A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model

Omar Hasan, Lionel Brunie, Elisa Bertino, *Fellow, IEEE,* and Ning Shang

***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. It requires an exchange of $O(n+log\ N)$ messages, 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 where users consume as well as provide resources, however, their trustworthiness is often unknown. The reputation of a user is computed as a 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 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.

O. Hasan and L. Brunie are with the University of Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France e-mail: omar.hasan@insa-lyon.fr, lionel.brunie@insa-lyon.fr.

E. Bertino is with the Department of Computer Science, Purdue University, IN 47907, USA e-mail: bertino@cs.purdue.edu.

N. Shang is with Qualcomm Inc., 5775 Morehouse Dr, San Diego, CA 92121, USA e-mail: nshang@qti.qualcomm.com.

In a previous paper [2], we presented the non-cryptographic k-shares decentralized privacy preserving reputation protocol for the semi-honest (also known as honest-but-curious) 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

We consider a multi-agent environment composed of N agents. Let us consider an agent t in the environment and refer to it as the target agent. Agent t has interacted with $n < N$ other agents in the environment who have assigned it private feedback values. We refer to these feedback providers as the source agents of the target agent t. The Malicious-k-shares protocol allows a querying agent q to compute the reputation of a target agent t as the mean of the private feedback values held by its source agents. The protocol aims to preserve the privacy of the source agents by preventing disclosure of their private feedback values under the malicious adversarial model (described in Section II-B). The novel contributions of our work are summarized below.

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 fellow feedback providers. This idea is central in our protocol and allows us to build a protocol that requires an exchange of only $O(n + log\ N)$ messages and $O(n^2 + log\ N)$ bytes of information, 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-C).

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 legal 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 under the malicious adversarial model. It requires the exchange of only $O(n + log\ N)$ messages and $O(n^2 + log\ N)$ bytes of information. Compare this to the protocol by Pavlov et al. [4] that requires $O(n^3 + N)$ messages and at least $O(n^3 + N)$ bytes of information 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. $(s, t) \in \mathbb{T}$, where $s, t \in \mathbb{A}$, implies that an agent s has a trust relationship toward an agent t.

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 $perform(s, t, \psi)$ outputs $true$ if agent t performs the action ψ anticipated by agent s, or it outputs $false$ if t does not perform the anticipated action. Let the subjective probability $P(perform(s, t, \psi) = true)$ denote agent s's belief that agent t will perform the action ψ.

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

When the context of trust (action ψ) is clear, we adopt the simplified notation f_{st} for $P(perform(s, t, \psi) = true)$. We can also refer to f_{st} as agent s's feedback about agent t.

An agent s is said to be a source agent of an agent t in the context of an action ψ if s has a trust relationship toward t in the context ψ. The set of all source agents of an agent t in context ψ is given as $\mathbb{S}_{t,\psi}$. The simplified notation $\mathbb{S}_t$ is used instead of $\mathbb{S}_{t,\psi}$ when the context ψ is clear.

Definition 2: **Reputation.** Let $\mathbb{S}_t = \{s_1 \dots s_n\}$ be the set of source agents of an agent t in context ψ. The reputation of agent t in context ψ is given as:

$$rep_{\oplus}(f_{s_1t} \dots f_{s_nt}) = \frac{\sum_{i=1}^{n} f_{s_it}}{n} \tag{1}$$

The function $rep_{\oplus}$ implements the reputation of an agent t as the mean of the feedback values of its source agents. The reputation of an agent t is denoted by $r_{t,\psi}$, or r_t when the context ψ is clear.

A limitation of the reputation protocol that we present in this paper is that it can only compute reputation as the mean of the feedback values. An alternative function that can be implemented by the protocol is the sum of the feedback values. However, other possible functions for computing reputation such as weighted mean and probability distribution are not supported by the current protocol.

We note that an advantage of reputation computed as mean is that it 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 feedback values. We derive the mean from the sum in order to normalize the reputation values. However, it is an interesting avenue for future work to explore efficient decentralized privacy preserving solutions for computing reputation as weighted mean, probability distribution, etc.

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 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 illegal 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 s's private data that agent s reveals to an agent u. Then agent u is said to preserve the privacy of agent s w.r.t. x, if (1) u does not use x to infer more information, and (2) u does not reveal x to any third party.

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

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

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) $TTP_{\mathbb{S}_t}$ is a participant, where $\mathbb{S}_t \equiv \mathbb{S}_{t,\psi}$ 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 $\mathbb{S}_t - \mathbb{M}$ and $TTP_{\mathbb{S}_t}$ are honest, (5) as part of the protocol, $TTP_{\mathbb{S}_t}$ 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 $s \in \mathbb{S}_t - \mathbb{M}$ may be revealed only to the $TTP_{\mathbb{S}_t}$.

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

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

D. Problem Definition

Let $\mathbb{S}_{t,\psi} = \{s_1 \ldots s_n\}$ 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 $f_{st} \equiv P(\mathit{perform}(s, t, \psi) = \mathit{true})$ from each agent $s \in \mathbb{S}_t$, 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 an additional 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 action that alters 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 $E_s(.)$ denote the encryption function with the public key PK_s of agent s in an asymmetric cryptosystem $\mathcal{C}$. The cryptosystem $\mathcal{C}$ is said to be additive homomorphic if we can compute $E_s(x+y)$, given only $E_s(x)$, $E_s(y)$, and PK_s. As an example, let us consider two integers, 3 and 4. A cryptosystem $\mathcal{C}$ is additive homomorphic if given only $E_s(3)$, $E_s(4)$, and PK_s, we are able to obtain $E_s(3+4) = E_s(7)$.

We use the Paillier cryptosystem [7] as it offers the following properties (in addition to additive homomorphic encryption) that allow us to construct a secure reputation protocol:

- **Randomized encryption.** Randomized encryption implies that an attacker cannot distinguish between the encryptions of different plaintexts even if the plaintexts and the key are known. For example, consider an attacker who is given two integers, 3 and 4, their encryptions, $E_s(3)$ and $E_s(4)$, and the encrypting public key PK_s. The attacker is unable to draw correspondence between the ciphertexts $E_s(3)$, $E_s(4)$, and the integers 3, 4. Cryptosystems that do not support randomized encryption (for example, RSA [8] without padding), always generate the same ciphertext for a given pair of plaintext and encryption key. Such cryptosystems are not suitable when the plaintext space is small (for example, a plaintext space such as $\{1, 2, 3, 4, 5\}$).
- **Non-interactive zero-knowledge proofs.** The Paillier cryptosystem allows construction of efficient non-interactive zero-knowledge proofs of set membership (Section III-B) and plaintext equality (Section III-C).

The additive homomorphic encryption property of the Paillier cryptosystem can be informally stated as: $E_s(\mathbb{m}_1) \times E_s(\mathbb{m}_2) = E_s(\mathbb{m}_1 + \mathbb{m}_2)$, where $\mathbb{m}_1$ and $\mathbb{m}_2$ are two plaintext messages. This implies that the multiplication of two ciphertexts gives the encrypted sum of their plaintexts.

