Multi-Party Trust Computation in Decentralized Environments

Tassos Dimitriou
Athens Information Technology
19.5 km Markopoulo Ave., 19002, Peania
Athens, Greece
Email: tdim@ait.edu.gr

Antonis Mihalas
Athens Information Technology, Greece
and
Aalborg University, Denmark
Email: amic@ait.edu.gr

***Abstract*—In this paper, we describe a decentralized privacy-preserving protocol for securely casting trust ratings in distributed reputation systems. Our protocol allows n participants to cast their votes in a way that preserves the privacy of individual values against both internal and external attacks. The protocol is coupled with an extensive theoretical analysis in which we formally prove that our protocol is resistant to collusion against as many as $n-1$ corrupted nodes in the semi-honest model.**

The behavior of our protocol is tested in a real P2P network by measuring its communication delay and processing overhead. The experimental results uncover the advantages of our protocol over previous works in the area; without sacrificing security, our decentralized protocol is shown to be almost one order of magnitude faster than the previous best protocol for providing anonymous feedback.

***Index Terms*—Decentralized Reputation Systems, Security, Voter Privacy, Anonymous feedback**

I. Introduction

During the last decade, the field of online communities has met great development. The main reasons of this evolution are not only the ease of comprehension and use, but also the large degree of accessible information and services. These aspects facilitate the exchange of information among unknown users while giving them the option to keep their identity hidden.

Nevertheless, the difficulty of gathering (reliable) evidence about unidentified transaction partners makes it hard to decide if a user is legitimate or corrupted, or to differentiate between a high and a low quality service provider. As a result, the topic of trust in computer networks is receiving significant attention in both the academic community and the e-commerce industry [11]. *Trust management* has been proposed by many researchers as a solution for providing a minimum level of security between two or more entities that belong to the same network and want to make reliable transactions or interactions with each other.

A well known technique that intends to help users avoid interacting with malicious or unreliable ones is the use of *reputation systems*. A reputation system rates the behavior of each user, based on the quality of the provided service(s), and gives appropriate information to the whole community in order to decide whether to *trust* or not a particular entity in the network.

However, one concern about reputation systems, which has received relatively little attention in the literature, is that of *feedback providers' privacy*. Although there are many reputation and trust establishment schemes, only some of them deal with the problem of securing the ratings (or votes) of participating nodes. This lack of privacy can lead to several problems, including the proper operation of the network. Additionally, the absence of schemes that provide privacy in decentralized environments, such as ad hoc networks, is even greater. For example, it has been observed in [14] that users of a reputation system may avoid providing honest feedback in fear of retaliation, if reputation scores cannot be computed in a privacy-preserving manner. In response to that, eBay has decided to change the feedback policy so that sellers can no longer leave negative/neutral feedback for buyers, claiming that "it will help buyers leave an honest feedback" [6].

Hence the development of reputation protocols that can be used to provide anonymous feedback is essential to the survivability of online communities and electronic marketplaces. In some sense, provision of anonymous feedback to a reputation system is analogous to that of anonymous voting in electronic elections. It potentially encourages truthfulness by guaranteeing secrecy and freedom from explicit or implicit influence.

Until now, many reputation and trust establishment schemes that preserve privacy have been proposed mostly for traditional (centralized) environments where there is a standard topology and the connectivity between nodes is not an issue. In contrast, there is a lack of research that targets *decentralized* environments such as ad hoc networks. These kinds of networks offer new challenges and opportunities for research for two main reasons: First, because collection of evidence is difficult due to the mobility or the resource constraints of the nodes that further restrain the trust evaluation process. Second, not only because submission of "votes" must be kept hidden from all nodes but also because it has to be distributed to the whole network due to lack of trusted authorities.

Contribution: In this work we present a protocol that preserves the privacy of votes in decentralized environments. The protocol allows n participants to securely cast their ratings in a way that preserves the privacy of individual votes against both internal and external attacks. More precisely, we analyze the protocol and prove that it is resistant to collusion even against up to $n-1$ corrupted insiders. The insights we obtain from this analysis allow us to refine the protocol and come up with a lighter version that is equally secure and uses only standard cryptographic mechanisms. This lighter protocol compares favorably with protocols for secure multi-party sum computation and we consider it as another important contribution of this work. Finally, the whole analysis is coupled with extensive experimental results that demonstrate the protocol's validity and efficiency over previous works in the area.

Organization of the paper: In Section II, we review some of the most important schemes that provide private trust ratings in decentralized environments. In Section III, we describe the problem of secure trust aggregation and we define the basic terms that we use in the rest of the paper. In Section IV, we present StR, our main protocol, while in Section V we provide a security discussion in which we show the resistance of our protocol against numerous attacks. Section VI describes the more efficient version of StR. In Section VII, we present experimental evidence that shows the effectiveness of our protocol, while in Section VIII, we elaborate on the applications that can benefit from the use of our scheme. Finally, Section IX concludes this paper.

II. Related Work

Although there are many reputation and trust establishment schemes, only some of them deal with the problem of securing the vote(s) of each individual node. The difficulties of building reputation systems that can also preserve privacy can be found in [4]. Furthermore, the absence of schemes that provide (partial) privacy in decentralized environments, such as ad hoc networks, is even bigger.

