Dynamic reputation-based trust computation in private networks
Technical Report IIIA-TR-2009-02

Jordi Nin[1], Barbara Carminati[2], Elena Ferrari[2] and Vicenç Torra[1]
[1]IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB s/n
08193 Bellaterra, (Catalonia, Spain)
E-mail: {jnin, vtorra}@iiia.csic.es
[2]Department of Computer Science and Communication
University of Insubria
22100 Varese, (Italy)
E-mail: {barbara.carminati, elena.ferrari}@uninsubria.it

January 27, 2009

Abstract

The use of collaborative networks services in general, and web based social networks (WBSN) services in particular, is today increasing and, therefore, the protection of the resources shared by network participants is becoming a crucial need. In a collaborative network, one of the main parameters on which access control relies is represented by trust and reputation, since access to a resource may or may not be granted on the basis of the trust/reputation of the requesting node. Therefore, the calculation of the trust of the nodes becomes a very important issue, mainly in business to business (BtoB) social networks, where trustworthy nodes can increase their benefits taking profit of their good reputation in the network. In order to address this point, in this paper we propose a mechanism to dynamically compute nodes trust, based on their past behavior. The key characteristic of our proposal is that trust is computed in a private way. This is obtained by anonymizing the local log files storing information about nodes actions.

1 Introduction

In several applications, such as peer-to-peer systems (13), social networks (4), or recommender systems (1) trust and reputation cover a key role. Although

the notion of trust is often associated with the one of reputation, there exists a relevant difference between the two concepts. As pointed out by (8), trust denotes whether (and possibly how much) a given entity A considers trustworthy another entity B. Therefore, trust expresses a personal opinion of A about B, and thus trust can be considered as a *subjective* (or *local*) measure of trustworthiness. By contrast, reputation denotes the trustworthiness of a given entity for all the entities in a network. Thus, reputation expresses the collective opinion of a community on one of its members, and thus it is an *objective* (or *global*) measure of trustworthiness. Additionally, an analysis of the related literature shows that there does not exist a unique definition of trust/reputation (12), whose definition may vary depending on the context and for which purposes they are used. For instance, in current Web-based social networks (WBSNs) (9), trust is a measure of how much a user trusts the other users in the network either with respect to a specific topic (*topical trust*) or in general (*absolute trust*), whereas in peer-to-peer systems the trustworthiness of a given peer mainly depends on its reliability in providing a given service. In contrast, when trust/reputation is used for access control/privacy protection (4; 5), trust/reputation should convey information about how much trustworthy a given user is not to reveal private or sensitive information to unauthorized users, and thus its purpose has some similarities with the notion of *security level* used in mandatory access control models (10).

Regardless of the specific notion of trust/reputation adopted, it is fundamental to devise mechanisms that help to automatically (or partially automatically) compute the trust/reputation. Although several proposal exists in this respect for peer-to-peer or social networks, we are not aware of similar proposals when trust/reputation is used for access control and privacy purposes. Therefore, in this paper, we propose a framework to address this issue. Our target scenario is a BtoB web-based social network where releasing of resources and personal/companies information is regulated by proper policies. However, the mechanisms we developed can be used in other kinds of *collaborative networks*, that is, networks where participants collaboratively carry out some tasks according to the network purpose (*e.g.*, releasing of resources in case of peer to peer networks, or establishment of new relationship in case of general purpose social networks). In our framework, reputation computation is based on the actions performed by a network node with regards to the releasing of both resources and private information (*e.g.*, relationships a node has with other nodes). The proposed mechanism exploits a local audit file to register actions performed by each user.

Obviously, public audit files raise serious privacy concerns (11; 17) because a participant might not agree in releasing (partial) information about the decisions he/she has made. For instance, a participant might prefer not to make public that he/she released some sensitive resources related to a patent of his/her company. To overcome this problem we propose a framework where information about decisions and actions performed by participants are available in an anonymous way. Here, the challenge is that of devising an anonymization strategy of the audit file such that details about the performed decision/action are kept

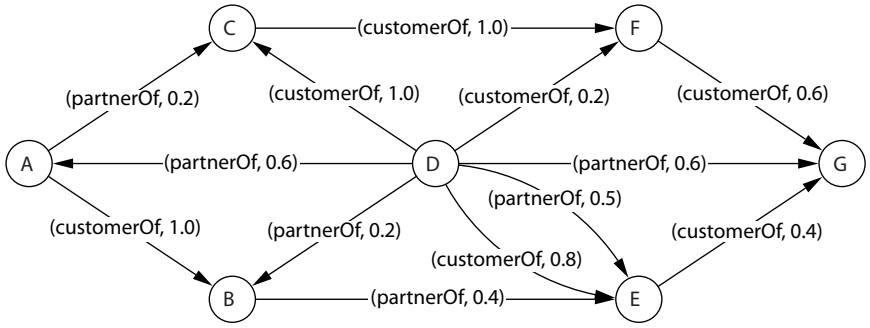

Figure 1: A portion of a social network. Labels associated with edges denote, respectively, the type and trust level of the corresponding relationship.

private but, at the same time, it is possible to determine whether the decision underlying the action is a correct or wrong one, with regards to the specified policies.

Even though there is a lot of research in the field of privacy enhancement, the work directly analyzing the privacy concerns of reputation seems to be limited, particularly in the field of access control. Usually, when privacy issues are addressed, trust computation services are centralized and the reputation information is stored in a trusted central repository (18). Obviously, this can be the bottleneck for the system. On the contrary, when trust computation services are distributed (2; 3), and scalability is not an issue, user privacy is disregarded. Up to our knowledge, our framework overcomes both problems, user privacy is ensured because sensitive information is encrypted and a trusted central repository is unnecessary because reputation information is distributed, each node stores its information.

The remainder of this paper is organized as follows. In Section 2 we explain some preliminary concepts about social networks, ElGamal encryption system and aggregation functions. Then, in Section 3 we describe the overall architecture of our framework, in Section 4 we explain our trust computation model. In Section 5, our audit system is described. Finally, the paper draws some conclusions and presents some future work.

2 Background

As pointed out in the introduction we consider, as reference scenario, a web-based social network (WBSN) where access control and privacy requirements are expressed through the specification of proper policies. Moreover, the proposed solution makes use of the homomorphic properties of the ElGamal encryption system and aggregation functions. For these reasons, in this section we provide a brief overview of all these notions.

2.1 Secure Web-based Social Networks

A common way to model a WBSN is as a directed labelled graph, where each node corresponds to a WBSN member and edges denote relationships between WBSN members. In particular, initial node of an edge denotes the WBSN member who established the corresponding relationship, the terminal node denotes the WBSN member who accepted to establish the relationship, whereas the label represents the type of the established relationship. Moreover, since a relevant feature of WBSNs is that relationships are characterized by a trust level, representing how much a given WBSN member considers trustworthy another member with whom he/she is establishing a relationship, we assume that each edge has a further label, in addition to relationship type, modelling the trust level. According to the considered WBSN model, throughout the paper we say that two WBSN members A and B *participate* in a relationship of a given type rt if there exists a path, consisting only of edges labelled with relationship type rt, connecting A with B. The length of such path corresponds to the *depth* of the corresponding relationship: if $depth = 1$, we say that the relationship is *direct*, whereas, if $depth > 1$, we say that the relationship is *indirect*. Let us consider, for instance, the WBSN depicted in Figure 1. D(avid) is a direct partner of E(ric), but by means of the relationship he has with B(ob), he is is also one of Eric's indirect partners. Therefore, we define the depth of a relationship of type rt between two nodes A and B as the length of the shortest path between A and B consisting only of edges labelled with rt. Thus, the depth of the 'partners of' relationship between D(avid) and B(ob) is 1. Moreover, in case of indirect relationships, the trust level corresponds to the multiplication of the trust levels associated with each edge in the corresponding path.[1] For instance, with regards to Figure 1, D(avid) is a direct customer of E(ric) with trust 0.8. By means of this relationship, D(avid) is also indirect customer of G(reg) with trust value equal to 0.32. Therefore, in our scenario we are assuming that relations are transitive. In the general purpose WBSN, this has no sense with some of the relationship types (*i.e.* father of), but in BtoB WBSN this assumption is possible because relations concern to economic connections as 'customer of' or 'partner of', and this kind of relations are in some way transitive.

