View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by WestminsterResearch

WestminsterResearch
http://www.westminster.ac.uk/research/westminsterresearch

Multi-Party Trust Computation in Decentralized Environments in the Presence of Malicious Adversaries

Tassos Dimitriou[1, 2]
Antonis Michalas[3, 4,*]

[1]Computer Technology Institute, Greece
[2]Computer Engineering Dept., Kuwait University, Kuwait
[3] Athens Information Technology, Greece
[4] Aalborg University, Denmark

*Now working in the Faculty of Science and Technology, University of Westminster, UK

NOTICE: this is the accepted version of a manuscript of an article that was accepted for publication in Ad Hoc Networks. It does not include copy-editing, formatting, technical enhancements and (if relevant) pagination. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Ad Hoc Networks, 15 (Apr), 2014 pp. 53-66. ISSN 1570-8705
doi:10.1016/j.adhoc.2013.04.013

© 2014. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/

The WestminsterResearch online digital archive at the University of Westminster aims to make the research output of the University available to a wider audience. Copyright and Moral Rights remain with the authors and/or copyright owners.

Whilst further distribution of specific materials from within this archive is forbidden, you may freely distribute the URL of WestminsterResearch: (http://westminsterresearch.wmin.ac.uk/).

In case of abuse or copyright appearing without permission e-mail repository@westminster.ac.uk

Multi-Party Trust Computation in Decentralized Environments in the Presence of Malicious Adversaries

Tassos Dimitriou*

Computer Technology Institute, Greece
and
Computer Engineering Dept., Kuwait University, Kuwait
Email: tassos.dimitriou@ieee.org

Antonis Michalas

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 both the semi-honest and malicious adversarial models.

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.

Keywords: Decentralized Reputation Systems, Security, Voter Privacy, Anonymous feedback

*Corresponding Author.

1. Introduction

During the last few years, online communities have met great development. The main reasons for this lie on the fact that comprehension and use have become easier, but also on the availability of information and accessibility of services. There is no doubt that we have become a society of sharing to the detriment of our personal privacy since we continue over-sharing without considering the privacy ramifications. Social media for example, provide a way to share every aspect of our life not only with people we know but with strangers as well. Corrupted users can take advantage of the freedom that they are given to create counterfeit identities and steal, damage or alter the information and data of legitimate users.

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 [1]. *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.

An established technique that aims at assisting users to avoid interacting with malicious or unreliable agents is the use of *reputation systems*. A reputation system assesses the behavior of users according to the quality of the service(s) provided, and reveals this information to the community in order to decide whether a network entity is *trustworthy* or not. A user's behavior and the conclusions of a reputation system regarding that user are referred to as evidence. Evidence can be collected offline as well as online and may be linked to already established trust relations. Hence, trust has a significant role in the decisions reached by network communities.

Nevertheless, *feedback providers' privacy* is a problem regarding reputation systems that has been scarcely answered in literature. Notwithstanding the existence of numerous reputation and trust establishment schemes, only a few deal with securing the ratings (or votes) of participating nodes. Lack of privacy of this type can lead to various problems, including the operation of the network itself. Furthermore, the lack of schemes providing privacy in decentralized environments, e.g. ad hoc networks, is even more accentuated. According to observations in [2], users of a reputation system may abstain from providing honest feedback since they are afraid of retaliation, in case reputation scores cannot be computed in a method that secures privacy. As

a result, eBay changed the feedback policy in order for sellers not to leave negative/neutral feedback for buyers, claiming that "it will help buyers leave an honest feedback" [3].

Consequently, developing reputation protocols that can be used to provide anonymous feedback is necessary for the survival of online communities and electronic marketplaces. One could say that providing anonymous feedback to a reputation system is analogous to that of anonymous voting in electronic elections. Secrecy provides freedom from explicit or implicit influence but also encourages truthfulness. Even though this freedom could potentially be exploited by dishonest feedback providers who tend to report exaggerated feedbacks, it seems highly beneficial to honest users, protecting the latter from being influenced by malicious behavior [4].

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. The contribution of this work is twofold. We first present a protocol that preserves the privacy of votes in decentralized environments under the semi-honest adversarial model. 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. While the previous two protocols work against semi-honest adversaries, we show how to extend these protocols to handle adversaries that can exhibit more malicious behavior. The whole analysis is coupled with extensive experimental results that demonstrate the protocol's validity and efficiency over previous works in the area.

Second, we present a list of attacks that can be applied to additive reputation systems. We use this analysis to guide us in the development of our secure protocol but also demonstrate the inefficiencies of existing systems. This comprehensive list of protocol flaws provides essential knowledge to protocol designers. By answering questions of the form "Did you know that sort of attack?" they can avoid common pitfalls and design even better feedback systems.

Organization of the paper. In Section 2, we review some of the most important schemes that provide private trust ratings in decentralized environments. In Section 3, 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 4, we describe two basic (yet insecure) protocols that will help us understand the vulnerabilities of existing systems through a series of attacks that can be applied to additive reputation protocols. In Section 5, we present StR, our main protocol, while in Section 5.1 we provide a security discussion in which we show the resistance of our protocol against numerous attacks. Section 6 describes the more efficient version of StR while in Section 7, we present experimental evidence that shows the effectiveness of our protocol. In Section 8 we present StR^M, an extension of StR, that provides security under the malicious adversarial model. In Section 9, we elaborate on the applications that can benefit from the use of our scheme. Finally, Section 10 concludes this paper.

2. 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 [5]. Furthermore, the absence of schemes that provide (partial) privacy in decentralized environments, such as ad hoc networks, is even bigger. In this section, we classify the most widely known decentralized systems that preserve the privacy of votes according to the adversarial model (semi-honest or malicious) used.

2.1. Protocols Under the Semi-Honest Model

The authors in [6] presented an inclusive analysis of trust and privacy in which they showed that these two notions are strongly related to each other. More precisely, they dealt with the questions of how much privacy is lost and how much trust is gained by revealing a specific credential and what

is the minimal degree of privacy that must be lost to gain a satisfactory amount of trust.

In [7], a secure framework was proposed to handle trust relationships in *super peer* networks. Privacy of the nodes is guaranteed with the use of threshold cryptography and homomorphic encryption. The use of a threshold cryptosystem ensures that no faulty or malicious node can decrypt the report submitted by a node. Furthermore, the sum of votes from all voters is calculated step by step in an encrypted manner and thus it is impossible for a malicious node to infer this result since each vote is encrypted with the public key of a trusted Certification Authority. The key difference from the work presented in this paper is the fact that super peers are trusted to handle the votes correctly, something that is not necessarily true here.

