2009 33rd Annual IEEE International Computer Software and Applications Conference

Computing Reputation for Collaborative Private Networks

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

Abstract

*The use of collaborative network services is increasing, therefore, the protection of the resources and relations shared by network participants is becoming crucial. One of the main issues in such networks is the evaluation of participant reputation, since network resources access may or may not be granted on the basis of the reputation of the requesting node. Therefore, the calculation of the reputation of the nodes becomes a very important issue. There are several reputation models presented in the literature. Some of these models (*e.g.*, Ebay or Sporas) are very simple and participants cannot express their preferences in the reputation computation process. On the contrary, there are other reputations models (*e.g.*, Reget or Fire) too complex to be applied when privacy is a primary concern.*

In this paper, we propose a new reputation model based on OWA and WOWA operators. The key characteristics of our proposal are that reputation is computed in a private way using the homomorphic properties of elGamal cryptosystem and it is possible to introduce user preferences inside reputation computation. We present the feasibility of this new reputation model by considering a Web-based Social Network scenario.

1. Introduction

In several applications, such as peer-to-peer systems [17], collaborative/social networks [7], or recommender systems [3] 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 [15], 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. In 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 therefore 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 [16], whose definition may vary depending on the context and for which purposes they are used. For instance, in current Web-based Social Networks [12], trust is a measure of how much a user trusts another node 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 is used for access control/privacy protection [7, 6], it 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 [11].

However, regardless of the specific notion of trust/reputation adopted, it is fundamental to devise mechanisms that help to automatically (or partially automatically) compute their value. In the literature, there are several proposals in this respect for a variety of scenarios. Probably the most widely used are those applied by eBay [2] and Amazon [1]. It is well-known that such approaches have several problems, such as the necessity of a central node [4] or the negative effect of considering all trust ratings in the same way [26]. For this reason, other (more complex) approaches have been proposed [9, 19, 20, 26]. However, none of them considers privacy threats related to trust computation. In contrast, our goal is to consider a scenario where trust is one of the parameters on which access control is based, and where the privacy of both network relationships and information about user decisions, which are instrumental for reputation computation, have to be protected (we refer to this scenario as *Collaborative Private Network*).

Even though there is a lot of research in the field of pri-

0730-3157/09 $25.00 © 2009 IEEE
DOI 10.1109/COMPSAC.2009.40

vacy enhancing technologies, the work directly analyzing privacy concerns in reputation systems seems to be limited, particularly when trust is used for access control purposes. Usually, when privacy issues are addressed, trust computation services are centralized and the reputation information is stored in a trusted central repository [25]. Obviously, this can be the bottleneck for the system. On the contrary, when trust computation services are distributed [4] user privacy is disregarded. We believe that the support for both these features it fundamental for the widespread adoption of collaborative platforms. With regard to reputation systems, there are few proposals for privacy of the raters, such as *e.g.* [14], where rates are interchanged using a public key encryption schema. In this protocol, network participants store their own reputation rates without any involvement from a central authority. However, it is mandatory a trusted third party (TTP) checks all the rates, therefore a new bottleneck arises.

In this paper, we revisit some of the reputation models presented in the literature describing the reasons because they are not suitable for our target scenario. For this reason, we describe a new reputation computation model, based on OWA [24] and WOWA [21] aggregation operators, that in combination with the multiplicative homomorphic properties of the ElGamal crypto-system [10] is able to compute user reputation in a private way using an encrypted public audit file in a complete decentralized way

The rest of this paper is organized as follows. In Section 2 we explain some preliminary concepts about ElGamal and aggregation functions. In Section 3 we review some well-known reputation models. In Section 4, we present our target scenario. Section 5 describes our reputation computation model for private collaborative networks and gives a detailed description of our audit system. Finally, in Section 7 the paper draws some conclusions and future work.