In this paper, we consider WBSNs where access control and privacy requirements are expressed and enforced according to the strategies proposed in (4; 5). Regarding access control, we assume that each resource to be shared in the WBSN is protected by a set of *access rules*, denoting the members authorized to access the resource in terms of the type, depth, and trust level of existing relationships in the network. According to the access control model proposed in (4), each access rule has the form (rsc,AC), where AC is a set of *access conditions* that need to be all satisfied in order to get access to the resource having rsc as id (*i.e.*, URI). Formally, an access condition is a tuple $ac = (v, rt, d\_max, t\_min)$, where v is the member, referred as *target node*, with whom the requestor of a

[1] In the literature there exist several alternative models to compute the trust value of indirect relationships. Since this is out of the scope of this paper, we have adopted the simplest one.

given resource must have a direct or indirect relationship to obtain the access, whereas rt, d_max, and t_min are, respectively, the type, maximum depth, and minimum trust level that the relationship must have.

Access control enforcement is performed based on a client-side approach according to which a requestor is authorized to access a resource only if he/she is able to demonstrate that he/she satisfies the access rules applying to the requested resource. As described in (5), this implies that the requestor has to provide to the owner a *proof* showing that there exist the relationships required by the specified access rules, with the required depth and trust level. In order to make a member able to generate a proof, we assume that each relationship is certified, that is, each relationship is encoded by a certificate, generated by the members participating in it and distributed by them to the other network's participants. According to this assumption, a proof of the existence of a relationship between two members is given by the set of certificates corresponding to each relationship in the path connecting them. Thanks to this set of certificates, which we refer to as *certificate path*, the owner is able to verify whether the relationship satisfies the constraints on depth and trust level stated in the access rule. Moreover, the model proposed in (5) also allows the specification of privacy requirements on the established relationships. Relationship privacy requirements are stated through *distribution rules*. A distribution rule on a relationship of type rt established by node A with node B, denoted in what follows as DR^{rt}_{AB}, is a triple of the form $(v, \overline{rt}, d_{max})$, stating that the only nodes authorized to be aware of the relationship of type rt established by A with B are those that have a direct/indirect relationship of type $\overline{rt}$ with v and maximum depth d_{max}. Certificates distribution is regulated by distribution rules. More precisely, the key idea is that the nodes that have established the relationship, say A and B, distribute the relationship certificate to all and only their direct neighbours that satisfy the distribution rules they have specified for that relationship. The relationship certificate is sent to each neighbour together with the corresponding distribution rule. Thus, each neighbour can access the distribution rule and decides whether he/she has to forward the certificate, and distribution rule, to his/her direct neighbours. More details can be found in (5).

2.2 ElGamal

The ElGamal (7) encryption system is a public key encryption algorithm which is based on the Diffie-Hellman (6) key agreement. This system is widely used in many applications (*e.g.*, the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems).

ElGamal encryption can be defined over any cyclic group $\mathbb{G}$. Its security depends upon the difficulty of a certain problem in $\mathbb{G}$ related to computing discrete logarithms. It is based on the following components:

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 generation. Choose $sk \in \mathbb{Z}_p^*$ at random, and publish $pk = g^{sk} mod\ p$.

Encyption. Encrypt message $m \in \mathbb{G}$. Take $r \in \mathbb{Z}_q^*$ at random, compute $R = g^r mod\ p$ and $s = m \cdot pk^r mod\ p$. The ciphertext is $c = (R, s)$. r is kept secret.

Decrypt. Given the secret key sk and the ciphertext $c = (R, s)$, the plain text is given by

$$m = \frac{s}{R^{sk}} mod\ p$$

ElGamal encryption system is multiplicatively homomorphic. That is, given $c_1 = (g^{r_1}, m_1 \cdot pk^{r_1})$ and $c_2 = (g^{r_2}, m_2 \cdot pk^{r_2})$ two encryptions of m_1 and m_2, we can obtain an encryption of m_1/m_2 as

$$c_1 \oslash c_2 = (\frac{g^{r_1}}{g^{r_2}}, \frac{m_1 \cdot pk^{r_1}}{m_2 \cdot pk^{r_2}}) = (g^{r_1 - r_2}, \frac{m_1}{m_2} \cdot pk^{r_1 - r_2}).$$

Thanks to this property, a user who has encrypted the same message twice, with ElGamal, can prove in a zero-knowledge way (14) that both plaintexts are equal. To do this, after having published the two encryptions c_1 and c_2, he has to publish the difference $r_1 - r_2$. Then, everybody can verify that the two plaintexts are equal (that is, $m_1/m_2 = 1$), by checking if

$$c_1 \oslash c_2 = (g^{r_1 - r_2}, pk^{r_1 - r_2}).$$

2.3 Aggregation functions

Aggregation functions (15) are numerical functions used for information fusion. They typically combine N numerical values supplied by N sources into a single datum. In this work, we will consider the Ordered Weighted Aggregation (OWA) operator. Two definitions exist, one applicable when the number of sources is known in advance, and another that do not require to know how many sources will be combined. We will use this latter definition, that uses fuzzy quantifiers.

Definition 1 *A function $Q : [0, 1] \to [0, 1]$ is a* regular monotonically non-decreasing fuzzy quantifier *(non-decreasing fuzzy quantifiers for short) if it satisfies: (i) $Q(0) = 0$; (ii) $Q(1) = 1$; (iii) $x > y$ implies $Q(x) \geq Q(y)$.*

Two examples of families of fuzzy quantifiers are given below.

$Q_1^\alpha(x) = x^\alpha$ for $\alpha = 1/5, 2/5, \cdots, \ldots, 10/5$

$Q_2^\alpha(x) = 1/(1 + e^{(\alpha - x)*10})$ for $\alpha = \{0, 0.1, \ldots 0.9\}$

Using fuzzy quantifiers, the OWA operator (16) is defined as follows.

Definition 2 *Let Q be a non-decreasing fuzzy quantifier, then $OWA_Q : \mathbb{R}^N \rightarrow \mathbb{R}$ is an* Ordered Weighting Averaging (OWA) operator *if*

$$OWA_Q(a_1, ..., a_N) = \sum_{i=1}^{N} (Q(i/N) - Q((i-1)/N))a_{\sigma(i)}$$

where σ is a permutation such that $a_{\sigma(i)} \geq a_{\sigma(i+1)}$.

The interest of the OWA operators is that they permit the user to aggregate the values giving importance to large (or small) values. In the case of the quantifiers given above, the smallest the α, the largest is the importance for the largest values being aggregated. In contrast, the largest the α, the lowest the importance of the largest values (and the largest the importance given to low values).

3 Overall framework

The key idea underlying our model for trust computation is the consideration that in the real world the trust value granted to a person/company is estimated on the basis of his/her *reputation*, which can be assessed taking in account the past behavior. Indeed, it is a matter of fact that people assign to a person with unfair behavior a bad reputation and, as a consequence, a low level of trust. The idea is to apply a similar rational to collaborative networks. That is, to estimate the trust level to be assigned to a participant on the basis of its reputation, given by its behavior with regards to all the nodes in the network. More concretely, a way to capture participant's behavior is to monitor the decisions and the actions performed by them once they have received requests of collaboration. For instance, the decision of releasing sensitive information to unauthorized participants obviously can be considered as a signal of bad behavior. A naive solution to make users able to monitor the behaviour of other participants is to record into a public *audit file* each decision made by network participants. This entry should store also information about the performed action, plus additional information making other participants able to infer whether a right or wrong decision has been made.

Obviously, public audit files raise serious privacy concerns because a participant might not agree in releasing information about the decisions he/she has made, even if these are signals of good behavior. To overcome this problem we propose a framework where information about decisions and actions performed by participants are available in an anonymous version. Here, the challenge is that of devising an anonymization strategy of the audit file such that details about the performed decision/action are kept private but, at the same time, it is possible to determine whether the decision underlying the action is a correct or wrong one, with regards to the specified policies.

The proposed framework assumes that each participant generates and makes available to all other network participants an *anonymized audit file*. In particular, to avoid malicious participants to omit or insert fake entries into the

anonymized audit file, we assume that this task is performed by a trusted module, provided directly by the network manager upon subscription, that each participant have to install in his/her computer computer. This module, called *auditor*, monitors all the decisions performed by the participant and for each of them it generates an entry to be inserted into the local anonymized audit file.

The maintenance of this anonymous audit file can be done once a week (or a month) as an off-line process. Then, the performance of this conversion process does not affect to the general network performance.

