A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model

Omar Hasan[1], Lionel Brunie[1], Elisa Bertino[2], and Ning Shang[3]
[1]University of Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
{omar.hasan,lionel.brunie}@insa-lyon.fr
[2]Department of Computer Science, Purdue University, IN 47907, USA
bertino@cs.purdue.edu
[3]Qualcomm Inc., 5775 Morehouse Dr, San Diego, CA 92121, USA
nshang@qualcomm.com

Technical Report
University of Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
June 2012

A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model

Omar Hasan*, Lionel Brunie*, Elisa Bertino†, and Ning Shang‡
*University of Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
{omar.hasan,lionel.brunie}@insa-lyon.fr
†Department of Computer Science, Purdue University, IN 47907, USA
bertino@cs.purdue.edu
‡Qualcomm Inc., 5775 Morehouse Dr, San Diego, CA 92121, USA
nshang@qualcomm.com

***Abstract*—Users hesitate to submit negative feedback in reputation systems due to the fear of retaliation from the recipient user. A privacy preserving reputation protocol protects users by hiding their individual feedback and revealing only the reputation score. We present a privacy preserving reputation protocol for the malicious adversarial model. The malicious users in this model actively attempt to learn the private feedback values of honest users as well as to disrupt the protocol. Our protocol does not require centralized entities, trusted third parties, or specialized platforms, such as anonymous networks and trusted hardware. Moreover, our protocol is efficient in terms of the number of messages exchanged: $O(n) + O(log\ N)$, where n and N are the number of users in the protocol and the environment respectively.**

***Index Terms*—Reputation, privacy, trust, decentralization, malicious adversarial model.**

I. Introduction

Reputation systems are a powerful security tool for modern distributed applications in which there are a massive number of users who consume as well as provide resources however their trustworthiness is unknown. The reputation of a user is computed as the function of feedback about them gathered from fellow users. A reputation system mitigates the disruptive effect of malicious users by assigning them low reputation scores and consequently by limiting their capabilities and influence in the application.

Examples of reputation systems include: (1) Reputation systems for online auction and e-commerce sites such as ebay.com and amazon.com, which identify fraudulent vendors. (2) iovation.com protects businesses from online fraud by using a reputation system to expose devices such as computers, tablets and smart phones that are associated with chargeback, identity theft, and account takeover attacks. (3) Reputation systems for online programming communities such as advogato.org and stackoverflow.com filter users who post spam.

The issue with most existing reputation systems is that the feedback provided by users is public. This makes users hesitant to submit negative feedback due to the fear of retaliation from the recipient user [1]. A privacy preserving reputation protocol protects users by hiding their individual feedback and revealing only the reputation score.

In a previous paper [2], we presented the non-cryptographic k-shares decentralized privacy preserving reputation protocol for the semi-honest adversarial model. The agents (who represent users) in that model are supposed to follow the protocol according to the specification. In this paper, we present the Malicious-k-shares protocol, which provides security under the stronger malicious adversarial model. The agents in this model are unrestricted in their behavior to learn private information and to disrupt the protocol.

A. Contributions

In the Malicious-k-shares protocol, an agent can preserve its privacy by partially trusting on only k fellow feedback providers, where k is much smaller than $n-1$, the size of the set of all fellow feedback providers. This idea is central in our protocol and allows us to build a protocol that requires only $O(n) + O(log\ N)$ messages, where n and N are the number of agents in the protocol and the environment respectively. This approach improves on the classic approach (as employed by Gudes et al. [3] and Pavlov et al. [4]) where an agent is required to partially trust on all $n-1$ fellow feedback providers to preserve its privacy which results in high communication complexity. In this paper, we use three real and large trust graphs to demonstrate that a high majority of agents can find k sufficiently trustworthy agents in a set of $n-1$ fellow feedback providers such that k is very small compared to $n-1$ (Section V-B).

Agents in our protocol can quantify the risk to their privacy before submitting their feedback. This allows us to extend the protocol such that agents can abstain if the risk to their privacy is above the desired threshold. We show using the three real trust graphs that even if agents abstain, accurate reputation values can be computed from the feedback submitted by only those agents whose privacy is preserved (Section V-D).

The Malicious-k-shares protocol prevents malicious agents from taking two actions that are particularly challenging to address in a decentralized privacy preserving reputation system without relying on trusted third parties: (1) A malicious agent can take advantage of private feedback and submit a value that is outside the valid interval for feedback. For example, a malicious agent can submit a value such as -99 when the

feedback interval is $[0, 1]$. (2) A malicious agent can make erroneous computations, for example, it can report a random number instead of reporting a correct sum.

The protocol addresses the above challenges through innovative constructions (described in Section IV-A) based on set-membership and plain-text equality non-interactive zero-knowledge proofs and an additive homomorphic cryptosystem. To the best of our knowledge, our protocol is the most efficient decentralized additive privacy preserving reputation protocol in terms of number of messages exchanged under the malicious adversarial model. It requires exchange of only $O(n) + O(log\ N)$ messages. Compare this to the protocol by Pavlov et al. [4] that requires $O(n^3) + O(N)$ messages using similar building blocks.

B. Outline

In Section II, we give a general framework for decentralized privacy preserving reputation systems in the malicious adversarial model. In Section III, we describe some building blocks that we utilize in the construction of our protocol. In Section IV, we present our proposal for a new decentralized privacy preserving reputation protocol. We also analyze the security and the complexity of the protocol in this section. In Section V, we use three real trust graphs to experimentally evaluate two hypotheses that the protocol is based on. In Section VI, we give a comparison of our protocol with other reputation systems in the literature. We conclude and present directions for future work in Section VII.

II. Framework

In this section, we establish a framework that allows us to describe and analyze the protocol in Section IV. However, the reader may skip directly to Section IV-A for a quick overview of the protocol without delving into the specifics of the framework.

A. Agents, Trust, and Reputation

We model our environment as a multi-agent environment. An agent represents a user. Let $\mathbb{A}$ denote the set of all agents in the environment. $|\mathbb{A}| = N$.

We subscribe to the definition of trust by sociologist Diego Gambetta [5], which characterizes trust as binary-relational, directional, contextual, and quantifiable as subjective probability. Our formal definition of trust attempts to capture each of these characteristics.

Let $\mathbb{D}$ denote an asymmetric binary relation on the set $\mathbb{A}$. Let $\mathbb{T} \subseteq \mathbb{D}$ be the set of all existing trust relationships between agents. $(a, b) \in \mathbb{T}$, where $a, b \in \mathbb{A}$, implies that an agent ***a*** has a trust relationship towards an agent ***b***.

Let Ψ denote the set of all actions. Examples of actions include: "prescribe correct medicine", "deliver product sold online", "preserve privacy", etc.

Let $perform$ denote a function, such that $perform : \mathbb{T} \times \Psi \rightarrow \{true, false\}$. The function $perform(a, b, \psi)$ outputs $true$ if agent ***b*** performs the action ψ anticipated by agent ***a***, or it outputs false if ***b*** does not perform the anticipated action. Let the subjective probability $P(perform(a, b, \psi) = true)$ denote agent ***a***'s belief that agent ***b*** will perform the action ψ.

Definition 1: **Trust.** The trust of an agent ***a*** in an agent ***b*** is given as the triple $\langle a\mathbb{T}b, \psi, P(perform(a, b, \psi) = true)\rangle$, where $a, b \in \mathbb{A}$, $(a, b) \in \mathbb{T}$, $\psi \in \Psi$, and $P(perform(a, b, \psi) = true) \in [0, 1]$.

When the context of trust (action ψ) is clear, we adopt the simplified notation l_{ab} for $P(perform(a, b, \psi) = true)$. We can also refer to l_{ab} as agent ***a***'s feedback about agent ***b***.

An agent ***a*** is said to be a source agent of an agent ***b*** in the context of an action ψ if ***a*** has trust in ***b*** in the context ψ. The set of all source agents of an agent ***b*** in context ψ is given as $S_{b,\psi}$. The simplified notation S_b is used instead of $S_{b,\psi}$ when the context is clear.

Definition 2: **Reputation.** Let $S_t = \{a_1 \dots a_n\}$ be the set of source agents of an agent t in context ψ. The reputation of agent t in context ψ is given as:

$$rep_{\oplus}(l_{a_1 t} \dots l_{a_n t}) = \frac{\sum_{i=1}^{n} l_{a_i t}}{n} \tag{1}$$

The function $rep_{\oplus}$ implements the reputation of an agent t as the mean of the feedback values of its source agents, which is an intuitive statistic. The eBay reputation system (ebay.com), which is one of the most successful reputation systems, represents reputation as the simple sum of all feedback. We derive the mean from the sum in order to normalize the reputation values. The reputation of an agent t is denoted by $r_{t,\psi}$, or r_t when the context is clear.

Definition 3: **Reputation Protocol.** Let Π be a multi-party protocol. Then Π is defined as a Reputation Protocol, if (1) the participants of the protocol include: a querying agent q, a target agent t, and all n source agents of t in the context ψ, (2) the inputs include: the feedback of the source agents in context ψ, and (3) the output of the protocol is: agent q learns the reputation $r_{t,\psi}$ of agent t.

B. Adversary

We refer to the coalition of dishonest agents as the adversary. In this paper, we propose a solution for the malicious adversarial model. Malicious agents actively attempt to learn private information of honest agents as well as to disrupt the protocol. Specifically, malicious agents may (1) refuse to participate in the protocol, (2) prematurely abort the protocol, (3) selectively drop messages that they are supposed to send, (4) tamper with the communication channels, (5) wiretap the communication channels, and (6) provide incorrect information (for example, provide out of range values as their inputs, make incorrect computations).

C. Privacy

Definition 4: **Preservation of Privacy (by an Agent).** Let x be an agent **a**'s private data that agent **a** reveals to an agent **b**. Then agent **b** is said to preserve the privacy of agent **a** w.r.t. x, if (1) **b** does not use x to infer more information, and (2) **b** does not reveal x to any third party.

Let action $\rho =$ "preserve privacy".

Definition 5: **Trusted Third Party (TTP).** Let $S \subseteq \mathbb{A}$ be a set of n agents, and $TTP_S \in \mathbb{A}$ be an agent. Then TTP_S is a Trusted Third Party (TTP) for the set of agents S if for each $a \in S$, $P(perform(a, TTP_S, \rho) = true) = 1$.