2. Background

2.1 ElGamal

The ElGamal [10] encryption system is a public key encryption algorithm which is based on the Diffie-Hellman [8] 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:

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

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

Specifically, to compute such a proof, let $H : \{0,1\} \rightarrow \mathbb{Z}_q$ be a cryptographic hash function. The prover acts as follows.

1. Choose at random $u_1, u_2 \in \mathbb{Z}_q$, compute $T_1 = g^{u_1}$, $T_2 = g^{u_2}$ and $T_3 = pk^{u_1 - u_2}$.
2. Compute $h = H(g, R_1, R_2, s_1, s_2, T_1, T_2, T_3)$.
3. Compute $w_1 = u_1 - r_1 h$ and $w_2 = u_2 - r_2 h$.
4. Define the proof as (h, w_1, w_2).

To verify the correctness of a proof (h, w_1, w_2) with respect to ciphertexts $c_1 = (R_1, s_1)$ and $c_2 = (R_2, s_2)$ and public key pk, one checks if

$$h = H\left(g, R_1, R_2, s_1, s_2, g^{w_1} R_1^h, g^{w_2} R_2^h, pk^{w_1 - w_2} \cdot (s_1/s_2)^h \right).$$

Using standard techniques (see [5, 23] for some similar proofs), it can be proved that this protocol enjoys all the required zero-knowledge properties [13]. In particular, no information about the random values r_1, r_2 and about the common plaintext encrypted in c_1 and c_2 is leaked.

2.2 Aggregation functions

Aggregation functions [22] are numerical functions used for information fusion. They typically combine N numerical values supplied by N sources into a single datum. Probably the most widely used aggregation functions are the arithmetic and the weighted mean. The former gives the same importance to all data sources, whereas the latter gives a different importance (reliability) to each data source.

In this work, we will consider a different family of aggregation functions. First of all, we consider the Ordered Weighted Aggregation (OWA) operator [24]. Two definitions exist for OWA, 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 adopt this latter definition, that uses fuzzy quantifiers.

Definition 1 *A function* $Q : [0,1] \rightarrow [0,1]$ *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* $Q(x) \geq Q(y)$.

Using fuzzy quantifiers, the OWA operator [24] 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 is, the largest is the importance for the largest values being aggregated. In contrast, the largest the α is, the lowest the importance of the largest values (and the largest the importance given to low values) is.

However in many cases, it is necessary to model situations in which one is interested in taking into account both the importance of the values and the importance of the information sources. The second aggregation function considered in this work is the WOWA operator [21]. It was introduced to deal with this kind of situations.

Definition 3 *Let* Q *be a non-decreasing fuzzy quantifier and let* p *be a weighting vector of dimension* N*; then,* $WOWA_{p,Q} : \mathbb{R}^N \rightarrow \mathbb{R}$ *is a* Weighted Ordered Weighting Averaging (WOWA) operator *of* dimension N *if:*

$$WOWA_{p,Q}(a_1, ..., a_N) = \sum_{i=1}^{N} \omega_i a_{\sigma(i)}$$

where σ *is defined as in the case of the OWA operator, and the weight* ω_i *is defined as:*

$$\omega_i = Q(\sum_{j \leq i} p_{\sigma(j)}) - Q(\sum_{j < i} p_{\sigma(i)}).$$

3 Reputation Models

Surely the most widely used reputation models are those applied in eBay [2] and Amazon [1]. Both these models are implemented as a centralized rating system so that users can rate and learn about the reputation of the other network participants. For instance in Ebay, after each interaction, a network user can rate its partner with a trust value $t_i \in [-1, 1]$, the largest is the value the best is the reputation. Such ratings are stored in a central node and reputation R is computed as $R = \sum t_i$. These models are very intuitive and easy to apply, however, they are too simple (in terms of their trust ratings and the way they are aggregated) for collaborative networks. Another important problem of these models is that new users start with a reputation $R = 0$, then if a network participant has a low reputation (lower than 0), it is better for such user to change his/her (old) identity for a new identity with reputation equal to 0.