4 Reputation-based trust evaluation model

In this section, we introduce the trust computation model used by our framework. As introduced in Section 3, the underlying idea is that the trust T_{AB} associated by a user A to a user B is evaluated based on A's reputation R_A, automatically computed considering A's behavior. We believe the adopted model to compute trust is orthogonal to the scope of this paper, and as such we assume that the trust value T_{AB} associated by user A to user B is given by a generic function $T()$ which takes as a mandatory input the reputation value R_A.Therefore, in what follows we focus on reputation computation.

When a participant performs all decisions in accordance with the specified distribution and access rules, he/she has a good behavior and therefore he/she has to be assigned a good reputation (the reputation value has to be maximum and, thus, equal to 1). In contrast, if a participant does not make a correct decision, his/her reputation level should be lower.

To formalize the model, we need to first introduce the possible wrong decisions a participant can make in the reference social network scenario. These are identified by considering the two main requests of collaboration a participant receives in a social network: (a) the request for *releasing a resource*, according to the specified access control rules, and (b) the request for *distributing a certificate*, on the basis of the specified distribution rules. Then, we have a wrong (bad) decision when a user decides to deny a owned or delegated resource to an authorized user (*e.g.*, a user does not release a resource *rsc* even if the requestor provides a correct proof which matches the specified access rules). We call this wrong decision *denial of resource releasing* (DRR). A similar wrong decision can be made with regard to certificate distribution, when a user decides not to distribute a certificate to some of his/her neighbors, even if they are authorized to receive it according to the specified distribution rules. We call this wrong decision *denial of certificate distribution* (DCD). A further kind of wrong decisions come when a user decides to disseminate resources/certificates to non-authorized users. For instance, a user can release a resource *rsc* when the provided proof is not correct. In this case, we refer to an *unauthorized resource dissemination* (URD). A similar wrong decision can be done with certificates. In this case, we have an *unauthorized certificate dissemination* (UCD).

In devising the model for reputation computation, we consider that not all wrong decisions have the same relevance. For example, in the case of denial of

certificate distribution (DCD), not to disseminate a certificate whose distribution rule has depth equal to 4 has not the same importance as not distribute a certificate with depth equal to 1. Indeed, the consequence of the first decision is more serious than the second one, since it affects more users of the social network. Similar considerations hold for other wrong decisions. To model the relevance of wrong decisions, we further classify them according to five dimensions: three related to requests of releasing a resource, or as we denote here, related to *access control decisions*, and two related to certificate distribution.

In relation to *access control decisions*, let us consider the decision of a user A about releasing or not to user B a resource `rsc`. Let us assume that `AR` is the access rule applying to `rsc`, and that `crt path` is the certificate path provided by user B as a proof. Then, the first dimension, denoted by $\texttt{AC}_\texttt{t}$ and called *trust dimension*, corresponds to the difference between the minimum trust required in `AR` and the trust computed based on the path extracted from `crt path`. Note that when $\texttt{AC}_\texttt{t}$ is lower than zero and the resource has not been released, A commits a `DRR`. In contrast, if $\texttt{AC}_\texttt{t}$ is greater than zero and the resource has been released, A commits a `URD`. The second dimension, referred to as *depth dimension*, denoted as $\texttt{AC}_\texttt{d}$, is defined as the difference between the depth required in `AC` and the depth of the path extracted from `crt path`. The third dimension, called *path dimension* denoted as $\texttt{AC}_\texttt{p}$, is used to model situations as the next one. Let us consider an access rule consisting only of the following access condition $AC = (B, cof, 3, 0.5)$, and let $p = (A \rightarrow_{cof} B)(B \rightarrow_{pof} E)$ be the path extracted from `crt path`. Obviously, this `crt path` is not a valid proof, since the *pof* relationship is not referred by the access rule, and the last node in the path is not the target node specified in the access condition. Formally, $\texttt{AC}_\texttt{p}$ is an integer value ranging from 0 to 3: 0 if the path extracted from `crt path` is correct or adding one for each of the following problems in the path: (a) there exists in the path at least a relationship whose type is not equal to the one required in `AC`; (b) the first node (resp. last node) in the path is not equal to the requestor (resp. the target node) in `AC`; (c) there exists in the path at least a relationship r such that the node whose established it is not equal to the node with which the relationship preceding r has been established.

Similar dimensions are also defined for decisions made as a consequence of a request for certificate distribution. More precisely, let us consider the decision of a user A to disseminate to user B the certificate `cert`, to which the distribution rule $\texttt{DR}_\texttt{rel}$ is applied. Moreover, let `crt path` be the set of certificates corresponding to the path that `cert` has traversed to reach A. Then, we define a depth dimension related to the certificate distribution decision, denoted with $\texttt{CD}_\texttt{d}$, as the difference between the maximum depth required in $\texttt{DR}_\texttt{rel}$ and the depth of the path extracted from `crt path`. Finally, we define the path dimension, denoted as $\texttt{CD}_\texttt{p}$, as for $\texttt{AC}_\texttt{p}$.

In order to combine these dimensions, values have to be in a common domain. In practice, we map all such values in the $[0, 1]$ interval. Formally, the path dimension is normalized dividing the values by 3 (the maximum difference allowed in the previous definition) and the depth dimension is normalized dividing the corresponding values by a constant (the maximum difference expected;

in case of larger values than expected, the dimension is set to one). Note that the trust dimension is already in the $[0, 1]$ domain.

Example 1 *Let us consider the social network in Figure 1, and suppose that D(avid) administers the resource* $\mathtt{rsc}_1$ *with access rule* $\mathtt{AR}_{\mathtt{rsc}_1} = (\mathtt{rsc}_1, \{(\mathtt{G}, \mathtt{pof}, 2, 0.25)\})$. *Suppose that B(ob) wishes to access* $\mathtt{rsc}_1$ *and that he provides a certificate path* `crt path` *containing the path* $\mathtt{p} = (\mathtt{B} \rightarrow_{(\mathtt{pof},0.4)} \mathtt{E})(\mathtt{E} \rightarrow_{(\mathtt{cof},0.4)} \mathtt{G})$. *Once the proof is received by David, he decides to send the resource* $\mathtt{rsc}_1$ *to Bob. Afterwards, David's auditor computes the different dimensions for access control decisions. First, it computes* $\mathtt{AC}_\mathtt{t} = 0.09$ *(*$\mathtt{p}$*'s trust is lower than the minimum trust required in* $\mathtt{AR}_{\mathtt{rsc}_1}$*), then it computes* $\mathtt{AC}_\mathtt{d} = 0$ *(*$\mathtt{p}$*'s depth is equal to the maximum depth required in* $\mathtt{AR}_{\mathtt{rsc}_1}$*) and, finally,* $\mathtt{AC}_\mathtt{p} = 1$ *(*i.e.*, the provided proof presents a problem: the relationship type cof is not allowed). These results imply that Bob is not allowed to access* $\mathtt{rsc}_1$ *because Bob's request does not fulfill two of the three access control dimensions. If David sends the resource* $\mathtt{rsc}_1$ *to Bob, this is an example of unauthorized resource dissemination (URD).*

Let us now formalize how we combine all the wrong decisions taken by user A to calculate his/her overall reputation value R_A. To do so, we first aggregate the values in each dimension and then combine all the dimensions. The aggregation in each dimension is based on the OWA operator (defined in Section 2.3) that permits to represent either the case of assigning larger importance to the most serious decisions (the ones with larger values), or to assign the same importance to all wrong decisions.

Definition 3 *Let A be a user, let* $\mathtt{AC}_{\mathtt{SET}_\mathtt{A}}$ *be the set of access control decisions made by A. Let* $\mathtt{DRR}_\mathtt{t} \subseteq \mathtt{AC}_{\mathtt{SET}_\mathtt{A}}$ *be the set of DRR wrong decisions with respect to trust (whose* $\mathtt{AC}_\mathtt{t}$ *trust dimension is lower than 0). Let* $\mathtt{URD}_\mathtt{t} \subseteq \mathtt{AC}_{\mathtt{SET}_\mathtt{A}}$ *be the set of URD wrong decisions with respect to trust (whose* $\mathtt{AC}_\mathtt{t}$ *trust dimension is greater than 0). Then, the set of wrong decisions related to access control with respect to trust is defined as* $\mathtt{WD}_{\mathtt{t}_\mathtt{A}} := DRR_t \cup URD_t$, *and the aggregated value* $\mathtt{AGt}_{\mathtt{AC}_{\mathtt{SET}_\mathtt{A}}}$ *of wrong decisions in* $\mathtt{WDt}_\mathtt{A}$ *with respect to the trust dimension is given by the following formula:*

$$\mathtt{AGt}_{\mathtt{AC}_{\mathtt{SET}_\mathtt{A}}} = \mathtt{OWA}_\mathtt{Q}(\mathtt{wd}_1, \ldots, \mathtt{wd}_{|\mathtt{WD}_{\mathtt{t}_\mathtt{A}}|})$$

where Q is a non-decreasing fuzzy quantifier, and $\mathtt{wd}_\mathtt{i}$ *are the absolute values of the* $\mathtt{AC}_\mathtt{t}$ *in* $\mathtt{WD}_{\mathtt{t}_\mathtt{A}}$ *(note that as OWA is a symmetric function, the ordering of the* $\mathtt{wd}_\mathtt{i}$ *is irrelevant).*

This definition aggregates the values for the trust dimension. The same applies to the other dimensions. In this way, we obtain aggregated values for the five dimensions presented above. We denote them by: $AGt_{AC_{SET_A}}$, $AGd_{AC_{SET_A}}$, $AGp_{AC_{SET_A}}$, $AGd_{CD_{SET_A}}$, $AGp_{CD_{SET_A}}$.