We define *security threshold* as a parameter that can be assigned a value in $[0, 1]$ according to the security needs of an application. A value of the *security threshold* closer to 1 indicates a stricter security requirement. We consider as *high* any probability greater than or equal to the *security threshold*, and as *low* any probability less than $1-$ *security threshold*.

We adopt the Ideal-Real approach [6] to define privacy preserving reputation protocols.

Definition 6: **Ideal Privacy Preserving Reputation Protocol.** Let Π be a reputation protocol (Definition 3). Then Π is an ideal privacy preserving reputation protocol under a given adversarial model, if: (1) the inputs of all n source agents of t are private, (2) TTP_{S_t} is a participant, where $S_t = S_{t,\psi}$ is the set of all source agents, (3) $m < n$ of the source agents (given as set $\mathbb{M}$) and agents q and t are considered to be dishonest, however, q wishes to learn the correct output, (4) agents $S_t - M$ and TTP_{S_t} are honest, (5) as part of the protocol, TTP_{S_t} receives the private inputs from the source agents and outputs the reputation $r_{t,\psi}$ to agent q, and (6) over the course of the protocol, the private input of each agent $a \in S_t - \mathbb{M}$ may be revealed only to the TTP_{S_t}.

In an ideal privacy preserving reputation protocol, it is assumed that for each agent $a \in S_t - \mathbb{M}$, the adversary does not gain any more information about the private input of agent **a** from the protocol other than what he can deduce from what he knows before the execution of the protocol and the output, with probability $P(perform(a, TTP_{S_t}, \rho) = true) = 1$.

Definition 7: **Real Privacy Preserving Reputation Protocol.** Let $\mathcal{I}$ be an ideal privacy preserving reputation protocol (Definition 6). Then $\mathcal{R}$ is a real privacy preserving reputation protocol w.r.t. $\mathcal{I}$, if: (1) $\mathcal{R}$ has the same parameters (participants, private inputs, output, adversary, etc.) as $\mathcal{I}$, except that there is no TTP_{S_t} as a participant (2) with high probability, the adversary learns no more info. about the private input of any agent **a** than it can learn in protocol $\mathcal{I}$.

D. Problem Definition

Let $S_{t,\psi} = \{a_1 \dots a_n\}$ be the set of all source agents of agent t in the context of action ψ. Find a reputation protocol Π, which takes private input $l_{at} \equiv P(perform(a, t, \psi) = true)$ from each agent $a \in S_t$, and outputs the reputation $r_{t,\psi}$ of the target agent t to a querying agent q. Reputation is computed as $rep_{\oplus}$. Agents q, t, and m of the source agents are considered to be dishonest, where $m < n$. However, q wishes to learn the correct output and therefore does not take any actions that alter the output. The reputation protocol Π is required to be decentralized and secure under the malicious adversarial model. If computing $r_{t,\psi}$ is not possible due to the disruptive actions of certain agents, then the protocol outputs the identity of those agents to the querying agent q.

III. Building Blocks

A. Additive Homomorphic Cryptosystem

Let $E_a(.)$ denote the encryption function with the public key PK_a of agent ***a*** in an asymmetric cryptosystem $\mathcal{C}$. The cryptosystem $\mathcal{C}$ is said to be additive homomorphic if we can compute $E_a(x+y)$, given only $E_a(x)$, $E_a(y)$, and PK_a. As an example, let's consider two integers, 3 and 4. A cryptosystem $\mathcal{C}$ is additive homomorphic if given only $E_a(3)$, $E_a(4)$, and PK_a, we are able to obtain $E_a(3+4) = E_a(7)$. We use the Paillier additive homomorphic cryptosystem.

B. Zero-Knowledge Proof of Set Membership

Let $S = \{\mathfrak{m}_1, \dots, \mathfrak{m}_p\}$ be a public set of p messages, and $E(\mathfrak{m}_i)$ an encryption of $\mathfrak{m}_i$ with a prover's public key, where i is secret. A zero-knowledge proof of set membership allows the prover to convince a verifier that $E(\mathfrak{m}_i)$ encrypts a message in S. In a non-interactive version of the zero-knowledge proof of set membership, we abstract the part of the proof generated by the prover as the function $setMembershipZKP(E(\mathfrak{m}_i), S)$, abbreviated as $smzkp(E(\mathfrak{m}_i), S)$. Our implementation of a non-interactive zero-knowledge proof of set membership inspired by Baudron et al. [7] is given in Figure 7.

C. Zero-Knowledge Proof of Plaintext Equality

Let $E_1(\mathfrak{m})$ and $E_2(\mathfrak{m})$ be encryptions of a message $\mathfrak{m}$ with the public key of agents 1 and 2 respectively. A zero-knowledge proof of plaintext equality allows a prover to convince a verifier that $E_1(\mathfrak{m})$ and $E_2(\mathfrak{m})$ encrypt the same message. In a non-interactive version of the zero-knowledge proof of plaintext equality, we abstract the part of the proof generated by the prover as the function $plaintextEqualityZKP(E_u(\mathfrak{m}), E_v(\mathfrak{m}))$, abbreviated as $pezkp(E_u(\mathfrak{m}), E_v(\mathfrak{m}))$. Our implementation of a non-interactive zero-knowledge proof of plaintext equality inspired by Baudron et al. [7] is given in Figure 8.

D. Source Managers

We define a *source manager* of an agent t as a fellow agent who maintains the set S_t. The idea of source managers is inspired by *score managers* in EigenTrust [8]. When a source agent assigns feedback to a target agent t, it reports that event to each of its source managers. The source managers add the source agent to the set S_t that they each maintain. A Distributed Hash Table (DHT), such as Chord [9], is used to locate the source managers. To learn a set $\hat{S}_t \supset S_t$, a querying agent can retrieve the set maintained by each of the source managers and take a union of the sets. The querying agent will learn $\hat{S}_t \supset S_t$ as long as at least one of the source managers is honest.

IV. The Malicious-k-shares Protocol

A. Protocol Overview

In the Malicious-k-shares protocol, each source agent ***a*** relies on at most k agents to preserve his privacy. Agent ***a***

selects these k agents based on his own knowledge of their trustworthiness in the context of preserving privacy and sends each of them an additive share of his private feedback value. The advantages of this approach are twofold. Firstly, an agent is able to quantify and maximize the probability that its privacy will be preserved. This also allows us to extend the protocol such that an agent can abstain from providing feedback if the risk to its privacy is high. Secondly, limiting the number of shares to $k \ll n$, results in a protocol that requires only $O(n)$ messages for the computation of a reputation score. An additional $O(log\ N)$ messages are required for communication with the source managers.

In the Malicious-k-shares protocol, each source agent ***a*** must prove that it has generated correct shares, that is, the sum of all shares is a value that lies in the correct interval for feedback. An agent ***a*** sends each of the k trusted agents a share encrypted with the recipient agent's public key. The shares are relayed through the querying agent q. We would like q to add these shares using the additive homomorphic property, however, this is not possible because the shares are encrypted with different keys. As a solution, agent ***a*** also encrypts each of the shares with his own public key and submits them to q. Additionally, it submits a set-membership zero-knowledge proof that the sum of these shares belongs to the correct interval. The querying agent can verify the veracity of this claim by using the additive homomorphic property to add the set of shares encrypted with agent ***a***'s key and then by verifying the proof. It still remains to show that the original shares sent to the trusted agents are correct. To show this, agent ***a*** gives a plaintext-equality zero-knowledge proof for each share that shows that a share encrypted with the recipient's public key and a share encrypted with the sender's public key contain the same plaintext. Verification of the equality of all pairs of shares verifies that agent ***a*** indeed sent correct shares.

In the Malicious-k-shares protocol, each source agent ***a*** must prove that is has computed the correct sum of the shares. The querying agent q can compute the encrypted sum of the shares from the copies of the encrypted shares that it received and relayed to agent ***a***. However, q cannot decrypt the sum because it is encrypted with the recipient agent's public key. Agent ***a*** computes the sum and sends it to q encrypted with q's public key. Agent ***a*** also sends a plaintext-equality zero-knowledge proof that shows that the encrypted sum independently computed by q and the encrypted sum sent by agent ***a*** hold the same value. Verification of the proof convinces q that agent ***a*** computed the sum correctly.

B. Protocol Outline

The important steps of the protocol are outlined below.