B. Zero-Knowledge Proof of Set Membership

Let $\mathbb{F} = \{\mathtt{m}_1, \ldots, \mathtt{m}_p\}$ be a public set of p messages, and $E(\mathtt{m}_i)$ be an encryption of $\mathtt{m}_i$ with a prover's public key, where $\mathtt{m}_i$ is secret. A zero-knowledge proof of set membership allows the prover to convince a verifier that $E(\mathtt{m}_i)$ encrypts a message in $\mathbb{F}$.

A standard interactive zero-knowledge proof comprises of three moves, that is, three messages exchanged between the prover and the verifier. In the first move, the prover sends a cryptographic commitment to the verifier. In the second move, the verifier sends a random challenge to the prover to test the commitment. The third move is the prover's response to the random challenge of the verifier. An interactive zero-knowledge proof can be converted to a non-interactive zero-knowledge proof using the Fiat-Shamir heuristic [9]. In a non-interactive proof, there is only one move made by the prover in which he sends the cryptographic commitment as well as a hash of the commitment to the verifier. The proof can be verified with this supplemental information without the need for additional moves.

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 $setMembershipZKP(E(\mathtt{m}_i), \mathbb{F})$, abbreviated as $smzkp(E(\mathtt{m}_i), \mathbb{F})$.

The zero-knowledge proof of set membership is specified for the Paillier cryptosystem as follows: Let $(\mathtt{n}, g)$ be a prover's public key, $\mathbb{F} = \{\mathtt{m}_1, \ldots, \mathtt{m}_p\}$ be a public set of p messages, and $c = g^{\mathtt{m}_i} \cdot \mathtt{r}^{\mathtt{n}} \ mod \ \mathtt{n}^2$ an encryption of $\mathtt{m}_i$, where i is secret, and $\mathtt{r}$ is a random integer. A zero-knowledge proof of set membership allows the prover to convince a verifier that c encrypts a message in $\mathbb{F}$.

An interactive zero-knowledge proof of set membership for the Paillier cryptosystem is described by Baudron et al. [10]. Our non-interactive adaptation of the proof, using the Fiat-Shamir heurisitic, is given in Figure 6 (Appendix B).

C. Zero-Knowledge Proof of Plaintext Equality

Let $E_u(\mathtt{m})$ and $E_v(\mathtt{m})$ be the encryptions of a message $\mathtt{m}$ with the public key of agents u and v respectively. A zero-knowledge proof of plaintext equality allows a prover to convince a verifier that $E_u(\mathtt{m})$ and $E_v(\mathtt{m})$ 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 $plaintextEqualityZKP(E_u(\mathtt{m}), E_v(\mathtt{m}))$, abbreviated as $pezkp(E_u(\mathtt{m}), E_v(\mathtt{m}))$.

The zero-knowledge proof of plaintext equality is specified for the Paillier cryptosystem as follows: Let $(\mathtt{n}_1, g_1)$ and $(\mathtt{n}_2, g_2)$ be the public keys of agents 1 and 2 respectively. Given two encryptions $c_1 = g_1^{\mathtt{m}} \cdot \mathtt{r}_1^{\mathtt{n}_1} \ mod \ \mathtt{n}_1^2$ and $c_2 = g_2^{\mathtt{m}} \cdot \mathtt{r}_2^{\mathtt{n}_2} \ mod \ \mathtt{n}_2^2$, a zero-knowledge proof of plaintext equality allows a prover to convince a verifier that c_1 and c_2 encrypt the same message.

An interactive zero-knowledge proof of plaintext equality for the Paillier cryptosystem is described by Baudron et al. [10]. In Figure 7 (Appendix B), we present our non-interactive version of the proof, which has been obtained using the Fiat-Shamir heurisitic.

D. Source Managers

We define a *source manager* of an agent t as a fellow agent who maintains the set $\mathbb{S}_t$. The idea of source managers is inspired by *score managers* in EigenTrust [11]. 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 $\mathbb{S}_t$ that they each maintain. A Distributed Hash Table (DHT), such as Chord [12], is used to locate the source managers.

It is important to note that the source managers are considered to be potentially dishonest. To learn a set $\hat{\mathbb{S}_t} \supset \mathbb{S}_t$, 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 $\hat{\mathbb{S}_t} \supset \mathbb{S}_t$ as long as at least one of the source managers is honest. Let us consider that there is an even probability that any given source manager is either honest or dishonest. Then the probability that at least one of all η_t source managers of an agent t will be honest is $1 - \frac{1}{2^{\eta_t}}$. This probability is 75% at $\eta_t = 2$, 97% at $\eta_t = 5$, and 99% at $\eta_t = 7$.

IV. The Malicious-k-shares Protocol

A. Protocol Overview

In the Malicious-k-shares protocol, each source agent s relies on at most k agents to preserve his privacy. Agent s 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 an exchange of only $O(n + log \ N)$ messages and $O(n^2 + log \ N)$ bytes of information.

In the Malicious-k-shares protocol, each source agent s must prove that it has generated correct shares, that is, the sum of all shares is a value that lies in the legal interval for feedback. An agent s 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 s 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 legal 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 s'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 s 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 s indeed sent correct shares.

In the Malicious-k-shares protocol, each source agent s must prove that it 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 s. However, q cannot decrypt the sum because it is encrypted with the recipient agent's public key. Agent s computes the sum and sends it to q encrypted with q's public key. Agent s 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 s hold the same value. Verification of the proof convinces q that agent s computed the sum correctly.

B. Protocol Outline

The important steps of the protocol are outlined below. The steps 1, 7, 8, 13, and 14 are performed by the querying agent q. Whereas, the steps 2 – 6, and 9 – 12, are performed by each source agent $s \in \mathbb{S}_t$.