Example 2 *Let us consider the social network in Figure 1, and suppose that D(avid) only administers the resource* $\mathtt{rsc}_1$*, with the same access rule given in*

Example 1. Let us assume that David has participated in three different requests for his resource $\mathtt{rsc}_1$*. One of them is Bob's request described in Example 1. Another request is from F(red), that provides* $\mathtt{p} = (\mathtt{F} \rightarrow_{(\mathtt{cof},0.4)} \mathtt{G})$*. David's auditor computes all access control dimensions and as Fred does not fulfills all the conditions needed by the access rule* $\mathtt{AR}_{\mathtt{rsc}_1} = (\mathtt{rsc}_1, \{(\mathtt{G}, \mathtt{pof}, 2, 0.25)\})$ (i.e., *the path is not a valid proof), David decides not to send* $\mathtt{rsc}_1$ *to Fred. In the third request from A(lice), she provides* $\mathtt{p} = (\mathtt{A} \rightarrow_{(\mathtt{pof},0.2)} \mathtt{C})$*. At first sight, Alice's proof is not correct, because the proof only proves that Alice is a partner of C(arl). When David's auditor computes the dimensions for this access control decision, he founds that* $\mathtt{AC_t} = 0.05$*,* $\mathtt{AC_d} = 1$ *and* $\mathtt{AC_p} = 1$*.*

So, based on these values, Alice has not access to the resource $\mathtt{rsc}_1$*. Nevertheless, David decides to send the resource* $\mathtt{rsc}_1$ *to Alice. Accordingly, David's decision is a URD.*

Now, let us consider that B(ob) wants to create a new relationship with David. To compute David's trust, Bob has to compute David's reputation, i.e., *he has to compute all the dimensions of the access control and certificate distribution actions. For instance, to compute* $\mathtt{AGt}_{\mathtt{AC}_{\mathtt{SET_D}}}$*, Bob has to decide which quantifier Q wants to use. Bob decides to use* $\mathtt{Q}_1^1$ *because this quantifier assigns a similar importance to all wrong decisions made by David. In this way, wrong decisions with respect to* $\mathtt{AC_t}$ *(with values* $\mathtt{AC_t} = \{0.09, 0.05\}$*) are aggregated. Thus,* $\mathtt{AGt}_{\mathtt{AC}_{\mathtt{SET_D}}}$ *is equal to* 0.07 *(Definition 3).*

We can now formalize the formula for the automatic computation of the reputation value.

Definition 4 *Let A be a user, let* $AGt_{AC_{SET_A}}$*,*
$AGd_{AC_{SET_A}}$ *and* $AGp_{AC_{SET_A}}$ *be the aggregated values for trust dimension, depth dimension and path dimension for access control decisions, respectively, and let* $AGd_{CD_{SET_A}}$ *and* $AGp_{CD_{SET_A}}$ *be the aggregated values for depth dimension and path dimension for certificate distribution decisions, respectively. The reputation value* R_A *of user A is defined as:*

$$\mathtt{R_A} = 1 - \tfrac{1}{5}(\mathtt{AGt}_{\mathtt{AC}_{\mathtt{SET_A}}} + \mathtt{AGd}_{\mathtt{AC}_{\mathtt{SET_A}}} + \mathtt{AGp}_{\mathtt{AC}_{\mathtt{SET_A}}} + \mathtt{AGd}_{\mathtt{CD}_{\mathtt{SET_A}}} + \mathtt{AGp}_{\mathtt{CD}_{\mathtt{SET_A}}})$$

5 Audit files

As introduced in Section 3, the auditor of a given user A monitors each decision made by A and stores it into a private audit file. A different entry is generated to each distinct decision. According to our approach, the auditor also creates an anonymized audit file available to all users of the WBSN, by considering each entry in the private audit file. In the next section, we formally introduce the information stored in the private and anonymized files, whereas we postpone to Section 5.2 a discussion on how, by using this information a user is able to determine whether a decision has been correctly evaluated.

5.1 Audit file generator

Let us start to consider the private audit file. Since participants in the social network can make two different types of decisions according to the received request (*i.e.*, request for resource releasing and request for certificate distribution), the audit file contains two different types of entries. The first type contains information about the decision made by a node A when he/she receives an access request. We call this entry *access request entry*. To model this type of decision, the entry contains information about the access request, the access rule associated with the requested resource,[2] as well as the certificate path provided as proof by the requestor. The formal definition of the access request entry is the following.

Definition 5 *Let* $\mathtt{AcR}$*=(*$\mathtt{rsc}$*,* $\mathtt{r}$*,* $\mathtt{AR}$*) be an access request, where* $\mathtt{rsc}$ *is the identifier of the requested resource,* $\mathtt{r}$ *is the identifier of the requestor, and* $\mathtt{AR}$ *is the access rule applied to resource* $\mathtt{rsc}$*. The corresponding access request entry is defined as:*

$$\mathtt{e_{ac}} = \langle \mathtt{AcR},\ \mathtt{crt\ path}, \mathtt{t_p},\ \mathtt{dec} \rangle$$

where $\mathtt{crt\ path}$ *is the certificate path provided by the requestor* $\mathtt{r}$ *as a proof,* $\mathtt{t_p}$ *is the trust value computed from the path extracted from* $\mathtt{crt\ path}$*, and* $\mathtt{dec}$ *is a boolean value set to 1 if the resource* $\mathtt{rsc}$ *has been released to requestor* $\mathtt{r}$*, set to 0, otherwise.*

Example 3 *Consider the social network in Figure 1, and suppose again that D(avid) administers the resource with identifier* $\mathtt{rsc_1}$ *with the access rule* $\mathtt{AR}_{\mathtt{rsc_1}} = (\mathtt{rsc_1}, \{(\mathtt{G}, \mathtt{pof}, 2, 0.25)\})$*. B(ob) wishes to have access to* $\mathtt{rsc_1}$ *and he provides a* $\mathtt{crt\ path}$ *from which the following path is extracted:* $p = (B \rightarrow_{(pof,0.4)} E)(E \rightarrow_{(cof,0.4)} G)$*. David decides to send the resource* $\mathtt{rsc_1}$ *to Bob, therefore David's auditor generates the following access request entry:*

$$\mathtt{e_{ac}} = \langle (\mathtt{rsc_1}, \mathtt{B}, (\mathtt{rsc_1}, \{(\mathtt{G}, \mathtt{pof}, 2, 0.25)\})),$$
$$(\mathtt{B} \rightarrow_{(\mathtt{pof},0.4)} \mathtt{E})(\mathtt{E} \rightarrow_{(\mathtt{cof},0.4)} \mathtt{G}), 0.16, 1 \rangle$$

The other kind of entries contained in the audit file are related to the decisions made by a node A when he/she receives a request to distribute a certificate. This type of entry, called *certificate distribution entry*, contains information about the certificate to be distributed, the user who established the corresponding relationship, the distribution rule associated with it, as well as information about the path the certificate has traversed to reach A. The formal definition of the certificate distribution entry is the following.