1) **Initiation.** The protocol is initiated by a querying agent q to determine the reputation $r_{t,\psi}$ of a target agent t. Agent q retrieves $S_t \equiv S_{t,\psi}$, which is the set of source agents of agent t in the context ψ. Agent q verifies S_t from the source managers of t. Agent q then sends S_t to each source agent $a \in S_t$.
2) **Select Trustworthy Agents.** Each agent $a \in S_t$ selects k other agents in S_t. Let's refer to these agents selected by ***a*** as the set $U_a = \{u_{a,1} \dots u_{a,k}\}$. Agent ***a*** selects these agents such that: $P(perform(a, u_{a,1}, \rho) = false) \times \dots \times P(perform(a, u_{a,k}, \rho) = false)$ is low. That is, the probability that all of the selected agents will collude to breach agent ***a***'s privacy is low.
3) **Prepare Shares.** Agent ***a*** then prepares $k+1$ shares of its secret feedback value l_{at}. The shares, given as: $x_{a,1} \dots x_{a,k+1}$, are prepared in the following manner: The first k shares are random numbers uniformly distributed over a large interval (for example, $[0, 2^{32}-1]$). The last share is selected as follows: $x_{a,k+1} = (l_{at} - \sum_{i=1}^{k} x_{a,i})\ mod\ M$, where M is a publicly known modulus.
   The preparation of the shares in this manner implies that: $(\sum_{i=1}^{k+1} x_{a,i})\ mod\ M = l_{at}$. That is, the sum of the shares $mod\ M$ is equal to the feedback value. The sum of the shares, $\sum_{i=1}^{k+1} x_{a,i}$, lies in the interval $[(h_a \times M), (h_a \times M) + L]$, where $h_a = (\sum_{i=1}^{k+1} x_{a,i})\ div\ M$, and $l_{at} \in [0, L]$.
   Since each of the $k+1$ shares is a number uniformly distributed over a large interval, no information about the secret can be learned unless all of the shares are known.
4) **Encrypt Shares.** Agent ***a*** then encrypts each of the $k+1$ shares with its own public key to obtain: $E_a(x_{a,1}) \dots E_a(x_{a,k+1})$. It also encrypts each share $x_{a,i}$ with the public key of agent $u_{a,i}$, for $i \in \{1 \dots k\}$, to obtain: $E_{u_{a,1}}(x_{a,1}) \dots E_{u_{a,k}}(x_{a,k})$.
5) **Generate Zero-Knowledge Proofs.** Agent ***a*** computes: $\beta_a = (E_a(x_{a,1}) \times \dots \times E_a(x_{a,k+1}))\ mod\ \mathfrak{n}_a^2$, where $\mathfrak{n}_a$ is the RSA modulus in the public key of agent ***a***. The result of this product is the encrypted sum of agent ***a***'s shares, that is $\beta_a = E_a(\sum_{i=1}^{k+1} x_{a,i})$ (due to the additive homomorphic property).
   Agent ***a*** then generates one non-interactive set membership zero-knowledge proof: $smzkp(\beta_a, [(h_a \times M), (h_a \times M) + L])$. The proof proves to a verifier that the ciphertext β_a encrypts a value that lies in the interval $[(h_a \times M), (h_a \times M) + L]$. In other words, the proof shows that the ciphertext contains a valid feedback value (considering $mod\ M$).
   Agent ***a*** also generates k non-interactive plaintext equality zero-knowledge proofs. Each proof $pezkp(E_a(x_{a,i}), E_{u_{a,i}}(x_{a,i}))$, where $i \in \{1 \dots k\}$, proves to a verifier that the two ciphertexts, one encrypted with the public key of ***a*** and the other encrypted with the public key of $u_{a,i}$, contain the same plaintext.
   A verifier who verifies these zero-knowledge proofs will be convinced that agent ***a*** has prepared the shares such that they add up to a correct feedback value. Moreover, the verifier will be assured that the shares destined for ***a***'s trustworthy agents correspond to those correct shares.
6) **Send Encrypted Shares and Proofs.** Agent ***a*** sends all encrypted shares, that is, $E_a(x_{a,1}) \dots E_a(x_{a,k+1})$ and $E_{u_{a,1}}(x_{a,1}) \dots E_{u_{a,k}}(x_{a,k})$, as well as all zero-knowledge proofs, that is, $smzkp(\beta_a, [(h_a \times M), (h_a \times$

$M) + L])$ and $pezkp(E_a(x_{a,i}), E_{u_{a,i}}(x_{a,i}))$, $i \in \{1 \dots k\}$, to agent q.

7) **Verify the Proofs.** Agent q independently computes β_a and verifies the proofs received from each agent ***a***. Their verification confirms that agent ***a*** has prepared the shares correctly. Agent q receives and verifies the proofs of all source agents before proceeding to the next step.
8) **Relay the Encrypted Shares.** Agent q relays to each agent ***a***, the encrypted shares received for it from agents who considered it trustworthy. That is, each encrypted share $E_{u_{v,j}}(x_{v,j})$, prepared by an agent v for agent $u_{v,j}$, is relayed to agent $u_{v,j}$.
The shares are relayed through agent q, therefore, any agent who drops a message would be easily identified. However, agent q does not learn any of the shares by relaying them since the shares are encrypted with the public keys of the destination agents.
9) **Compute Sum of the Shares.** Each agent ***a*** receives the encrypted shares of the agents who considered it trustworthy. Agent ***a*** computes γ_a as the product of those encrypted shares along with the ciphertext of its own $(k+1)$'th share $x_{a,k+1}$. Due to the additive homomorphic property, γ_a is an encryption of the sum of the plaintexts of those shares. Agent ***a*** decrypts γ_a to obtain the plaintext sum σ_a.
Adding the $(k+1)$'th share provides security in the case when ***a*** receives only one share. If there is no $(k+1)$'th share to add, then agent q would learn the received share. Secrecy of the $(k+1)$'th share itself is not critical to the security of the protocol.
10) **Encrypt the Sum.** Agent ***a*** then encrypts σ_a with agent q's public key to obtain $E_q(\sigma_a)$.
11) **Generate Zero-Knowledge Proof.** Agent ***a*** then generates a non-interactive plaintext equality zero-knowledge proof: $pezkp(\gamma_a, E_q(\sigma_a))$. The proof proves to a verifier that the two ciphertexts, one encrypted with the public key of ***a*** and the other encrypted with the public key of q, contain the same plaintext.
Agent q, who can independently compute γ_a, can be convinced by this proof that $E_q(\sigma_a)$ contains the correct sum of the shares.
12) **Send Encrypted Sum and Proof.** Agent ***a*** sends the encrypted sum $E_q(\sigma_a)$ and the zero-knowledge proof $pezkp(\gamma_a, E_q(\sigma_a))$ to agent q.
13) **Verify the Proof.** Agent q independently computes γ_a and verifies the zero-knowledge proof received from each agent ***a***. Its verification confirms that the agent has computed the sum of the shares correctly. Agent q receives and verifies the proofs of all source agents before proceeding to the next step.
14) **Compute Reputation.** Agent q decrypts $E_q(\sigma_a)$ to obtain σ_a for each agent $a \in S_t$. Agent q then computes $r_{t,\psi} = ((\sum_{a \in S_t} \sigma_a) \ mod \ M)/n$.

C. Protocol Specification

The protocol is specified in Figures 1 and 2. Table IV summarizes the notation used in the protocol specification.

For the purpose of this protocol, we consider feedback values to be integers in the range $[0, L]$ (for example, $[0, 10]$). The reputation computed by the protocol can be normalized to the interval $[0, 1]$ by dividing the result by L.

Let M be a publicly known modulus, such that $M > L$, and $\forall t \in A : \sum_{a \in S_t} l_{at} < M$. Moreover, M is sufficiently smaller than $2^{\mathbb{k}}$, where $\mathbb{k}$ is the security parameter — the length in bits of the RSA modulus $\mathbb{n}$ in the cryptographic keys of the agents (for example, $\mathbb{k} = 2048$, and $M = 2^{80}$).

Let $[0, X]$ be a large interval (for example, $[0, 2^{32} - 1]$).

To generate the zero-knowledge proof $setMembershipZKP(\beta_a, [(h_a \times M), (h_a \times M) + L])$ in step 10 of the event PREP, an agent ***a*** requires the randomization $\mathbb{r}_{\beta_a}$ of the encryption β_a, which can be computed as follows: $\mathbb{r}_{\beta_a} = \mathbb{r}_{a,1} \times \ldots \times \mathbb{r}_{a,k+1}$, where $\mathbb{r}_{a,i}$ is the randomization used for the encryption of $E_a(x_{a,i})$.

To generate the zero-knowledge proof $plaintextEqualityZKP(\gamma_a, E_q(\sigma_a))$ in step 4 of the event VERIFIED_SHARES, an agent ***a*** requires the randomization $\mathbb{r}_{\gamma_a}$ of the encryption γ_a, which can be computed as follows: $\mathbb{r}_{\gamma_a} = (g^{-\sigma_a} \cdot \gamma_a)^{1/\mathbb{n}_a \ mod \ (p-1)(q-1)} \ mod \ \mathbb{n}_a$, where g and $\mathbb{n}_a$ are in the public key of ***a***, and p and q are in the secret key, $\mathbb{n}_a = pq$.

The protocol uses the following functions: **timestamp()** – Returns current time. **random(α,β)** – Returns a random number uniformly distributed over the interval $[\alpha, \beta]$. **set_of_trustworthy(*a*, S)** – Returns a set of agents $U_a = \{u_{a,1} \dots u_{a,k}\}$, where $U_a \subseteq S$. The set U_a is selected such that: $P(perform(a, u_{a,1}, \rho) = false) \times \ldots \times P(perform(a, u_{a,k}, \rho) = false)$ is low.

D. Security Analysis – Correctness

First, we consider the **semi-honest model**. In the protocol Malicious-k-shares (PREP: lines 2 – 4), each agent $a \in S_t$ prepares the shares $x_{a,1} \dots x_{a,k+1}$, such that:

$$\sum_{i=1}^{k+1} x_{a,i} = (l_{at} + (h_a \times M)) \ mod \ M \tag{2}$$

The sum of all shares of all source agents can be given as:

$$\sum_{a \in S_t} \sum_{i=1}^{k+1} x_{a,i} = \sum_{a \in S_t} ((l_{at} + (h_a \times M)) \ mod \ M) \tag{3}$$

In the protocol (PREP: line 14, SHARES: line 10), each agent $a \in S_t$ sends its share $x_{a,i}$ to another agent in S_t through q, where $i \in \{1 \dots k\}$. Each agent $a \in S_t$ computes σ_a, which is the sum of the received shares and its own $(k+1)$'th share (VERIFIED_SHARES: lines 1 – 2). We deduce that $\sum_{a \in S_t} \sigma_a$ is the sum of all shares received by all agents (that is, $\sum_{a \in S_t} \sum_{i=1}^{k} x_{a,i}$) and all $(k+1)$'th shares (that is, $\sum_{a \in S_t} x_{a,k+1}$).

