Efficient Privacy Preserving Protocols for Decentralized Computation of Reputation

Omar Hasan
INSA Lyon, France
omar.hasan@insa-lyon.fr

Elisa Bertino
Purdue University, IN, USA
bertino@cs.purdue.edu

Lionel Brunie
INSA Lyon, France
lionel.brunie@insa-lyon.fr

ABSTRACT

We present three different privacy preserving protocols for computing reputation. They vary in strength in terms of preserving privacy, however, a common thread in all three protocols is that they are fully decentralized and efficient. Our protocols that are resilient against semi-honest adversaries and non-disruptive malicious adversaries have linear and loglinear communication complexity respectively. We evaluate our proposed protocols on data from the real web of trust of Advogato.org.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Security, Algorithms, Performance, Human Factors

Keywords

trust, reputation, privacy, decentralized

1. INTRODUCTION

Reputation systems represent a key technology for securing distributed applications from misuse by dishonest entities. A reputation system computes reputation scores of the entities in the system, which helps single out those that are exhibiting less than desirable behavior. Examples of reputation systems may be found in several application domains: E-commerce websites such as eBay (ebay.com) and Amazon (amazon.com) use their reputation systems to discourage fraudulent activities. The EigenTrust [10] reputation system enables peer-to-peer file sharing systems to filter out peers who provide inauthentic content. The web-based community of Advogato.org uses a reputation system [13] for spam filtering.

The reputation score of a target entity is a function of the feedbacks provided by other entities. Thus an accurate reputation score is possible only if the feedbacks are accurate. However, it has been observed that the users of a reputation system may avoid providing honest feedback. The reasons for such behavior include fear of retaliation from the target entity or mutual understanding that a feedback value would be reciprocated. eBay originally allowed buyers and sellers to assign each other positive, neutral or negative feedback. A study [15] of eBay's reputation system revealed that there is a high correlation between buyer and seller feedback and over 99% of the feedback is positive. As discussed in [14], this could either imply that mutually satisfying transactions are in fact the norm or that the users are not providing honest feedback due to the above cited reasons. The actual cause was obvious as the latter when eBay recently revised its reputation system [6] citing that "*... the [earlier] feedback system made some buyers reluctant to hold sellers accountable. For example, buyers fear retaliatory feedback from sellers if they leave a negative.*" Now sellers are not permitted to assign negative or neutral feedback to the buyers.

A more general solution to the problem of lack of honest feedback is computing reputation scores in a privacy preserving manner. A privacy preserving protocol for computing reputation scores operates such that the individual feedback of any entity is not revealed to other entities in the system. The implication of private feedback is that there are no consequences for the feedback provider and thus he is uninhibited to provide honest feedback. In this paper we focus on privacy preserving protocols for decentralized additive reputation systems. These reputation systems are characterized by the absence of a central authority and the computation of reputation scores in an additive manner.

The privacy preserving protocols for computing reputation that have been discussed in the literature, either rely on specialized hardware, are centralized, or have high communication complexity associated with them. For instance, a protocol described in [14] has a communication complexity of $O(n^2)$, where n is the number of feedback providers for a target entity. Further literature is discussed in section 8.

In this paper, we present three different protocols with varying strengths in terms of preserving privacy. The common thread in all three protocols is that they are fully decentralized and efficient. Our protocol that is resilient against non-disruptive malicious adversaries has loglinear communication complexity. A recent protocol with similar strengths by Pavlov et al. [14] is quadratic. We evaluate our proposed protocols on data from the web of trust of Advogato.org. To the best of our knowledge, this is the first work to evaluate a decentralized privacy preserving protocol for computing reputation on data from a real and large web of trust.

2. PRELIMINARIES

In this section we introduce several definitions that lay the foundation for the work presented in subsequent sections.

2.1 Decentralized Additive Reputation Systems

The reputation system that we present in this paper can be classified as a *Decentralized Additive Reputation System.* We quote the following definition from [14]:

Decentralized Additive Reputation System. *Reputation System R is said to be a Decentralized Additive Reputation System if it satisfies two requirements:*

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

In a Decentralized Additive Reputation System, the reputation of an agent is in essence computed as follows: the agents that have local feedback about that agent, collaborate and contribute their local feedback values to arrive at the sum of those values. The sum is considered the reputation value of the agent.

Summation of local feedbacks about an entity to compute it's global reputation is an approach adopted by several reputation systems including the successful eBay reputation system (ebay.com). The advantage of this approach is that it is intuitive and thus the meaning of a reputation value is easily understood by the users.

Please note that although the eBay reputation system is additive, it is not decentralized. A centralized reputation system such as eBay is appropriate for the web based e-commerce application, which is inherently centralized. Our solution is thus not intended to be a privacy preserving alternative to the eBay reputation system. Our solution targets decentralized applications, examples of which include peer-to-peer file sharing, MANETs, etc.

2.2 Preserving Privacy

Here we define what we mean by preserving the privacy of an agent in a Decentralized Additive Reputation System.

Preserving Privacy. *Let a be an agent that contributes its local feedback about an agent t, as part of a protocol to compute the reputation of agent t. Then the privacy of agent a is said to be preserved if during or after the execution of the protocol, no other agent in the system is able to learn agent a's local feedback.*

In this paper we will consider two types of attacks that can lead to the leakage of an agent's local feedback values:

Type 1 Attack. An agent a, as part of a protocol to compute the reputation of an agent t, exchanges intermediate information with other agents. Those agents either individually or as a group of colluders try to derive the private local feedback of agent a about agent t from those intermediate values.

Type 2 Attack. An adversary observes the reputation of agent t immediately before and after agent a updates its local feedback about agent t. Since the reputation is computed in an additive manner, the adversary can learn about agent a's private local feedback as: $\delta l = r'_t - r_t$, where δl is the difference between agent a's previous and current local feedback about agent t, and r_t and r'_t are the reputation values of agent t before and after the update respectively. If agent a assigned feedback to agent t for the first time, then δl is equal to its complete feedback about agent t.

To the best of our knowledge this is the first work which addresses the Type 2 attack in a decentralized additive reputation system.