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 $\mathbb{S}_t \equiv \mathbb{S}_{t,\psi}$, which is the set of source agents of agent t in the context ψ. Agent q verifies $\mathbb{S}_t$ from the source managers of t. Agent q then sends $\mathbb{S}_t$ to each source agent $s \in \mathbb{S}_t$.
2) **Select Trustworthy Agents.** Each agent $s \in \mathbb{S}_t$ selects k other agents in $\mathbb{S}_t$. Let us refer to these agents selected by s as the set $\mathbb{U}_s = \{u_{s,1} \dots u_{s,k}\}$. Agent s selects these agents such that: $P(\neg perform(s, u_{s,1}, \rho)) \times \dots \times P(\neg perform(s, u_{s,k}, \rho))$ is low. That is, the probability that all of the selected agents will collude to breach agent s's privacy is low.
3) **Prepare Shares.** Agent s then prepares $k+1$ shares of its secret feedback value f_{st}. The shares, given as: $x_{s,1} \dots x_{s,k+1}$, 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: $x_{s,k+1} = (f_{st} - \sum_{i=1}^{k} x_{s,i}) \ mod \ M$, where M is a publicly known modulus.
The preparation of the shares in this manner implies that: $(\sum_{i=1}^{k+1} x_{s,i}) \ mod \ M = f_{st}$. That is, the sum of the shares $mod \ M$ is equal to the feedback value. The sum of the shares, $\sum_{i=1}^{k+1} x_{s,i}$, lies in the interval $[(h_s \times M), (h_s \times M) + F]$, where $h_s = (\sum_{i=1}^{k+1} x_{s,i}) \ div \ M$, and $f_{st} \in [0, F]$.
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 s then encrypts each of the $k+1$ shares with its own public key to obtain: $E_s(x_{s,1}) \dots E_s(x_{s,k+1})$. It also encrypts each share $x_{s,i}$ with the public key of agent $u_{s,i}$, for $i \in \{1 \dots k\}$, to obtain: $E_{u_{s,1}}(x_{s,1}) \dots E_{u_{s,k}}(x_{s,k})$.
5) **Generate Zero-Knowledge Proofs.** Agent s computes: $\beta_s = (E_s(x_{s,1}) \times \dots \times E_s(x_{s,k+1})) \ mod \ \mathtt{n}_s^2$, where $\mathtt{n}_s$ is the RSA modulus in the public key of agent s. The result of this product is the encrypted sum of agent s's shares, that is $\beta_s = E_s(\sum_{i=1}^{k+1} x_{s,i})$ (due to the additive homomorphic property).
Agent s then generates one non-interactive set membership zero-knowledge proof: $smzkp(\beta_s, [(h_s \times M), (h_s \times M) + F])$. The proof proves to a verifier that the ciphertext β_s encrypts a value that lies in the interval $[(h_s \times M), (h_s \times M) + F]$. In other words, the proof shows that the ciphertext contains a valid feedback value (considering $mod \ M$).
Agent s also generates k non-interactive plaintext equality zero-knowledge proofs. Each proof $pezkp(E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i}))$, where $i \in \{1 \dots k\}$, proves to a verifier that the two ciphertexts, one encrypted with the public key of s and the other encrypted with the public key of $u_{s,i}$, contain the same plaintext. A verifier who verifies these zero-knowledge proofs will be convinced that agent s 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 s's trustworthy agents correspond to those correct shares.
6) **Send Encrypted Shares and Proofs.** Agent s sends all encrypted shares, that is, $E_s(x_{s,1}) \dots E_s(x_{s,k+1})$ and $E_{u_{s,1}}(x_{s,1}) \dots E_{u_{s,k}}(x_{s,k})$, as well as all zero-knowledge proofs, that is, $smzkp(\beta_s, [(h_s \times M), (h_s \times M) + F])$ and $pezkp(E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i}))$, $i \in \{1 \dots k\}$, to agent q.
7) **Verify the Proofs.** Agent q independently computes β_s and verifies the proofs received from each agent s. Their verification confirms that agent s 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 $u \in \mathbb{S}_t$, the encrypted shares received for it from agents who considered it trustworthy. That is, each encrypted share $E_{u_{s,i}}(x_{s,i})$, prepared by an agent s for agent $u_{s,i}$, is relayed to agent $u_{s,i}$.
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 $s \in \mathbb{S}_t$ receives the encrypted shares of the agents who considered it trustworthy. Agent s computes γ_s as the product of those encrypted shares along with the ciphertext of its own $(k+1)$'th share $x_{s,k+1}$. Due to the additive homomorphic property, γ_s is an encryption of the sum of the plaintexts of those shares. Agent s decrypts γ_s to obtain the plaintext sum σ_s.
Adding the $(k+1)$'th share provides security in the case when s 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 s then encrypts σ_s with agent q's public key to obtain $E_q(\sigma_s)$.
11) **Generate Zero-Knowledge Proof.** Agent s then generates a non-interactive plaintext equality zero-knowledge proof: $pezkp(\gamma_s, E_q(\sigma_s))$. The proof proves to a verifier that the two ciphertexts, one encrypted with the public key of s and the other encrypted with the public key of q, contain the same plaintext.
    Agent q, who can independently compute γ_s, can be convinced by this proof that $E_q(\sigma_s)$ contains the correct sum of the shares.
12) **Send Encrypted Sum and Proof.** Agent s sends the encrypted sum $E_q(\sigma_s)$ and the zero-knowledge proof $pezkp(\gamma_s, E_q(\sigma_s))$ to agent q.
13) **Verify the Proof.** Agent q independently computes γ_s and verifies the zero-knowledge proof received from each agent s. 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 $E_q(\sigma_s)$ to obtain σ_s for each agent $s \in \mathbb{S}_t$. Agent q then computes $r_{t,\psi} = ((\sum_{s \in \mathbb{S}_t} \sigma_s)\ mod\ M)/n$.

C. Protocol Specification

The protocol is specified in Figures 1 and 2. Tables IV and V summarize the notation used in the protocol specification.

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

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

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

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

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

The protocol uses the following functions: **timestamp**() – Returns current time. **random**(ξ,ζ) – Returns a random number uniformly distributed over the interval $[\xi, \zeta]$, where $\xi < \zeta$. **set_of_trustworthy**(s, $\mathbb{S}$) – Returns a set of agents $\mathbb{U}_s = \{u_{s,1} \ldots u_{s,k}\}$, where $\mathbb{U}_s \subseteq \mathbb{S}$. The set $\mathbb{U}_s$ is selected such that: $P(\neg perform(s, u_{s,1}, \rho)) \times \ldots \times P(\neg perform(s, u_{s,k}, \rho))$ is low.

Protocol: Malicious-k-shares
Participants: Agents: q, t, and the agents in the set $\mathbb{S}_t \equiv \mathbb{S}_{t,\psi} = \{s_1 \ldots s_n\}$. Agents q, t, and a subset of $\mathbb{S}_t$ 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 agent $s \in \mathbb{S}_t$ has a private input $f_{st} \equiv P(perform(s, t, \psi) = true)$.
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: Agent t maintains $\mathbb{S}_t \equiv \mathbb{S}_{t,\psi}$, 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:

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

agent t receives the tuple $(\text{REQUEST\_FOR\_SOURCES}, \psi)$ from agent q
```
1  send tuple $(\text{SOURCES}, \psi, \mathbb{S}_t)$ to $q$
```