In [16], the authors considered the problem of distributed reputation management in large systems of autonomous and heterogeneous agents. In such systems, it is generally inadvisable to assume that there exists trustworthy entities who can declare the trustworthiness of different users. Instead, both the reputation of users and the ratings they provide are stored locally and known only to the corresponding entity. The challenge therefore is to compute the reputation while maintaining private data. Three works that deal with the problem of computing ratings in decentralized reputation systems can be found in [13], [9], [5].

Pavlov *et al.* [13] showed that when $n-1$ malicious nodes collude with the querying node to reveal the vote of the remaining node then perfect privacy is not feasible. Furthermore, they proposed three protocols that allow voting to be privately provided in decentralized additive reputation systems. The first protocol is not resilient against collusion of nodes and can be used when dishonest peers are not an issue. The other two protocols are based on a *probabilistic* peers' selection scheme and are resistant to collusion of up to $n-1$ peers only with a certain degree of probability.

Hasan *et al.* [9] proposed a privacy preserving protocol under the semi-honest adversarial model. It's main difference from Pavlov's protocols is that each U_i sends her shares to at most $k < n-1$ nodes that are considered "trustworthy" by U_i. During initialization, the querying agent A_q sends to each U_i the whole list of participating users U. Each U_i selects up to k nodes from U in such a way that the probability that all the selected nodes will collude to break U_i's privacy, is low. Then it splits the vote into k shares and distributes it among the k trustworthy agents. The role of A_q is simply to accumulate the shares into a collective vote.

Dolev *et al.* [5] proposed two main decentralized schemes where the number of messages exchanged is proportional to the number n of participants (however, the length of each message is $O(n)$). The first protocol AP (and its weighted variant WAP) assumes that the querying agent A_q is not compromised while the next protocol, namely MPKP (and its weighted variant MPWP) assumes that any node can act maliciously. Apart from that, all the proposed schemes rely on the use of homomorphic encryption. More precisely, the AP and WAP protocols are based on the Paillier cryptosystem [12], while the more secure MPKP and MPWP are based on the Benaloh cryptosystem [2]. It is exactly this dependency that makes decryption cumbersome. The weakness of Dolev's protocols is the fact that unnecessary and inefficient computations are taking place.

One cannot help but notice the relevance of this problem to *secure multi-party computation*, where a number of distributed entities collaborate to compute a common function of their inputs while preserving the privacy of these inputs. One such well known example is the secure sum protocol [3], which uses randomization to securely compute the sum of the individual inputs. This protocol is a natural fit for the problem at hand but it suffers from a number of attacks and falls prey to honest-but-curious insiders which can easily infer the private input of *any* entity.

The protocols in [13], [9], [5] can be thought as attempts to recover from the security inefficiencies of secure sum, properly applied to the context of reputation management. Our final protocol, shown in Section VI, not only improves upon these schemes but can also be applied directly for secure sum computation, refining earlier results in this area [15].

III. Problem Statement & Definitions

We start by providing a definition of decentralized additive reputation systems as described in [13].

Definition 1: A Reputation System R is said to be a Decentralized Additive Reputation System if it satisfies the following two requirements:

1) Feedback collection, combination and propagation are implemented in a decentralized way.
2) Combination of feedbacks provided by nodes is calculated in an additive manner.

In this paper, we focus on the following problem:

Problem Statement: *A querying node A_q, receives a service request from a target node A_t. Since A_q has incomplete information about A_t, she asks other nodes in the network to give their votes about A_t. Let $U = \{U_1, \cdots, U_n\}$ be the set of all nodes that will provide an opinion to A_q. The problem is to find a way that each vote (v_i) remains private while at the same time A_q would be in position of understanding what voters, as a whole, believe about A_t, by evaluating the sum of all votes ($\sum_{i=1}^{n} v_i$).*

Similar to existing work in the area, all the protocols that are presented in this paper assume that the adversary is *semi-honest*. In the semi-honest adversarial model, malicious nodes correctly follow the protocol specification. However, nodes overhear all messages and may attempt to use them in order to learn information that otherwise should remain private. Semi-honest adversaries are also called *honest-but-curious*.

For the needs of our protocol, we assume that the reader is familiar with the concept of public key cryptography. Each node (A_q, U_i, $i \in [1,n]$) has generated a public/private key pair (k_{A_q}/K_{A_q}, k_{U_i}/K_{U_i}). The private key is kept secret, while the public key is shared with the rest of the nodes. These keys will be used to secure message exchanges between the nodes, hence the communication lines between parties are assumed to be secure. Our first protocol also relies on the use of homomorphic encryption for the collection of votes by the querying agent A_q. The vote of U_i concerning A_t is denoted by v_i.

Definition 2 (Homomorphic Encryption): Let $E(.)$ be an encryption function. We say that $E(.)$ is additive homomorphic iff for two messages m_1, m_2 the following holds:

$$E(m_1) \cdot E(m_2) = E(m_1 + m_2).$$

The notation $E(.)$ will refer to the results of the application of an homomorphic encryption function (as per Definition 2) that A_q can decrypt with her private key. Pailler's Cryptosystem [12] is an example of cryptosystem where the trapdoor mechanism is based on such a homomorphic function.

IV. Splitting the Random values (StR)

In this section, we present our main protocol (StR) in which we use both homomorphic encryption and random numbers to secure the privacy of votes for each node.