Protocol: Malicious-k-shares
Participants: Agents: q, t, and $S_t \equiv S_{t,\psi} = \{a_1 \ldots a_n\}$. Agents q, t, and a subset of $S_{t,\psi}$ of size $m < n$ are considered to be dishonest, however, q wishes to learn the correct output (and therefore does not disrupt the protocol). $n \geq 3$.
Input: Each source agent ***a*** has a private input $l_{at} \equiv P(perform(a, t, \psi) = true)$.
Output: Agent q learns $r_{t,\psi}$, the reputation of agent t in the context ψ, or agent q learns the identity of the agents who disrupt the protocol.
Setup: Each agent ***a*** maintains $S_a \equiv S_{a,\psi}$, the set of its source agents in the context ψ. All communication takes place over authenticated point-to-point channels that are resistant to wire-tapping and tampering.
Events and Associated Actions (for an Agent *a*):

need arises to determine r_t
▷ initiate query
1 send tuple $(\text{REQUEST\_FOR\_SOURCES}, \psi)$ to t
2 receive tuple $(\text{SOURCES}, \psi, S_t)$ from t
3 verify S_t from the source managers of t
4 retrieve public key PK_w of each agent $w \in S_t$ from a certificate authority
5 $S'_t \leftarrow S_t$ ▷ initialize the set of agents who are expected to send their shares
6 $\theta \leftarrow 0$ ▷ a cumulative sum for computing reputation
7 $V_w \leftarrow \phi$, for each agent $w \in S_t$ ▷ initialize the sets of encrypted shares
8 $s \leftarrow$ timestamp()
9 send tuple $(\text{PREP}, q, t, s, S_t)$ to each agent $w \in S_t$

tuple $(\text{REQUEST\_FOR\_SOURCES}, \psi)$ **received from agent** q
1 send tuple $(\text{SOURCES}, \psi, S_a)$ to q

tuple $(\text{PREP}, q, t, s, S_t)$ **received from agent** q
▷ select trustworthy agents
1 $U_a \leftarrow$ set_of_trustworthy(***a***, $S_t - \{a\}$)
▷ prepare shares
2 **for** $i \leftarrow 1$ **to** k
3 **do** $x_{a,i} \leftarrow \text{random}(0, X)$
4 $x_{a,k+1} \leftarrow (l_{at} - \sum_{i=1}^{k} x_{a,i}) \; mod \; M$
5 $h_a \leftarrow (\sum_{i=1}^{k+1} x_{a,i}) \; div \; M$
▷ retrieve public keys
6 retrieve the public key of each $u \in U_a$ and the public key of q from a certificate authority
▷ encrypt shares
7 encrypt $x_{a,1} \ldots x_{a,k+1}$ with the public key of ***a*** to obtain $E_a(x_{a,1}) \ldots E_a(x_{a,k+1})$
8 encrypt $x_{a,1} \ldots x_{a,k}$ with the public key of $u_{a,1} \ldots u_{a,k}$ respectively to obtain $E_{u_{a,1}}(x_{a,1}) \ldots E_{u_{a,k}}(x_{a,k})$
▷ generate zero-knowledge proofs
9 $\beta_a \leftarrow (E_a(x_{a,1}) \times \ldots \times E_a(x_{a,k+1}))$ mod $\mathbb{n}_a^2$
10 generate $\text{setMembershipZKP}(\beta_a, [(h_a \times M), (h_a \times M) + L])$
11 **for** $i \leftarrow 1$ **to** k
12 **do** generate $\text{plaintextEqualityZKP}(E_a(x_{a,i}), E_{u_{a,i}}(x_{a,i}))$
▷ send the encrypted shares and the proofs to q
13 $\overrightarrow{\mathcal{I}_a} \leftarrow \langle U_a, E_a(x_{a,1}), \ldots, E_a(x_{a,k+1}), E_{u_{a,1}}(x_{a,1}), \ldots, E_{u_{a,k}}(x_{a,k}), h_a, \text{setMembershipZKP}(\beta_a, [(h_a \times M), (h_a \times M) + L]), \text{plaintextEqualityZKP}(E_a(x_{a,1}), E_{u_{a,1}}(x_{a,1})), \ldots, \text{plaintextEqualityZKP}(E_a(x_{a,k}), E_{u_{a,k}}(x_{a,k}))\rangle$
14 send tuple $(\text{SHARES}, q, t, s, \overrightarrow{\mathcal{I}_a})$ to agent q

Fig. 1: Protocol: Malicious-k-shares

Protocol: Malicious-k-shares (contd.)

tuple $(\text{SHARES}, q, t, s, \overrightarrow{\mathcal{I}_v})$ **received from an agent** $v \in S_t$
▷ verify the set membership proof
1 $\beta_v \leftarrow (E_v(x_{v,1}) \times \ldots \times E_v(x_{v,k+1}))$ mod $\mathbb{n}_v^2$
2 verify $\text{setMembershipZKP}(\beta_v, [(h_v \times M), (h_v \times M) + L])$
▷ verify the plaintext equality proofs
3 **for** $i \leftarrow 1$ **to** k
4 **do** verify $\text{plaintextEqualityZKP}(E_v(x_{v,i}), E_{u_{v,i}}(x_{v,i}))$
▷ manage the sets of encrypted shares to be relayed
5 **for** $i \leftarrow 1$ **to** k
6 **do** $V_{u_{v,i}} \leftarrow V_{u_{v,i}} \cup \{E_{u_{v,i}}(x_{v,i})\}$
▷ subtract v from the set of agents who are yet to send their shares
7 $S'_t \leftarrow S'_t - \{v\}$
▷ if shares have been received from all source agents then relay the shares
8 **if** $S'_t = \phi$
9 **then** $S'_t \leftarrow S_t$ ▷ initialize the set of agents who are yet to send their sum
10 send tuple $(\text{VERIFIED\_SHARES}, q, t, s, V_w)$ to each agent $w \in S_t$

tuple $(\text{VERIFIED\_SHARES}, q, t, s, V_a)$ **received from agent** q
▷ compute sum of the shares
1 $\gamma_a \leftarrow ((\prod_{c \in V_a} c) \times E_a(x_{a,k+1}))$ mod $\mathbb{n}_a^2$
2 $\sigma_a \leftarrow D_a(\gamma_a)$
▷ encrypt the sum
3 encrypt σ_a with the public key of q to obtain $E_q(\sigma_a)$
▷ generate zero-knowledge proof
4 generate $\text{plaintextEqualityZKP}(\gamma_a, E_q(\sigma_a))$
▷ send the encrypted sum and the proof to agent q
5 send tuple $(\text{AGGREGATE}, q, t, s, E_q(\sigma_a), pezkp(\gamma_a, E_q(\sigma_a))$ to q

tuple $(\text{AGGREGATE}, q, t, s, E_q(\sigma_v), pezkp(\gamma_v, E_q(\sigma_v))$ **received from an agent** $v \in S_t$
▷ verify the proof
1 $\gamma_v \leftarrow ((\prod_{c \in V_v} c) \times E_v(x_{v,k+1}))$ mod $\mathbb{n}_v^2$
2 verify $\text{plaintextEqualityZKP}(\gamma_v, E_q(\sigma_v))$
▷ decrypt the sum
3 $\sigma_v \leftarrow D_q(E_q(\sigma_v))$
▷ compute intermediate sum for reputation
4 $\theta \leftarrow \theta + \sigma_v$
▷ subtract v from the set of agents who are yet to send their sum
5 $S'_t \leftarrow S'_t - \{v\}$
▷ if sum has been received from all source agents, compute reputation
6 **if** $S'_t = \phi$
7 **then** $r_{t,\psi} \leftarrow (\theta \; mod \; M)/n$

Fig. 2: Protocol: Malicious-k-shares (contd.)

$$\sum_{a \in S_t} \sigma_a = \sum_{a \in S_t} \sum_{i=1}^{k} x_{a,i} + \sum_{a \in S_t} x_{a,k+1} \quad (4)$$

$$= \sum_{a \in S_t} \sum_{i=1}^{k+1} x_{a,i} \quad (5)$$

In the protocol, θ is computed as follows (*need arises to determine* r_t: line 6, AGGREGATE: line 4):

$$\theta = \sum_{a \in S_t} \sigma_a \quad (6)$$

From equations 3, 4, and 6:

$$\theta = \sum_{a \in S_t} ((l_{at} + (h_a \times M)) \; mod \; M) \quad (7)$$

$$(\theta \; mod \; M)/n = ((\sum_{a \in S_t} ((l_{at} + (h_a \times M)) \; mod \; M)) \; mod \; M)/n \quad (8)$$

Since $l_{at} \leq L < M$ for each $a \in S_t$, and $\sum_{a \in S_t} l_{at} < M$, we get:

$$(\theta \; mod \; M)/n = (\sum_{a \in S_t} l_{at})/n \quad (9)$$

In the protocol (AGGREGATE: line 7), agent q learns the reputation of agent t in the context ψ as:

$$r_{t,\psi} = (\theta \; mod \; M)/n \quad (10)$$

From equations 9, 10, and equation 1, we conclude that agent q learns the correct reputation of agent t in the context ψ under the semi-honest model.

Now, we consider the **malicious model**. Malicious agents may **(1)** refuse to participate in the protocol, **(2)** prematurely abort the protocol, **(3)** selectively drop messages that they

are supposed to send, **(4)** tamper with the communication channels, **(5)** wiretap the communication channels, and **(6)** provide incorrect information (for example, provide out of range values as their inputs, make incorrect computations).

Agent q**.**

Agent q wishes to learn the correct output therefore it would not take any of the actions 1 to 4, and 6. Wiretapping the communication channels has no effect on the correctness of the protocol.

Each Agent $a \in S_t$**.**

Each agent $a \in S_t$ communicates exclusively with agent q. If an agent ***a*** takes any of the actions 1 to 3, it would be exposed as malicious to agent q. *Note:* Agent q can then remove the malicious agent from the set of source agents and restart the protocol. Eventually, only those agents who do not take actions 1 to 3 will remain in the set of source agents.

An agent $a \in S_t$ is unable to tamper with the communication channels since we assume that all communication takes place over authenticated point-to-point channels that are resistant to tampering. Since each agent $a \in S_t$ communicates exclusively with agent q, it will be exposed as malicious if it does not conform to these requirements.

Wiretapping the communication channels has no effect on the correctness of the protocol.

The first tuple of information that an agent $a \in S_t$ provides to agent q is: $(\text{SHARES}, q, t, s, \overrightarrow{\mathcal{I}_a})$, where $\overrightarrow{\mathcal{I}_a} = \langle\ U_a,\ E_a(x_{a,1}),\ \ldots,\ E_a(x_{a,k+1}),\ E_{u_{a,1}}(x_{a,1}),\ \ldots,\ E_{u_{a,k}}(x_{a,k}),\ \text{setMembershipZKP}(\beta_a, L),\ \text{plaintextEqualityZKP}(E_a(x_{a,1}),\ E_{u_{a,1}}(x_{a,1})),\ \ldots,\ \text{plaintextEqualityZKP}(E_a(x_{a,k}),\ E_{u_{a,k}}(x_{a,k}))\ \rangle$.