an agent $s \in \mathbb{S}_t$ receives the tuple $(\text{PREP}, q, t, \tau, \mathbb{S}_t)$ from agent q
```
    ▷ select trustworthy agents
1   $\mathbb{U}_s \leftarrow \text{set\_of\_trustworthy}(s, \mathbb{S}_t - \{s\})$
    ▷ prepare shares
2   for $i \leftarrow 1$ to $k$
3       do $x_{s,i} \leftarrow \text{random}(0, X)$
4   $x_{s,k+1} \leftarrow (f_{st} - \sum_{i=1}^{k} x_{s,i})\ mod\ M$
5   $h_s \leftarrow (\sum_{i=1}^{k+1} x_{s,i})\ div\ M$
    ▷ retrieve public keys
6   retrieve the public key of each $u \in \mathbb{U}_s$ and the public key of $q$
    from a certificate authority
    ▷ encrypt shares
7   encrypt $x_{s,1} \ldots x_{s,k+1}$ with the public key of $s$
    to obtain $E_s(x_{s,1}) \ldots E_s(x_{s,k+1})$
8   encrypt $x_{s,1} \ldots x_{s,k}$ with the public key of $u_{s,1} \ldots u_{s,k}$
    respectively to obtain $E_{u_{s,1}}(x_{s,1}) \ldots E_{u_{s,k}}(x_{s,k})$
    ▷ generate zero-knowledge proofs
9   $\beta_s \leftarrow (E_s(x_{s,1}) \times \ldots \times E_s(x_{s,k+1}))$ mod $\mathbb{n}_s^2$
10  generate $\text{setMembershipZKP}(\beta_s, [(h_s \times M), (h_s \times M) + F])$
11  for $i \leftarrow 1$ to $k$
12      do generate $\text{plaintextEqualityZKP}(E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i}))$
    ▷ send the encrypted shares and the proofs to $q$
13  $\overrightarrow{\mathcal{I}_s} \leftarrow \langle \mathbb{U}_s, E_s(x_{s,1}), \ldots, E_s(x_{s,k+1}), E_{u_{s,1}}(x_{s,1}), \ldots, E_{u_{s,k}}(x_{s,k}),$
      $h_s, \text{setMembershipZKP}(\beta_s, [(h_s \times M), (h_s \times M) + F]),$
      $\text{plaintextEqualityZKP}(E_s(x_{s,1}), E_{u_{s,1}}(x_{s,1})), \ldots,$
      $\text{plaintextEqualityZKP}(E_s(x_{s,k}), E_{u_{s,k}}(x_{s,k}))\rangle$
14  send tuple $(\text{SHARES}, q, t, \tau, \overrightarrow{\mathcal{I}_s})$ to $q$
```

Fig. 1: Protocol: Malicious-k-shares

D. Security Analysis – Correctness

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.

In this section we analyze correctness in the context of the messages sent by the source agents under the malicious adversarial model. Correctness under the semi-honest model is analyzed in detail in the extended technical report [13].