In [8], 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 exist 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 [4, 9, 10].

Pavlov *et al.* [4] 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 there are no dishonest peers. 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.* [10] proposed two main decentralized schemes where the number of messages exchanged is proportional to the number n of partic-

ipants (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 [11], while the more secure MPKP and MPWP are based on the Benaloh cryptosystem [12]. 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 [13], 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 [4, 9, 10] can be thought as attempts to recover from the security inefficiencies of secure sum, properly applied to the context of reputation management. Our protocol, shown in Section 6, not only improves upon these schemes but can also be applied directly for secure sum computation, refining earlier results in this area [14].

2.2. Protocols Under the Malicious Model

Dolev *et al.* [15] proposed CEBP, a protocol that is functioning under the malicious adversarial model in the sense that the query agent A_q can easily verify that all submitted votes lies in a certain range. Moreover, authors manage to avoid the use of zero-knowledge proofs and thus avoid complex computations, since at the last round A_g receives a list with all the individual votes in a random order. However, their protocol is using commutative encryption schemes, like the Pohlig-Hellman scheme [16]. Existing commutative encryption schemes in general do not provide formal methods of security [17], and may lead to security breaches in real world applications. Furthermore, the protocol does not support robustness since users, especially the last one (U_n) can change all the votes of the previous $n-1$ users and thus send to A_q a false voting list.

Pavlov *et al.* presented a decentralized privacy preserving scheme [4] for the malicious case as well. The protocol is based on Pedersen's [18] verifiable secret sharing scheme to support validity checking of the feedback values

provided by voters. In other words, they provide a mechanism to ensure that reputation ratings lie within a predefined range. The main disadvantage of the protocol is the fact that it requires $O(n^3)$ messages primarily due to a costly witness selection scheme. In addition to that, there is an insufficient description of the protocol. For example, there is no explanation regarding the zero-knowledge proofs that the protocol requires. Also, it is not clear if a vote can belong to any interval $[a, b]$ or should be bounded to a smaller one (e.g $[0, 1]$) which would change the required computations for the verifier of a vote.

In [19], Hasan *et al.* presented the Malicious-k-shares protocol, a distributed privacy preserving reputation protocol for the malicious adversarial model. The protocol is more efficient in comparison with Pavlov's and is based on set-membership and plain-text equality non-interactive zero knowledge proofs and an additive homomorphic cryptosystem. The main drawback though is the fact that one cannot sustain that the protocol has a decentralized behavior since the query agent A_q acts like a central authority since all messages are transfered to her and then she forwards them to the actual receivers. In addition to that, at the second step of the protocol, each U_i selects k other agents in such a way that the probability that all of the selected agents will collude to breach agent U_i's privacy is low. However, it is not clear how does this witness selection scheme effects computation complexity of the whole protocol.

3. Problem Statement & Definitions

We start by providing a definition of decentralized additive reputation systems as described in [4].

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, the protocols that are presented in Sections 5-6 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*. In Section 8, we build upon the previous protocols to devise a scheme that can withstand more malicious behavior; the adversaries not only may attempt to disrupt protocol execution but can even bias the results in order to lead to wrong outcomes.

Protocol Setup: We assume that the reader is familiar with the concept of public key cryptography. For the needs of our protocols, 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. It is also assumed that nodes are familiar with the public keys of nodes they interact with. 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 [11] is an example of cryptosystem where the trapdoor mechanism is based on such a homomorphic function. The semantic security of Pailler's cryptosystem is proved under the decisional composite residuosity assumption: Given $N = pq$, it is hard to decide whether an element in $\mathbb{Z}_{N^2}$ is an $N - th$ power of an element in $\mathbb{Z}^*_{N^2}$.

4. Toy Protocols

In this section we present two protocols, the second protocol being a more "secure" version of the first one. This exposition has mostly pedagogical character. We describe these protocols in order to expose their vulnerabilities and thus create a starting point that will help us design our main protocol. In the course of this process we will be able to characterize the different types of attacks that can be applied to decentralized reputation systems thus helping researchers avoid common pitfalls while designing secure reputation systems. Our final protocol will provide resistance against these attacks, thus successfully preserving the privacy of submitted trust ratings.

4.1. Toy Protocol 1

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 adds its vote to the sum of previous votes by using the homomorphic property of Paillier's cryptosystem [11]. At the end, the last node U_n sends to A_q the sum of all n votes encrypted with the public key of A_q. Upon reception, A_q decrypts $E(\sum_{i=1}^{n} v_i)$, finds the sum of all votes and divides by the number of voters n to find the average value.

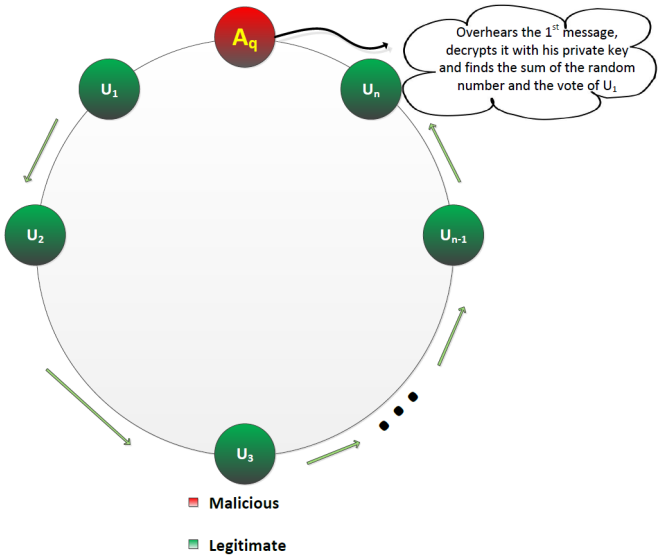

Figure 1: Querying Node Attack

Analysis. This toy protocol (which is reminiscent of the AP protocol of [10]) is rather simple and achieves privacy of the submitted votes only if A_q is not considered malicious. External attackers have no chance recovering the votes since they are encrypted with A_q's public key. However, because A_q can overhear all messages, she can decrypt them one by one and find all the individual votes. More precisely, A_q decrypts the first message $E(v_1)$ and finds the vote v_1 of U_1, then decrypts the second message $E(v_1 + v_2)$ subtracts v_1 and finds v_2. By doing this n times, she can find all $v_i, i \in [1, n]$ and thus break the privacy of the protocol. We call this type of attack the *Querying Node Attack* (see also Figure 1 for a graphical representation). Notice that even in the case where A_q cannot overhear all messages, this protocol still falls prey to other attacks by malicious voters, however we delay the description of these attacks for the next section.