2.3 Adversarial Models

We identify three types of adversarial agents in the context of preserving privacy in a reputation system:

Semi-Honest Agents. Semi-honest agents always correctly follow the protocol for computing the reputation of an agent. However, semi-honest agents are curious about the private local feedbacks of other agents. They will use intermediate information received during the protocol and any other information that they can receive through legitimate means to derive the local feedbacks of other agents. Semi-honest agents may also collude to satisfy this goal.

Non-Disruptive Malicious Agents. Malicious agents are not bound to conform to the protocol. They may deviate from the protocol as and when they deem necessary. They may participate in extra-protocol activities, devise sophisticated strategies, and collude to learn the private local feedbacks of other agents. Non-Disruptive Malicious agents have a single objective: to learn private feedbacks of other agents. They do not intend to disrupt the normal function of the protocol other than to achieve this objective.

Disruptive Malicious Agents. Disruptive malicious agents have exactly the same capabilities as non-disruptive malicious agents in terms of learning the private feedbacks of other agents. This implies that if non-disruptive malicious agents are unable to learn the private feedback of a particular agent then disruptive malicious agents cannot do so either. What differentiates disruptive malicious agents from those who are non-disruptive is that the former have objectives beyond learning the private feedbacks of other agents. Their objectives may range from gaining illegitimate advantage over other agents by tampering with the protocol to completely denying the services of the reputation system to other agents.

Protocols that preserve privacy under the first two models are provided in this paper. A protocol that is resilient against disruptive malicious adversaries is identified as future work. We do however discuss some possible techniques for a protocol for the third adversarial model in section 9.

2.4 Data Perturbation

Data perturbation is a technique for hiding a data item by adding noise to it. The noise added is sufficiently large in order to make the derivation or estimation of the data item from the resulting sum highly improbable.

We quote the Data Perturbation Assumption from [5] as follows (variable r in the original definition is given here as variable y):

Data Perturbation Assumption. *If an input is $x \in X$, we assume that $x + y$ effectively preserves the privacy of x if y is a secret random number uniformly distributed in a domain F, where $|F| \gg |X|$.*

As an example, let's consider that a value $x = 0.5 \in [-1, 1]$ is to be hidden. If we add a secret random number $y =$

$-3.2 \in [-10, 10]$ to x, then the sum $x + y = -2.7$. In this case it is impossible to learn the value of x from the sum.

The data perturbation technique is well established in several domains including privacy-preserving data mining [1], [17], and secure two party [5], [7] and multi-party [7] computation.

With data perturbation there is some probability that x will not be hidden properly. In the above example, if $x = 1$ and the secret random number turns out to be $y = 10$, then the sum would be $x + y = 11$, which would give away the value of x. Please see section 7.2 for a heuristic that suppresses such occurrences.

3. THE BASIC FRAMEWORK OF THE REPUTATION SYSTEM

In this section we present the basic framework of our decentralized additive reputation system.

The reputation system comprises of N agents. The set of agents in the system is given as $A = \{a_i : 1 \leq i \leq N\}$.

After two agents interact, they each may assign the other a feedback value. A feedback value represents one agent's local view of the trustworthiness of another agent. The feedback value assigned by an agent a to an agent t is given as $l_{at} \in [-1, 1]$. The choice of feedback values is real numbers between -1 and 1, which allows infinite resolution for expressing trust. -1 implies "minimum trust", 0 implies "neutral trust", and 1 implies "maximum trust".

$r_t \in \mathbb{R}$ represents the global reputation value of an agent t. Higher values indicate higher reputation.

There is no central authority in the system. Feedback values are stored locally by the agents who assigned them. For example, a feedback value l_{at} is stored by the agent a. The global reputation values are transient.

When an agent q wishes to determine the reputation r_t of an agent t, we refer to agent q as the *querying agent* and to agent t as the *target agent*. The agents that have assigned feedback to agent t are called the *source agents* and they are given as the set $S_t = \{s : s \in A \wedge l_{st} \text{ exists}\}$. $n_t = |S_t|$ is the number of source agents for agent t.

To determine the reputation of agent t, agent q initiates a reputation computation protocol, which at minimum involves the source agents and terminates with q learning the current reputation of agent t. The protocols that we discuss in this paper compute the reputation in an additive manner.

In each of the next three sections we describe a different reputation computation protocol. The three protocols are given in the order of their strength in preserving the privacy of agents with the last one being the strongest.

4. PROTOCOL 1: SECURE SUM

The secure sum protocol [4], [17] is a well known protocol that computes the sum of local values of multiple sites without revealing the local value of any site. The protocol is clearly applicable to the problem at hand of computing the reputation of an agent in an additive manner while preserving the privacy of local feedback values. However, the secure sum protocol in its simple form has several shortcomings which we will highlight after describing the protocol. We include the secure sum protocol here to establish a sense of the challenges faced in developing a privacy preserving reputation computation protocol for a decentralized additive reputation system. A variant of the secure sum protocol adapted for computing the reputation of a target agent t in our reputation system is described below.

Each agent a maintains S_a, the set of its source agents. That is, each agent maintains a set of the agents that have interacted with it and have reported assigning feedback to it.

The steps of the protocol are as follows:

1. The querying agent q retrieves S_t from the target agent t.
2. Agent q creates an ordered list of the agents in S_t, which is given as the vector $\overrightarrow{S_t} = (s_1, s_2, \ldots, s_n)$, where $s_1, s_2, \ldots, s_n$ refer to the agents in S_t and $n = |S_t|$.
3. Agent q sends the vector $\overrightarrow{S_t}$ and y to agent s_1. y is a random number in $[-Y, Y]$.
4. Agent s_1 upon receipt of the vector $\overrightarrow{S_t}$ and y, computes $v_{s_1} = y + l_{s_1 t}$. Agent s_1 sends $\overrightarrow{S_t}$ and v_{s_1} to s_2.
5. Each subsequent agent s_i that receives the vector $\overrightarrow{S_t}$ and $v_{s_{i-1}}$, computes $v_{s_i} = v_{s_{i-1}} + l_{s_i t}$, and sends the vector $\overrightarrow{S_t}$ and v_{s_i} to s_{i+1}.
6. The last agent s_n upon receipt of the vector $\overrightarrow{S_t}$ and $v_{s_{n-1}}$, computes $v_{s_n} = v_{s_{n-1}} + l_{s_n t}$, and sends v_{s_n} to q.
7. q computes $r_t = v_{s_n} - y$.