Sporas [26] extends the eBay and Amazon models by introducing a new method for trust aggregation and fixing the problems with negative reputation values. Specifically, it does not store all the ratings but rather it updates the global reputation value of a network participant according to its most recent rating. More formally, reputation is computed according to Definition 4 above.

Definition 4 *Let* A *and* B *be two network users. Let us assume that at time i, user* A *interacts with user* B*. After such interaction user* B *modifies the reputation value of* A *using the following formula:*

$$R^A_{i+1} = R^A_i + \frac{1}{\Theta}\sum_{j=1}^{i} \Phi(R^A_j) R^B_i (t_{i+1} - E(t_{i+1}))$$

$$\Phi(R) = 1 - \frac{1}{1+e^{\frac{-(R-D)}{\sigma}}} \quad E(t_{i+1}) = \frac{R_i}{D}$$

where Θ *is a constant greater than 1,* t_i *stands for the trust value given by user* B*,* R^B_i *stands for the the reputation of user* B*,* D *is the range of the reputation values and* σ *is the acceleration factor of the damping function* $\Phi(R)$.

In Sporas, the initial reputation of a new user is equal to 0 and it can increase up to the maximum of D. As trust values are always positive ($t_i \in [0.1, 1]$), it is guaranteed that no user can ever have a reputation value lower than the reputation of a new user. Then, it is useless for a network participant to change his/her identity. The damping function $\Phi(R)$ ensures that reputation of trustworthy participants are more robust against temporary malicious attacks, in other words, one network user cannot reduce the reputation of another user adding low trust ratings. The main drawbacks of Sporas are that it is a centralized approach as Ebay/Amazon and it only considers direct interactions (rates) when reputation is calculated, disregarding in this way other network information, such as the role of the node in the network.

To solve these two drawbacks, other methods, such as for instance Regret [20], Fire [9] or Repage [19], have been proposed in the literature. On the one hand, to avoid the presence of a central node, these models decentralize reputation computation, and each user stores his/her ratings into a local log file. On the other hand, These models also separate direct ratings (*i.e.*, auctions evaluation) from indirect ratings (*i.e.*, witness evaluations or neighbors evaluations) and role ratings (*e.g.*, playing the same role in a company influences trust computation). The goal of considering different data sources for reputation computation is to increase the quality of the reputation computation reducing the effect of the noise in ratings. These models aggregate the different data sources using the arithmetic or the weighted mean.

These models improve the quality of the previously discussed reputation models and they fix the problem of the central node. However to be applied, it is necessary that all network relationships are public. As we have introduced before, our target scenario are collaborative networks where relationships may be kept private. Therefore, these recent models are unsuitable for this scenario. For this reason, in Section 5 we will present a model specifically coinceved for private collaborative networks.

4 Private Collaborative Networks

A common way to model a collaborative network is as a directed labelled graph, where nodes correspond to network members and edges denote relationships between two different members. In particular, the initial node of an edge denotes the member who established the corresponding relationship and the terminal node denotes the member who accepted to establish the relationship, whereas the label represents the type of the established relationship. Since a relevant feature of collaborative networks is that relationships are characterized by a trust level, representing how much a given member considers trustworthy another member with whom he is establishing a relationship, it is assumed that each edge has a further label, modeling the trust level t ($t \in [0, 1]$). Two members A and B are in a relationship of a given type rt if there is a path, consisting only of edges labelled with the 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* (it is assumed that relations are transitive).

In our reference collaborative private network, access control and privacy requirements are expressed and enforced according to the model presented in [7, 6], that we briefly summarize in what follows. As far as access control is concerned, each resource to be shared in the network 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. 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 rsc. 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 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. For the sake of simplicity, in this paper we assume that access control rules are composed by only one access condition. Moreover, we assume that each resource is administered by the owner and by the users to which the owner possibly delegates the administration. Since the delegation model is out of the scope of this paper, we simply assume that once a user has been authorized to access a given resource, he/she becomes automatically authorized to administer that resource, that is, to grant or deny accesses on it to other participants. Other delegation models can be used as well without requiring a modification of the proposed framework.