4.2. Toy Protocol 2

This is an extension of the protocol of the previous section. In this protocol (shown in Figure 2), we propose a solution that can be used to overcome the *Querying Node Attack*. However, this protocol is still not secure as we will describe in while.

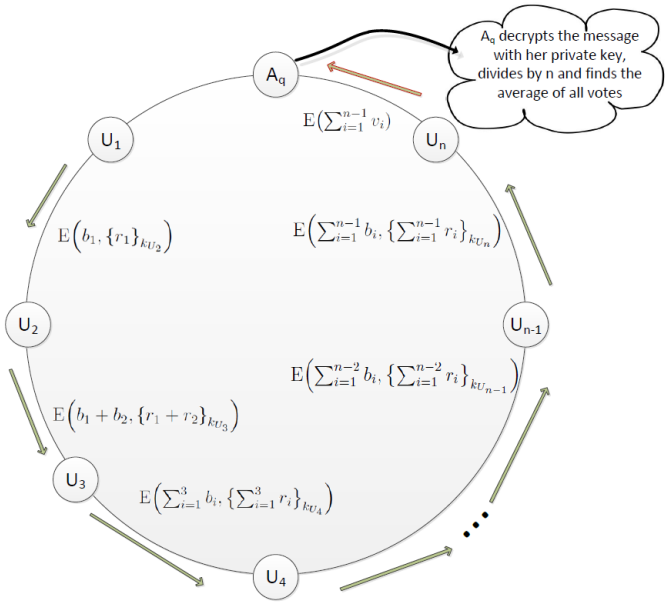

Figure 2: Toy Protocol 2

During the initialization stage, A_q sends to each node a list which contains only the previous and the next node in U. U_1 who is the first node in the list, generates a random number r_1. Then she adds her vote v_1 for A_t to r_1 and encrypts the result $b_1(= v_1 + r_1)$ with k_{A_q} to obtain $E_q(b_1)$. At this point, U_1 sends to the next node U_2 the message $\langle E_q(b_1), E_2(r_1)\rangle$.

U_2 calculates $b_2(= v_2 + r_2)$ adds it to $E_q(b_1)$ and finds r_1 by decrypting $E_2(r_1)$ with K_{U_2}. Once she finds r_1, she calculates $r_1 + r_2$, encrypts it with k_{U_3} and sends the following message to U_3: $\langle E_q(b_1 + b_2), E_3(r_1 + r_2)\rangle$. By extrapolation, the last node will receive the following message

$$\left\langle E_q\left(\sum_{i=1}^{n-1} b_i\right), E_n\left(\sum_{i=1}^{n-1} r_i\right)\right\rangle.$$

U_n decrypts $E_n\left(\sum_{i=1}^{n-1} r_i\right)$ with K_{U_n} and sends back to A_q the message $E_q(\sum_{i=1}^{n-1} b_i - \sum_{i=1}^{n-1} r_i + v_n) = E_q(\sum_{i=1}^{n} v_i)$. Upon reception, A_q decrypts $E_q(\sum_{i=1}^{n} v_i)$, divides it by n and finds the average of submitted votes.

Analysis. This protocol is more secure than the first one. Every node U_i instead of adding v_i to $E(.)$, generates a random number r_i and use it to mask its vote v_i. This means that even if A_q is malicious and overhears all messages, she cannot find the individual votes, since she does not know the random numbers that have been used to hide these votes. However, this protocol is still vulnerable to numerous attacks by malicious insiders, as we describe below.

- **First Node Attack (Figure 3):** In this scenario, A_q and the second voter U_2 are considered malicious. So, A_q and U_2 can collaborate in order to find the vote v_1 of U_1. When U_2 receives $\langle E_q(b_1), E_2(r_1)\rangle$, she decrypts $E_2(r_1)$ with her private key (K_{U_2}) and sends to A_q the random number r_1 that U_1 generated in the previous step. Now, A_q can easily find $v_1 = b_1 - r_1$ since she knows both r_1 and b_1 (by decrypting $E_q(b_1)$).

- **Last Node Attack:** By symmetry, A_q and the penultimate voter U_{n-1} are considered malicious. This means that A_q and U_{n-1} can cooperate in order to find the vote v_n of U_n. When U_{n-1} sends $\left\langle E_q\left(\sum_{i=1}^{n-1} b_i\right), E_n\left(\sum_{i=1}^{n-1} r_i\right)\right\rangle$ to U_n, she informs A_q about the value of $\sum_{i=1}^{n-1} r_i$. Thus, A_q can find $\sum_{i=1}^{n-1} v_i$, since $\sum_{i=1}^{n-1} v_i = \sum_{i=1}^{n-1} b_i - \sum_{i=1}^{n-1} r_i$. As we mentioned before, A_q receives from U_n the value $E_q(\sum_{i=1}^{n} v_i)$ which decrypts and finds $\sum_{i=1}^{n} v_i$. Now, A_q can find v_n by calculating $v_n = \sum_{i=1}^{n} v_i - \sum_{i=1}^{n-1} v_i$.

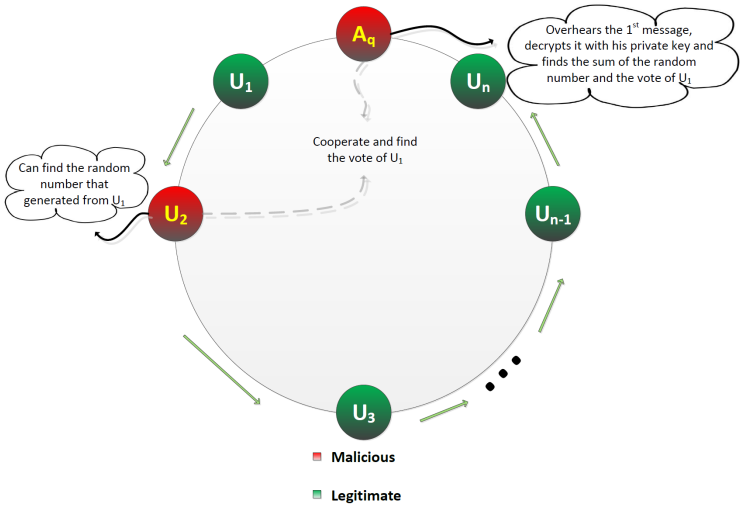

Figure 3: First Node Attack

If A_q is malicious and has only two compromised nodes in U, then she will place them in such a way that can achieve both last node and first node attacks. That way, A_q will manage to find the maximum number (v_1 and v_n) of individual votes that she can. Another alternative is for A_q to use these two malicious nodes to find the vote of any user U_i by "sandwiching" U_i between two malicious ones as explained below.