The privacy of s_1's local feedback value is preserved as it sends $y + l_{s_1 t}$ to s_2. Given the data perturbation assumption, since y is added to $l_{s_1 t}$, s_2 is unable to determine $l_{s_1 t}$ from the sum. Similarly when any agent s_i sends $v_{s_{i-1}} + l_{s_i t} = y + \sum_{j=1}^{i-1} l_{s_j t} + l_{s_i t}$ to s_{i+1}, data perturbation prevents s_{i+1} from learning $l_{s_i t}$ or any $l_{s_j t}$.

Agent q receives $y + \sum_{i=1}^{n} l_{s_i t}$ and computes the result by subtracting y. Assuming $n \geq 2$ and that the local feedback values are uniformly distributed over their range, it is highly improbable for q to be able to distinguish the individual local feedback values of the source agents.

We observe that this protocol preserves privacy against the type 1 attack only when agents are not colluding. If agents s_{i-1} and s_{i+1} collude, they can compute agent s_i's local feedback value as follows: $l_{s_i t} = v_{s_i} - v_{s_{i-1}}$. Agent s_i has no control over who s_{i-1} and s_{i+1} are in this protocol. Agent q, who establishes the order of $\overrightarrow{S_t}$, can set up any agent for the privacy of that agent to be breached in this manner.

Since the reputation value is the deterministic sum of all local feedback values, it is also straightforward to mount the type 2 attack against any agent.

Considering these issues, it is clear that the reputation computation protocol based on secure sum does not preserve privacy against either type of attack under any of the three adversarial models.

The protocol requires $n + 1$ messages to be exchanged thus the complexity of the protocol in terms of messages exchanged is $O(n)$, where n is the number of source agents of a target agent t.

5. PROTOCOL 2: RESILIENCY AGAINST SEMI-HONEST ADVERSARIES

We now present a reputation computation protocol that preserves privacy against both types of attacks under the semi-honest adversarial model. A brief summary of the protocol is given below followed by a formal description in the next subsection. As in protocol 1, each agent a maintains S_a, the set of its source agents.

- The protocol is initiated by a querying agent q to determine the reputation of a target agent t. Agent q retrieves S_t from t and initiates the *forwards* round by sending $S = S_t$ and $r = 0$ to an agent randomly selected from S_t.
- The receiving agent adds its local feedback value and a random number $y \in [-Y, Y]$ to r. After removing itself from S, the agent sends the updated S and r to the agent in S that it trusts the most to respect its privacy. The protocol continues with the *forwards* round in this manner until the last agent in S updates r and sends it to a pre-trusted or seed agent.
- The seed agent generates a random number $x \in [-Y, Y]$ and then selects $n = |S_t|$ numbers such that the sum of those numbers is equal to x. It sends each of those numbers to distinct agents in S_t. The seed then initiates the *backwards* round by sending $S = S_t$ and r to a randomly selected agent in S_t.
- The receiving agent removes the random number y from r that it added to it in the *forwards* round. The agent adds to r, the number that it received from the seed. The agent then removes itself from S and sends the updated S and r to the agent in S that it trusts the most. However, it selects an agent that is different from the agent that it selected in the *forwards* round. The *backwards* round continues in this manner until the last agent in S updates r and then sends it to q.
- The value of r that q receives is the sum of the local feedback values of all agents in S_t and x. This value of r is considered the reputation value of agent t.

Proof of privacy is provided in section 5.3, however, the general ideas behind the defenses of this protocol against the two types of attacks are summarized as follows:

Type 1 Attack. Each agent exchanges information with five agents during the protocol. All five of those agents must collude to learn the local feedback value of the agent. This can be highly improbable since two of those agents are trustworthy agents selected by the agent himself and another one is a highly trusted seed agent.

Type 2 Attack. The true sum of the local feedback values of all agents in S_t is never learned by any agent. The result of the protocol is a value that is probabilistically close to the true sum. This is achieved by the random number added by the seed agent. Thus simply observing a reputation value before and after an update, does not reveal the local feedback of the updater agent. This type of attack can be successful if the seed agent colludes with agent q, however, this has low probability given that the seed agent is highly trusted.

The key innovation in this protocol is that an agent himself selects trustworthy agents whom he wants to share intermediate information with. The advantages of this approach are twofold: Firstly, since the agent himself selects the agents whom to trust, he can choose them such as to maximize the probability that his privacy will be preserved. Choosing the agents whom to trust also allows an agent to be aware of the exact value of that probability (please see section 5.3 for detail). Secondly, since each agent exchanges messages with a constant number of other agents, the communication complexity of the protocol is linear. This is in contrast to the protocol presented by Pavlov et al. [14] for a similar adversarial model, which requires each agent to exchange messages with all other agents in the protocol resulting in quadratic communication complexity. Another innovation in our protocol is the presence of seed agents, which help in preventing the type 2 attack. Additionally, we also evaluate our protocol on data from a real and large web of trust (section 7).

5.1 Formal Description

As in the secure sum protocol, each agent a maintains S_a, the set of its source agents.

Some of the agents in the system are identified as seed agents. The set of seed agents is given as $D = \{d_i : d_i \in A \wedge 1 \leq i \leq N\}$. The set D is universally known by all agents in the system. The concept of seed agents is used in many successful reputation systems including Advogato [13] and EigenTrust [10]. Seed agents are typically those agents who joined the system at its inception and are thus known to have been thoroughly vetted and highly trustworthy. The trustworthiness of seed agents is universally considered in the system to be at least $0.99 \in [-1, 1]$. This is a reasonable assumption, given that in the practical and very successful Advogato reputation system, the seed agents are considered 100% trustworthy.

Each query is uniquely identified by a sequence of three variables q, t, p, where q is the querying agent, t is the target agent, and p is the timestamp when the query was initiated.