The correctness of the first four elements of the tuple and the set U_a can be trivially verified by agent q. The remaining information pertains to the shares prepared by agent ***a***. The shares have been prepared correctly if the following conditions hold true: **(1)** the shares add up to a value in $[(h \times M), (h \times M) + L]$; **(2)** $E_{u_{a,1}}(x_{a,1}),\ \ldots,\ E_{u_{a,k}}(x_{a,k})$ encrypt the same shares as $E_a(x_{a,1}),\ \ldots,\ E_a(x_{a,k})$ respectively; **(3)** $E_{u_{a,1}}(x_{a,1}),\ \ldots,\ E_{u_{a,k}}(x_{a,k})$ are encrypted with the public keys of agents $u_{a,1} \ldots u_{a,k}$ respectively.

The first condition holds true for an agent ***a*** if the verification of $\text{setMembershipZKP}(\beta_a, [(h_a \times M), (h_a \times M) + L])$ by agent q is successful. Agent q can verify the proof since it can independently compute β_a (due to the additive homomorphic property of the cryptosystem), L and M are publicly known, and h_a is provided by agent ***a***. An incorrect value of h_a will result in failure of the verification of the zero-knowledge proof. A zero-knowledge proof that shows membership in an interval with an incorrect h_a has no effect on the final output of the protocol since it is computed as $mod\ M$.

The second and third conditions hold true for an agent ***a*** if the verification of each $\text{plaintextEqualityZKP}(E_a(x_{a,i}), E_{u_{a,i}}(x_{a,i}))$ by agent q is successful, where $i \in \{1 \ldots k\}$. Agent q can verify these proofs since it can independently retrieve the public keys of agents ***a*** and $u_{a,1} \ldots u_{a,k}$ from a certificate authority.

If the verification of the one set-membership zero-knowledge proof and the k plaintext-equality zero-knowledge proofs provided by an agent ***a*** succeeds, it implies that agent ***a*** has provided correct information pertaining to the shares that it prepared. Otherwise, agent ***a*** can be considered as malicious.

The second tuple of information that an agent $a \in S_t$ provides agent q is: $(\text{AGGREGATE}, q, t, s, E_q(\sigma_a), pezkp(\gamma_a, E_q(\sigma_a))$.

The correctness of the first four elements of the tuple can be trivially verified by agent q. The remaining information pertains to the sum σ_a. The sum has been computed correctly if the following condition holds true: γ_a and $E_q(\sigma_a)$ encrypt the same plaintext.

The condition holds true for an agent ***a*** if the verification of $pezkp(\gamma_a, E_q(\sigma_a))$ by agent q is successful. Agent q can verify the proof since it can independently compute γ_a (due to the additive homomorphic property of the cryptosystem) and it can independently retrieve the public key of agent ***a*** from a certificate authority.

If the verification of the plaintext-equality zero-knowledge proof provided by an agent ***a*** succeeds, it implies that agent ***a*** has provided correct information pertaining to the sum σ_a. Otherwise, agent ***a*** can be considered as malicious.

Agent t**.**

We assume that agent q is able to retrieve the correct S_t from the source managers of agent t.

It follows that in the protocol Malicious-k-shares (Figure 1), agent q either learns the correct reputation of agent t in the context ψ, or learns the identity of a malicious agent who has disrupted the protocol, under the malicious adversarial model.

E. Security Analysis – Privacy

First, we consider the **semi-honest model**. Let's consider an agent $a \in S_t$. Agent ***a*** prepares the shares $x_{a,1} \ldots x_{a,k+1}$ of its secret feedback value l_{at}. The first k shares $x_{a,1} \ldots x_{a,k}$ are random numbers uniformly distributed over a large interval. The final share, $x_{a,k+1} = (l_{at} - \sum_{i=1}^{k} x_{a,i})\ mod\ M$, is also a number uniformly distributed over a large interval since it is a function of the first k shares which are random numbers. Thus, individually each of the shares does not reveal any information about the secret feedback value l_{at}. Moreover, no information is learned about l_{at} even if up to k shares are known, since their sum would be some random number uniformly distributed over a large interval. The only case in which information can be gained about l_{at} is if all $k+1$ shares are known. Then, $l_{at} = (\sum_{i=1}^{k+1} x_{a,i})\ mod\ M$.

We now analyze if the $k+1$ shares of an agent ***a*** can be learned by the adversary from the protocol.

Agent ***a*** sends each share $x_{a,i}$ only to agent $u_{a,i}$, where $i \in \{1 \ldots k\}$. Although, agent q relays each share $x_{a,i}$ from agent ***a*** to agent $u_{a,i}$, agent q or any third agent is unable to learn the share $x_{a,i}$ since it is sent encrypted with agent $u_{a,i}$'s public key as $E_{u_{a,i}}(x_{a,i})$. Only agent $u_{a,i}$ is able to decrypt $E_{u_{a,i}}(x_{a,i})$ and obtain $x_{a,i}$.

Each agent $u_{a,i}$ computes $\sigma_{u_{a,i}}$, which is the sum of all shares that it receives and its own final share $x_{u_{a,i},k+1}$. Even if agent ***a*** is the only agent to send agent $u_{a,i}$ a share, $\sigma_{u_{a,i}} = x_{a,i} + x_{u_{a,i},k+1}$. That is, the sum of agent ***a***'s share and agent $u_{a,i}$'s final share. Consequently, $\sigma_{u_{a,i}}$ is a number uniformly distributed over a large interval. Thus, when agent $u_{a,i}$ sends

this number to agent q, it is impossible for q to distinguish the individual shares from the number. Therefore, each share $x_{a,i}$ that agent $\boldsymbol{a}$ sends to agent $u_{a,i}$ will only be known to agent $u_{a,i}$. Unless, agent $u_{a,i}$ is dishonest. The probability that agent $u_{a,i}$ is dishonest, that is, it will attempt to breach agent $\boldsymbol{a}$'s privacy is given as: $P(perform(a, u_{a,i}, \rho) = false)$.

To learn the first k shares of agent $\boldsymbol{a}$, all agents $u_{a,1} \dots u_{a,k}$ would have to be dishonest. The probability of this scenario is given as: $P(perform(a, u_{a,1}, \rho) = false) \times \ldots \times P(perform(a, u_{a,k}, \rho) = false)$.

Even in the above scenario, the adversary does not gain information about l_{at}, without the knowledge of agent $\boldsymbol{a}$'s final share $x_{a,k+1}$. However, agent $\boldsymbol{a}$ has to send $\sigma_a = x_{a,k+1} + \sum_{v \in J_a} x_v$, and agent $\boldsymbol{a}$ has no control over the $\sum_{v \in J_a} x_v$ part of the equation. Therefore, we assume that agent q learns the final share of agent $\boldsymbol{a}$.

Thus the probability that the protocol will not preserve agent $\boldsymbol{a}$'s privacy can be stated as: $P(perform(a, u_{a,1}, \rho) = false) \times \ldots \times P(perform(a, u_{a,k}, \rho) = false)$. We assume that the agents $u_{a,1} \dots u_{a,k}$ are selected such that this probability is low. Therefore, with high probability, the adversary learns no more information about l_{at} than it can learn in the ideal protocol with what it knows before the execution of the protocol and the outcome.

Now, we consider the **malicious model**. Malicious agents may **(1)** refuse to participate in the protocol, **(2)** prematurely abort the protocol, **(3)** selectively drop messages that they are supposed to send, **(4)** tamper with the communication channels, **(5)** wiretap the communication channels, and **(6)** provide incorrect information (for example, provide out of range values as their inputs).

Privacy of Agent q and Agent t.

Agent q and agent t are not required to contribute any private information during the protocol.

Privacy of Each Agent $a \in S_t$.

Attack 1. Refuse to Participate in the Protocol. If a source agent v refuses to participate in the protocol, it has no effect on the privacy of any agent $\boldsymbol{a}$ since agent v must receive a share of agent $\boldsymbol{a}$'s private information before it can attack its privacy.

Agent t's refusal to participate also has no effect on the protocol. Agent q may retrieve S_t directly from agent t's source managers.

Agent q does not refuse to participate in the protocol since it wishes to learn the correct output of the protocol.

Attack 2. Prematurely Abort the Protocol. If a source agent v prematurely aborts the protocol before receiving the shares, it has no effect on the privacy of any agent $\boldsymbol{a}$ since agent v must receive a share of agent $\boldsymbol{a}$'s private information before it can attack its privacy. The other scenario is that the source agent v prematurely aborts the protocol after receiving a share of agent $\boldsymbol{a}$'s private information. In that case, all first k shares of agent $\boldsymbol{a}$ must still be known to breach $\boldsymbol{a}$'s privacy. Thus prematurely aborting the protocol does not give an agent $v \in S_t$ any advantage in learning agent $\boldsymbol{a}$'s private information.

If agent t aborts the protocol before providing S_t, agent q may retrieve S_t directly from agent t's source managers. Therefore agent t's disruption has no effect on the protocol.

Agent q has no incentive to prematurely abort the protocol since it wishes to learn the correct output of the protocol, which is not learned until after the last step.

Attack 3. Selectively Drop Messages. If a source agent v selectively drops messages or parts of messages, it has no effect on the condition that all first k shares of agent $\boldsymbol{a}$ must be known to breach an agent $\boldsymbol{a}$'s privacy. Thus, this is another action that does not give an agent $v \in S_t$ any advantage in learning agent $\boldsymbol{a}$'s private information.

Agent t may not provide S_t or may provide only a subset, however, that has no effect on the protocol since q also retrieves and verifies S_t from agent t's source managers.

Agent q does not selectively drop messages since it wishes to learn the correct output of the protocol. *Note:* Please see the discussion on Attack 6 for the case where agent q may relay incorrect shares or may not relay them at all.

Attack 4. Tamper with the Communication Channels. We assume that all communication takes place over authenticated point-to-point channels that are resistant to tampering.

Attack 5. Wiretap the Communication Channels. We assume that all communication takes place over authenticated point-to-point channels that are resistant to wiretapping.

Attack 6. Provide Incorrect Information. If a source agent v provides incorrect information, it has no effect on the condition that all first k shares of agent $\boldsymbol{a}$ must be known to breach an agent $\boldsymbol{a}$'s privacy. Agent v provides no information to agent $\boldsymbol{a}$ or agent q that would result in agent $\boldsymbol{a}$ divulging any extra information.

Agent t may provide an incorrect S_t, however, that has no effect on the protocol since q also retrieves and verifies S_t from agent t's source managers.

