into the anonymous path structure a certain number of fake encrypted relationships. As such, a node receiving from one of the owner's neighbours a request of collaboration cannot know, by inspecting the received data structure, the length of the path traversed so far.

This strategy avoids inferences from intermediate nodes. However, we have also to consider the inferences the owner could do by reading the anonymous path. This may happens when the anonymous path contains two items, that is, two anonymous relationships. Let us consider, for example, the path A→M→T in Figure 1, where A is the resource owner and T is the requestor. The corresponding anonymous path contains the anonymous relationship corresponding to relationship (A,M), generated by $A$, and the anonymous relationship for (M,T), generated by M. Thus, by accessing the anonymous path, the owner understands that M has a direct relationship with T of the type specified in the collaboration request, even if the corresponding anonymous relationship is anonymized. Moreover, it can also infer the trust of the relationship connecting M to T, since it knows the trust level of relationship (A, M) and the product of this trust level with the trust of relation (M,T). As such, with paths of depth 2, the relationship/trust privacy of owner neighbours may be violated. However, it is important to note, that this kind of inference is possible in any relationship-based access control enforcement. Indeed, if the owner knows that a given user has been granted the access to one of its resources protected by an access rule that requires a relationship of depth 2, the owner could infer that one of its neighbours has a direct relationship of that type with the requestor. Let us consider, for example, Figure 1 and assume that node M states for a given resource $rid$ the following access rule `({rid, (M,fof,2,0.1)})`. Moreover, let assume that M knows that C has been authorized to access $rid$. Since M has a unique $fof$ relationship with T, M is able to infer that T has a direct $fof$ relationship with C.

To manage this situation, we can extend our protocol such that the neighbour whose privacy can be violated by the above-described inference is aware about this violation and it can decide whether agreeing or not to the relationship disclosure. This can be obtained by assuming that, once received by the requestor, the anonymous path/anonymous trust value is forwarded directly to the TP. TP can verify the length of

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339*

the anonymous path, by simply counting the numbers of anonymous relationships it contains, and, in case of path of length 2, TP can ask to the interested owners neighbour whether it agrees or not in the disclosure of its relationship with the requestor. Accordingly with neighbour decision, the TP will forward the received messages to the owner or not. Actually, in order to make the TP able to compute the length of the anonymous path and to eventually contact one of the owner neighbours, it has to be aware of the neighbour that has to be contacted and of the number of fake anonymous relationships the owner has inserted in the path. Both this information can be safely provided to the TP by assuming the owner encrypts both these values with the public key of TP and attaches the resulting encrypted values to the request of collaboration.

## V. Conclusions

In this paper, we have presented a protocol on support of privacy-aware access control in WBSNs, based on a collaboration of selected nodes in the network. The protocol exploits the homomorphic properties of the ElGamal cryptosystem to ensure relationships and trust privacy. The access control protocol relies on the use of the anonymous path and anonymous trust value data structures that make the owner able to verify whether a path satisfies an access rule without knowing the exact relationships and trust levels composing the path. Such data structures also prevent the nodes taking part in the collaboration to acquire information on the relationships/trust levels in the path.

We plan to extend the work reported in this paper along several directions. First, we plan to implement the protocol to test the feasibility of the proposed method for different social network topologies and application domains. We also plan to investigate how our method should be extended to work under the malicious model.

## Acknowledgements

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

## References

[1] B. Ali, W. Villegas, and M. Maheswaran, "A trust based approach for protecting user data in social networks", in *Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative research (CASCON'07)*, 2007, pp. 288–293.

[2] S. Berteau, "Facebook's misrepresentation of beacon's threat to privacy: tracking users who opt out or are not logged in", Security Advisor Research Blog. http://community.ca.com/blogs/securityadvisor/archive/2007/11/29/facebook-s- misrepresentation-of-beacon-s-threat-to-privacy-tracking-users-who-opt-out-or- are-not-logged-in.aspx.

[3] B. Carminati and E. Ferrari, "Privacy-aware collaborative access control in web-based social networks", in *Proc. of the 22nd IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC2008), London, UK, July 2008*.

[4] B. Carminati, E. Ferrari, and A. Perego, "Enforcing access control in web-based social networks", *ACM Trans. on Information & System Security*, in press.

[5] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "A semantic web based framework for social networks", in *Proc. of the 14th ACM Symposium on Access Control Technologies (SACMAT'09), Stresa, Italy*, ACM, Ed., 2009.

[6] L. Chen, "Facebook's feeds cause privacy concerns. the amherst student", Available at: http://halogen.note.amherst.edu/ astudent/2006-2007/issue02/news/01.html., October 2006.

[7] C. P. Commission, "Social networking and privacy", http://www.privcom.gc.ca/information/social/index_e.asp, 2007.

[8] F. T. Commission, "Social networking sites: a parents guide", http://www.ftc.gov/bcp/edu/pubs/consumer/tech/tec13.shtm, 2007.

[9] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Tran. on Information Theory*, vol. 22, no. 644-654, 1976.

[10] J. Domingo-Ferrer, A. Viejo, F. Sebé, and U. González-Nicolás, "Privacy homomorphisms for social networks with private relationships", *Computer Networks Journal, Elsevier*, vol. 52, no. 15, pp. 3007-3016, 2008.

[11] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[12] EPIC Center, "Epic - social networking privacy", http://epic.org/privacy/socialnet/default.html, 2008.

[13] J. A. Golbeck and J. Hendler, "Inferring binary trust relationships in Web-based social networks", *ACM Transactions on Internet Technology*, vol. 6, no. 4, pp. 497–529, Nov. 2006.

[14] G. Hogben, "Security issues and recommendations for online social networks", Position Paper 1, European Network and Information Security Agency (ENISA).

[15] S. R. Kruk, S. Grzonkowski, H.-C. Choi, T. Woroniecki, and A. Gzella, "D-FOAF: distributed identity management with access rights delegation", in *Proceedings of the 1st Asian Semantic Web Conference (ASWC 2006)*. LNCS 4185, 2006.

[16] M. A. Philip W. L. Fong and Z. Zhao, "A privacy preservation model for facebook-style social network systems", in *Proceedings of the 14th European Symposium on Research In Computer Security (ESORICS'09), Saint Malo, France, September, 2009*, LNCS, Ed.

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8339*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8339*