Each agent $s \in \mathbb{S}_t$ communicates exclusively with agent q. If an agent s takes any of the actions 1 to 3 (Section II-B), it

```
Protocol: Malicious-$k$-shares (contd.)

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

an agent $s \in \mathbb{S}_t$ receives the tuple $(\text{VERIFIED\_SHARES}, q, t, \tau, \mathbb{V}_s)$ from agent $q$
      ▷ compute sum of the shares
 1    $\gamma_s \leftarrow ((\prod_{c \in \mathbb{V}_s} c) \times E_s(x_{s,k+1}))$ mod $\mathfrak{n}_s^2$
 2    $\sigma_s \leftarrow D_s(\gamma_s)$
      ▷ encrypt the sum
 3    encrypt $\sigma_s$ with the public key of $q$ to obtain $E_q(\sigma_s)$
      ▷ generate zero-knowledge proof
 4    generate $\text{plaintextEqualityZKP}(\gamma_s, E_q(\sigma_s))$
      ▷ send the encrypted sum and the proof to $q$
 5    send tuple $(\text{AGGREGATE}, q, t, \tau, E_q(\sigma_s), pezkp(\gamma_s, E_q(\sigma_s)))$ to $q$

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

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

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 $s \in \mathbb{S}_t$ 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 $s \in \mathbb{S}_t$ 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 $s \in \mathbb{S}_t$ provides to agent q is: $(\text{SHARES}, q, t, \tau, \overrightarrow{\mathcal{I}_s})$, where $\overrightarrow{\mathcal{I}_s} = \langle\ \mathbb{U}_s,\ E_s(x_{s,1}),\ \ldots,\ E_s(x_{s,k+1}),\ E_{u_{s,1}}(x_{s,1}),\ \ldots,\ E_{u_{s,k}}(x_{s,k}),\ \text{setMembershipZKP}(\beta_s, F),\ \text{plaintextEqualityZKP}(E_s(x_{s,1}),\ E_{u_{s,1}}(x_{s,1})),\ \ldots,\ \text{plaintextEqualityZKP}(E_s(x_{s,k}),\ E_{u_{s,k}}(x_{s,k}))\ \rangle$.

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

The first condition holds true for an agent s if the verification of $\text{setMembershipZKP}(\beta_s, [(h_s \times M), (h_s \times M) + F])$ by agent q is successful. Agent q can verify the proof since it can independently compute β_s (due to the additive homomorphic property of the cryptosystem), F and M are publicly known, and h_s is provided by agent s. An incorrect value of h_s 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_s 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 s if the verification of each $\text{plaintextEqualityZKP}(E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i}))$ 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 s and $u_{s,1} \ldots u_{s,k}$ 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 s succeeds, it implies that agent s has provided correct information pertaining to the shares that it prepared. Otherwise, agent s can be considered as malicious.

The second tuple of information that an agent $s \in \mathbb{S}_t$ provides agent q is: $(\text{AGGREGATE}, q, t, \tau, E_q(\sigma_s), pezkp(\gamma_s, E_q(\sigma_s)))$.

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_s. The sum has been computed correctly if the following condition holds true: γ_s and $E_q(\sigma_s)$ encrypt the same plaintext.

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

E. Security Analysis – Privacy

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

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 s in Malicious-k-shares than it can learn in the ideal protocol, with high probability: $1 - (P(\neg \mathit{perform}(s, u_{s,1}, \rho)) \times \ldots \times P(\neg \mathit{perform}(s, u_{s,k}, \rho)))$.

In this section we analyze privacy only in the context of attack 6 in which an agent provides illegal information (Section II-B) under the malicious adversarial model. Privacy under the semi-honest model and under the other attacks of the malicious model is analyzed in detail in the extended technical report [13].

If a source agent $u \in \mathbb{S}_t$ provides illegal information, it has no effect on the condition that all first k shares of agent s must be known to breach agent s's privacy. Agent u provides no information to agent s or agent q that would result in agent s divulging any extra information.

Agent t may provide an illegal $\mathbb{S}_t$, however, that has no effect on the protocol since q also retrieves and verifies $\mathbb{S}_t$ 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 $\mathbb{S}_t$ itself in order to attack an agent $s \in \mathbb{S}_t$. The set may be created such that it contains all dishonest agents except agent s who is under attack. However, we assume that $P(\neg \mathit{perform}(s, u_{s,1}, \rho)) \times \ldots \times P(\neg \mathit{perform}(s, u_{s,k}, \rho))$ is low. That is, there exist trustworthy agents in the protocol such that agent s receives a high enough privacy guarantee.

VERIFIED_SHARES: Agent q may substitute the shares sent by other agents to an agent s 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 s's $(k+1)$'th share. This has no effect on the privacy of agent s 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 may be extended such that an agent s 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.

1) An Attack on the Ideal Protocol: We describe an attack in which the adversary attempts to determine the private feedback of a source agent over the course of two reputation queries. Consider the scenario when a new agent s is added to the set of source agents $\mathbb{S}_t$. Let $\mathbb{S}_t^{'} = \mathbb{S}_t \cup \{s\}$. Let the reputation of the target agent t be r_t and $r_t^{'}$ for the set of source agents $\mathbb{S}_t$ and $\mathbb{S}_t^{'}$ respectively. A querying agent q that queries the reputation of the target agent t with both sets of source agents can compute the private feedback of agent s as $f_{st} = (r_t^{'} \times (n+1)) - (r_t \times n)$, where $n = |\mathbb{S}_t|$. A similar attack can also determine the private feedback of an existing source agent that drops out from the set of source agents.

The ideal protocol (Definition 6) is vulnerable to this attack. Consequently, the Malicious-k-shares protocol is also vulnerable as it emulates the ideal protocol. We note that this is a general issue for all protocols (including Pavlov et al. [4] and Gudes et al. [3]) that can produce the sum of the feedback values of the following sets of source agents: $\mathbb{S}_t$, and $\mathbb{S}_t \cup \{s\}$ or $\mathbb{S}_t - \{u\}$, where $s \notin \mathbb{S}_t$ and $u \in \mathbb{S}_t$.

We have previously described this attack in our paper [2] on the non-cryptographic k-shares decentralized privacy preserving reputation protocol for the semi-honest model. Additionally, we have discussed three possible defenses to counter this attack in our previous work. These solutions can also be applied to the protocol described in the current work. We briefly describe the three solutions below and refer the reader to our previous work for further details.

1) Each source agent retrieves a random number from a trusted random number generator and adds it to its feedback. The sum of the random numbers perturbs the reputation score with a different value each time the reputation is calculated.
2) Two or more source agents who trust each other in the context of preserving privacy form a trusted subset. The agents in a trusted subset submit feedback in tandem, that is, they either all submit their feedback or none of them does. Thus, at best, an attacker can only learn the cumulative sum of multiple feedback values instead of an individual feedback value.
3) A source agent probabilistically decides whether to participate in the protocol or to abstain from the protocol. The effect achieved is that the attacker can no longer deterministically create the set of agents who will submit their feedback about a given target agent.

F. Complexity Analysis

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

Tuple	**Occur-rences**	**IDs**	**Cipher-texts**	**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	$O(n)$	$O(n^2)$	$O(kn)$	$O(n)$	$O(kn)$