During the initialization step, A_q creates the set U with all voters, orders them in a circle $A_q \to U_1 \to \cdots \to U_n$ and sends to each U_i the identity of its successor in the circle. Each U_i splits its random number r_i into n pieces and shares one with the rest of the nodes. Then, it creates a *blinded* vote and adds it to the sum of previous votes by using the homomorphic property of Paillier's cryptosystem [12]. At the end, the last node U_n forwards to A_q the sum of all n votes encrypted with the public key of A_q. Upon reception, A_q decrypts the result and finds the sum of all votes. A detailed description of StR follows below. Figure 1 illustrates the two rounds of StR.

First round

During the initialization step, A_q sends to all nodes the list of all voters U. Each node U_i generates a random number r_i and splits it into n integers in such a way that the i^{th} share will be *encrypted* with the public key of U_i. So, if U_1 has generated a random number r_1, the shares will be

$$r_1 = r_{1,1}, \{r_{1,2}\}_{k_{U_2}}, \cdots, \{r_{1,n-1}\}_{k_{U_{n-1}}}, \{r_{1,n}\}_{k_{U_n}}.$$

The next step for each U_i is to distribute the shares to the remaining $n-1$ nodes in U. This means that each U_i will receive the following $n-1$ messages

$$\{r_{1,i}\}_{k_{U_i}}, \cdots, \{r_{i-1,i}\}_{k_{U_i}}, \cdots, \{r_{n-1,i}\}_{k_{U_i}}, \{r_{n,i}\}_{k_{U_i}}.$$

Since all $n-1$ numbers that U_i received are encrypted with her public key, she decrypts them and calculates the blinded vote b_i which is equal to

$$b_i = v_i + r_i - (\sum_{j=1}^{n} r_{j,i}). \tag{1}$$

When all nodes (in parallel) compute their blinded votes, the second round begins.

Second round

U_1 calculates $E(b_1)$ and sends it to U_2. U_2 adds b_2 to $E(b_1)$ by using the additive homomorphic property ($E(b_1) \cdot E(b_2) = E(b_1 + b_2)$) of Paillier's cryptosystem and sends $E(b_1 + b_2)$ to U_3. At the end of this round A_q will receive from U_n the following: $E(\sum_{i=1}^{n} b_i) = E(\sum_{i=1}^{n} v_i)$. Upon reception, A_q decrypts the message, finds the sum of all votes and divides by n in order to find the average of votes. A concise description of StR is shown in Algorithm 1.

Algorithm 1 StR Protocol

```
$A_q$ generates and distributes $U = \{U_1, \cdots U_n\}$
Round 1 - All nodes in parallel
for all $U_i \in U$ do
    $U_i$ generates $r_i$.
    $U_i$ calculates the $n$-shares: $r_i = r_{i,1} + \ldots + r_{i,n}$
    for all $U_j \in U \setminus \{U_i\}$ do
        $U_i$ sends $\{r_{i,j}\}_{k_{U_j}}$ to $U_j$
    end for
    $U_i$ receives all shares destined to it and calculates the blinded vote $b_i = v_i + r_i - \left(\sum_{j=1}^{n} r_{j,i}\right)$.
end for
Round 2 - All nodes sequentially
for $i = 1$ to $n$ do
    $U_i$ obtains $\prod_{j=1}^{i-1} E(b_j)$ from $U_{i-1}$ (or $E(0)$ from $A_q$, if $i = 1$).
    $U_i$ encrypts $b_i$ with $k_{A_q}$ to obtain $E(b_i)$.
    $U_i$ calculates the homomorphic product $\prod_{j=1}^{i-1} E(b_j) \cdot b_i$
    $U_i$ sends $\prod_{j=1}^{i} E(b_j) = E(\sum_{j=1}^{i} b_j)$ to $U_{i+1}$ (or $E(\sum_{i=1}^{n} v_i)$ to $A_q$, if $i = n$).
end for
```

V. Security Analysis

In this section we analyze the behavior of StR in the presence of corrupted agents. First, we will consider the case of a well-behaving query agent A_q. Such an agent respects the privacy of participating users and does not form malicious coalitions with corrupted agents in the set U (however, among the agents in U there can be corrupted ones). Then, in Section VI, we will proceed to discuss the case where A_q is malicious as well. This will also lead to the development of an even more efficient but equally secure version of StR.

Theorem 1 (Uncompromised A_q): **Assume an honest-but-curious adversary $\mathcal{ADV}$ corrupts at most $k < n$ users out of those in the set U. Then $\mathcal{ADV}$ cannot infer any information about the votes of the legitimate users.**

Proof. **We will prove the robustness of the protocol by reducing its security to the semantic security property of the encryption function $E(\cdot)$. A cryptosystem is called semantically secure, if it is infeasible for a computationally-bounded adversary to derive significant information about a message (plaintext) when given only its ciphertext and the corresponding public encryption key. An equivalent definition for semantic security is that of *ciphertext indistinguishability* [8]. Indistinguishability under Chosen Plaintext Attacks is defined by a game in which an attacker generates two messages m_0 and m_1 and has to determine which of the two messages was chosen by an encryption oracle with probability significantly greater than $1/2$ (i.e. better than random guessing).**

We will prove the privacy of the StR protocol using a standard simulation argument. In particular, we will show that for any adversary that corrupts (or controls) a subset of the participating users, there exists a simulator that, given the corrupted parties data and the final result, can generate a view that, to the adversary, it is *indistinguishable* from a real execution of the protocol. This guarantees that whatever information the adversary can obtain after attacking the protocol can be actually generated by herself, using the simulator. As a result, no useful information about legitimate users' data is leaked (see also Chap. 7 of [7]).