Each agent maintains three vectors $\overrightarrow{\mathcal{A}}$, $\overrightarrow{\mathcal{X}}$, and $\overrightarrow{\mathcal{Y}}$ for storing the variables $a_{(q,t,p)} \in A$, $x_{(q,t,p)} \in [-Y, Y]$, and $y_{(q,t,p)} \in [-Y, Y]$ respectively.

The agents communicate using messages. Each message comprises of a tuple whose first element identifies the type of the message.

All agents in the system are driven by a common protocol and thus exhibit homogeneous behavior. The protocol for an agent a in the system is given in appendix A as a collection of events and associated actions.

5.2 Correctness

Theorem *If all agents properly follow Protocol 2, then at the completion of a query,* $r_t = \sum_{a \in S_t} l_{at} + x$.

PROOF. Please see appendix B. □

5.2.1 Accuracy of Reputation Values

The addition of x implies that the result of the query deviates from the true sum by a random value on the interval $[-Y, Y]$. We discuss two approaches of quantifying the difference between the actual reputation value and the perturbed reputation value: 1) absolute difference, and 2) relative difference.

The absolute difference is given as: *absolute difference* = |*actual reputation* − *perturbed reputation*|. Since $x \in [-Y, Y]$, $\Rightarrow$ *absolute difference* $\leq Y$.

The relative difference is expressed as: *relative difference* = |*actual reputation* − *perturbed reputation*| / *actual reputation*, where *actual reputation* $\neq 0$. From the previous equations: *relative difference* $\leq$ (Y / *actual reputation*). The bound on relative difference is inversely proportional to the actual reputation. As the actual reputation increases, the bound on relative difference will decrease.

We argue that absolute difference is a more objective interpretation of the difference between actual and perturbed reputations. The bound for absolute difference remains constant (Y) and is independent of the reputation values. Whereas, depending on the actual reputation value, the bound for relative difference can vary between 0 and ∞ (both exclusive). In terms of absolute difference, the accuracy of a reputation value is given as $\pm Y$.

Please refer to section 7.4 for further discussion on the accuracy of reputation values computed by the protocol.

5.3 Preserving Privacy

If a trust relationship exists between two agents a and k, then $l_{ak} \in [-1, 1]$ is interpreted as the amount of trust a has in k to not attack it to learn its private data. Let's say that $Z_a = \{z : z \in A \wedge z \text{ will attack } a\}$ is the set of all agents in A who will attack a if given the opportunity. Then we can also state that l_{ak} is the amount of trust a has in k to not belong to Z_a. The relationship between l_{ak} and the probability $P(k \in Z_a)$ is assumed to be as follows:

$$P(k \in Z_a) = \begin{cases} 1 - l_{ak} & \text{if } l_{ak} \geq 0 \\ 1 & \text{if } l_{ak} < 0 \\ 1 & \text{if } l_{ak} \text{ does not exist} \end{cases}$$

Since by definition agents are curious, if agent a does not have a positive trust relationship with agent k, it is assumed that k will attack a to learn its private data.

For a seed agent d, $P(d \in Z_a) = 1 - 0.99 = 0.01$.

Theorem *If the agents who participate in Protocol 2 are semi-honest, then at the completion of a query, the probability that a type 1 attack will reveal the local feedback value of an agent $a \in S_t$, who is not the last agent in the* forwards *or the* backwards *round, is: $P(a_{(f,out)} \in Z_a) \times P(a_{(b,out)} \in Z_a) \times P(d \in Z_a)$.*

PROOF. Please see appendix B. □

Theorem *If the agents who participate in Protocol 2 are semi-honest, then at the completion of a query, the probability that a type 1 attack will reveal the local feedback value of an agent $a \in S_t$ is at most $P(d \in Z_a)$.*

PROOF. Please see appendix B. □

Theorem *If the agents who participate in Protocol 2 are semi-honest, then the probability that a type 2 attack will reveal the local feedback value of an agent $a \in S_t$ is at most $P(d \in Z_a)$.*

PROOF. Please see appendix B. □

Under both types of attacks, the probability that the privacy of agent a's local feedback value will be preserved is at least: $1 - P(d \in Z_a) = 99\%$.

Please note that in the case of type 1 attack, an agent does not rely solely on a seed agent for it's privacy unless it is unable to find other trustworthy agents over the course of the protocol. However, as we observe in the experiment in section 7.3 conducted on a real and large web of trust, a large majority of the agents is able to find trustworthy agents thus avoiding total reliance on the seed agent.

Even though the seed agents are highly trustworthy and their effectiveness has been demonstrated in systems such as EigenTrust and Advogato, it is possible that an agent might not feel comfortable sharing its feedback when it has to rely solely on a seed agent for its privacy. A simple extension to the protocol which enables agents to abstain from providing feedback is as follows: Due to the absence of trustworthy agents or due to any other reason if an agent is unwilling to contribute its feedback, it can provide dummy feedback of value 0 and indicate to the querying agent or alternatively all agents in the protocol that it has abstained from providing its real feedback. The agent can participate in the rest of the protocol as usual.

The privacy guarantee for a type 2 attack relies solely on a seed agent. However, since to the best of our knowledge this work is the first attempt to a solution for the type 2 attack in a decentralized additive reputation system, we believe that it is a step towards stronger privacy guarantees. We can also make the following enhancement to the protocol to eliminate total reliance on a seed agent in the case of a type 2 attack: Let's assume that when an agent assigns feedback to a target agent, $\lfloor \frac{|S_a|}{3} \rfloor = \gamma$, where γ is some constant. The agent contributes its real feedback only if ($\lfloor \frac{|S_a|}{3} \rfloor = \gamma \wedge |S_a| \ mod\ 3 = 0) \vee \lfloor \frac{|S_a|}{3} \rfloor > \gamma$. This implies that a new source agent contributes its feedback only when there are at least two other agents contributing their values for the first time. Thus a type 2 attack is unable to differentiate between the feedbacks provided by the three new source agents. This solution is complementary to the existence of the seed agent since it is also probabilistic in terms of preserving privacy. A number higher than 3 would increase the probability of privacy being preserved while decreasing the rate at which new feedback effects the reputation.