Definition 6 *Let user A be the owner of the audit file. Let* $\mathtt{CdR}$*=(*$\mathtt{cert}$*,* $\mathtt{rl}$*,* $\mathtt{DR}$*,* $\mathtt{s}$*,* $\mathtt{crt\ path}$*) be a request for certificate distribution, where* $\mathtt{cert}$ *is the certificate*

[2] In what follows, for simplicity, we suppose that each resource is protected by a single access rule.

to be distributed, $\mathtt{rl}$ *is the corresponding relationship,* $\mathtt{DR_{rl}}$ *is the associated distribution rule,* $\mathtt{s}$ *is the A's neighbour sending the request for distribution, and* $\mathtt{crt\ path}$ *is the certificate path corresponding to the path certificate* $\mathtt{cert}$ *has traversed to reach A. The corresponding certificate distribution entry is defined as:*

$$\mathtt{e_{cd}} = \langle \mathtt{CdR},\ \mathtt{r},\ \mathtt{rt_{(A,r)}},\ \mathtt{dec} \rangle$$

where $\mathtt{r}$ *is the identifier of A's neighbour,* $\mathtt{rt_{(A,r)}}$ *is the type of the relationship between A and* $\mathtt{r}$*, and* $\mathtt{dec}$ *is a boolean value set to 1 if the certificate* $\mathtt{cert}$ *has been released to neighbour* $\mathtt{r}$*, set to 0, otherwise.*

Example 4 *Consider the social network in Figure 1, and suppose that D(avid) and C(arl) create a relationship rl of type pof and trust equal to 0.6. Let* $\mathtt{cert}^{\mathtt{pof}}_{\mathtt{D,C}}$ *be the certificate generated for relationship rl. Let* $\mathtt{DR}^{\mathtt{pof}}_{\mathtt{D,C}} = (\mathtt{D}, \mathtt{pof}, 2)$ *be the distribution rule for* $\mathtt{cert}^{\mathtt{pof}}_{\mathtt{D,C}}$*. E(ric) receives the certificate* $\mathtt{cert}^{\mathtt{pof}}_{\mathtt{D,C}}$ *from David. Suppose that Eric sends such certificate to G(reg). Once the certificate* $\mathtt{cert}^{\mathtt{pof}}_{\mathtt{D,C}}$ *is released, Eric's auditor generates the following certificate distribution entry:*

$$\mathtt{e_{cd}} = \langle (\mathtt{cert}^{\mathtt{pof}}_{\mathtt{D,C}}, \mathtt{rl}, \{(\mathtt{D}, \mathtt{pof}, 2)\}, \mathtt{D},$$
$$\{(\mathtt{D} \rightarrow_{(\mathtt{pof},0.5)} \mathtt{E})\}), \mathtt{G}, \mathtt{cof}, 1 \rangle$$

According to the proposed strategy, the anonymized audit file is generated by anonymizing each entry of the audit file, that is, the access request and the certificate distribution entries. In the following, we introduce the formal definition of the anonymized access request entry and the anonymized distribution certificate entry. In both the definitions, we exploit notation $\mathtt{C}(\mathtt{R}, \mathtt{text})$ to denote the ElGamal encryption of $\mathtt{text}$ by using the random number r.

Let us start to define the anonymized access request entry.

Definition 7 *Let user A be the owner of an audit file f, and let* $\mathtt{e_{ac}}$ *be an access request entry stored into f. The corresponding anonymized access request entry is defined as*

$$\mathtt{ae_{ac}} = \langle \mathtt{C_r},\ \mathtt{C_{tr}},\ \mathtt{C_t},\ \mathtt{CRT},\ \mathtt{t_p}, \mathtt{t_{min}},\ \mathtt{d_{max}},\ \mathtt{dec},\ \mathtt{C_p} \rangle$$

where $\mathtt{C_r} = \mathtt{C}(\mathtt{R_r}, \mathtt{r})$ *is the encryption, using the random number* $\mathtt{r_r}$*, of the requestor identifier stored into the* $\mathtt{AcR}$ *component of* $\mathtt{e_{ac}}$*;* $\mathtt{C_{tr}} = (\mathtt{R_{tr}}, \mathtt{AcR.AR.tr})$ *is the encryption, using the random number* $\mathtt{r_{tr}}$*, of the target node of the access rule* $\mathtt{AR}$ *contained into the* $\mathtt{AcR}$ *component of* $\mathtt{e_{ac}}$*;* $\mathtt{C_t} = (\mathtt{R_t}, \mathtt{e_{ar}.t_p})$ *is the encryption, using the random number* $\mathtt{r_t}$*, of the trust value contained into the* $\mathtt{t_p}$ *component of* $\mathtt{e_{ac}}$*;* $\mathtt{CRT}$ *is a set containing for each possible relationship type* $\mathtt{rt_j}$ *in the access rule* $\mathtt{AR}$ *of the* $\mathtt{e_{ac}.AcR}$ *component a distinct encryption* $\mathtt{C_{rt_j}} = \mathtt{C}(\mathtt{R_{rt_j}}, \mathtt{rt_j})$ *using the random number* $\mathtt{r_{rt_j}}$*;* $\mathtt{t_p}$ *is the trust value associated with the path extracted from* $\mathtt{crt\ path}$*;* t_{min} *is the minimum trust required by* $\mathtt{e_{ar}.AcR.AR}$*,* $\mathtt{d_{max}}$ *is the maximum depth required by* $\mathtt{e_{ar}.AcR.AR}$*;* $\mathtt{dec}$ *is a boolean value set to 1*

if the resource `rsc` *has been released to requestor* `r`*, set to 0, otherwise and* $\mathtt{C_p}$ *is the anonymous path structure defined on path extracted from* `crt path` *(see Definition 9 below).*

In contrast, the definition of the anonymized certificate distribution entry is given in what follows.

Definition 8 *Let user A be the owner of the audit file f, and let* $\mathtt{e_{cd}}$ *be a certificate distribution entry stored into f. The corresponding anonymized certificate distribution entry is defined as:*

$$\mathtt{ae_{cd}} = \langle \mathtt{C_r},\ \mathtt{C_a},\ \mathtt{CRT},\ \mathtt{rt}(\mathtt{A_r}),\ \mathtt{d_{max}},\ \mathtt{dec},\ \mathtt{C_p} \rangle$$

where $\mathtt{C_r} = \mathtt{C}(\mathtt{R_r}, \mathtt{r})$ *is the encryption, using the random number* $\mathtt{r_r}$*, of the identifier of the node who established the relationship;* $\mathtt{C_a} = \mathtt{C}(\mathtt{R_a}, \mathtt{A})$ *is the encryption of the identifier of A using the random number* $\mathtt{r_a}$*;* `CRT` *is a set containing for each possible relationship type* $\mathtt{rt_j}$ *in the distribution rule* `CdR.DR` *a distinct encryption* $\mathtt{C_{rt_j}} = \mathtt{C}(\mathtt{R_{rt_j}}, \mathtt{rt_j})$ *using the random number* $\mathtt{r_{rt_j}}$*;* `rt` *contains the third component of* $\mathtt{e_{cd}}$*;* $\mathtt{d_{max}}$ *is the maximum depth of the distribution rule contained in the* `CdR` *component of* $\mathtt{e_{cd}}$*;* `dec` *is a boolean value set to 1 if the certificate* `cert` *has been released to neighbour* `r`*, set to 0, otherwise and* $\mathtt{C_p}$ *is the anonymous path structure defined on path extracted from* `crt path` *(see Definition 9 below).*

Both the anonymized entries make use of the *anonymous path structure*. This structure is the building block of the proposed approach to verify whether the decision corresponding to an entry is correct or not. To verify the correctness of a decision, it is essential to check the structure of the path extracted by the certificate path `crt path` received as proof. In general, notwithstanding of the type of the received request of collaboration, access control or certificate distribution request, in order to determine whether the decision *dec* was made correctly, we need to check the following characteristics of the path p extracted from the received certificate path `crt path`: (a) whether all relationships in p have type equal to the one required in the `AR` access rule, or in the `DR` distribution rule; (b) whether the first node (resp. last node) of the first relationship (resp. last relationship) in p is equal to the requestor (resp. the target node in `AR`) in case of access request, or it is equal to the node who established the relationship (resp. the node who has sent the certificate) in case of certificate distribution; (c) whether all relationships r in p have the node whose established it equal to the node with which the relationship preceding r has been established; in particular, if r is the first relationship of p, the node must be equal to the requestor in case of access request, or it must be equal to the node that has generated the certificate in case of certificate distribution; (d) whether p's depth is less than or equal to the maximum depth required in `AR` or `DR` (e) only in case of access request, whether p's trust is greater than or equal to the minimum trust required in `AR`.

Thus, the anonymous path structure has been devised to make a user able to verify the above mentioned conditions. In particular, the basis of such anonymous verifications are provided by the homomorphic property of the ElGamal crypto-system. To exploit this property for comparing two texts, it is necessary to know the difference between the random numbers used in the encryption of both texts. Recall that each time that a message is encrypted with the ElGamal, a new random number $r \in \mathbb{Z}_q^*$ (kept private) is generated, and $R = g^r mod\ p$ (public). Then, the encryption of a message m is computed as $s = m \cdot pk^r mod\ p$.