- **Sandwich Attack (Figure 4):**

  A generalization of the previous attacks is when a (malicious) A_q arranges the voters in such a way that a legitimate node U_i will always be *between* two malicious ones U_{i-1} and U_{i+1}. This way, A_q can use values from adjacent malicious nodes to calculate the random number r_i that was used to blind the vote v_i of U_i. A_q can thus find all the votes of legitimate nodes in the set. The first and last node attacks described previously are simple variants of the sandwich attack in which A_q acts as one of the two malicious nodes.

5. 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.

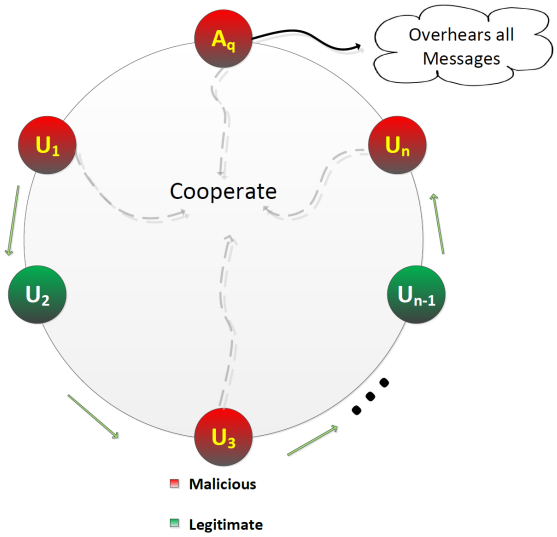

Figure 4: Sandwich Attack

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 [11]. 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 5 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

$$r_1 = r_{1,1}, E_2(r_{1,2}), \ldots, E_{n-1}(r_{1,n-1}), E_n(r_{1,n}).$$

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

$$E_i(r_{1,i}), \ldots, E_i(r_{i-1,i}), \ldots, E_i(r_{n-1,i}), E_i(r_{n,i}).$$

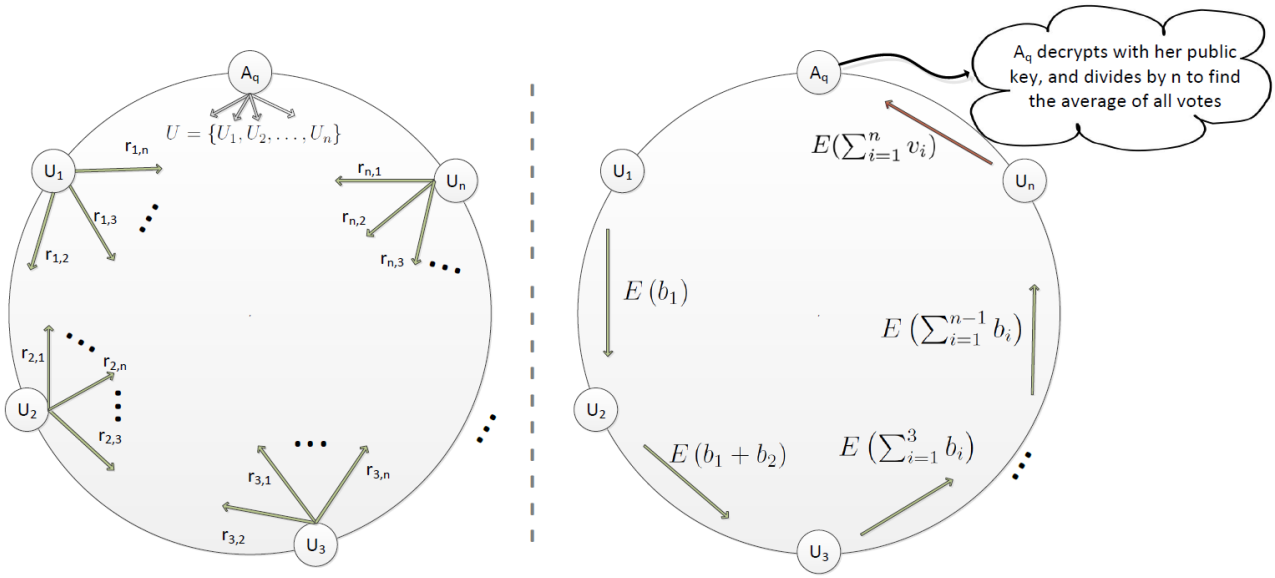

Figure 5: The two rounds of StR.

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 $E_q(b_1)$ and sends it to U_2. U_2 adds b_2 to $E_q(b_1)$ by using the additive homomorphic property ($E_q(b_1) \cdot E_q(b_2) = E_q(b_1 + b_2)$) of Paillier's cryptosystem and sends $E_q(b_1 + b_2)$ to U_3. At the end of this round A_q will receive from U_n the following: $E_q(\sum_{i=1}^{n} b_i) = E_q(\sum_{i=1}^{n} v_i)$. 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.

5.1. 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

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 $E_j(r_{i,j})$ 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 $\prod_{j=1}^{i-1} E_q(b_j)$ from U_{i-1} (or $E_q(0)$ from A_q, if $i = 1$).
  U_i encrypts b_i with k_{A_q} to obtain $E_q(b_i)$.
  U_i calculates the homomorphic product $\prod_{j=1}^{i-1} E_q(b_j) \cdot b_i$
  U_i sends $\prod_{j=1}^{i} E_q(b_j) = E_q(\sum_{j=1}^{i} b_j)$ to U_{i+1} (or $E_q(\sum_{i=1}^{n} v_i)$ to A_q, if $i = n$).
end for

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 6, 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* [20]. 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 [21]).

Let $C = \{U_{i_1}, U_{i_2}, \ldots, U_{i_k}\}$ denote the set of compromised users, where $k < n$. Consider the information available to protocol users in C: this includes 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

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

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. □

6. 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 [14]. In particular, in [14], a distributed protocol is presented that requires $O(n^2)$ message exchanges that must be *sequentially* executed, one after the other, by a set of nodes ordered in a ring. Our protocol is completely parallelized and does not even require placing the nodes around such a ring.

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

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 $r_{i,j}$ 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 $\sum_{i=1}^{n} b_i = \sum_{i=1}^{n} v_i$.

Theorem 2 (Compromised A_q). *Assume an honest-but-curious adversary $\mathcal{ADV}$ corrupts A_q and at most $k < n-1$ users out 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 6).