5.4 Communication Complexity

For n source agents, the protocol requires $n+1$ messages in the *forwards* round, $n+1$ messages in the *backwards* round, n messages from the seed agent to the source agents, and 2 messages between the querying agent and the target agent. The total number of messages required is $3n + 4$, thus the complexity of the protocol in terms of number of messages exchanged is $O(n)$. This is in contrast to the complexity of $O(n^2)$ of the protocol secure under the semi-honest model described in [14].

In terms of bandwidth used, our protocol requires transmission of $O(n^2)$ number of agent IDs and $O(n)$ number of integers over the course of a query. In contrast, the protocol given in [14] requires transmission of $O(n^2)$ number of agent IDs as well as $O(n^2)$ number of integers. In practice, our protocol would also economize on bandwidth due to the fewer number of connections that it requires to be established between agents (linear vs. quadratic in [14]).

6. PROTOCOL 3: RESILIENCY AGAINST NON-DISRUPTIVE MALICIOUS ADVERSARIES

In this section we present protocol 3, which is an extended version of the protocol 2 introduced in the previous section. Protocol 3 preserves privacy against the type 1 and type 2 attacks under the non-disruptive malicious adversarial model.

Protocol 2 assumes that all agents would follow the protocol properly. However, non-disruptive malicious agents are not bound to conform to the protocol. They can deviate from the protocol as well as take actions that are outside the protocol in attempt to learn local feedbacks of other agents. We anticipate the following actions that non-disruptive malicious agents could take to sabotage protocol 2.

1. A non-disruptive malicious agent could eavesdrop on the communication of an agent in S_t and learn all the messages that it exchanges with other agents over the course of a query.
2. Agent q could drop agents from S_t, keeping only those agents who are colluding with it along with one non-colluding agent who is under attack. To gain unfair advantage, agent t could also drop the agents from S_t whom he thinks might have rated him poorly.
3. Agent q or an agent in S_t could drop agents from S before they have participated in the query, keeping only those agents who are colluding with it along with one non-colluding agent who is under attack.

6.1 Extensions to Protocol 2

Protocol 3 adds the following extensions to Protocol 2 to make it resistant to the malicious actions described above.

6.1.1 Secure Communication

Eavesdropping is prevented by requiring all messages to be exchanged via secure communication, which can be achieved through a protocol such as TLS (Transport Layer Security) or SSL (Secure Sockets Layer).

6.1.2 Source Managers

The set S_a is no longer maintained by agent a. In Protocol 3, the set S_a is maintained for agent a by two or more other agents in the system independently of each other. Those agents are called the *source managers* of agent a.

When a source agent assigns feedback to a target agent, it reports that event to each of the source managers of the target agent. The source managers add the source agent to the set S_t that they each maintain for the target agent t.

Agent q retrieves the set S_t from the source managers of agent t. It is possible that a number of the source agents are colluding with agent t and thus drop agents from S_t as desired by t. To counter this problem, an agent that needs the set S_t, retrieves it from all the source managers of agent t and then takes the union of all those sets to get the final S_t. Thus even if a single source manager is honest, the final set S_t would include all source agents of agent t.

To retrieve S_t from a source manager of agent t in Protocol 3, agent q sends the tuple $(\text{REQUEST\_FOR\_TUPLE}, q, t, p)$ to the source manager. The source manager returns a signed credential which includes S_t and (q, t, p). Agent q creates a vector $\overrightarrow{\mathcal{P}}$ that includes this credential retrieved from all source managers of agent t. The simple set S_t that is part of messages in Protocol 2 is replaced by the vector $\overrightarrow{\mathcal{P}}$ in Protocol 3. Each agent, participating in a query identified by (q, t, p), that receives this vector can derive the final S_t by taking the union of all sets in the credentials. Each agent who receives $\overrightarrow{\mathcal{P}}$ verifies that it includes the credential from all source managers of t and that each credential is signed by the issuing source manager. This measure prevents agent q from dropping agents from S_t.

We now discuss how the source managers of an agent t are located.

Score managers of a peer in the EigenTrust [10] reputation system are other peers that compute its reputation score. The following excerpt from [10] describes how score managers are assigned and located in EigenTrust.

> To assign score managers, we use a distributed hash table (DHT), such as CAN or Chord. DHTs use a hash function to deterministically map keys such as file names into points in a logical coordinate space. At any time, the coordinate space is partitioned dynamically among the peers in the system such that every peer covers a region in the coordinate space. Peers are responsible for storing (key, value) pairs the keys of which are hashed into a point that is located within their region.
>
> In our approach, a peer's score manager is located by hashing a unique ID of the peer, such as its IP address and TCP port, into a point in the DHT hash space. The peer which currently covers this point as part of its DHT region is appointed as the score manager of that peer. All peers in the system which know the unique ID of a peer can thus locate its score manager.

We implement source managers in the same manner as described above for score managers in EigenTrust. Peers in EigenTrust correspond to agents in our system.

6.1.3 Verifiable Participation

To prevent an agent from maliciously dropping other agents from the set S, Protocol 3 implements the following measures:

A new element, vector $\overrightarrow{\mathcal{Q}}$ is added to the tuples of the FORWARDS and BACKWARDS messages.

The vector $\overrightarrow{\mathcal{Q}}$ is empty in the first FORWARDS message sent out by the querying agent. An agent $a \in S_t$ processes a FORWARDS message the same as in Protocol 2. However, it also adds a signed credential $C_a^{forwards}$ to the vector $\overrightarrow{\mathcal{Q}}$ before sending it out. The content of $C_a^{forwards}$ is the sequence $(\mathcal{F}, q, t, p)$, where $\mathcal{F}$ is a constant. Each agent that receives a FORWARDS message verifies that for any agent k that is in S_t but not in S, the credential $C_k^{forwards}$ with the correct q, t, and p is present in the vector $\overrightarrow{\mathcal{Q}}$. This ensures that agents cannot be arbitrarily dropped by non-disruptive malicious agents in the *forwards* round.