One of the main characteristics of the access control model presented in [7] is that access control enforcement is performed client-side. According to this approach, 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. This implies that the requestor has to provide the owner a *proof* showing that there exist the relationships required by the specified access rules, with the required depth and trust level. For this reason, each relationship is certified by the members participating in it and the certificate is distributed to the other network 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.

To enforce privacy requirements on network relationships, each certificate has an associated *distribution rule* controlling its distribution, stating the set of users authorized to know the existence of the corresponding relationship. For the sake of simplicity, here, we consider that certificates are network resources, therefore the certificate access is enforced in the same way than for the other network resources. Therefore, an intruder cannot have access neither the user relationships nor the network resources.

5 Private Reputation Computation

Now, we introduce two different reputation computation models that can be applied to the scenario described before. The first one does not consider temporal information, *i.e.* past actions have the same importance than recent ones, while in the second model such information is considered.

5.1 Basic Reputation Model

When a participant performs all decisions in accordance with the specified 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 correctly enforce access control rules, his/her reputation level should be lower.

To formalize the model, we need to introduce the possible wrong decisions a participant can make in the reference scenario. These are identified by considering the main request of collaboration a participant receives in a private collaborative network: the request for *releasing a resource*, according to the specified access control rules. Then, we have a wrong decision when a user decides to deny a owned or delegated resource to an authorized user (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 further kind of wrong decisions comes when a user decides to disseminate resources to non-authorized users. For example, a user releases a resource rsc when the provided proof is not correct. We call this wrong decision *unauthorized resource dissemination* (URD).

In devising the model for reputation computation, we consider that not all wrong decisions have the same relevance. To model the relevance of wrong decisions, we classify them according to three dimensions. 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 AC_t and called *trust dimension*, corresponds to the difference between the minimum trust required in AR and the trust computed on the basis of the path extracted from $crt\ path$. Note that when AC_t is lower than zero and the resource has not been released, A performs a *DRR*. In contrast, if AC_t is greater than zero and the resource has been released, A performs a *URD*. As we are only interested in the negative values when A performs a *DRR* action or positive ones when A performs a *URD* action, we can compute the absolute value of AC_t, therefore AC_t is always in the $[0, 1]$ interval. The second dimension, referred to as *depth dimension*, denoted as AC_d, is defined as the difference between the depth required in AC and the depth of the path extracted from $crt\ path$, AC_d is unbound and therefore its interval is $[0, \infty]$. The third dimension, called *path dimension* denoted as AC_p, is used to model situations as the next one. Let us consider an access rule consisting only of the following access condition $AC = (B, rt_1, 3, 0.5)$, and let $p = (A \rightarrow_{rt_1} B)(B \rightarrow_{rt_2} E)$ be the path extracted from $crt\ path$. Obviously, this $crt\ path$ is not a valid proof, since the rt_2 relationship is not referred by the access rule, and the last node in the path is not the node specified in the access condition. Formally, AC_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.

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 value 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.

Let us now formalize how we combine all the wrong decisions taken by a 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 2.2) 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 5 *Let A be a user, let AC_{SET_A} be the set of access control decisions made by A. Let $DRR_t \subseteq AC_{SET_A}$ be the set of DRR wrong decisions with respect to trust (whose AC_t trust dimension is lower than 0). Let $URD_t \subseteq AC_{SET_A}$ be the set of URD wrong decisions with respect to trust (whose AC_t trust dimension is greater than 0). Then, the set of wrong decisions related to access control with respect to trust is defined as $WD_{t_A} = DRR_t \cup URD_t$, and the aggregated value $AGt_{AC_{SET_A}}$ of wrong decisions in WDt_A with respect to the trust dimension is given by the following formula:*

$$AGt_{AC_{SET_A}} = OWA_Q(wd_1, \ldots, wd_{|WD_{t_A}|})$$

where Q is a non-decreasing fuzzy quantifier, and wd_i are the absolute values of the AC_t in WD_{t_A} (as OWA is a symmetric function, the ordering of the wd_i is irrelevant).