The anonymous path structure is defined as follows:

Definition 9 *Let A be a node in the network. Let* `crt path` *be a certificate path contained into an access request entry of the audit file of A (resp. into the* `CdR` *component of a certificate distribution entry). Let p be the path extracted from* `crt path`*. Let d be the depth of the path extracted from* `crt path`*.*

The anonymous path C_p generated from p is defined as follows:

$$\mathtt{C_p} = \langle \mathtt{C_{cp}}, \mathtt{D_t} \rangle$$

where:

- $\mathtt{C_{cp}}$ *is a set containing for each relationship* $\mathtt{r_j}$ *in p, $j = 1 \ldots d$ a different tuple of the form:*

  $$\mathtt{C_{cp}} = \langle \mathtt{C_{o_j}},\ \mathtt{C_{d_j}},\ \mathtt{C_{rt_j}},\ \mathtt{C_{t_j}},\ \mathtt{D_{o_j}},\ \mathtt{D_{d_j}},\ \mathtt{D_{rt_j}} \rangle$$

  where $\mathtt{C_{o_j}} = \mathtt{C}(\mathtt{R_{o_j}}, \mathtt{o_{id_j}})$ *is the encryption of the identifier of the user with which* $\mathtt{r_j}$ *is established using the random number* $\mathtt{r_{o_j}}$*;* $\mathtt{C_{d_j}} = \mathtt{C}(\mathtt{R_{d_j}}, \mathtt{d_j})$ *is the encryption of the identifier of the user who established* $\mathtt{r_j}$ *using the random number* $\mathtt{r_{d_j}}$*;* $\mathtt{C_{rt_j}} = \mathtt{C}(\mathtt{R_{rt_j}}, \mathtt{rt_j})$ *is the encryption of the type of relationship* $\mathtt{r_j}$ *using the random number* $\mathtt{r_{rt_j}}$*;* $\mathtt{C_{t_j}} = \mathtt{C}(\mathtt{R_{t_j}}, \mathtt{t_j})$ *is the encryption of the trust value of* $\mathtt{r_j}$ *using the random number* $\mathtt{r_{t_j}}$*. Moreover, we set* $\mathtt{D_{o_j}}$ *as the difference* $\mathtt{r_{o_j}} - \mathtt{r_{o_{j-1}}}$ *for $j = 2, \ldots, d$;* $\mathtt{D_{d_j}}$ *as the difference* $r_{d_{j+1}} - r_{d_j}$ *for $j = 1, \ldots, d-1$;* $\mathtt{D_{rt_j}}$ *as the difference* $r_{rt_j} - CRT.r_{rt_i}$ *for $j = 1, \ldots, d$ where* $\mathtt{CRT.r_{rt_i}}$ *is the random number used in the encryption of the corresponding relationship type in the* `CRT` *component of the anonymized access request entry or anonymized certificate distribution entry. The differences* $\mathtt{D_{o_0}}$ *and* $\mathtt{D_{d_n}}$ *are computed as* $\mathtt{D_{o_0}} = \mathtt{r_{o_j}} - \mathtt{r_r}$ *and* $\mathtt{D_{d_n}} = \mathtt{r_{d_j}} - \mathtt{r_{tr}}$ *respectively.*

- $\mathtt{D_t}$ *is equal to* $\mathtt{D_t} = \mathtt{r_t} - (\sum_{\mathtt{j=0}}^{\mathtt{d}} \mathtt{r_{t_j}})$*, where* $\mathtt{r_t}$ *is the randomness used to encrypt the trust of* `crt path` *in the anonymized access request entry or anonymized certificate distribution entry.*

Example 5 *Consider the social network in Figure 1, let* $\mathtt{e_{ac}}$ *be an access request entry corresponding to the access request explained in Example 3. Then, the corresponding anonymous access request entry is as follows:*

$$\mathtt{ae_{ac}} = \langle \mathtt{C}(\mathtt{R_r}, \mathtt{B}), \mathtt{C}(\mathtt{R_{tr}}, \mathtt{G}), \mathtt{C}(\mathtt{R_t}, 0.16),$$
$$\{(\mathtt{C}(\mathtt{R_{rt_1}}, \mathtt{pof}))\}, 0.16, 0.25, 2, \mathtt{C_p}\rangle$$

where $\mathtt{C_p}$ *is as follows:*

$$\mathtt{C_p} = \langle\langle \mathtt{C}(\mathtt{B}, \mathtt{r_{o_1}}), \mathtt{C}(\mathtt{E}, \mathtt{r_{d_1}}), \mathtt{C}(\mathtt{pof}, \mathtt{r_{rt_1}}), \mathtt{C}(0.4, \mathtt{r_{t_1}}),$$
$$(\mathtt{r_{0_1}} - \mathtt{r_r}), (\mathtt{r_{d_2}} - \mathtt{r_{d_1}}), (\mathtt{r_{rt_1}} - \mathtt{CRT.r_{rt_1}}), \mathtt{C}(\mathtt{E}, \mathtt{r_{o_2}}),$$
$$\mathtt{C}(\mathtt{G}, \mathtt{r_{d_2}}), \mathtt{C}(\mathtt{cof}, \mathtt{r_{rt_2}}), \mathtt{C}(0.4, \mathtt{r_{t_2}}), (\mathtt{r_{0_2}} - \mathtt{r_{0_1}}),$$
$$(\mathtt{r_{tr}} - \mathtt{r_{d_2}}), (\mathtt{r_{rt_2}} - \mathtt{CRT.r_{rt_1}})\rangle, (\mathtt{r_t} - (\mathtt{r_{t_1}} + \mathtt{r_{t_2}}))\rangle$$

Appendix A presents an algorithm for generating, given a private entry, the corresponding anonymized version. The anonymization process is based on Definitions 7, 8 and 9. Thus, the first part of the algorithm is devoted to generate the anonymized version of an access control entry. The requestor identifier (line 2), target node identifier (line 3), trust value of the path extracted from the certificate path `crt path` (line 4) and the set of possible relationship types of the access rule (lines 5-7) and are encrypted. Trust value of the path (line 8), and Minimum trust (line 9) and maximum depth (line 10) of the access rule are stored as a plain text, since this does not reveal sensitive information. The second part of the algorithm is devoted to generate the anonymized version of a certificate distribution entry. According to Definition 8, the requestor identifier (line 13), owner identifier (line 14) and the set of possible relationship types of the certificate distribution (lines 15-17) are encrypted. The relationship type between the owner and the sender of the certificate (line 18) and maximum depth allowed by the certification distribution rule (line 19) are stored as plain text. The last part of the algorithm is devoted to generate the anonymous path structure of `crt path`. Here, for each relationship j in the path p extracted from `crt path`, the components of $\mathtt{C_{cp}}$ described in Definition 9 are encrypted (line 21-37). Note that the algorithm also calculates the differences needed to check the correctness of the path and encrypt the trust of the path p (lines 38-39).

5.2 Audit file verification

In this section, we show how the data contained in an anonymized entry can be used by any WBSN participant to determine whether the corresponding action is compliant with the specified access and distribution rules. We recall that, this information is then exploited to automatically evaluate the user's reputation (see Section 4). In general, to verify whether an anonymized access control entry $\mathtt{ae_{ac}}$ or an anonymized certificate distribution entry $\mathtt{ae_{cd}}$ refers to a right or wrong decision, it is necessary to check the certificate path received together with the request of collaboration to which the entry refers to.

Since to protect user privacy, in both the anonymized entries, information about the certificate path are encrypted exploiting the ElGamal encryption scheme (see Definitions 7, 8 and 9), it is possible to verify the abovementioned conditions by exploiting the multiplicative homomorphic property

of this scheme. In particular, in order to verify whether the above conditions hold or not a user has to check that some of the encrypted values in the path correspond to the same plaintext.

According to ElGamal scheme, this can be done in the following way: to verify that an ElGamal ciphertext $C_1 = (R_1, s_1)$, obtained with randomness r_1, and a ciphertext $C_2 = (R_2, s_2)$, obtained with randomness r_2, correspond to the same plaintext, given the difference $r_1 - r_2$, one just has to check if

$$C_1 \oslash C_2 = (\frac{R_1}{R_2}, \frac{s_1}{s_2}) = (g^{r_1 - r_2}, pk^{r_1 - r_2})$$

where g is the generator of the group $\mathbb{G}$ and pk is the public key of the audit file owner.