Table I presents an analysis of the complexity of the Malicious-k-shares protocol. The column "Occurrences" analyzes the number of messages exchanged. Whereas, the columns "IDs", "Ciphertexts", "SMZKPs", and "PEZKPs" analyze the bandwidth usage of the protocol.

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 $O(n) + O(log\ N) = O(n + log\ N)$, 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 usage, the protocol requires transmission of the following amount of information: $O(n^2)$

agent IDs, $O(kn)$ ciphertexts, $O(n)$ non-interactive zero-knowledge proofs of set membership, $O(kn)$ non-interactive zero-knowledge proofs of plaintext equality, and an additional $O(log\ N)$ messages of constant size for the DHT lookup.

The size of the IDs, the ciphertexts, and the PEZKPs (Section III-C, Figure 7) is constant. Moreover, the size of the SMZKPs (Section III-B, Figure 6) is also constant, given that $p = |\mathbb{F}|$ is constant. Thus, the complexity of the protocol in terms of bytes of information transferred can be stated as $O(n^2)+O(kn)+O(n)+O(kn)+O(log\ N) = O(n^2+log\ N)$. We observe in Section V-B that $k \ll n$.

Moreover, k can be considered as a constant in the protocol. k can be set as a system-wide constant. Alternatively, in the extended version of the protocol where agents can abstain, the querying agent can be given the choice to set k. The trade-off between a lower and a higher value of k is possible lower participation from the source agents and higher bandwidth usage respectively.

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.

The first hypothesis places emphasis on the notion that a source agent can find k sufficiently trustworthy agents that enable it to preserve its privacy, even if k is much smaller than $n-1$. The fact that a source agent is able to preserve its privacy with k trustworthy agents, where $k \ll n-1$, has already been validated in Section IV-E.

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.

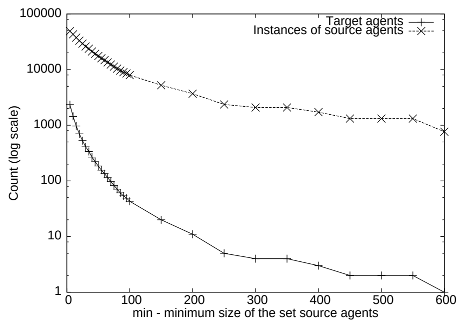
B. Experiment 1

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

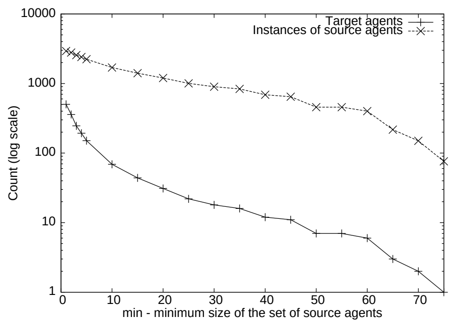
2) Setup: The maximum number of fellow source agents that an agent can trust on is $n-1$. A fraction of the size of this set can be stated as $\kappa \times (n-1)$, where $\kappa \in [0, 1]$. For our experiments, we equate $k = \lceil \kappa \times (n-1) \rceil$. This allows us to use κ (kappa) to vary 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 s is preserved if $P(\neg perform(s, u_{s,1}, \rho)) \times \ldots \times P(\neg perform(s, u_{s,k}, \rho))$ is low, which is less than or equal to 0.1 in our case. $u_{s,1} \ldots u_{s,k}$ are the agents that agent s trusts.

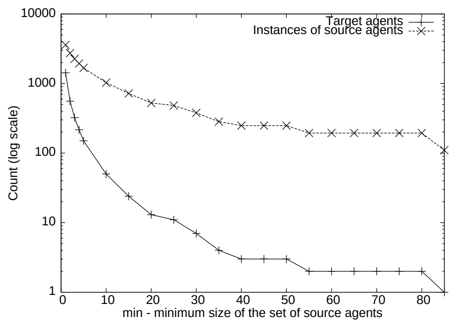
3) Analysis: In the results of the experiment on the Advogato trust graph (Figure 4a), we observe that for $min = 25$, the privacy of 71% of the instances of source agents is preserved when $\kappa = 0.01$. That is, 71% of the source agents find sufficiently trustworthy agents among only 1% of their 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



(a) Advogato

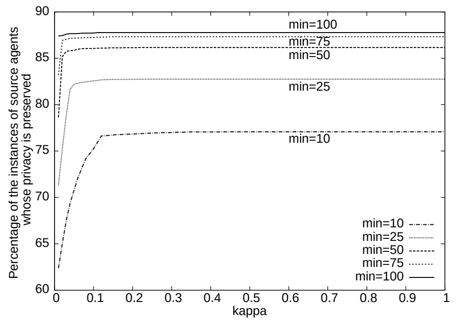


(b) Squeak

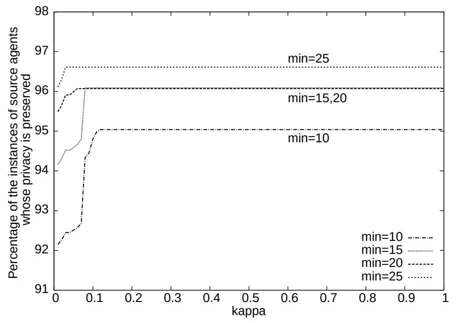


(c) Robots

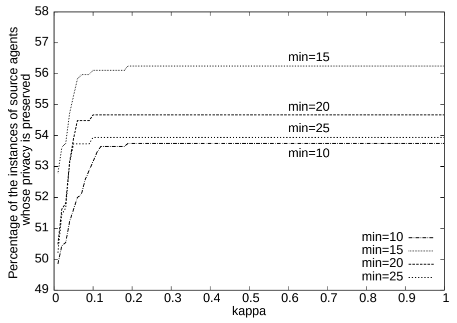
Fig. 3: Distribution of the potential target agents and the instances of source agents



(a) Advogato

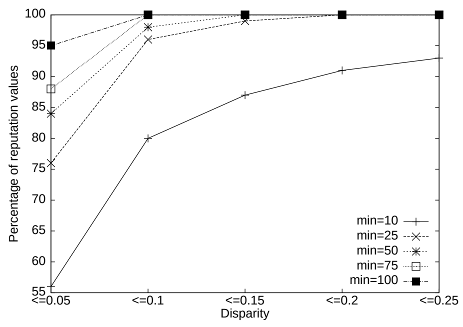


(b) Squeak

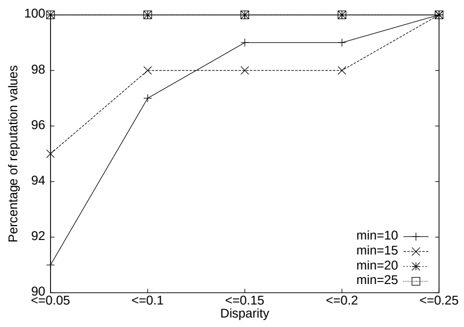


(c) Robots

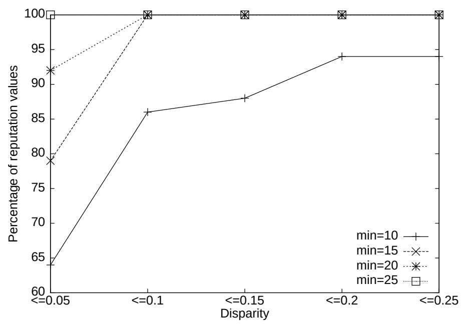
Fig. 4: Effect of increasing κ on the percentage of instances of source agents whose privacy is preserved



(a) Advogato



(b) Squeak



(c) Robots

Fig. 5: Disparity

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. It 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.

The privacy of a source agent s is preserved if $P(\neg perform(s, u_{s,1}, \rho)) \times \ldots \times P(\neg perform(s, u_{s,k}, \rho))$ is low, that is, if the probability that all the k fellow source agents that s considers trustworthy will turn out to be dishonest is low. Thus, whether the privacy of a source agent s will be preserved depends on whether s can find k sufficiently trustworthy agents among fellow source agents. However, in certain cases, an agent s is unable to find k sufficiently trustworthy agents even if $kappa = 1$ (that is, $k = n - 1$) and even if there are more than 100 source agents in the protocol. This is because the agent s either does not have trust relationships toward its fellow source agents or the relationships are not strong enough. The functions in Figure 4 do not converge to 100% due to the existence of such agents whose privacy cannot be preserved no matter how large is k or how high is min. However, we observe in Section V-C that such agents are in the minority.

A high percentage of the agents are able to find sufficiently trustworthy agents among their fellow source agents in order to preserve their privacy in real trust graphs. We note that protocols in the literature such as Pavlov et al. and Gudes et al. that rely on all $n-1$ fellow source agents of an agent to preserve its privacy will also fail to protect the privacy of the agents who cannot preserve their privacy in our protocol. This is because all $n-1$ fellow source agents are not trustworthy enough. On the contrary, our protocol can be extended to allow agents to abstain from submitting feedback thus protecting their privacy.

C. *Experiment 2*

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 $\mathbb{B} \subset \mathbb{S}_t$ and $\mathbb{S}_t$ 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 $\mathbb{S}_t$ 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 $\mathbb{S}_t - \mathbb{B}$.

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 $\mathbb{S}_t - \mathbb{B}$. 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 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 5a), 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 5b) and Robots (Figure 5c) 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

Pavlov et al. [4] propose a decentralized privacy preserving reputation protocol for the malicious adversarial model. The protocol comprises of two steps: (1) The first step is the execution of a witness (feedback provider) selection scheme, which guarantees the inclusion of a certain number of honest witnesses as participants. (2) The second step is the decentralized computation of the reputation as the sum of the feedback values. According to our analysis, the protocol requires an exchange of $O(n^3+N)$ messages and at least $O(n^3+N)$ bytes of information, where n is the number of witnesses selected and N is the number of all potential witnesses. The witness selection scheme requires $O(N)$ messages. The decentralized computation of the reputation uses Pederson Verifiable Secret Sharing. Each witness sends messages to every other $n-1$ participating witnesses. The result is an additional exchange of $O(n^3)$ messages. In contrast, our protocol requires an exchange of only $O(n + log\ N)$ messages and $O(n^2 + log\ N)$ bytes of information. In addition to innovative cryptographic constructions, an important reason for the lower complexity is that each agent in our protocol selects k fellow agents based on their trustworthiness, where $k \ll n$. This eliminates the need for a costly witness selection scheme as well as the need for each agent to exchange messages with $n-1$ fellow agents. Another key difference is that our protocol allows entities to quantify and to minimize the risk to their privacy before feedback is submitted. 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 even with k as very small compared to $n-1$.