Let $C = \{U_{i_1}, U_{i_2}, \ldots, U_{i_k}\}$ denote the set of compromised users, where $k < n$. Let also $view^C$ denote the views of the protocol for all users in C, including their votes $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$, their random numbers $\{r_{i_1}, r_{i_2}, \ldots, r_{i_k}\}$ and the sequence of messages $E(\sum_{j=1}^{i_1} b_j), \ldots, E(\sum_{j=1}^{i_k} b_j)$ received by each one of them during the second round of the protocol, where by definition

$$b_i = v_i + r_i - (\sum_{j=1}^{n} r_{j,i}).$$

A simulator has access to the shares of the random numbers $r_{i,j}, i \neq j$ that ended up in corrupted users during the first round but cannot possibly generate the exact sequence of encrypted sums since it does not know the private data of legitimate users. So, the simulator will have to replace the private data with random quantities α_i as indicated below

$$b_i' = \begin{cases} b_i, & \textbf{if } U_i \textbf{ is corrupted/compromised} \\ \alpha_i, & \textbf{otherwise,} \end{cases}$$

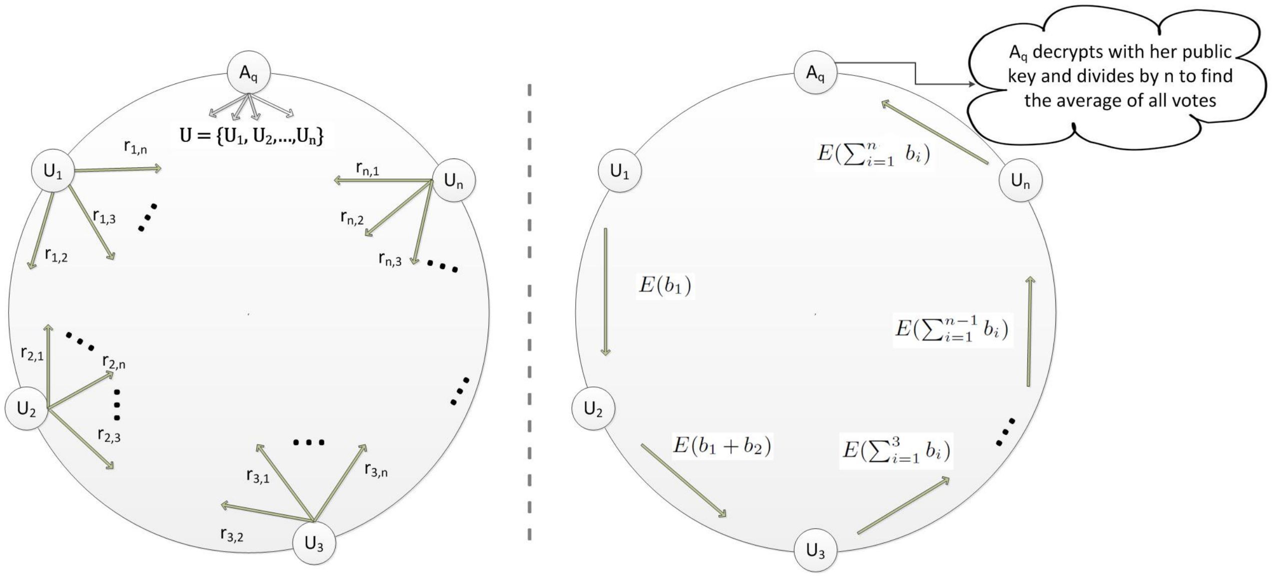

Fig. 1. The two rounds of StR.

and compute $E(b'_i)$ for all $i = 1, \ldots, n$. The simulator can now replace $E(\sum_{j=1}^{i_l} b_j)$ with $E(\sum_{j=1}^{i_l} b'_j)$.

To complete the analysis we need to argue that if there exists an adversary $\mathcal{A}$ that distinguishes between the encryption of the observed values $E(\sum_{j=1}^{i_l} b_j)$ and the random ones $E(\sum_{j=1}^{i_l} b'_j)$ produced by the simulator, then there is an adversary $\mathcal{B}$ that can attack the semantic security of $E(\cdot)$.

Such an attacker $\mathcal{B}$ would operate as follows: Its input is a sequence of values $E(x_i)$, $i = 1, \ldots, n$ and its goal is to determine whether the values x_i correspond to the values provided by the users, or is simply a sequence of random values α_i. Adversary $\mathcal{B}$, using the homomorphic property of $E()$, computes $E(\sum_{j=1}^{i_l} x_j)$ and provides the encryption of the partial sums $E(\sum_{j=1}^{i_1} x_j), \ldots, E(\sum_{j=1}^{i_k} x_j)$ as input to $\mathcal{A}$. It then returns whatever answer $\mathcal{A}$ returns.