Agent q sends two types of messages to source agents: PREP, and VERIFIED_SHARES.

PREP: Agent q may create S_t itself in order to attack an agent $a \in S_t$. The set may be created such that it contains all dishonest agents except agent $\boldsymbol{a}$ who is under attack. However, we assume that $P(perform(a, u_{a,1}, \rho) = false) \times \ldots \times P(perform(a, u_{a,k}, \rho) = false)$ is low. That is, there exist trustworthy agents in the protocol such that agent $\boldsymbol{a}$ receives a high enough privacy guarantee.

VERIFIED_SHARES: Agent q may substitute the shares sent by other agents to an agent $\boldsymbol{a}$ with shares that it has created itself. Agent q may also not relay a share at all. In both these cases, the best outcome for q would be to learn agent $\boldsymbol{a}$'s $(k+1)$'th share. This has no effect on the privacy of agent $\boldsymbol{a}$ since agent q is still unable to learn its first k shares. Each of those shares is encrypted and can only be decrypted by its destination agent.

The protocol Malicious-k-shares is a real privacy preserving reputation protocol (Definition 7) under the malicious model, because: **(1)** Malicious-k-shares has the same parameters as the ideal protocol (except the TTP), and **(2)** the adversary does not learn any more information under the malicious adversarial model about the private input of any agent $\boldsymbol{a}$ in Malicious-k-shares than it can learn in the ideal protocol, with high probability: $1 - (P(perform(a, u_{a,1}, \rho) = false) \times \ldots \times P(perform(a, u_{a,k}, \rho) = false))$.

The protocol may be extended such that an agent $\boldsymbol{a}$ is

allowed to abstain if the privacy guarantee is not sufficient. The extension would be as follows: The agent who wishes to abstain would generate the shares such that their sum equals zero. The abstaining agent would inform the querying agent that it has abstained, and would prove that the sum of the shares equals zero.

The reader may refer to [2] for discussions on the privacy of trust relationships and attacks on the ideal protocol.

F. Complexity Analysis

TABLE I: Protocol Malicious-k-shares – Complexity.

Tuple	Occurrences	IDs	Ciphertexts	SMZKPs	PEZKPs
REQUEST _FOR _SOURCES	1				
SOURCES	1	n			
PREP	n	$n \times n = n^2$			
SHARES	n	kn	kn	n	kn
VERIFIED _SHARES	n		kn		
AGGREGATE	n		n		n
Total	$4n+2$	$n+n^2+kn$	$2kn+n$	n	$kn+n$
Complexity	$O(n)$	$O(n^2)$, for $k \ll n$	$O(n)$, for $k \ll n$	$O(n)$	$O(n)$, for $k \ll n$

The protocol requires $O(n)$ messages to be exchanged. The protocol also performs a DHT lookup in the initiation phase, which requires an additional $O(log\ N)$ messages (assuming Chord). Thus the total number of messages exchanged is $O(n)+O(log\ N)$, where n is the number of source agents in the protocol and N is the total number of agents in the system respectively.

In terms of bandwidth used, the protocol requires transmission of the following amount of information: $O(n^2)$ agent IDs, $O(n)$ ciphertexts, $O(n)$ non-interactive zero-knowledge proofs of set membership, and $O(n)$ non-interactive zero-knowledge proofs of plaintext equality.

V. Experimental Evaluation

We conduct experiments to evaluate the following two hypotheses that the Malicious-k-shares protocol is based on:

1) A source agent can preserve its privacy by trusting on only k fellow source agents, where k is much smaller than $n-1$, the size of the set of all fellow source agents.
2) Accurate reputation values can be computed even if the source agents whose privacy can not be preserved abstain and thus do not provide their feedback values.

A. Datasets

A trust graph can be defined as a weighted directed graph $G = (\mathbb{A}, \mathbb{T}, \mathbb{F})$, in which the set of vertices corresponds to the set of agents $\mathbb{A}$, the set of edges corresponds to the set of binary trust relationships $\mathbb{T}$, and the set of weights of the edges is given as a set of feedback values $\mathbb{F}$.

We use three real trust graphs as the datasets for our experiments. These three trust graphs have been independently evolved by the communities of advogato.org, squeak.org, and robots.net. The members of each of these communities rate each other in the context of being active and responsible members of the community. A common element between the three sites is that they use the same reputation system and thus offer the same set of feedback values. The choice of feedback values are *master*, *journeyer*, *apprentice*, and *observer*, with *master* being the highest level in that order. The trust graphs were obtained from the site trustlet.org on May 30, 2012.

Table II lists the number of users, the number of ratings, and the distribution of the ratings in each of the three trust graphs. Figure 3 shows the distribution of the potential target agents in each trust graph according to the minimum size of the set of their source agents. The graphs in Figure 3 also plot the instances of source agents in the trust graphs.

TABLE II: Trust Graphs.

	Advogato	Squeak	Robots
No. of users	14,020	766	16,620
No. of ratings	56,652	2,928	3,593
Ratings / user	4.04	3.82	0.22
master ratings	31.9%	31.8%	35.4%
journeyer ratings	40.0%	32.0%	26.0%
apprentice ratings	18.7%	33.2%	35.2%
observer ratings	9.4%	3.0%	3.4%

The members of the communities are expected to not post spam, not attack the reputation system, etc. Thus we consider that the context "be a responsible member of the community" comprises of the context "be honest". Since we quantify trust as probability, we heuristically substitute the four feedback values of the trust graphs as follows: $master = 0.99$, $journeyer = 0.70$, $apprentice = 0.40$, and $observer = 0.10$.

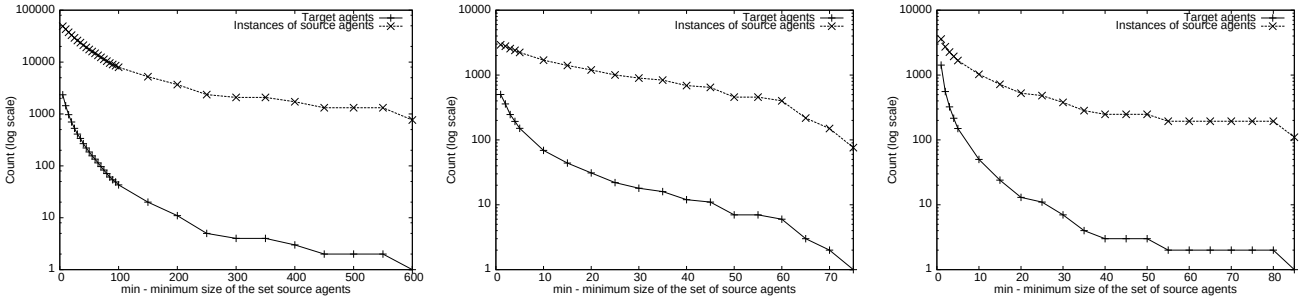
For the experiments, we define the lowest acceptable probability that privacy will be preserved as 0.90. This implies that a set of two trustworthy agents must include either one *master* rated agent or two *journeyer* rated agents for this threshold to be satisfied.

B. Experiment 1

1) Objective: Observe the effect of increasing the value of k on the percentage of the instances of source agents whose privacy is preserved.

2) Setup: The maximum number of fellow source agents that an agent can trust on is $n-1$. A fraction of the size of this set can be stated as $\kappa \times (n-1)$, where $\kappa \in [0,1]$. For our experiments, we equate $k = \lceil \kappa \times (n-1) \rceil$. This allows us to use κ (kappa) to vary the value of k as a fraction of $n-1$.

We query the reputation of all agents with at least min source agents. We vary κ from 0.01 to 1 with an increment of 0.01 and observe the percentage of the instances of source agents whose privacy is preserved. The set of experiments is run with $min \in \{10, 25, 50, 75, 100\}$ for the Advogato trust graph and with $min \in \{10, 15, 20, 25\}$ for the Squeak and Robots trust graphs. As discussed in Section IV-E, the privacy of a source agent $\boldsymbol{a}$ is preserved if $P(perform(a, u_{a,1}, \rho) = false) \times \ldots \times P(perform(a, u_{a,k}, \rho) = false)$ is low, that is less than or equal to 0.1 in our case. $u_{a,1} \ldots u_{a,k}$ are the set of agents that agent $\boldsymbol{a}$ trusts on.



(a) Advogato (b) Squeak (c) Robots

Fig. 3: Distribution of the potential target agents and the instances of source agents

3) Analysis: In the results of the experiment on the Advogato trust graph (Figure 4a), we observe that for $min = 25$, the privacy of 71% of the instances of source agents is preserved when $\kappa = 0.01$. That is, 71% of the source agents find sufficiently trustworthy agents among only 1% of their fellow source agents in order to preserve their privacy. The percentage is 82% at $\kappa = 0.04$ at which stage the function nearly converges and there is no significant improvement in the percentage by increasing κ any further. Convergence is reached at $\kappa = 0.03$ for the functions of $min = 50$ and above. Even for $min = 10$, convergence is reached at the fairly low value of $\kappa = 0.12$. It is thus evident that in the Advogato trust graph, a source agent can preserve its privacy by trusting on only k fellow source agents, where k is much smaller than $n-1$, the size of the set of all fellow source agents. Raising k over a certain threshold offers no advantage. This conclusion is also supported by the results of the experiments on the Squeak (Figure 4b) and Robots (Figure 4c) trust graphs. The Robots trust graph is quite sparse as compared to the other two graphs. The Robots trust graph has an average user to ratings ratio of only 0.22 compared to the Advogato and Squeak trust graphs that have a ratio of 4.04 and 3.82 respectively. Yet the percentage of the instances of source agents whose privacy is preserved converge very early in the Robots trust graph as well.

C. Experiment 2

1) Objective: Observe the effect of increasing the minimum size (min) of the set of source agents on the percentage of the instances of source agents whose privacy is preserved.

2) Setup: We vary min and observe the percentage of the instances of source agents whose privacy is preserved. The set of experiments is run with values of min from 5 upwards. κ is defined as in Experiment 1. We query the reputation of all agents with $\kappa \in \{0.05, 1\}$.