Gudes et al. [3] present a protocol for the malicious adversarial model that augments their Knots reputation system [14] with privacy preserving features. The Knots reputation system is a personalized reputation system, which implies that feedback is collected only from the entities whom the querying entity trusts. The protocol by Gudes et al. eliminates the need for a witness selection scheme but still requires $O(n^3)$ messages for the decentralized computation of reputation.

The decentralized reputation system proposed by Kinateder and Pearson [15] requires a Trusted Platform Module (TPM) chip at each agent. The TPM enables an agent to demonstrate that it is a valid agent and a legitimate member of the reputation system without disclosing its true identity. This permits the agent to provide feedback anonymously. A later system by Kinateder et al. [16] avoids the hardware modules, however, it requires an anonymous routing infrastructure at the network level. Our approach does not mandate hardware modules or specialized platforms such as anonymous networks.

Androulaki et al. [17] propose a reputation scheme for pseudonymous peer-to-peer systems in anonymous networks. Users in such systems interact only through disposable pseudonyms such that their true identity is not revealed. The reputation protocol has two key objectives: (1) unlinkability between pseudonyms and true identities, and (2) unforgeability, that is, users are unable to forge good reputation. These objectives are achieved primarily by using e-cash [18], [19], a cryptographic digital currency that offers anonymity and unforgeability. Reputation is awarded in the form of e-coins called *repcoins*. The querying of reputation requires a constant number of messages. The weakness of the system is that it requires the presence of a centralized entity called the *bank*. Additionally, the system also requires that all communication take place over an anonymous network, such as a network using Onion routing [20]. In contrast, our protocol is truly decentralized and does not require a specialized platform such as an anonymous network to operate. Schiffner et al. [21] also present privacy preserving reputation protocols based on e-cash and anonymous networks, however, their solution is also centralized.

The reputation system described by Kerschbaum [22] is another centralized privacy preserving reputation system. Kerschbaum introduces the requirement of authorizability, which implies that only the users who have had a transaction with a ratee are allowed to rate him. This property prevents users who have not transacted with a ratee from assigning him feedback and thus possibly reduces the impact of attacks such as bad mouthing and self promotion.

Bethencourt et al. [23] propose a centralized reputation system based on proof systems for bilinear groups, key private encryption, and their new cryptographic primitive called signatures of reputation. A user in the reputation system can verify that the reputation of a target user is composed of feedback provided by distinct feedback providers. The objective is to prevent colluding users from augmenting each other's reputation by taking advantage of privacy. The reputation system achieves this property while maintaining the anonymity of all users.

A number of surveys on reputation systems appear in the literature that discuss the issues of reputation systems including privacy. Notable surveys on reputation systems include those by Jøsang at el. [24], Hoffman et al. [25], and Pinyol and Sabater-Mir [26].

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 the 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.

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

System	Architecture	Building Blocks	Complexity (Messages)
Malicious-k-shares	D	Zero-knowledge proofs, Homomorphic encryption	$O(n+log\ N)$
Pavlov et al. [4] (WSS-2)	D	Verifiable secret sharing, Discrete log commitment	$O(n^3+N)$
Gudes et al. [3] (Scheme 3)	D	Random permutation, Verifiable secret sharing, Discrete log commitment	$O(n^3+N)$
Kinateder and Pearson [15]	D	Trusted hardware platform, Digital signatures	Not Provided
Androulaki et al. [17]	C	E-cash, Blind signatures, Anonymous networks	$O(1)$
Steinbrecher [27]	C	Pseudonym and identity management	$O(1)$
Schiffner et al. [21]	C	E-cash, Anonymous networks	$O(1)$
Kerschbaum [22]	C	Homomorphic encryption, Cryptographic pairings	$O(1)$
Bethencourt et al. [23]	C	Signatures of reputation, Proof systems for bilinear groups, Key private encryption	$O(1)$

D – Decentralized, C – Centralized

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 illegal 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 communication complexity.

Our experiments on three real and large trust graphs demonstrate the validity of the 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 decentralized 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.

Appendix 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
c	A ciphertext
F	A positive integer constant. $f_{st} \in [0, F]$. For example, $F = 10$.
f_{st}	The feedback of a source agent s about a target agent t
h_s	The quotient when the sum of the shares of an agent s is divided by M. $h_s = (\sum_{i=1}^{k+1} x_{s,i})\ div\ M$.
$\overrightarrow{\mathcal{I}_s}$	A vector that contains the encrypted shares and the proofs sent by an agent s to the agent q
k	Constant k is the number of agents that each source agent s selects to send shares to. $k \ll n$.
$\Bbbk$	The security parameter. The length in bits of the RSA modulus in the cryptographic keys of the agents. For example, $\Bbbk = 2048$.
M	A publicly known modulus. $M > F$. $\forall t \in A : \sum_{s\in\mathbb{S}_t} f_{st} < M$. $M \ll 2^{\Bbbk}$. For example, $\Bbbk = 2048$, $M = 2^{80}$.
m	The number of dishonest source agents in $\mathbb{S}_t$. $m < n$.
n	The cardinality of the set $\mathbb{S}_t$. $n = \lvert\mathbb{S}_t\rvert$.
$\mathfrak{n}_s$	The RSA modulus in the public key of an agent s
PK_s	The public key of an agent s
q	The querying agent
$r_t \equiv r_{t,\psi}$	The reputation of an agent t in the context ψ

TABLE V: Notation used in Malicious-k-shares. (contd.)

Notation	Description
$\mathbb{S}_t \equiv \mathbb{S}_{t,\psi}$	The set of source agents of agent t in the context ψ
$\mathbb{S}'_t$	An intermediate set that is initialized to $\mathbb{S}_t$. The set of agents who are expected to send their shares and sums to agent q.
s	A source agent. $s \in \mathbb{S}_t$.
t	The target agent
$\mathbb{U}_s$	The set of fellow source agents that an agent s selects as trustworthy
u	A source agent. $u \in \mathbb{S}_t$.
$\mathbb{V}_u$	The set of encrypted shares that agent q receives from other agents and then relays to agent u
X	A large positive integer constant. $x_{s,i} \in [0, X]$. For example, $X = 2^{32} - 1$.
$x_{s,i}$	The i^{th} share of an agent s
β_s	The encrypted sum of an agent s's shares. $\beta_s = E_s(\sum_{i=1}^{k+1} x_{s,i})$.
γ_s	The encrypted sum of the shares received by an agent s and agent s's $(k+1)$'th share $x_{s,k+1}$
θ	A cumulative sum for computing reputation
σ_s	The sum of the shares received by an agent s and agent s's $(k+1)$'th share $x_{s,k+1}$
τ	A timestamp
ψ	An action. The context for trust.

Appendix B
Non-Interactive Zero-Knowledge Proofs

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

Prover

1) Prover picks at random z in $\mathbb{Z}^*_{\mathbb{n}}$
2) Prover randomly picks $p-1$ values e_j in $\mathbb{Z}_{\mathbb{n}}$, where $j \neq i$
3) Prover randomly picks $p-1$ values $\mathbb{v}_j$ in $\mathbb{Z}^*_{\mathbb{n}}$, where $j \neq i$
4) Prover computes $\mathbb{u}_j = \mathbb{v}_j^{\mathbb{n}} \cdot (g^{\mathbb{m}_j}/c)^{e_j} \ mod\ \mathbb{n}^2$, where $j \neq i$, and $\mathbb{u}_i = z^{\mathbb{n}} \ mod\ \mathbb{n}^2$
5) Prover computes $e = hash(\langle \mathbb{u}_1 \ldots \mathbb{u}_p \rangle)$
6) Prover computes $e_i = e - \sum_{j \neq i} e_j \ mod\ \mathbb{n}$
7) Prover computes $\mathbb{v}_i = z \cdot \mathbb{r}^{e_i} \cdot g^{(e - \sum_{j \neq i} e_j)/\mathbb{n}} \ mod\ \mathbb{n}$