Thus, provided with the differences between the random numbers exploited during the ElGamal encryption, an user is able to verify the certificate path correctness. According to Definition 9, all these differences are available in the second component of the anonymous path structure.

Formally, action verification and path reviewing process are described in Algorithms A and A in Appendix A. Given in input an anonymous entry, these algorithms compute all the action dimensions needed to compute the final participant reputation. These algorithms first checks if a given action is an action with a wrong decision or not, this check is done by the Algorithm A. Algorithm A checks all the possible combinations of dimensions producing a denial of service or an unauthorized dissemination (for both access control and certificate distribution actions). Here, it is necessary to know the participant decision (*i.e.* the *d*ec variable). Recall that, If a resource has to be released or a certificate distributed *dec* will be equal 1, 0 otherwise. For instance, lines 2-4 check if an access control decision is a denial of resource releasing (DRR), verifying if the path trust is lower than the minimum trust or the path depth is larger than the maximum depth of the corresponding rule and the participant decision is equal to 1. Note that, the path dimension is verified in lines 8-18 where all the components of each certificate of the certificate path `cer path` are checked.

Once the wrong decisions has been retrieved, the user can exploit the corresponding anonymous entries to compute the aggregate values introduced in Section 4, and then to compute the reputation value to being associated with the owner of the audit file. This is done in Algorithm A. This algorithm normalizes all components of wrong decisions (lines 5-9) and computes the final reputation value using the OWA operator (line 10).

6 Conclusions

In this paper, we have presented a complete framework to dynamically compute the reputation of a collaborative network participant, when trust is used to access control and privacy protection pourposes. The main characteristic of our approach is that reputation computation is done in a private way. We have also

described in details how to implement our framework in a Web-based Social Network.

As future work, we plan to develop a java-based implementation of the proposed framework in order to test its performance.

Acknowledgements

Partial support by the Spanish MEC (projects ARES – CONSOLIDER INGENIO 2010 CSD2007-00004 – and eAEGIS – TSI2007-65406-C03-02), the Generalitat de Catalunya (grant 2005-SGR-00093), and the European Community under the QUATRO Plus project (SIP-2006- 211001) is acknowledged. Jordi Nin wants to thank the Spanish Council for Scientific Research (CSIC) for his I3P grant.

References

[1] Adomavicius, G., Tuzhilin, A., (2005), Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6): 734749.

[2] Blaze, M., Feigenbaum, J., Lacy, J., (1996), Decentralized trust management, IEEE Conference on Security and Privacy.

[3] Blaze, M., (1999), Using the KeyNote trust management system, AT&T Research Labs.

[4] Carminati, B., Ferrari, E., Perego, A., (2006), Rule-based access control for social networks, LNCS, volume: 4278, pages: 1734-1744, Springer-Verlag.

[5] Carminati, B., Ferrari, E., Perego, A., (2007), Private relationships in social networks, Private Data Management 2007 workshop of the ICDE 2007.

[6] Diffie, W., Hellman M. E., (1976), New Directions in Cryptography, IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp: 644-654.

[7] ElGamal, T., (1985), A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Trans. on Information Theory, volume: 31, number: 4, 1985, pages: 469-472.

[8] Jøsang A., Ismail R. and Boyd C., (2007), A Survey of Trust and Reputation Systems for Online Service Provision. Decision Support Systems, volume: 43, number: 2, 2007, pages: 618-644.

[9] Golbeck J.A. and Hendler J., (2006), Inferring Binary Trust Relationships in Web-based Social Networks, ACM Transactions on Internet Technology, volume: 6, number: 4; 2006, pages: 497–529.

[10] Ferrari, E., Thuraisingham, B., Secure Database Systems, Advanced Database Technology and Design, Artech House, Inc., 2000, pages: 353-403.

[11] Korolova A., Motwani, R., Nabar, S., Xu, Y, (2008), Link Privacy in Social Networks, Proc. of ICDE 2008.

[12] Mui, L., (2002), PhD thesis: Computational models of trust and reputation: agents, evolutionary games and social networks, MIT.

[13] Nakajima, Y., Goudarzi, A., Enokido, T., Takizawa, M., (2008), Subjective and Objective Approaches to Obtaining Trustworthiness of Peers, AINA Workshops 2008: 313-318.

[14] Quisquater, J.J., Guillou, L.C., Berson, T., (1990), How to Explain Zero-Knowledge Protocols to Your Children. Advances in Cryptology - CRYPTO '89, pages: 628-631.

[15] Torra, V., Narukawa, Y. (2007) Modeling decisions: information fusion and aggregation operators, Springer.

[16] Yager, R. R., (1993), Families of OWA operators, Fuzzy Sets and Systems, volume: 59, pages: 125-148.

[17] Ylitalo, K., Kortesniemi, Y., (2005), Privacy in distributed reputation management. Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks, page: 63 - 71

[18] Zacharia, G., (1999), Collaborative Reputation Mechanisms for Online Communities, Phd. Thesis, Massachusetts Institute of Technology

A Algorithms

Require: $e = private\ entry$ and $e_t = entry\ type$.

if $(e_t = ac)$ **then**
  $ae.C_r = C(R_r, e.AcR.r)$ with the random number r_r
  $ae.C_{tr} = C(R_{tr}, e.AcR.AR.r_{tr})$ with the random number r_{tr}
  $ae.C_t = C(R_t, e.t_p)$ with the random number r_{t_p}
  for $(j = 0;\ j \leq e.AcR.AR.rt.length;\ j++)$ **do**
    $ae.CRT = ae.CRT \cup C(R_{rt_j}, e.AcR.AR.rt_j)$ with the random number r_{rt_j}
  end for
  $ae.t_p = e.t_p$
  $ae.t_{min} = e.AcR.AR_{rsc}.t_{min}$
  $ae.d_{max} = e.AcR.AR_{rsc}.d_{max}$
end if
if $(e_t = cd)$ **then**
  $ae.C_r = C(R_r, e.CdR.rl.r)$ with the random number r_r
  $ae.C_{owner} = C(R_a, owner)$ with the random number r_a
  for $(j = 0;\ j \leq e.CdR.DR.rt.length;\ j++)$ **do**
    $ae.CRT = ae.CRT \cup C(R_{rt_j}, e.CdR.DR.rt_j)$ with the random number r_{rt_j}
  end for
  $ae.rt = e.CdR.DR.rt$
  $ae.d_{max} = e.CdR.DR.d_{max}$
end if
for $(j = 0;\ j \leq e.crt\ path.length;\ j++)$ **do**
  $C_{o_j} = C(R_{o_j}, o_j)$ with the random number r_{o_j}
  $C_{d_j} = C(R_{d_j}, d_j)$ with the random number r_{d_j}
  $C_{rt_j} = C(R_{rt_j}, rt_j)$ with the random number r_{rl_j}
  if $(j = 0)$ **then**
    $D_o = r_r - r_{o_0}$
  else
    $D_o = r_{o_j} - r_{o_{j-1}}$
  end if
  if $(j = e.crt\ path.length)$ **then**
    $D_{l_j} = r_d - r_{d_j}$
  else
    $D_d = r_{d_j} - r_{d_{j-1}}$
  end if
  $D_{rt_j} = r_{rt_j} - e.CRT.r_{rt_j}$
  $ae.C_p = ae.C_p \cup \langle C_{o_j}, C_{d_j}, C_{rt_j}, D_o, D_d, D_{rt} \rangle$
end for
$D_t = r_t - (\sum_{j=0}^{e.crt\ path.length} r_{t_j})$
$ae.C_p = ae.C_p \cup \langle D_t \rangle$
return $ae = anonymous\ entry$.

Require: AE_{ac} = *set of access request entries*, AE_{cd} = *set of certificate distribution entries.*

1: $WDt_{ac}, WDd_{ac}, WDp_{ac} = dimensions\ for\ access\ control\ decisions$
2: $WDd_{cd}, WDp_{cd} = dimensions\ for\ certificate\ distribution\ decisions$
3: $Access\_control\_action\_verification(AE_{ac}, WDt_{ac}, WDd_{ac}, WDp_{ac})$
4: $Certificate\_distribution\_action\_verification(AE_{dc}, WDd_{cd}, WDp_{cd})$
5: $WDt_{ac} = Normalize(DRR_t) \cup Normalize(URD_t)$
6: $WDd_{ac} = Normalize(DRR_d) \cup Normalize(URD_d)$
7: $WDp_{ac} = Normalize(DRR_p) \cup Normalize(URD_p)$
8: $WDd_{cd} = Normalize(DCD_t) \cup Normalize(UCD_t)$
9: $WDp_{cd} = Normalize(DCD_p) \cup Normalize(UCD_p)$
10: $R = 1 - (OWA_Q(WDt_{ac}) + OWA_Q(WDd_{ac}) + OWA_Q(WDp_{ac}) + OWA_Q(WDd_{cd}) + OWA_Q(WDp_{cd}))/5$
11: **return** R

Require: $ae = anonymous\ entry$, $e_t = anonymous\ entry\ type$, $DS = boolean$, $US = boolean$.