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

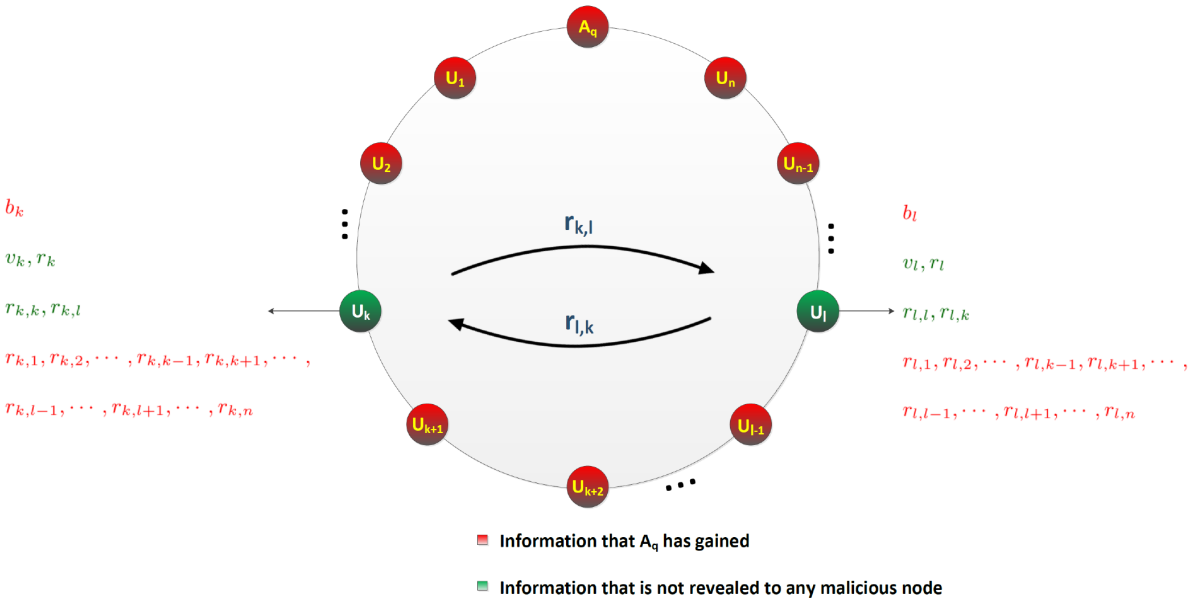

Figure 6: Robustness up to $n-1$ malicious nodes

to the two nodes in Figure 6).

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

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

and

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

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

$$b_k = \textcircled{v_k} + \sum_{j \neq k,l} (r_{k,j} - r_{j,k}) + \textcircled{r_{k,l} - r_{l,k}} \qquad (4)$$

and

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

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. □

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 $U \cup \{A_q\}$. This observation serves as a nice introduction to the following attack.

6.1. Alone in the List Attack

We conclude this section by considering one more attack that can be thought as equivalent to the case where $n-1$ nodes are compromised. Hence there can be no real defense against this scenario.

If A_q is malicious she can ask each node from U to give their vote *separately* (i.e. the cardinality of U will be one). By doing so, she will be able to find the value of all individual votes and thus easily break their privacy. This attack is a special case of having $n-1$ nodes compromised, in which only one user is legitimate. In such a scenario, a corrupted A_q will create a list with only one voter (U_1) which means that she will be able to break the privacy of U_1 . We believe that for this kind of attack there cannot be a complete solution and thus we propose two simple "countermeasures" that could just give more possibilities to U_1 to protect her privacy. In those situations where a node U_1 is the only one in the list, she can ask a "friendly"[1] node U_1' to give her vote about A_t. U_1' encrypts her vote v_1' with k_{U_1} and sends it to U_1. U_1 decrypts it, adds v_1 to v_1' and sends to A_q the sum of the two votes $v_1 + v_1'$. This way, it is impossible for A_q to find the individual votes, unless of course U_1' is also malicious.

[1]The term 'friendly' refers to two nodes that have a previously-established connection and both trust each other to some extent.

The above-mentioned solution can be also used in cases where $|U| > 1$ and $n-1$ nodes are compromised. Assume that U_l is the only legitimate node in U. At the initialization step where U_l receives the list U, she checks to see if U contains any node that U_l trusts. In the case where U_l does not find any "trustworthy" node or she suspects that the rest of the nodes are compromised, she invites a node that does not belong to U to give her vote about A_t. As before, A_q cannot infer the vote of U_l unless of course the helper node is also malicious. In some sense, we decrease the number of malicious nodes from $n-1$ to $n-2$ where StR provides protection.

7. Experimental Results

This section presents the implementation of StR, as well as a comparison with Dolev's Multiple Private Keys Protocol (MPKP) [10]. In order to prove the effectiveness of StR, we implemented both protocols in Java and we used JADE 4.0.1 [22] 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.

7.1. 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 7 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.

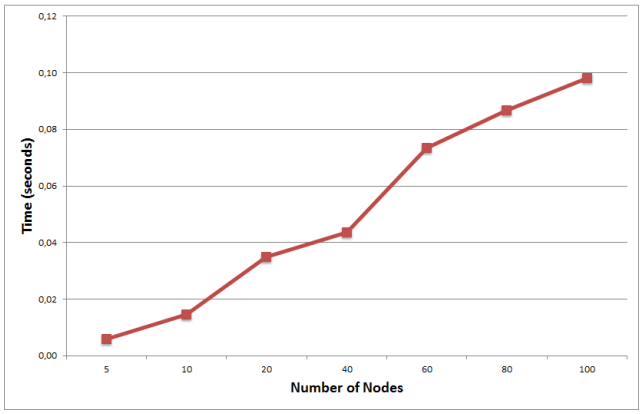

Figure 7: Processing time required by StR

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. Despite this inefficiency, we treat both times as *comparable* and we focus only on the communications aspects of both protocols.

7.2. Communication Delay

7.2.1. 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 8 illustrates the delay in seconds as a function of the number of nodes n.

7.2.2. 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

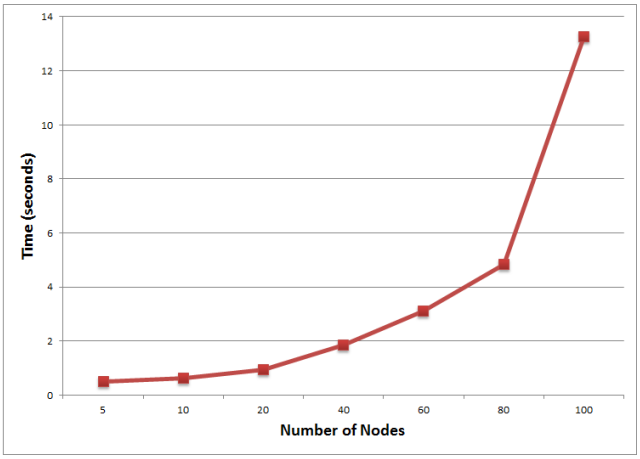

Figure 8: Communication Delay of first round of StR

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.