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

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

Definition 6 *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, depth and path dimension, respectively. The reputation value R_A of user A is defined as:*

$$R_A = 1 - \frac{1}{3}(AGt_{AC_{SET_A}} + AGd_{AC_{SET_A}} + AGp_{AC_{SET_A}})$$

5.2 Damping Reputation Model

In the basic reputation model introduced before, the weight (importance) of a wrong decision is assigned taking into account only its value. This allows us to consider as more important for reputation computation wrong decisions with a large impact. For example, not to release a resource whose access rule has depth equal to 4 has not the same importance as not to release a resource 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. However, as we have explained in Section 3, it is also important to take into account the time when the wrong decision has been done, since past actions usually are less important than more recent actions.

To easily take into account this temporal dimension in our reputation model, we propose to use the WOWA operator instead of the OWA operator, when trust values are aggregated. To do this, we can use the same fuzzy quantifier applied in the basic model, of course the interpretation of such quantifier is exactly the same as in the OWA operator, but we have to define a weighting vector p. This weighting vector will have the same interpretation than the weighting vector of the weighted mean: it measures the importance of each data source.

In our scenario the importance of each data source (wrong action) is given by the time the action has been done. Then, we can built the weighting vector p using a damping function Φ, as it was introduced in Sporas. Considering the weighting vector p we can re-formulate the aggregation of the trust values using the following definition.

Definition 7 *Let A be a user, let AC_{SET_A} be the set of access control decisions made by A. Let $DRR_t \subseteq AC_{SET_A}$ be the set of DRR wrong decisions with respect to trust (whose AC_t trust dimension is lower than 0). Let $URD_t \subseteq AC_{SET_A}$ be the set of URD wrong decisions with respect to trust (whose AC_t trust dimension is greater than 0). Then, the set of wrong decisions related to access control with respect to trust is defined as $WD_{t_A} = DRR_t \cup URD_t$, and the aggregated value $AGt_{AC_{SET_A}}$ of wrong decisions in WDt_A with respect to the trust dimension is given by the following formula:*

$$AGt_{AC_{SET_A}} = WOWA_{p,Q}(wd_1, \ldots, wd_{|WD_{t_A}|})$$

where Q is a non-decreasing fuzzy quantifier, p is weighting vector, and wd_i are the absolute values of the AC_t in WD_{t_A}. Note that WOWA is not a symmetric function, then the ordering of the wd_i is now relevant, in this case wrong decisions have to be sorted considering when the decision was taken (from the oldest to the newest).

The reputation value can then be computed as before.

6 Audit files

In order to compute reputation, each user has to store the decisions made with regards to access control into a private audit file. For privacy purposes, After that, each node has also to create an anonymized audit file available to all users. Here, we formally introduce the information stored into the private and anonymized files. Then, we discuss how, by using the anonymized information, a user is able to determine whether a decision has been correctly evaluated.

6.1 Audit File Generator

Let us start to consider the private audit file. Since we are interested in computing reputation according to the way a user answers to an access control request, the audit file has to contain information about the decisions made by a node when he/she receives such requests. To model this type of decisions, each entry contains information about the access request, the access rule associated with the requested resource as well as the certificate path provided as proof by the requestor. The formal definition of the private audit file entry is the following.