Obviously $\mathcal{B}$ would be able to break the semantic security of $E()$ with the same probability that $\mathcal{A}$ could distinguish between the real views and the random values produced by the simulator. Since $E()$ is assumed to be semantically secure, such $\mathcal{A}$ cannot exist. Hence the security of the StR protocol is guaranteed provided at most $k < n$ users are compromised, but A_q is not. □

VI. A More Efficient StR

In this section we will consider the case where node A_q is compromised as well. Since A_q knows the private key and A_q has been compromised by $\mathcal{ADV}$ (or is member of the colluding group), A_q can simply decrypt any communicated message. Hence we cannot rely on the semantic security property of the underlying cryptosystem. The semantic security of the cryptosystem protects the nodes from seeing intermediate results but it is the added randomness which keeps $\mathcal{ADV}$ from obtaining those intermediate values. In this scenario the security is therefore solely based on the *randomness* which is used to blind the individual votes.

To see this, observe that in the second round of StR, homomorphic encryption is used to compute the sum of the blinded votes, $\sum_i b_i$, around the ring. However, a compromised A_q can learn these values by collaborating with a set of malicious agents. Hence homomorphic encryption does not offer any real benefit and can be dropped entirely! This also suggests that during the second round the nodes can send the blinded votes *directly* to A_q without having to go around the ring, thus increasing the efficiency of the algorithm, as we will see in the experimental section. The new protocol is shown in Algorithm 2. Round 2 is a degenerate one and can clearly be combined with Round 1.

The more efficient StR also provides an improvement over previous protocols in the field of secure multi-party sum computation [15]. In particular, in [15], a distributed protocol is presented that requires $O(n^2)$ *sequential* computations around a ring of nodes. Our protocol is completely parallelized and does not even require placing the nodes around a ring.

Algorithm 2 Improved StR
```
$A_q$ generates and distributes $U = \{U_1, \cdots U_n\}$
**Round 1 - All nodes in parallel**
**for all** $U_i \in U$ **do**
    $U_i$ generates $r_i$.
    $U_i$ calculates the $n$-shares: $r_i = r_{i,1} + \ldots + r_{i,n}$
    **for all** $U_j \in U \setminus \{U_i\}$ **do**
        $U_i$ sends $\{r_{i,j}\}_{k_{U_j}}$ to $U_j$
    **end for**
    $U_i$ waits until it receives *all* shares destined to it and calculates the blinded vote $b_i = v_i + r_i - \left(\sum_{j=1}^{n} r_{j,i}\right)$.
**end for**
**Round 2 - All nodes in parallel**
**for** $i = 1$ to $n$ **do**
    $U_i$ sends $b_i$ to $A_q$
**end for**
Upon reception of all votes, $A_q$ computes $\sum_{i=1}^{n} b_i = \sum_{i=1}^{n} v_i$
```

In what follows we prove the security of the more efficient version of StR.

Theorem 2 (Compromised A_q): **Assume an honest-but-curious adversary $\mathcal{ADV}$ corrupts A_q and at most $k < n-1$ users out**

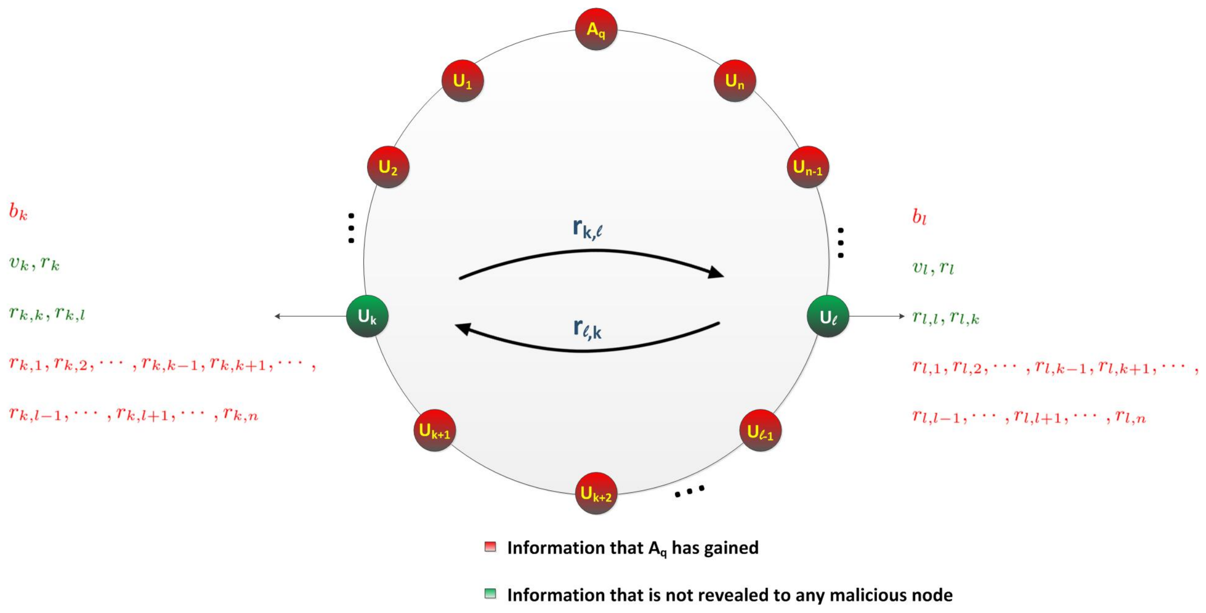
A_q
U_1
U_n
U_2
U_{n-1}
$r_{k,\ell}$
$r_{\ell,k}$
U_k
U_ℓ
U_{k+1}
$U_{\ell-1}$
U_{k+2}
b_k
v_k, r_k
$r_{k,k}, r_{k,l}$
$r_{k,1}, r_{k,2}, \cdots, r_{k,k-1}, r_{k,k+1}, \cdots,$
$r_{k,l-1}, \cdots, r_{k,l+1}, \cdots, r_{k,n}$
b_l
v_l, r_l
$r_{l,l}, r_{l,k}$
$r_{l,1}, r_{l,2}, \cdots, r_{l,k-1}, r_{l,k+1}, \cdots,$
$r_{l,l-1}, \cdots, r_{l,l+1}, \cdots, r_{l,n}$
🟥 Information that A_q has gained
🟩 Information that is not revealed to any malicious node