We have summarized these findings in Figure 9. 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.

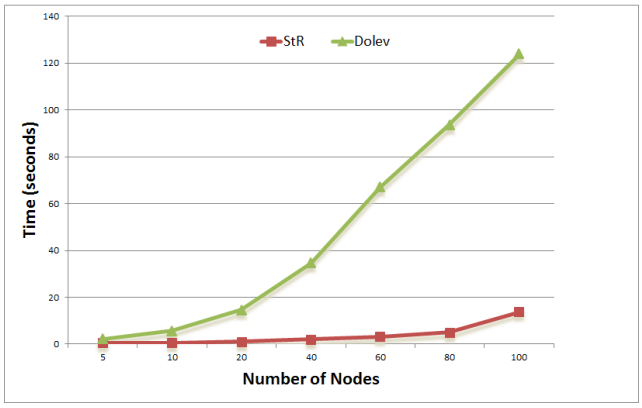

Figure 9: Communication Delay for StR and Dolev *et al.* protocols

8. StR^M: Beyond honest-but-curious behavior

The main drawback of the protocol described in the Section 6 is the fact that it is effective only under the semi-honest model. However, if we wish to prevent real malicious behavior, we have to build protocols that will assume that every adversary acts under the malicious model. It is obvious that in comparison to the semi-honest model, secure protocols within the malicious model enhance security. However, it is important to note that a malicious model may provide tighter security, at the expense of a greater computational costs. In this section, we present an extension of *StR* that effectively manages malicious adversaries, adversaries that may provide dishonest input to bias the protocol or try to disrupt protocol execution.

The new protocol, StR^M, is based on the improved StR shown in Algorithm 2. However, to make this protocol resistant to malicious attacks, we need to augment it with certain cryptographic operations that will allow us to argue about its correctness in the malicious case. The two sub-protocols that will be used by StR^M are zero-knowledge proofs of *plaintext equality* and *set membership*.

8.1. Sub-protocols

In a zero-knowledge proof of plaintext equality (*ZK-PEQ*), a prover convinces a verifier that two messages which are encrypted under different public keys correspond to the same plaintext message. In our case, we will be encrypting messages using the Pailier cryptosystem. So, if $E_i(m)$ and $E_j(m)$ are the encryptions of the message m using the public keys of users i and j, respectively, then a prover can convince a verifier that these ciphertexts

correspond to the same plaintext m. This operation is very critical to the correctness of StR^M as it will allow node U_i to convince the remaining nodes that the shares it sends to the other parties are the same like the ones used in the construction of its random number and its blinded vote.

Such a protocol for plaintext equality is described in [23]. This protocol can be made *non-interactive* by making the challenge of the verifier equal to the hash of the protocol messages. A description of the protocol can be found in the Appendix. In what follows we will denote by *ZK-PEQ*$(E_i(m), E_j(m))$ an execution of the protocol on ciphertexts $E_i(m)$ and $E_j(m)$ encrypted with the public keys of nodes i and j, respectively.

Another useful building block is a protocol that can be used to prove in zero knowledge that a ciphertext encrypts a message that lies in a predefined range R of values. This is required in order to ensure that each voter cannot bias the sum of the secret votes by sending votes that are not within some allowable set of values. Such a protocol for plaintext equality is described in [23]. A non-interactive version of the protocol can be found in the Appendix. In what follows we will denote by *ZK-RANGE*$(R, E(v))$ an execution of the protocol where a prover convinces a verifier that $E(v)$ is the encryption of a message v from the range R.

8.2. Description of StR^M

We are now ready to proceed with the description of StR^M. Our goal would be to make the protocol resistant to adversaries that will not conform to protocol specifications. Such malicious adversaries may attempt to deviate from the protocol in order to violate the privacy of other participants. In particular, they may (*i*) refuse to participate in certain protocol steps or drop messages that are supposed to forward, (*ii*) provide incorrect values in order to bias the final result, and (*iii*) modify protocol messages or tamper with communication channels in order to gain an advantage over well behaving users. In what follows we describe in detail how the protocol manages to address these issues. As a result misbehaving users can be *detected* and can be penalized (say by adding them to a blacklist or even removing them from future consideration), which will also affect their reputation in the community.

Let A_q be the querying agent and let $U = \{U_1, U_2, \ldots, U_n\}$ be the set of users providing feedback to A_q. During the initialization phase of the protocol, each user i picks n random numbers $r_{i,1}, r_{i,2}, \ldots, r_{i,n}$. These will be used to blind its vote v_i later on. Next, user i encrypts its vote v_i and the $r_{i,j}$'s to produce $E_i(v_i)$ and $E_i(r_{i,j})$, respectively. It also encrypts each share $r_{i,j}$ with the public key of user j to produce the encryptions $E_j(r_{i,j})$.

Algorithm 3 StR^M 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 and calculates the n-shares $r_{i,1}, \ldots, r_{i,n}$
  U_i computes $E_i(v_i)$, $E_i(r_{i,j})$ & $E_j(r_{i,j}))$
  for all $U_j \in U \setminus \{U_i\}$ **do**
    U_i sends $E_i(r_{i,j})$ and $E_j(r_{i,j})$ to U_j
    U_i proves that v_i is valid using protocol *ZK-RANGE*$(R, E_i(v_i))$
    U_i proves that $D_i(E_i(r_{i,j})) = D_j(E_j(r_{i,j}))$ by using protocol *ZK-PEQ*$(E_i(r_{i,j}), E_j(r_{i,j}))$
  end for
  U_i waits until it receives *all* encrypted shares $E_i(r_{j,i})$ destined to it and calculates the encrypted blinded vote $E_i(b_i) = E_i(v_i)\frac{\prod_{j\neq i} E_i(r_{i,j})}{\prod_{j\neq i} E_i(r_{j,i})} = E_i(v_i + \sum_{j\neq i} r_{i,j} - \sum_{j\neq i} r_{j,i})$
end for
Round 2 - All nodes in parallel
for $i = 1$ to n **do**
  U_i Computes $E_q(b_i)$ and sends to A_q the values $E_q(b_i)$, $E_i(b_i)$ & *ZK-PEQ*$(E_q(b_i), E_i(b_i))$
  to prove that $D_q((E_q(b_i)) = D_i(E_i(b_i))$
end for
A_q computes $E_i(b_i)$ itself from the shares published in the first round and
verifies that all shares were incorporated correctly
A_q decrypts $E_q(b_i)$ & and computes $\sum_{i=1}^{n} b_i = \sum_{i=1}^{n} v_i$

User i then proceeds to send these values to all participants of the protocol (without loss of generality we will assume that this step can be implemented by a bulletin board where participants may post messages that can be seen by everybody). It then goes on to prove in zero knowledge that (i) its vote v_i lies in the specified range using protocol *ZK-RANGE*$(R, E_i(v_i))$, and (ii) the plaintext equality of the ciphertexts $E_i(r_{i,j})$ and $E_j(r_{i,j})$ using protocol *ZK-PEQ*$(E_i(r_{i,j}), E_j(r_{i,j}))$. This last part is necessary in order to ensure any third party that these ciphertexts correspond to the encryption of the share $r_{i,j}$ using the public keys of nodes i and j, respectively. It then sends the encrypted share $E_j(r_{i,j})$ to U_j as in the simplified StR (Algorithm 2).