Similar steps are taken in the *backwards* round. The seed agent sends out an empty $\overrightarrow{\mathcal{Q}}$. In addition to the regular processing of a BACKWARDS message, an agent $a \in S_t$ adds a signed credential $C_a^{backwards}$ to the vector $\overrightarrow{\mathcal{Q}}$ before sending it out. The content of $C_a^{backwards}$ is the sequence $(\mathcal{B}, q, t, p)$, where $\mathcal{B}$ is a constant. Verification is done by each agent in the same manner as in the *forwards* round, thus also preventing any agents maliciously being dropped from S in the *backwards* round.

6.2 Communication Complexity

The querying agent and each of the source agents need to perform a DHT lookup to locate the target agent's source managers. Considering a DHT such as Chord [16], which requires $O(log\ N)$ messages for a lookup, the number of additional messages required by protocol 3 is $(n+1)\cdot O(log\ N)$, or $O(n\ log\ N)$. The communication complexity of protocol 3 is thus: $O(n) + O(n\ log\ N)$, or $O(n\ log\ N)$.

Figure 1 compares protocol 3 with the protocol by Pavlov et al. [14] that is resilient against non-disruptive malicious adversaries. Our protocol performs better after $n = 13$ for $N = 11,558$ (Advogato.org) and after $n = 19$ for $N = 1,000,000$.

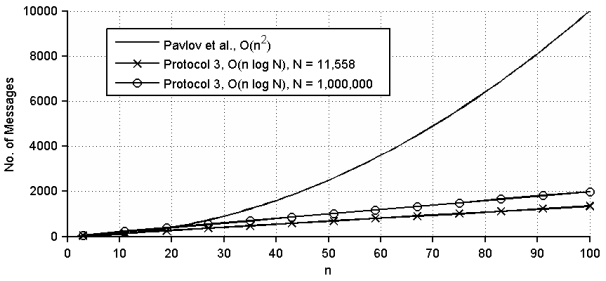

Figure 1: Protocol 3 vs Pavlov et al.

Please note that protocol 3 is not resilient against *disruptive* malicious agents, who could disrupt the system by dropping messages or by adding values that are out of range. However, such an action would still not reveal private feedbacks of other agents. An efficient protocol that is resilient against disruptive malicious agents is identified as future work. Please see section 9 for a discussion of some possible techniques for a protocol resilient against disruptive malicious agents.

7. EXPERIMENTS

We conduct three experiments to examine different aspects of Protocol 2 (which is also the foundation for protocol 3). The data set and the implementation of the experiments are described in the next two sections. The final three sections give details of the experiments.

7.1 Data Set

The data set that we use for our experiments is the real web of trust of Advogato.org [13]. Advogato.org is a web-based community of open source software developers. A major focus of the site is a peer rating system. The members of the site rate each other in terms of their trustworthiness. The choice of feedback values are *master*, *journeyer* and *apprentice*, with *master* being the highest level in that order. The result of these ratings among members is a rich web of trust, which comprises of $11,558$ users and $51,119$ trust ratings.

The instance of the Advogato web of trust referenced in this paper was retrieved on November 19, 2007 by crawling the Advogato.org web site with a script that we wrote in Python. To conform the Advogato web of trust to our framework, we substitute its three feedback values as follows: $master = 1.0$, $journeyer = 0.66$, and $apprentice = 0.33$. The Advogato rating system does not offer any feedback values for neutral or negative trust. Advogato.org identifies four seed users (*raph*, *miguel*, *mako*, *alan*), who are considered the most trustworthy users in the system.

The Advogato web of trust may be viewed as a directed weighted graph, with users as the vertices and trust ratings as the directed weighted edges of the graph. The number of vertices with no outgoing edges is $5,832$ and the number of vertices with no incoming edges is $5,548$.

7.2 Implementation

The experiments have been implemented as individual Java programs. Each program starts off by creating agent objects and references that correspond to the vertices and the edges of the Advogato web of trust respectively. The program then initiates queries according to the algorithm of the experiment and logs the required data to file. The user *cbz* is randomly selected as the querying agent for the experiments. The set of seed agents is made up of the four users identified by Advogato as seeds.

We choose Y as 2, which implies that the random numbers used in the protocol lie on the interval $[-2, 2]$. We add the following heuristic to the *forwards* round of the protocol (l is the local feedback value of an agent and y is the random number added to it for data perturbation): if $|l + y| > Y$, y is regenerated until the condition holds false. This heuristic inhibits instances of data perturbation which fail to completely hide the local feedback value. The heuristic also allows us to use $Y = 2$ instead of a higher number.

7.3 Experiment: Probability that Privacy will be Preserved

We conduct this experiment to observe the effectiveness of the protocol in preserving the privacy of agents in a real web of trust.

Algorithm: The querying agent queries the reputation of every other agent in the environment (a total of $11,557$ agents). Over the course of successful queries, we consider every instance of a source agent a that is not the last agent in either the *forwards* or the *backwards* round. The following information is logged for all such instances of source agents: t, a, $P(a_{(f,out)} \in Z_a)$, and $P(a_{(b,out)} \in Z_a)$.

Results: Queries succeed for $3,761$ target agents since the rest of the agents have less than 2 source agents. Over the course of successful queries, the number of instances of source agents is $45,109$. As discussed in theorem B, the probability that a type 1 attack will reveal the local feedback value of an agent a is given as: $P(a_{(f,out)} \in Z_a) \times P(a_{(b,out)} \in Z_a) \times P(d \in Z_a)$. The trustworthiness of seed agents is universally considered as at least 0.99, which implies that $P(d \in Z_a) \leq 0.01$ for all instances of source agents. The probability that the privacy of a source agent will be preserved is the complement of the probability that its local feedback value will be revealed. The probability that privacy will be preserved is computed for all instances of source agents. The frequency distribution of the probabilities is given in table 1.

Discussion: The probability that the privacy of a source agent will be preserved is always at least 99%. This is made possible due to the participation of a seed agent in each query. A high percentage (68.2%) of source agents are able to find trustworthy agents among fellow source agents in the *forwards* and/or the *backwards* round. This results in a probability that is higher than the default. A significant

Table 1: Probability that privacy will be preserved.

Probability	Count	Percentage (Total: 45,109)
99.00%	14,354	31.8%
99.33%	3,068	6.8%
99.55%	774	1.7%
99.66%	7,313	16.2%
99.77%	2,102	4.7%
99.88%	5,679	12.6%
100.00%	11,819	26.2%

percentage (26.2%) of instances of source agents receive a 100% guarantee that their privacy will be preserved.