3) Analysis: In the results of the experiment on the Advogato trust graph (Figure 5a), we observe that for $min = 25$ and above, the privacy of over 82% of the instances of source agents is preserved. This percentage rises to over 86% for $min = 50$ and above. From $min = 450$ and above, this percentage is over 95%. Even at $min = 5$, the privacy of over 64% of the instances of source agents is preserved. It is thus clear that in the Advogato trust graph, the privacy of a high percentage of source agents can be preserved. The percentage rises sharply by increasing min up to 25 and then stabilizes up to 400. Thus any agent who participates as a source agent in the query of the reputation of a target agent with at least 25 source agents has over 82% probability of finding sufficiently trustworthy agents to preserve its privacy. Please note that the value of κ as 0.05 is quite conservative. The experiment run with $\kappa = 1$ gives fairly similar results, especially with $min = 25$ and above. Similar results are obtained from the experiment on the trust graph of Squeak (Figure 5b) where we observe that the privacy of over 90% of instances of source agents is preserved for $min = 10$ and above. The experiment on the Robots trust graph (Figure 5b) yields different results with the privacy of instances of source agents in the range of 46 to 57 percent being preserved. This could be attributed to the relative sparsity of the ratings in the trust graph.

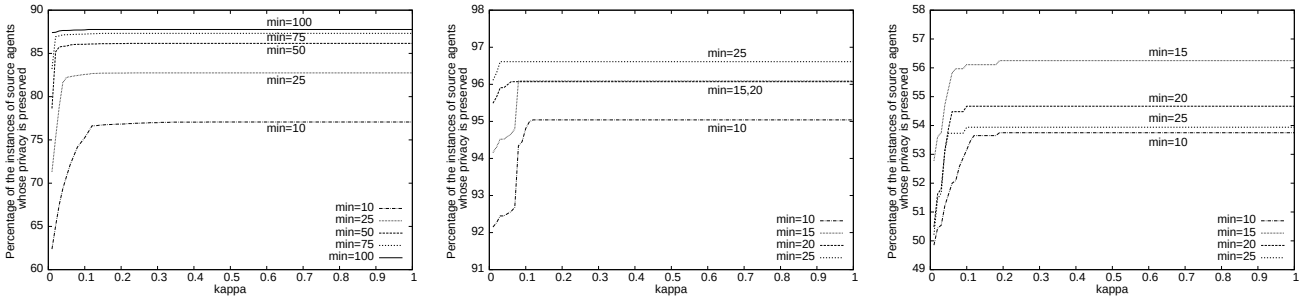
D. Experiment 3

1) Objective: Observe the accuracy of the reputation values computed when source agents whose privacy can not be preserved abstain and thus do not provide their feedback values.

2) Setup: Let $\mathbb{B}$ be the set of source agents that abstain and thus do not provide their feedback values, where $\mathbb{B} \subset S_t$ and S_t is the set of all source agents of the target agent t. Let r_t be the reputation computed using feedback from all source agents in S_t and let r'_t be the reputation computed using feedback from only the agents who do not abstain, that is, the agents in the set $S_t - \mathbb{B}$.

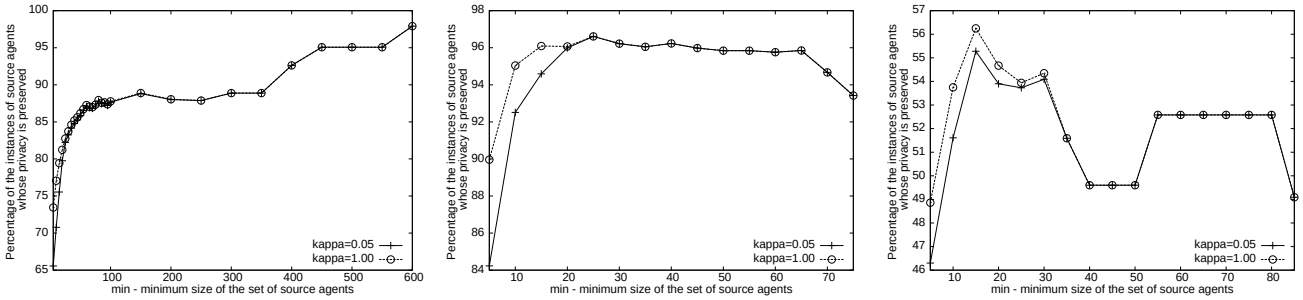
We define the disparity of a reputation value as $|r_t - r'_t|$. That is, the absolute difference between the reputation computed with all source agents and the reputation computed with only the source agents in $S_t - \mathbb{B}$. The disparity ranges from 0 to 1. The lower the disparity, the more accurate is the reputation. A disparity of 0 means that a reputation value computed with less than all source agents is exactly the same as it would be if computed with all source agents.

We compute the reputation of all target agents with at least min source agents twice. Firstly, with all source agents submitting their feedback. Secondly, with only those source agents submitting feedback whose privacy can be preserved. We then compute the disparity between the two values of reputation for each target agent. We count the number of



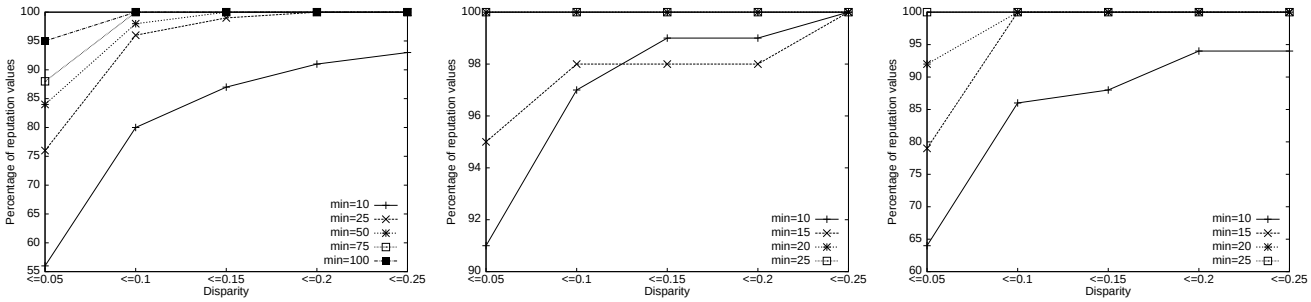
(a) Advogato (b) Squeak (c) Robots

Fig. 4: Effect of increasing κ on the percentage of instances of source agents whose privacy is preserved



(a) Advogato (b) Squeak (c) Robots

Fig. 5: Effect of increasing min on the percentage of instances of source agents whose privacy is preserved



(a) Advogato (b) Squeak (c) Robots

Fig. 6: Disparity

instances of reputation values where disparity is less than the values in $\{0.05, 0.1, 0.15, 0.20, 0.25\}$ respectively.

3) Analysis: In the results of the experiment on the Advogato trust graph (Figure 6a), we observe that for $min = 25$, the disparity of over 76% of reputation values is less than or equal to the low value of 0.05. Over 96% of reputation values have a disparity of less than or equal to just 0.1. For $min = 75$ and above, the disparity of 100% of the instances of reputation values is less than or equal to the fairly low value of 0.15. Thus, it is evident that even if source agents whose privacy can not be preserved abstain, the reputation of a high percentage of target agents can still be calculated with high accuracy as the mean of the feedback values. This inference is supported by the results of the experiments on the Squeak (Figure 6b) and Robots (Figure 6c) trust graphs. For $min = 25$, 100% of all reputation values have a disparity of less than or equal to 0.05 in both the Squeak and the Robots trust graphs.

VI. Related Work

Table III presents a comparison of our protocol with other reputation systems in the literature. The comparison illustrates that our protocol is the most efficient in terms of number of messages exchanged in decentralized environments. Moreover, our protocol does not require trusted third parties or specialized platforms, such as anonymous networks and trusted hardware.

VII. Conclusion and Future Work

In this article, we have presented a privacy preserving reputation protocol for the malicious adversarial model. The protocol counters attacks by malicious agents such as submitting

TABLE III: Protocol Malicious-k-shares – Comparison.

System	Architecture	Building Blocks	Complexity (Messages)
Malicious-k-shares	D	Zero-knowledge proofs, Homomorphic encryption	$O(n) + O(log\ N)$
Pavlov et al. [4] (WSS-2)	D	Verifiable secret sharing, Discrete log commitment	$O(n^3) + O(N)$
Gudes et al. [3] (Scheme 3)	D	Random permutation, Verifiable secret sharing, Discrete log commitment	$O(n^3) + O(N)$
Kinateder and Pearson [10]	D	Trusted hardware platform, Digital signatures	Not Provided
Androulaki et al. [11]	C	E-cash, Blind signatures, Anonymous networks	$O(1)$
Steinbrecher [12]	C	Pseudonym and identity management	$O(1)$
Schiffner et al. [13]	C	E-cash, Anonymous networks	$O(1)$
Kerschbaum [14]	C	Homomorphic encryption, Cryptographic pairings	$O(1)$
Bethencourt et al. [15]	C	Signatures of reputation, Proof systems for bilinear groups, Key private encryption	$O(1)$

invalid feedback values or making erroneous computations. The characteristics that differentiate the protocol from other protocols in the literature include: (1) full decentralization, (2) no need for trusted third parties and specialized platforms, (3) low complexity in terms of messages exchanged.

Our experiments on three real and large trust graphs demonstrate the validity of two hypotheses that the Malicious-k-shares protocol is based on: (1) A source agent can preserve its privacy by trusting on only k fellow source agents, where k is much smaller than $n-1$, the size of the set of all fellow source agents. (2) Accurate reputation values can be computed even if the source agents whose privacy can not be preserved abstain and thus do not provide their feedback values.

As future work, we aim to develop a privacy preserving reputation system that can counter the slandering and self-promotion attacks. In slandering, a user submits unjustifiable negative feedback to intentionally malign the reputation of a rival user. In self-promotion, a user achieves the inverse by submitting highly positive feedback to artificially increase his or a friend's reputation. Privacy prevents accountability of such users. Our goal is to develop a system that preserves the privacy of users yet exposes users who mount these attacks.

REFERENCES

[1] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," *Volume 11 of Advances in Applied Microeconomics*, pp. 127–157, 2002.

[2] O. Hasan, L. Brunie, and E. Bertino, "Preserving privacy of feedback providers in decentralized reputation systems," *Computers & Security*, 2011, http://dx.doi.org/10.1016/j.cose.2011.12.003.