Fig. 2. Robustness up to $n-1$ malicious nodes

of those in the set U. Then $\mathcal{ADV}$ cannot infer any information about the votes of the legitimate users.

Proof. **Here, we consider the extreme case where *all* nodes collaborate with a corrupted A_q *except* for two nodes U_k, U_l which are considered legitimate (Figure 2).**

To prove that StR protects the privacy of legitimate users, even if A_q is compromised, we need to look at the data exchanged in StR. Recall that during the first round, each node will receive $n-1$ shares from the remaining nodes of U. Since $n-2$ nodes are compromised, at the end of round one, the adversary will know all the $n \cdot (n-2)$ shares of the $n-2$ compromised nodes plus the $n-4$ shares that U_k and U_l have sent to the compromised ones.

From the four remaining shares, $r_{k,k}$ and $r_{l,l}$ will be known only to U_k and U_l, since these are part of the shares they keep for the calculation of their blinded votes b_k, b_l. Additionally, the last two remaining shares $(r_{l,k}, r_{k,l})$ will be known only to U_k, U_l since they are encrypted with their corresponding public keys and then exchanged between them. Since we have assumed that these two nodes are legitimate, they will not reveal the value of these shares to any other node (compromised information is shown next to the two nodes in Figure 2).

To ease the analysis, in the following expressions we have circled the variables that the adversary has *not* been able to compromise:

$$b_k = \enclose{circle}{v_k} + \enclose{circle}{r_k} - (r_{1,k} + \cdots + \enclose{circle}{r_{k,k}} + \cdots + \enclose{circle}{r_{l,k}} + \cdots + r_{n,k}) \tag{2}$$

and

$$b_l = \enclose{circle}{v_l} + \enclose{circle}{r_l} - (r_{1,l} + \cdots + \enclose{circle}{r_{l,l}} + \cdots + \enclose{circle}{r_{k,l}} + \cdots + r_{n,l}). \tag{3}$$

However, considering the fact that r_k, r_j are equal to the sum of the corresponding shares, i.e. $r_k = \sum_{j=1}^{n} r_{k,j}$ and $r_l = \sum_{j=1}^{n} r_{l,j}$, we obtain that

$$r_k - r_{k,k} = \sum_{j \neq k} r_{k,j} \quad \text{and} \quad r_l - r_{l,l} = \sum_{j \neq l} r_{l,j}.$$

Plugging these last two expressions to Equations (2) and (3), we obtain

$$b_k = \enclose{circle}{v_k} + \sum_{j \neq k,l} (r_{k,j} - r_{j,k}) + \enclose{circle}{r_{k,l} - r_{l,k}} \tag{4}$$

and

$$b_l = \enclose{circle}{v_l} + \sum_{j \neq k,l} (r_{l,j} - r_{j,l}) - \enclose{circle}{r_{k,l} - r_{l,k}}. \tag{5}$$

Treating the last term $(r_{k,l} - r_{l,k})$ as a single unknown quantity, we see that it is impossible to correctly calculate the exact values v_k, v_l since the adversary, even with the help of A_q, ends up with a system of two equations and three unknown variables (the case is analogous when there are more than 2 legitimate users). We conclude that the protocol remains secure as long as there exist at least two nodes that are legitimate. $\square$

Finally, observe that in both cases (Theorems 1 and 2) StR offers an equivalent level of security as long as *there are at least two nodes which are not corrupted*. In the first case, this is A_q plus another agent from the set U. In the second case, these are two nodes from U. Thus, a legitimate node can be sure of its private vote if and only if there is at least one more legitimate node in the set $U + \{A_q\}$.

VII. Experimental Results

This section presents the implementation of StR, as well as a comparison with Dolev's Multiple Private Keys Protocol (MPKP) [5]. In order to prove the effectiveness of StR, we implemented both protocols in Java and we used JADE 4.0.1 [1] for the communication of the agents. Since we wanted our experiments to be as close to reality as possible, we setup different JADE agents in different computers. All agents (nodes of the protocols) were connected to the Internet through a NetFasteR IAD 2 router over a 24Mbps ADSL line.

Our experiments aimed at analyzing two main performance metrics; processing time and communication overhead.