U_i then waits until it receives all encrypted shares $E_i(r_{j,i})$, destined to it. Using the homomorphic property of the Paillier cryptosystem it combines these shares with its encrypted vote and the shares $E_i(r_{i,j})$ it sent to the

other users in the previous step to compute the product

$$p_i = E_i(v_i)\frac{\prod_{j\neq i} E_i(r_{i,j})}{\prod_{j\neq i} E_i(r_{j,i})} = E_i(v_i + \sum_{j\neq i} r_{i,j} - \sum_{j\neq i} r_{j,i}) = E_i(b_i), \qquad (6)$$

where $b_i = v_i + \sum_{j\neq i} r_{i,j} - \sum_{j\neq i} r_{j,i}$ is the blinded vote.

It then encrypts b_i with the public key of A_q to produce the ciphertext $E_q(b_i)$ and sends A_q both p_i and $E_q(b_i)$ along with a plaintext equality proof *ZK-PEQ*$(p_i, E_q(b_i))$, thus demonstrating that these correspond to the same plaintext b_i. As A_q *itself* (or *any* other agent for that matter) can compute the product p_i from the encrypted values published in the first round, it concludes that all shares were incorporated correctly by user i in producing b_i. After verifying this for every i, A_q decrypts the received blinded votes $E_q(b_i)$ and computes the sum $\sum_{i=1}^{n} b_i = \sum_{i=1}^{n} v_i$. A concise description of StR^M is shown in Algorithm 3. The shaded parts in the protocol highlight the extra operations needed compared to Algorithm 2 in order to offer protection against malicious adversaries.

8.3. Security analysis of StR^M

In Sections 5 and 6 we proved the resistance of our protocol against numerous attacks regarding the privacy of individual votes. More precisely, we showed that even if A_q and up to $n-2$ voters are compromised, our protocol protects the privacy of the remaining legitimate ones through secret splitting. In the case of StR^M, we entirely relied on the algorithm of the improved StR (Algorithm 2) with the addition of some cryptographic mechanisms that offer protection in the case of malicious adversaries. The use of these cryptographic mechanisms does not affect the main operation of the algorithm, hence the privacy of individual voters is also successfully protected in the case of StR^M even if A_q and $n-2$ nodes are corrupted.

The main drawback of StR is its inability to ensure that (i) ratings are provided correctly, and (ii) that are within a predefined range. StR^M circumvents this vulnerability with the use of zero knowledge proofs. The zero knowledge proofs are characterized by the existence of public algorithms that can distinguish between valid and invalid encrypted texts. The main advantage of public verifiable schemes is the fact that the validity of the shares distributed by an agent can be verified by anyone, not just the owner; thus anyone can verify that the protocol run correctly and that each voter acted according to the specifications of the protocol. Furthermore, during the verification process, the original data are never revealed to anyone, even those that take part in the process. As a result, A_q or any other agent

can verify that submitted values from the participating nodes are valid. This is done with the use of plaintext equality proofs for the distributed shares and the set membership proofs for the submitted votes. Hence not only malicious behavior is detected but also the correct computation of the results is guaranteed.

8.4. Comparison of Improved StR with StR^M

StR^M is based on the improved StR with the addition of some cryptographic mechanisms that make the protocol resistant to malicious behavior. However, while these mechanisms improve the robustness of the scheme, they increase the number of messages that each agent needs to create and exchange. More precisely, during the first round of the improved StR each node sends $n-1$ shares to the rest of the participants, and just one message (b_i) to the query agent during the second round. Thus every node sends a total of n messages overall, for a total of n^2 messages.

During the first round of StR^M, each node sends $2n-2$ encrypted messages. Additionally, one range proof is used to prove that a vote lies within a certain range and $n-1$ plaintext equality proofs to prove that the ciphertexts $E_i(r_{i,j})$ and $E_j(r_{i,j})$ are equal. For the plaintext equality proof, each agent has to send a message that contains five random numbers. For a range proof that a vote v belongs to an interval $[a, b]$, a prover must send a message consisting of $3|a-b|$ random numbers. During the second round, each node sends to the query agent two encrypted messages as well as a plaintext equality proof. In total, each agent sends $3|a-b|+7n$ messages which is equal to $(7+o(1))n$ if we make the reasonable assumption that the range $|a-b|$ is $o(n)$, say from 0 to 100. This makes the protocol about 7 times slower than the improved StR but still faster than Dolev's *et al.* algorithm for the semi-honest case, as Figure 9 clearly demonstrates.

9. 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 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 the 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.

10. 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. Finally, while the previous protocols worked well in

the semi-honest model, we were also able to extend these protocols to handle the case of malicious adversaries, i.e. adversaries that can deviate from the protocol steps by dropping messages, refusing to participate, tampering with communications or providing out-of-range values in order to break the secrecy of votes of the remaining participants.

References

[1] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, pp. 618–644, Mar. 2007.

[2] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," in *The Economics of the Internet and E-Commerce, volume 11 of Advances in Applied Microeconomics*, SFCS '85, (Washington, DC, USA), pp. 383–395, IEEE Computer Society, 1985.

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

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

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Reputation in P2P Anonymity Systems," in *In Workshop on Economics of Peer-to-Peer Systems*, 2003.

[6] L. Lilien and B. Bhargava, "Privacy and trust in online interactions," pp. 85–122, IGI Global, 2009.

[7] T. Dimitriou, G. Karame, and I. Christou, "Supertrust - a secure and efficient framework for handling trust in super peer networks," *9th International Conference on Distributed Computing and Networking (ICDCN 2008)*, pp. 350–362, 2008.

[8] B. Yu and M. P. Singh, "Detecting deception in reputation management," in *Proceedings of the 2nd International joint Conference on Autonomous agents and multiagent systems*, AAMAS '03, (New York, NY, USA), pp. 73–80, ACM, 2003.