[3] E. Gudes, N. Gal-Oz, and A. Grubshtein, "Methods for computing trust and reputation while preserving privacy," in *Proc. of DBSec'09*, 2009.

[4] E. Pavlov, J. S. Rosenschein, and Z. Topol, "Supporting privacy in decentralized additive reputation systems," in *Proceedings of the Second International Conference on Trust Management (iTrust 2004)*, Oxford, UK, 2004.

[5] D. Gambetta, *Trust: Making and Breaking Cooperative Relatioins*. Department of Sociology, University of Oxford, 2000, ch. Can We Trust Trust?, pp. 213 – 237.

[6] O. Goldreich, *The Foundations of Crypto. - Vol. 2*. Cambridge Univ. Press, 2004.

[7] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard, "Practical multi-candidate election system," in *Proc. of PODC'01*, 2001.

[8] S. D. Kamvar, M. T. Schlosser, and H. GarciaMolina, "The eigentrust algorithm for reputation management in p2p networks," in *Proc. of the 12th Intl. Conf. on World Wide Web (WWW 2003)*, Budapest, Hungary, May 2003.

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. of the 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001, pp. 149–160.

[10] M. Kinateder and S. Pearson, "A privacy-enhanced peer-to-peer reputation system," in *Proc. of the 4th Intl. Conf. on E-Commerce and Web Technologies*, 2003.

[11] E. Androulaki, S. G. Choi, S. M. Bellovin, and T. Malkin, "Reputation systems for anonymous networks," in *Proc. of PETS'08*, 2008.

[12] S. Steinbrecher, "Design options for privacy-respecting reputation systems," in *Security and Privacy in Dynamic Environments*, 2006.

[13] S. Schiffner, S. Clau, and S. Steinbrecher, "Privacy and liveliness for reputation systems," in *Proc. of EuroPKI'09*, 2009, pp. 209 – 224.

[14] F. Kerschbaum, "A verifiable, centralized, coercion-free reputation system," in *Proc. of the 8th ACM workshop on privacy in the e-society (WPES'09)*, 2009.

[15] J. Bethencourt, E. Shi, and D. Song, "Signatures of reputation: Towards trust without identity," in *Proc. of the Intl. Conf. on Financial Cryptography (FC '10)*, 2010.

A Notation Used in Malicious-k-shares

TABLE IV: Notation used in Malicious-k-shares.

Notation	Description
$\mathbb{A}$	The set of all agents in the environment
a	A source agent. $a \in S_t$.
c	A ciphertext
h_a	The quotient when the sum of the shares of an agent $\boldsymbol{a}$ is divided by M. $h_a = (\sum_{i=1}^{k+1} x_{a,i})\ div\ M$.
$\overrightarrow{T_a}$	A vector that contains the encrypted shares and the proofs sent by an agent $\boldsymbol{a}$ to the agent q
k	Constant k is the number of agents that each source agent $\boldsymbol{a}$ selects to send shares to. $k \ll n$.
$\mathbb{k}$	The security parameter. The length in bits of the RSA modulus $\mathfrak{n}$ in the cryptographic keys of the agents. For example, $\mathbb{k} = 2048$.
L	A positive integer constant. $l_{at} \in [0, L]$. For example, $L = 10$.
l_{at}	The feedback of a source agent $\boldsymbol{a}$ about a target agent t
M	A publicly known modulus. $M > L. \forall t \in A : \sum_{a \in S_t} l_{at} < M$. $M \ll 2^{\mathbb{k}}$. For example, $\mathbb{k} = 2048$, $M = 2^{80}$.
m	The number of dishonest source agents in S_t. $m < n$.
n	The cardinality of the set S_t. $n = \lvert S_t \rvert$.
$\mathfrak{n}$	The RSA modulus in the public key of an agent
$\mathfrak{n}_a$	The RSA modulus in the public key of an agent $\boldsymbol{a}$
PK_a	The public key of an agent $\boldsymbol{a}$
q	The querying agent
$r_t \equiv r_{t,\psi}$	The reputation of an agent t in the context ψ
$S_t \equiv S_{t,\psi}$	The set of source agents of agent t in the context ψ
S'_t	An intermediate set that is initialized to S_t. The set of agents who are expected to send their shares and sums to agent q.
s	A timestamp
t	The target agent
U_a	The set of fellow source agents that an agent $\boldsymbol{a}$ selects as trustworthy
u	A source agent. $u \in S_t$.
V_w	The set of encrypted shares that agent q receives from other agents and then relays to agent w
v	A source agent. $v \in S_t$.
w	A source agent. $w \in S_t$.
X	A large positive integer constant. $x_{a,i} \in [0, X]$. For example, $X = 2^{32} - 1$.
$x_{a,i}$	The i^{th} share of an agent $\boldsymbol{a}$
β_a	The encrypted sum of an agent $\boldsymbol{a}$'s shares. $\beta_a = E_a(\sum_{i=1}^{k+1} x_{a,i})$.
γ_a	The encrypted sum of the shares received by an agent $\boldsymbol{a}$ and agent $\boldsymbol{a}$'s $(k+1)$'th share $x_{a,k+1}$
θ	A cumulative sum for computing reputation
σ_a	The sum of the shares received by an agent $\boldsymbol{a}$ and agent $\boldsymbol{a}$'s $(k+1)$'th share $x_{a,k+1}$
ψ	An action. The context for trust.

B Non-Interactive Zero-Knowledge Proofs

Protocol: Non-Interactive-ZKP-Set-Membership
Participants: A prover and a verifier.
Input: Prover: $\mathfrak{n}$, g, $\mathfrak{m}_i$, p, $\mathfrak{r}$, $c = g^{\mathfrak{m}_i} \cdot \mathfrak{r}^{\mathfrak{n}}\ mod\ \mathfrak{n}^2$. Verifier: $\mathfrak{n}$, g, p, c.
Output: The verifier is convinced that c encrypts a message in S.
Setup: Public knowledge: A set $S = \{\mathfrak{m}_1, \ldots, \mathfrak{m}_p\}$, and the prover's public key $(\mathfrak{n}, g)$. $hash(x)$ is a cryptographic hash function secure against a computationally PPT bounded adversary.
Steps:

Prover

1) Prover picks at random ρ in $\mathbb{Z}^*_{\mathfrak{n}}$
2) Prover randomly picks $p-1$ values e_j in $\mathbb{Z}_{\mathfrak{n}}$, where $j \neq i$
3) Prover randomly picks $p-1$ values v_j in $\mathbb{Z}^*_{\mathfrak{n}}$, where $j \neq i$
4) Prover computes $u_j = v_j^{\mathfrak{n}} \cdot (g^{\mathfrak{m}_j}/c)^{e_j}\ mod\ \mathfrak{n}^2$, where $j \neq i$, and $u_i = \rho^{\mathfrak{n}}\ mod\ \mathfrak{n}^2$
5) Prover computes $e = hash(\langle u_1 \ldots u_p \rangle)$
6) Prover computes $e_i = e - \sum_{j \neq i} e_j\ mod\ \mathfrak{n}$
7) Prover computes $v_i = \rho \cdot \mathfrak{r}^{e_i} \cdot g^{(e - \sum_{j \neq i} e_j)/\mathfrak{n}}\ mod\ \mathfrak{n}$
8) Move 1: Prover sends u_j, v_j, e_j, where $j \in \{1 \ldots p\}$, to the verifier

Verifier

1) Verifier computes $e = hash(\langle u_1 \ldots u_p \rangle)$
2) Verifier checks that $e = \sum_j e_j\ mod\ \mathfrak{n}$
3) Verifier checks that $v_j^{\mathfrak{n}} = u_j \cdot (c/g^{\mathfrak{m}_j})^{e_j}\ mod\ \mathfrak{n}^2$ for each $j \in \{1 \ldots p\}$

Fig. 7: Protocol: Non-Interactive Zero-Knowledge Proof of Set Membership

Protocol: Non-Interactive-ZKP-Plaintext-Equality
Participants: A prover and a verifier.
Input: Prover: $\mathfrak{n}_1$, g_1, $\mathfrak{n}_2$, g_2, $\mathfrak{m}$, $\mathfrak{r}_1$, $\mathfrak{r}_2$, $c_1 = g_1^{\mathfrak{m}} \cdot \mathfrak{r}_1^{\mathfrak{n}_1}\ mod\ \mathfrak{n}_1^2$, $c_2 = g_2^{\mathfrak{m}} \cdot \mathfrak{r}_2^{\mathfrak{n}_2}\ mod\ \mathfrak{n}_2^2$. Verifier: $\mathfrak{n}_1$, g_1, $\mathfrak{n}_2$, g_2, c_1, c_2.
Output: The verifier is convinced that c_1 and c_2 encrypt the same message.
Setup: Public knowledge: The public keys $(\mathfrak{n}_1, g_1)$ and $(\mathfrak{n}_2, g_2)$. $hash(x)$ is a cryptographic hash function secure against a computationally PPT bounded adversary.
Steps:

Prover

1) Prover picks at random ρ in $[0, 2^{\mathbb{k}}[$
2) Prover randomly picks $s_1 \in \mathbb{Z}^*_{\mathfrak{n}_1}$ and $s_2 \in \mathbb{Z}^*_{\mathfrak{n}_2}$
3) Prover computes $u_j = g_j^{\rho} \cdot s_j^{\mathfrak{n}_j}\ mod\ \mathfrak{n}_j^2$, for each $j \in \{1, 2\}$
4) Prover computes $e = hash(\langle u_1, u_2 \rangle)$
5) Prover computes $z = \rho + \mathfrak{m} \cdot e$
6) Prover computes $v_j = s_j \cdot \mathfrak{r}_j^e\ mod\ \mathfrak{n}_j$, for each $j \in \{1, 2\}$
7) Move 1: Prover sends z, u_1, u_2, v_1, v_2 to the verifier

Verifier

1) Verifier computes $e = hash(\langle u_1, u_2 \rangle)$
2) Verifier checks that $z \in [0, 2^{\mathbb{k}}[$
3) Verifier checks that $g_j^z \cdot v_j^{\mathfrak{n}_j} = u_j \cdot c_j^e\ mod\ \mathfrak{n}_j^2$ for each $j \in \{1, 2\}$

Fig. 8: Protocol: Non-Interactive Zero-Knowledge Proof of Plaintext Equality