Definition 8 *Let AcR=(rsc, ts, r, AR) be an access request, where rsc is the identifier of the requested resource, ts is a timestamp, r is the identifier of the requestor, and AR is the access rule applied to resource rsc. The corresponding access request entry is defined as:*

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

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

According to the proposed reputation computation strategy, the anonymized audit file is generated by anonymizing each entry of the audit file. In the following, we introduce the formal definition of the anonymized access request entry. In this definition, we exploit notation $C(R, text)$ to denote the ElGamal encryption of $text$ by using the random number r. Recall that to encrypt a message using elGamal, we take randomly a value (in our case r) and we compute $R = g^r mod\, p$ (g and p are the parameters of elGamal defined in Section 2.1). R is needed to decrypt the ciphertext $text = m \cdot pk^r mod\, p$ (pk is the public key of the user).

Definition 9 *Let A be the owner of an audit file f, and let e_{ac} be an access request entry stored into f. The corresponding anonymized access request entry is defined as:*

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

where $C_r = C(R_r, r)$ is the encryption, using the random number r_r, of the requestor identifier stored into $e_{ac}.AcR$[1]*; $C_{tr} = (R_{tr}, AcR.AR.v)$ is the encryption, using the random number r_{tr}, of the target node of the access rule $e_{ac}.AcR.AR$; $C_t = (R_t, e_{ar}.t_p)$ is the encryption, using the random number r_t, of the trust value $e_{ac}.t_p$; $C_{rt} = (R_{rt}, AcR.AR.rt)$ is the encryption, using the random number r_{rt}, of the relationship type of the access rule $e_{ac}.AcR.AR$; t_p is the trust value associated with the path extracted from $crt\ path$; t_{min} is the minimum trust required by $e_{ar}.AcR.AR$, d_{max} is the maximum depth required by $e_{ar}.AcR.AR$; dec is a boolean value set to 1, if the resource rsc has been released to requestor r, set to 0, otherwise, ts is the timestamp, and C_p is the anonymous path structure defined on the path extracted from $crt\ path$ (Definition 10).*

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 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; (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); (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; (d) whether p's depth is less than or equal to the maximum depth required in AR; (e) 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 is 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 a message is encrypted with 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 10 *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. 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 $C_p = \langle C_{cp}, D_t \rangle$ where:*

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

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

  where $C_{o_j} = C(R_{o_j}, o_{id_j})$ is the encryption of the identifier of the user with which r_j is established using the random number r_{o_j}; $C_{d_j} = C(R_{d_j}, d_j)$ is the encryption of the identifier of the user who established r_j using the random number r_{d_j}; $C_{rt_j} = C(R_{rt_j}, rt_j)$ is the encryption of the type of relationship r_j using the random number r_{rt_j}; $C_{t_j} = C(R_{t_j}, t_j)$ is the encryption of the trust value of r_j using the random number r_{t_j}. Moreover, we set D_{od_j} as the difference $r_{o_j} - r_{d_{j-1}}$ for $j = 2, \ldots, d-1$; D_{rt_j} as the difference $r_{rt_j} - r_{rt}$ for $j = 1, \ldots, d$ where r_{rt} is the random number used in the encryption of the relationship type in the anonymized access request entry. Remember that these differences are needed to use elGamal as a zero-knowledge proof (see Section 2.1 for more details).

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

As all entries are encrypted is impossible for an intruder to discover from the anonymized log files some precise information on the performed actions.

6.2 Audit file verification

In this section, we show how the data contained in an anonymized entry can be used by any network participant to determine if the corresponding action is compliant with the specified access rules. We recall that, this information is then exploited to evaluate user's reputation (see Section 5). In general, to verify whether an anonymized access control entry ae_{ac} 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.

---

[1] Here and in what follows, we use the dot notation to denote specific components within a tuple.

Since to protect user privacy, information about the certificate path are encrypted exploiting the ElGamal encryption scheme (see Definitions 9 and 10), it is possible to verify the aforementioned 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. Such proof can be done following the protocol described in Section 2.1.

Due to lack of space, the complete algorithms and some examples are omitted, but they can be found in [18].