This experiment does not cover instances of source agents who are last in either the *forwards* or the *backwards* round. However, please note that as discussed in section 5.3, the probability that their privacy will be preserved is also at least 99%. A simple extension to the protocol is also suggested in the same section which enables agents to abstain from contributing their feedback when they do not receive a sufficient privacy guarantee.

7.4 Experiment: Accuracy of the Reputation Scores

In the following experiment we observe the effect of adding the random variable $x \in [-Y, Y]$ (for perturbation) on the accuracy of the reputation scores.

Algorithm: The querying agent queries the reputation of every other agent. The following information is logged for each of the target agents: t, actual reputation of t (without the addition of x), perturbed reputation of t (computed by the protocol, with the addition of x).

Results: Figure 2 depicts a scatter plot of the actual reputation values and the perturbed reputation values. The 80 target agents with actual scores that range from 60.33 to 726.97 have been omitted to provide a better resolution of the more densely populated area. The perturbed reputation values of all agents (including the 80 agents not plotted) are within ± 2 of their corresponding actual reputation values.

The average difference between the actual and the perturbed reputation values is 1.00 (rounded down to two decimal places).

Discussion: The results follow the discussion in section 5.2.1. The absolute difference between the actual reputation of a target agent and the corresponding perturbed reputation is at most Y (2 in this case). The addition of the random variable $x \in [-2, 2]$ does not have a drastic effect on reputation values in the case of this data set. Actual reputation values that may be interpreted as low, remain low after being perturbed. Similarly, reputation values that are relatively high, stay high.

7.5 Experiment: Load on Agents

In this experiment we observe the load that an agent has to bear to fulfill its role as a source agent. The experiment sets up the following scenario: in a unit of time, every agent in the system initiates one query to learn the reputation of a random unknown agent. We would like to know: 1) within that unit of time, the number of queries that an agent has to participate in as a source agent, 2) the relationship of that load to the number of feedbacks assigned by the agent

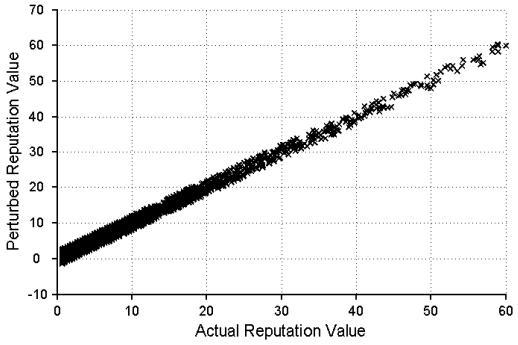

Figure 2: Actual reputation vs perturbed reputation.

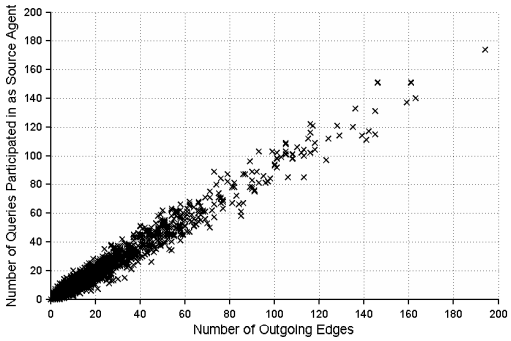

Figure 3: Load on agents.

(number of outgoing edges).

Algorithm: Each of the 11,558 agents in the system, randomly selects a previously unknown agent and queries its reputation. The following information is recorded for each agent: the number of queries that it participates in as a source agent over the course of the experiment.

Results: The number of successful queries is 3,649. The rest of the queries do not succeed since their target agents have less than 2 source agents. Figure 3 shows a scatter plot of the number of outgoing edges of an agent and the number of queries that it participates in as a source agent. The linear correlation (Pearson) between the two variables is 0.99 (rounded down to two decimal places). The average number of queries that an agent has to participate in is 0.68 per feedback assigned. There are 5,832 agents who have 0 outgoing edges. They have not been considered for computing this average as they have not assigned any feedback.

Discussion: Clearly, there is a recurring cost associated with each feedback that an agent assigns. However, this does not imply that agents would avoid assigning feedback. Generally, a strong incentive in reputation systems for assigning feedback is that other agents reciprocate. An agent that assigns no feedback and thus receives no feedback would find itself in the undesirable state of having no reputation.

8. RELATED WORK

Our work shares many similarities with the work by Pavlov et al. [14], which also focuses on decentralized additive reputation systems. However, their protocol that is resilient

against non-malicious disruptive adversaries requires $O(n^2)$ messages for n source agents. In our protocol, agents exchange messages with a constant number of agents which leads to a tighter bound. Moreover, we identify the type 2 attack and present a solution for it. Additionally, we also provide experimental evaluation of our proposal. Pavlov et al. also present a protocol that is resilient against malicious disruptive adversaries. That protocol has a communication complexity of $O(n^3)$.

A number of privacy preserving reputation systems are based on the premise that a trusted hardware module is present at every agent. A decentralized system proposed by Kinateder and Pearson [11] 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. Voss et al. [18] and Bo et al. [3] also present decentralized systems which are based on similar lines, however they both suggest using smart cards as the trusted hardware modules. A later system by Kinateder et al [12] avoids the hardware modules, however it requires anonymous routing infrastructure at the network level. These systems clearly differ from our approach, which does not mandate specialized platforms.

Several privacy preserving reputation systems have the concept of e-cash as their basis. One such system is suggested for centralized environments by Ismail et al. [9]. Later, in [8], they also propose a decentralized version. However, an essential part of their architecture is a "trusted third party" – a central authority, which creates a single point of failure. A reputation system for decentralized anonymous networks which makes use of e-cash is presented by Androulaki et al. [2]. However, due to the presence of a central "bank" in the system, it also suffers from the problem of a single point of failure. In comparison, our architecture does not have this limitation. Please note that there is no bound on the number of seed agents in our system.

None of the works cited above provide evaluation of the proposed systems with data from a real web of trust.