[9] O. Hasan, L. Brunie, and E. Bertino, "k-shares: A privacy preserving reputation protocol for decentralized environments.," in *SEC* (K. Rannenberg, V. Varadharajan, and C. Weber, eds.), vol. 330 of *IFIP Advances in Information and Communication Technology*, pp. 253–264, Springer, 2010.

[10] 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," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, (New York, NY, USA), pp. 1460–1465, ACM, 2010.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes.," in *EUROCRYPT* (J. Stern, ed.), vol. 1592 of *Lecture Notes in Computer Science*, pp. 223–238, Springer, 1999.

[12] J. Benaloh, "Dense probabilistic encryption," in *In Proceedings of the Workshop on Selected Areas of Cryptography*, pp. 120–128, 1994.

[13] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Explor. Newsl.*, vol. 4, pp. 28–34, Dec. 2002.

[14] R. Sheikh, B. Kumar, and D. K. Mishra, "A distributed k-secure sum protocol for secure multi-party computations," *CoRR*, vol. abs/1003.4071, 2010.

[15] S. Dolev, N. Gilboa, and M. Kopeetsky, "Computing trust anonymously in the presence of curious users," in *In Proceedings of the International Symposium on Stochastic Models in Reliability Engineering, Life Science and Operations Management*, (Beer Sheva, Israel), Feb. 2010.

[16] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.)," *IEEE Trans. Inf. Theor.*, vol. 24, pp. 106–110, Sept. 2006.

[17] S. A. Weis, *New foundations for efficient authentication, commutative cryptography, and private disjointness testing.* PhD thesis, Cambridge, MA, USA, 2006. AAI0810110.

[18] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, (London, UK, UK), pp. 129–140, Springer-Verlag, 1992.

[19] O. Hasan, L. Brunie, E. Bertino, and N. Shang, "A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model," Tech. Rep. RR-LIRIS-2012-008, LIRIS UMR 5205 CNRS/INSA de Lyon/Universit Claude Bernard Lyon 1/Universit Lumire Lyon 2/cole Centrale de Lyon, June 2012.

[20] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270 – 299, 1984.

[21] O. Goldreich, "Foundations of cryptography," vol. 2, Cambridge University Press, 2004.

[22] F. Bellifemine, A. Poggi, G. Rimassa, and T. Italia, "Jade," 1999.

[23] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard, "Practical multi-candidate election system," in *Proceedings of the 20th annual ACM symposium on Principles of Distributed Computing*, PODC '01, (New York, NY, USA), pp. 274–283, ACM, 2001.

[24] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proceedings of the 17th Annual ACM symposium on Theory ofCcomputing*, STOC '85, (New York, NY, USA), pp. 291–304, ACM, 1985.

Appendix A. Zero-Knowledge Proofs

Zero-knowledge proofs were introduced by Goldwasser *et al.* [24] and are interactive protocols that allows a party (the prover) to convince another party (the verifier) that a statement is true without revealing any information except the fact that the statement is true. In this section we will present the non-interactive versions of the proofs that we used in the description of StR^M (Section 8). The proofs can be considered as an extension of the interactive protocols that were presented in [23].

Appendix A.1. Non-Interactive Proof of Plaintext Equality

In a zero-knowledge proof of equality we assume that $E_i(m)$ and $E_j(m)$ are encryptions of a message m with the public key of U_i and U_j respectively. In such a proof, a prover P can convince a verifier V that $D_i(E_i(m)) = m = D_j(E_j(m))$.

Let (N_i, g) be the public key of U_i where N_i is an RSA modulus $N_i = pq$ such that p and q primes. Let g be an integer of order multiple of N_i modulo N_i^2 and $\mathcal{H}$ a secure cryptographic hash function. The non-interactive version of the interactive protocol presented in [23] follows:

Algorithm 4 Non-Interactive Proof of Plaintext Equality

Prover (P)

Picks a random $\rho \in [0, 2^l)$

Randomly picks $s_i \in \mathbb{Z}^*_{N_i}$ and $s_j \in \mathbb{Z}^*_{N_j}$

Computes $u_i = g_i^{\rho} s_i^{N_i} \bmod N_i^2$ and $u_j = g_j^{\rho} s_j^{N_j} \bmod N_j^2$

Computes $e = \mathcal{H}(u_i, u_j)$

Computes $z = \rho + me$

Computes $v_i = s_i r_i^e \bmod N_i$ and $v_j = s_j r_j^e \bmod N_j$

Sends to V the following: z, u_i, u_j, v_i, v_j

Verifier (V)

Computes $e = \mathcal{H}(u_i, u_j)$

Validates that $z \in [0, 2^l)$

Validates that $g_i^z v_i^{N_i} = u_i E_i(m)^e \bmod N_i^2$ and $g_j^z v_j^{N_j} = u_j E_j(m)^e \bmod N_j^2$

Appendix A.2. Non-Interactive Range Proof

In a zero-knowledge range proof a prover P can convince a verifier V that an encrypted message is an element of a certain set S. More precisely, if we assume that $S = \{m_1, \ldots, m_p\}$ is a public set of p messages and $E_i(m)$ is an encryption of a message m with the public key of U_i then P can convince V that $E_i(m)$ encrypts a message in S.

The non-interactive version of the protocol presented in [23] follows:

Algorithm 5 Non-Interactive Range Proof

Prover (P)

Picks a random $\rho \in \mathbb{Z}^*_N$

Randomly picks $p-1$ values $\{e_j\}_{j \neq i} \in \mathbb{Z}_N$ & $p-1$ values $\{v_j\}_{j \neq i} \in \mathbb{Z}_N$

Computes $u_i = \rho^N \bmod N^2$ & $\left\{u_j = v_j^N (g^{m_j}/E_P(m))_j^e \bmod N_j^2\right\}$

Computes $e = \mathcal{H}\Big(\{u_j\}_{j \in \{1,\ldots,p\}}\Big)$

Computes $e_i = e - \sum_{j \neq i} e_j \bmod N$ and $v_i = \rho r^{e_i} g^{e_i/N} \bmod N$

Sends to V the following: $\{u_j, v_j, e_j\}_{j \in \{1,\ldots,p\}}$

Verifier (V)

Calculates $e = \mathcal{H}\Big(\{u_j\}_{j \in \{1,\ldots,p\}}\Big)$

Checks that $e = \sum_j e_j \bmod N$

Checks that $v_j^N = u_j (E_P(m)/g^{m_j})^{e_j} \bmod N^2$, $j \in \{1, \ldots, p\}$