7 Conclusions

In this paper, we have revisited some of the most well-known reputation models. As we have shown, none fulfills all requirements for computing user reputation in private collaborative networks. Therefore, we have presented two different reputation models that use OWA and WOWA operators to aggregate the different trust values obtained from the access control decisions made by a network participant and that are well suited for the private network scenario. We have also described how to develop the audit file generator needed to apply our reputation models.

We are currently developing an implementation of the proposed framework in order to test its performance and the effectiveness of the proposed reputation models.

Acknowledgements

Partial support by the Spanish MEC (projects ARES – CONSOLIDER INGENIO 2010 CSD2007-00004 – and eAEGIS – TSI2007-65406-C03-02), the European Community QUATRO Plus project (SIP-2006-211001) and the Italian Ministry of University, Education and Research ANONIMO project (PRIN-2007F9437X 004) are acknowledged.

References

[1] Amazon site, http://www.amazon.com.

[2] Ebay site, http://www.ebay.com.

[3] G. Adomavicius and A. Tuzhilin. 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):734–749, 2005.

[4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *EEE Conf. on Security and Privacy*, 1996.

[5] J. Camenisch. *Group signature schemes and payment systems based on the discrete logarithm problem*. PhD thesis, ETH Zurich. Diss. ETH No. 12520, 1998.

[6] B. Carminati, E. Ferrari, and A. Perego. Decentralized security framework for web-based social networks. *Int. J. of Information Security and Privacy*, 2(4):22–53, 2008.

[7] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in web-based social networks. *ACM Tran.on Information & System Security*, page in press, 2008.

[8] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Tran. on Information Theory*, 22(644-654), 1976.

[9] T. Dong-Huynha, N. Jennings, and N. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *Eu. Conf. of Artificial Intelligence*, 2004.

[10] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[11] E. Ferrari and B. Thuraisingham. *Advanced Database Technology and Design*, chapter Secure Database Systems, pages 353–403. Artech House, Inc., 2000.

[12] J. Golbeck and J. Hendler. Inferring binary trust relationships in Web-based social networks. *ACM Tran. on Internet Technology*, 6(4):497–529, 2006.

[13] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1), 1989.

[14] R. Ismail, C. Boyd, A. Jøsang, and S. Russell. Private reputation schemes for p2p systems. In *2nd Int. Workshop on Security In Information Systems*, pages 196–206, 2004.

[15] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[16] L. Mui. *Computational models of trust and reputation: agents, evolutionary games and social networks*. PhD thesis, Massachusetts Institute of Technology, 2002.

[17] Y. Nakajima, A. Goudarzi, T. Enokido, and M. Takizawa. Subjective and objective approaches to obtaining trustworthiness of peers. In *AINA Workshops*, pages 313–318, 2008.

[18] J. Nin, B. Carminati, E. Ferrari, and V. Torra. Dynamic reputation-based trust computation in private networks. Technical Report 02, IIIA-CSIC, 2009.

[19] J. Sabater, M. Paolucci, and R. Conte. Repage: Reputation and image among limited autonomous partners. *Artificial Societies and Social Simulation*, 9(2):3, 2006.

[20] J. Sabater and C. Sierra. Regret: reputation in gregarious societies. In *5th Int. Conf. on Autonomous Agents*, 2001.

[21] V. Torra. The weighted owa operator. *Internation Journal of Intelligence Systems*, 12:153–166, 1997.

[22] V. Torra and Y. Narukawa. *Modeling decisions: information fusion and aggregation operators*. Springer, 2007.

[23] J. H. V. Daza and G. Sáez. On the computational security of a distributed key distribution scheme. *IEEE Transactions on Computers*, 57(8):1087–1097, 2008.

[24] R. R. Yager. Families of OWA operators. *Fuzzy Sets and Systems*, 59:125–148, 1993.

[25] G. Zacharia. *Collaborative Reputation Mechanisms for Online Communities*. PhD thesis, MIT, 1999.

[26] G. Zacharia and P. Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881–908, 2000.