1: $C_{tt} = 1$
2: **if** $(((ea.t_p < ea.t_{min})$ or $(ea.d_{max} < ea.C_p.length)$ or $(ae.C_r \neq ae.C_p.C_{ol_0})$ or $(ae.C_{tr} \neq ae.C_p.C_{ol_n}))$ and $(e_t = ac)$ and $(ea.dec = 1))$ **then**
3: US = true
4: **end if**
5: **if** $(((ea.d_{max} < ea.C_p.length)$ or $(ae.C_r \neq ae.C_p.C_{ol_0})$ or $(ae.C_{tr} \neq ae.C_p.C_{ol_n}))$ and $(e_t = cd)$ and $(ea.dec = 1))$ **then**
6: US = true
7: **end if**
8: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
9: **if** $((ae.C_p.C_{dl_i} \neq ae.C_p.C_{ol_{i+1}})$ or $(ae.C_p.C_{rtl_i} \neq ae.C_p.C_{rt_j}))$ and $(e_t = ac)$ and $(ea.dec = 1))$ **then**
10: $US = true$
11: **end if**
12: **if** $((ae.C_p.C_{dl_i} \neq ae.C_p.C_{ol_{i+1}})$ or $(ae.C_p.C_{rtl_i} \neq ae.C_p.C_{rt_j}))$ and $(e_t = cd)$ and $(ea.dec = 1))$ **then**
13: $US = true$
14: **end if**
15: **if** $(e_t = ac)$ **then**
16: $Ctt = C_{tt} \times ae.C_p.C_{tl_i}$
17: **end if**
18: **end for**
19: **if** $((ae.C_p.C_t \neq C_{tt})$ and $(e_t = ac)$ and $(ea.dec = 1))$ **then**
20: $US = tue$
21: **end if**
22: **if** $(((ea.t_p \geq ea.t_{min})$ and $(ea.d_{max} \geq ea.C_p.length)$ and $(ae.C_r = ae.C_p.C_{ol_0})$ and $(ae.C_{tr} = ae.C_p.C_{ol_n}))$ and $(e_t = ac)$ and $(ea.dec = 0))$ **then**
23: DS = true
24: **end if**
25: **if** $(((ea.d_{max} \geq ea.C_p.length)$ and $(ae.C_r = ae.C_p.C_{ol_0})$ and $(ae.C_{tr} = ae.C_p.C_{ol_n}))$ and $(e_t = cd)$ and $(ea.dec = 0))$ **then**
26: DS = true
27: **end if**
28: **return** US, DS

Require: $AE_{ac} = set\ of\ access\ request\ entries$,
$WDt_{ac}, WDd_{ac}, WDp_{ac} = dimensions\ for\ access\ control\ decisions.$

1: **for** $ae \in AE_{ac}$ **do**
2: DS = false, US = false, p = 0
3: Action_verification(ae,'ac',DS,US)
4: **if** $(US = true)$ **then**
5: **if** $(ea.t_p < ea.t_{min})$ **then**
6: $DRR_t = DRR_t \cup \langle |ae.t_{min} - ae.t_p| \rangle$
7: **end if**
8: **if** $(ea.d_{max} < ea.C_p.length)$ **then**
9: $DRR_d = DRR_d \cup \langle |ea.C_p.length - ea.d_{max}| \rangle$
10: **end if**
11: **if** $(ae.C_r \neq ae.C_p.C_{ol_0})$ or $(ae.C_{tr} \neq ae.C_p.C_{ol_n}))$ **then**
12: $p = p + 1$
13: **end if**
14: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
15: **if** $(ae.C_p.C_{dl_i} \neq ae.C_p.C_{ol_{i+1}})$ **then**
16: $p = p + 1$ break
17: **end if**
18: **end for**
19: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
20: **if** $(ae.C_p.C_{rtl_i} \neq ae.C_p.C_{rt_j})$ **then**
21: $p = p + 1$ break
22: **end if**
23: **end for**
24: **if** $(p \neq 0)$ **then**
25: $DRR_p = DRR_p \cup \langle p \rangle$
26: **end if**
27: **end if**
28: **if** $(DS = true)$ **then**
29: p = 0
30: **if** $(ea.t_p \geq ea.t_{min})$ **then**
31: $URD_t = URD_t \cup \langle |ae.t_{min} - ae.t_p| \rangle$
32: **end if**
33: **if** $(ea.d_{max} \geq ea.C_p.length)$ **then**
34: $URD_d = URD_d \cup \langle |ea.C_p.length - ea.d_{max}| \rangle$
35: **end if**
36: **if** $(ae.C_r = ae.C_p.C_{ol_0})$ or $(ae.C_{tr} = ae.C_p.C_{ol_n}))$ **then**
37: $p = p + 1$
38: **end if**
39: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
40: **if** $(ae.C_p.C_{dl_i} = ae.C_p.C_{ol_{i+1}})$ **then**
41: $p = p + 1$ break
42: **end if**
43: **end for**
44: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
45: **if** $(ae.C_p.C_{rtl_i} = ae.C_p.C_{rt_j})$ **then**
46: $p = p + 1$ break
47: **end if**
48: **end for**
49: **if** $(p \neq 0)$ **then**
50: $URD_p = URD_p \cup \langle p \rangle$
51: **end if**
52: **end if**
53: **end for**
54: **return** $WDt_{ac}, WDd_{ac}, WDp_{ac}$

Require: $AE_{cd} = set\ of\ certificate\ distribution\ entries$,
$WDd_{cd}, WDp_{cd} = dimensions\ for\ certificatedistribution\ decisions.$

1: **for** $ae \in AE_{cd}$ **do**
2: DS = false, US = false, p = 0
3: Action_verification(ae,"cd",DS,US)
4: **if** $(US = true)$ **then**
5: **if** $(ea.d_{max} < ea.C_p.length)$ **then**
6: $CDC_d = CDC_d \cup \langle |ea.C_p.length - ea.d_{max}| \rangle$
7: **end if**
8: **if** $(ae.C_r \neq ae.C_p.C_{ol_0})$ or $(ae.C_{tr} \neq ae.C_p.C_{ol_n}))$ **then**
9: $p = p + 1$
10: **end if**
11: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
12: **if** $(ae.C_p.C_{dl_i} \neq ae.C_p.C_{ol_{i+1}})$ **then**
13: $p = p + 1$ break
14: **end if**
15: **end for**
16: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
17: **if** $(ae.C_p.C_{rtl_i} \neq ae.C_p.C_{rt_j})$ **then**
18: $p = p + 1$ break
19: **end if**
20: **end for**
21: **if** $(p \neq 0)$ **then**
22: $DCD_p = DCD_p \cup \langle p \rangle$
23: **end if**
24: **end if**
25: **if** $(DS = true)$ **then**
26: p = 0
27: **if** $(ea.d_{max} \geq ea.C_p.length)$ **then**
28: $UCD_d = UCD_d \cup \langle |ea.C_p.length - ea.d_{max}| \rangle$
29: **end if**
30: **if** $(ae.C_r = ae.C_p.C_{ol_0})$ or $(ae.C_{tr} = ae.C_p.C_{ol_n}))$ **then**
31: $p = p + 1$
32: **end if**
33: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
34: **if** $(ae.C_p.C_{dl_i} = ae.C_p.C_{ol_{i+1}})$ **then**
35: $p = p + 1$break
36: **end if**
37: **end for**
38: **for** $(i = 0;\ i < ae.C_p.length;\ i++)$ **do**
39: **if** $(ae.C_p.C_{rtl_i} = ae.C_p.C_{rt_j})$ **then**
40: $p = p + 1$ break
41: **end if**
42: **end for**
43: **if** $(p \neq 0)$ **then**
44: $UCD_p = UCD_p \cup \langle p \rangle$
45: **end if**
46: **end if**
47: **end for**
48: **return** WDd_{cd}, WDp_{cd}