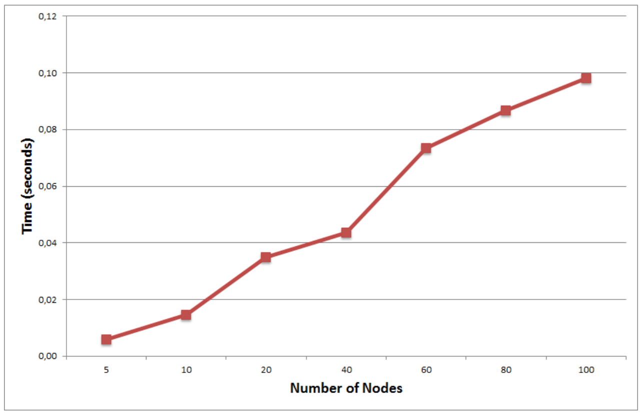

Fig. 3. Processing time required by StR

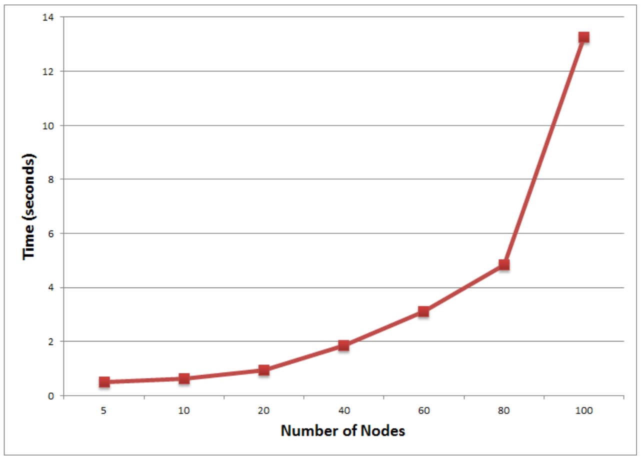

Fig. 4. Communication Delay of first round of StR

A. Processing Time

The first phase of our experiments involved measuring the processing time of StR. To this end, we measured the completion time for the following procedures:

- *Secure Random Number Generation*
- *Encryption/Decryption*

For the encryption and decryption, we used the RSA cryptosystem for encrypting the random shares with a key length equal to 1024 bits. Figure 3 displays the results following 1000 test runs in a computer with a 1.6GHz CPU and 1GB DDR RAM, where each node has to i) encrypt the $n-1$ shares to be transmitted, and ii) decrypt the $n-1$ shares received, where n ranges from $n=5$ to $n=100$. As is evident from the graph, the required processing time is negligible and does not constitute any real burden to nodes of the StR protocol.

Notice that this is *not* the case for Dolev *et al.*'s protocol. Decryption of the homomorphic values is inefficient because it requires a trial-end-error decryption in order to compute the encrypted trust ratings. This is due to the use of the Benaloh cryptosystem which does not allow for efficient decryption. Thus, processing time depends not only on n but also on the allowable *range* of trust values (details omitted due to space restrictions). Despite this inefficiency, we treat both times as *comparable* and we focus only on the communications aspects of both protocols.

B. Communication Delay

1) First Round:: By default, JADE uses the Message Transport Protocol (MTP) for the communication between nodes. During the first phase of our experiments, we wanted to measure the communication delay for the first round of StR. For that purpose, we created nodes in different computers that generated n encrypted shares (1024 bits long each); these were sent in parallel as single messages to each of the $n-1$ remaining nodes, where n was incremented from $n=5$ to 100 in steps of 5. As expected, the delay did not increase in a strictly linear manner, since the overhead processing of collecting the shares and computing the masked vote $b_i = v_i + r_i - (\sum_{j=1}^{n} r_{j,i})$ increased with the number of nodes. Figure 4 illustrates the delay in seconds as a function of the number of nodes n.

2) Second Round:: While in StR only one message (the blinded vote) is transmitted from each node to A_q, this is not the case for Dolev's protocol as each node must send to the next one in the ring the result of the homomorphic encryption. Thus, in this case, we wanted to calculate the communication delay of transmitting a message of size 1024 bits long (the result of the homomorphic encryption) between successive nodes in the list U. We have run 1000 experiments in our JADE platform and we have found that, on average, the time to sent a single message between two successive nodes is approximately equal to 0.115 seconds.

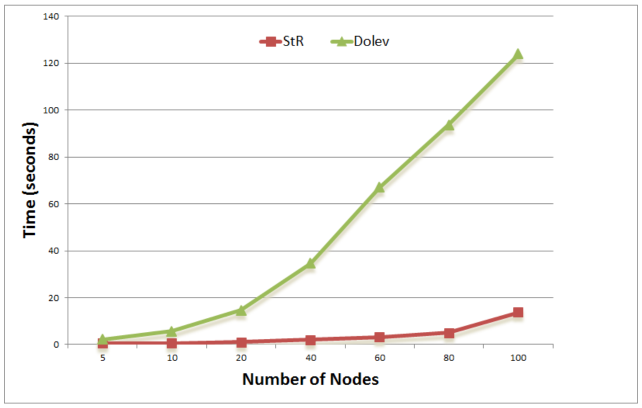

Fig. 5. Communication Delay for StR and Dolev *et al.* protocols

We have summarized these findings in Figure 5. This figure shows a comparison for the communication delay of *both* rounds of StR and Dolev's protocol. While both protocols show a quadratic behavior – Dolev's protocol *sequentially* propagates, for a total of n times, a large message of length $O(n)$, while in StR each node sends, in *parallel*, $(n-1)$ messages of size O(1) – StR outperforms Dolev's protocol. This is something to be expected since during the first round of StR time is saved by sending the shares in parallel and not sequentially. Additionally, during the second round time is saved by eliminating the need to visit the nodes in the ring. Thus, without sacrificing security, the communication delay of StR for a list of up to one hundred voters, is almost an order of magnitude smaller than that of Dolev's protocol (13.7sec vs. 124sec) and is expected to be magnified even further for larger values of n.