8) Move 1: Prover sends $\mathbb{u}_j$, $\mathbb{v}_j$, e_j, where $j \in \{1 \ldots p\}$, to the verifier

Verifier

1) Verifier computes $e = hash(\langle \mathbb{u}_1 \ldots \mathbb{u}_p \rangle)$
2) Verifier checks that $e = \sum_j e_j \ mod\ \mathbb{n}$
3) Verifier checks that $\mathbb{v}_j^{\mathbb{n}} = \mathbb{u}_j \cdot (c/g^{\mathbb{m}_j})^{e_j} \ mod\ \mathbb{n}^2$ for each $j \in \{1 \ldots p\}$

Fig. 6: Protocol: Non-Interactive Zero-Knowledge Proof of Set Membership

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments.

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*, vol. 31, no. 7, pp. 816 – 826, October 2012, 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.

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

Prover

1) Prover picks at random z in $[0, 2^{\mathbb{k}}[$
2) Prover randomly picks $\mathbb{s}_1 \in \mathbb{Z}^*_{\mathbb{n}_1}$ and $\mathbb{s}_2 \in \mathbb{Z}^*_{\mathbb{n}_2}$
3) Prover computes $\mathbb{u}_j = g_j^z \cdot \mathbb{s}_j^{\mathbb{n}_j} \ mod\ \mathbb{n}_j^2$, for each $j \in \{1, 2\}$
4) Prover computes $e = hash(\langle \mathbb{u}_1, \mathbb{u}_2 \rangle)$
5) Prover computes $z = z + \mathbb{m} \cdot e$
6) Prover computes $\mathbb{v}_j = \mathbb{s}_j \cdot \mathbb{r}_j^e \ mod\ \mathbb{n}_j$, for each $j \in \{1, 2\}$
7) Move 1: Prover sends z, $\mathbb{u}_1$, $\mathbb{u}_2$, $\mathbb{v}_1$, $\mathbb{v}_2$ to the verifier

Verifier

1) Verifier computes $e = hash(\langle \mathbb{u}_1, \mathbb{u}_2 \rangle)$
2) Verifier checks that $z \in [0, 2^{\mathbb{k}}[$
3) Verifier checks that $g_j^z \cdot \mathbb{v}_j^{\mathbb{n}_j} = \mathbb{u}_j \cdot c_j^e \ mod\ \mathbb{n}_j^2$ for each $j \in \{1, 2\}$

Fig. 7: Protocol: Non-Interactive Zero-Knowledge Proof of Plaintext Equality

[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] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, 1999.

[8] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, p. 120126, 1978.

[9] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO'86*, 1986.

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

[11] 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.

[12] 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.

[13] O. Hasan, L. Brunie, E. Bertino, and N. Shang, "A decentralized privacy preserving reputation protocol for the malicious adversarial model," LIRIS, France, http://liris.cnrs.fr/Documents/Liris-5609.pdf, Tech. Rep. RR-LIRIS-2012-008, 2012.

[14] N. Gal-Oz, E. Gudes, and D. Hendler, "A robust and knot-aware trust-based reputation model," in *Proceedings of the Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM 2008)*, 2008.

[15] 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.

[16] M. Kinateder, R. Terdic, and K. Rothermel, "Strong pseudonymous communication for peer-to-peer reputation systems," in *Proceedings of the 2005 ACM symposium on Applied computing*, 2005.

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

[18] D. Chaum, "Blind signatures for untraceable payments," in *Proc. Advances in Cryptology (CRYPTO '82)*, 1982.

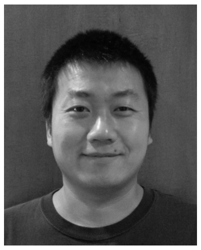
[19] ——, "Blind signature systems," in *Advances in Cryptology (CRYPTO'83)*, 1983.

[20] R. Dingledine, N. Mathewson, and P. F. Syverson, “Tor: The second-generation onion router,” in *Proceedings of the USENIX Security Symposium*, 2004.
[21] S. Schiffner, S. Clau, and S. Steinbrecher, “Privacy and liveliness for reputation systems,” in *Proc. of EuroPKI’09*, 2009, pp. 209 – 224.
[22] 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.
[23] 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.
[24] A. Josang, R. Ismail, and C. Boyd, “A survey of trust and reputation systems for online service provision,” *Decision Support Systems*, vol. 43, no. 2, pp. 618 – 644, March 2007.
[25] K. Hoffman, D. Zage, and C. Nita-Rotaru, “A survey of attack and defense techniques for reputation systems,” *ACM Computing Surveys*, vol. 41, no. 4, December 2009.
[26] I. Pinyol and J. Sabater-Mir, “Computational trust and reputation models for open multi-agent systems: a review,” *Artificial Intelligence Review*, 2011.
[27] S. Steinbrecher, “Design options for privacy-respecting reputation systems,” in *Security and Privacy in Dynamic Environments*, 2006.

Omar Hasan is Assistant Professor at the *Institut National des Sciences Appliquées (INSA)*, University of Lyon, France. His research interests include distributed systems, information privacy, and trust and reputation management. He received his PhD in computer science from INSA, University of Lyon. Prior to his current position, he was a researcher on the SOCEDA project of the French *Agence Nationale de la Recherche (ANR)*. Additionally, he was a visiting researcher at Purdue University, USA for approximately one year. He also holds four years of software engineering and R&D experience in the IT industry.

Lionel Brunie is Professor at the *Institut National des Sciences Appliquées (INSA)*, University of Lyon, France since 1998. He was previously a faculty member at the *Ecole Normale Supérieure (ENS)*, Lyon, France. He received his PhD in computer science in 1992 from Joseph Fourier University, Grenoble, France. In 1999, he created the INSA e-learning department which he led until 2002. Then from 2002 to 2006, he headed the Lyon doctoral school in computer science (300+ registered PhD students). In 2003, he co-founded the LIRIS lab in which he acted as deputy director in 2006-2007. In 2007, he co-founded the international doctoral college in “Multimedia Distributed and Pervasive Secure systems (MDPS)”. He leads the LIRIS DRIM research team which comprises of 10 permanent researchers and 20+ PhD students. His main topics of interest include security and privacy, data management in large scale and pervasive systems, collaborative information systems, and e-health applications. He has led numerous national and international research projects; he is the (co-)author of over 180 research papers; he has been member of over 70 scientific conference and workshop committees.

Elisa Bertino is Professor of Computer Science at Purdue University, and serves as research director of the Center for Education and Research in Information Assurance and Security (CERIAS) and Interim Director of Cyber Center (Discovery Park). Previously, she was a faculty member and department head at the Department of Computer Science and Communication of the University of Milan. Her main research interests include security, privacy, digital identity management systems, database systems, distributed systems, and multimedia systems. She is currently serving as chair of the ACM SIGSAC and as a member of the editorial board of the following international journals: IEEE Security & Privacy, IEEE Transactions on Service Computing, ACM Transactions on Web. She also served as editor in chief of the VLDB Journal and editorial board member of ACM TISSEC and IEEE TDSC. She co-authored the book “Identity Management - Concepts, Technologies, and Systems”. She is a fellow of the IEEE and a fellow of the ACM. She received the 2002 IEEE Computer Society Technical Achievement Award for outstanding contributions to database systems and database security and advanced data management systems and the 2005 IEEE Computer Society Tsutomu Kanai Award for pioneering and innovative research contributions to secure distributed systems.



Ning Shang is a product security engineer at Qualcomm Inc. He received his PhD degree in mathematics from Purdue University, West Lafayette, Indiana. Before coming to Qualcomm, he was a postdoc researcher in the Department of Computer Science at Purdue University and a software development engineer at Microsoft. His research interests include algorithmic number theory, curve-based cryptography, and design and implementation of security systems.