VIII. StR's Application Domain

After the Napster era, all successful P2P file sharing networks have developed a common element - they have become decentralized. The most successful networks such as Kazaa, LimeWire and Morpheus, although they provide a some form of legal protection to their users by giving them the opportunity to distribute software, songs, movies, etc., they cannot, effectively, protect them from downloading malicious software. For example, many users distribute jpeg files that are infected with malicious code. The result is that when a user opens the file, she sees an image, but at the same time the computer is infected with malicious code. The utilization of feedback/voting can provide a solution to these kinds of attacks, as a user will be notified that the other user or the requested file are considered as insecure. Furthermore, by also providing this feedback anonymously, the retaliation between nodes/users can be minimized. There are additional reasons for using anonymity in P2P networks:

- Material is legal but socially deplored, embarrassing or problematic in the individual's social world (for example, anonymity is seen as a key requirement for organizations like alcoholics, drug addicted, etc.)
- Fear of retribution (against whistleblowers, unofficial leaks, and activists who do not believe in restrictions on information or knowledge)
- Censorship at the local, organizational, or national level
- Personal privacy preferences such as preventing tracking or data mining activities.

Apart from that, anonymous feedback can also have a role in the education field, as students (especially in the eLearning field) will benefit from the ability to evaluate the services offered by the school/university. Students will greatly benefit from this form of transparent and reliable anonymous feedback. Furthermore, numerous web applications such as betterme.com and rypple.com offer members of different communities the opportunity to send anonymous feedback regarding their coworkers, classmates, friends, landlord, boss etc. This feedback can be processed by StR in order to improve the operation of the corresponding community.

IX. Conclusions

In this work we presented StR, a decentralized privacy-respecting scheme for securely casting trust ratings in additive reputation systems. Our protocol relied on the use of public key cryptography and homomorphic encryption and has been formally proved to be resistant to collusion even against as many as $n-1$ malicious insiders. In the course of this work, we have also presented a lighter, but equally secure protocol, that can be thought as an independent contribution to the field of secure multiparty sum computation. The effectiveness of StR was demonstrated by conducting extensive experiments measuring its communication delay and processing overhead in a real P2P network, showing its superior performance over the previous best protocol to date.

As part of our future research, we intend to enhance our protocol and consider defense mechanisms that will effectively manage malicious adversaries, adversaries that may provide dishonest input to bias the protocol or exhibit byzantine behavior based on Hoffman's *et al.* [10] attacker model, thus deviating from the designated honest-but-curious behavior examined here.

X. Acknowledgements

This work has been funded by the European Community's FP7 project SafeCity (Grant Agreement no: 285556).

References

[1] F. Bellifemine, A. Poggi, G. Rimassa, and T. Italia. Jade. Internal Tecnhical Report. http://jade.tilab.com/

[2] J. Benaloh. Dense probabilistic encryption. *Workshop on Selected Areas of Cryptography*, 1994.

[3] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin and M. Y. Zhu. Tools for Privacy Preserving Distributed Data Mining. In *ACM SIGKDD Explorations*, 2003.

[4] R. Dingledine, N. Mathewson, and P. Syverson. Reputation in p2p anonymity systems. *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[5] S. Dolev, N. Gilboa, and M. Kopeetsky. Computing multi-party trust privately: in $o(n)$ time units sending one (possibly large) message at a time. *ACM Symposium on Applied Computing (SAC '10)*, 2010.

[6] eBay. Buyer accountability. *http://pages.ebay.com/services/forum/sellerprotection.html*.

[7] O. Goldreich. Foundations of Cryptography. Volume 2. Cambridge University Press, 2004.

[8] S. Goldwasser and S. Micali. Probabilistic encryption. *Computer and System Sciences*, 28:270–299, 1984.

[9] O. Hasan, L. Brunie, and E. Bertino. k-shares: A privacy preserving reputation protocol for decentralized environments. *25th IFIP International Information Security Conference (SEC 2010)*, 2010.

[10] D. Z. K. Hoffman and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42:1:1:1–31, 2009.

[11] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43:618–644, Oct 2007.

[12] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology, EUROCRYPT '99*, pp.223–238, 1999.

[13] E. Pavlov, J. S. Rosenschein, and Z. Topol. Supporting privacy in decentralized additive reputation. *Second International Conference on Trust Management (iTrust 2004)*, pages 108–119, 2004.

[14] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *The Economics of the Internet and E-Commerce*, 11(3):129–158, 2002.

[15] Rashid Sheikh, Beerendra Kumar, Durgesh Kumar Mishra. A Distributed k-Secure Sum Protocol for Secure Multi-Party Computations. In *Journal of Computing*, Volume 2, Issue 3, March 2010.

[16] B. Yu and M. Singh. Detecting deception in reputation management. In *Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS' 03)*, 2003.