9. FUTURE WORK

We are currently working on the development of an efficient protocol that is resilient against disruptive malicious agents. We identify two main actions that disruptive malicious agents may take to disrupt the system: 1) drop messages, 2) add values that are out of range.

A provisional solution for the first problem is as follows: Each message is relayed by the querying agent. The sender agent, instead of sending a message directly to the recipient agent, must send the message to the querying agent. The querying agent then relays the message to the recipient agent. This enables the querying agent to identify any agent that drops a message. To prevent the querying agent from compromising the privacy of agents, each message is encrypted by the sender with the recipient's public key. This technique prevents agents from maliciously dropping messages, however, it requires a significant number of expensive cryptographic operations. The theoretical bound remains unchanged, however, in practice the solution would raise the number of messages in protocol 2 to approximately twice as before.

For a solution to the problem of out of range values, we are currently looking at homomorphisms and zero knowledge proofs, and secure voting protocols.

10. CONCLUSION

We presented novel privacy preserving protocols for computing reputation in decentralized environments under semi-honest and non-disruptive malicious adversarial models. The protocols draw their strength from elements that include data perturbation, presence of pre-trusted seed agents, and most importantly the ability of feedback providers to themselves select trustworthy agents that they want to share intermediate information with. Our protocol that is resilient against non-disruptive malicious adversaries has loglinear communication complexity. This makes the protocol more efficient than comparable protocols discussed in the literature. Moreover, our protocols are fully decentralized and do not suffer from any single points of failure. An experiment conducted on data from the real web of trust of Advogato.org demonstrates that the protocols preserve the privacy of agents with high success. An important direction for future work is the development of an efficient protocol that is resilient against disruptive malicious adversaries.

11. REFERENCES

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, 2000.

[2] E. Androulaki, S. G. Choi, S. M. Bellovin, and T. Malkin. Reputation systems for anonymous networks. In *Proc. of the 8th Privacy Enhancing Technologies Symp. (PETS 2008)*, 2008.

[3] Y. Bo, Z. Min, and L. Guohuan. A reputation system with privacy and incentive. In *Proc. of the 8th ACIS Intl. Conf. on Soft. Eng., AI, Networking, and Parallel/Distributed Comp. (SNPD'07)*, 2007.

[4] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, Jan. 2003.

[5] W. Du. *A Study of Several Specific Secure Two-Party Computation Problems*. PhD thesis, Purdue University, West Lafayette, IN, USA, 2001.

[6] eBay. Upcoming changes to feedback. http://pages.ebay.com/services/forum/new.html, 2008. Retrieved June 30, 2008.

[7] O. Goldreich. Secure multi-party computation. Working Draft, Version 1.4, 2002.

[8] R. Ismail, C. Boyd, A. Josang, and S. Russell. Private reputation schemes for p2p systems. In *Proc. of the 2nd Intl. Workshop on Security in Info. Systems*, 2004.

[9] R. Ismail, C. Boyd, A. Josang, and S. Russell. Strong privacy in reputation systems. In *Proc. of the 4th Intl. Workshop on Info. Security Apps. (WISA'03)*, 2004.

[10] 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)*, 2003.

[11] 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 Techs.*, 2003.

[12] M. Kinateder, R. Terdic, and K. Rothermel. Strong pseudonymous communication for peer-to-peer

reputation systems. In *Proc. of the 2005 ACM Symp. on Applied Computing*, 2005.

[13] R. Levien. Attack resistant trust metrics. Manuscript, University of California - Berkeley. www.levien.com/thesis/compact.pdf, 2002.

[14] E. Pavlov, J. S. Rosenschein, and Z. Topol. Supporting privacy in decentralized additive reputation systems. In *Proc. of the 2nd Intl. Conf. on Trust Management (iTrust 2004)*, 2004.

[15] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions. *The Economics of the Internet and E-Commerce. Vol. 11 of Advances in Applied Microeconomics*, pages 127–157, 2002.

[16] 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 2001Conf. on Apps., Technologies, Architectures, and Protocols for Computer Communications*, 2001.

[17] J. Vaidya and C. Clifton. Privacy-preserving data mining: Why, how, and when. *IEEE Security and Privacy*, 2(6):19–27, November 2004.

[18] M. Voss, A. Heinemann, and M. Muhlhauser. A privacy preserving reputation system for mobile information dissemination networks. In *Proc. of the 1st Intl. Conf. on Security and Privacy for Emerging Areas in Comm. Networks (SECURECOMM)*, 2005.

APPENDIX

A. PROTOCOL 2

need arises to determine r_t

▷ initiate query to determine r_t
1 send tuple (REQUEST_FOR_SOURCES) to t
2 receive tuple (SOURCES, S_t) from t
3 **if** $|S_t| \geq 2$
4 **then** $a_{(f,out)} \leftarrow$ random_element(S_t)
5 $q \leftarrow a$
6 $p \leftarrow$ timestamp()
7 $r \leftarrow 0$
8 send tuple (FORWARDS, q, t, p, r, S_t, S_t) to $a_{(f,out)}$

tuple (REQUEST_FOR_SOURCES) **received from agent** k
1 send tuple (SOURCES, S_a) to k

Figure 4: Protocol 2.

B. PROOFS

Theorem 1. *If all agents properly follow Protocol 2, then at the completion of a query,* $r_t = \sum_{a \in S_t} l_{at} + x$.

PROOF. In the *forwards* round, the tuple (FORWARDS, q, t, p, r, S, S_t) arrives once at each agent in S_t. When the querying agent initiates the query, $r = 0$, and when the tuple arrives at the seed, each $a \in S_t$ has added the values of its l_{at} and $y_{(q,t,p)}$ to it. Let's say that the set $S_t = \{a_1, a_2, \ldots, a_n\}$ and let's refer to the $y_{(q,t,p)}$ value of agent a_i as $y^{a_i}_{(q,t,p)}$. Then the value of r when it reaches the seed is $r = \sum_{i=1}^{n} l_{a_i t} + \sum_{i=1}^{n} y^{a_i}_{(q,t,p)}$. The seed sends $x_1, x_2, \ldots, x_n$ to $a_1, a_2, \ldots, a_n$ respectively. $\sum_{i=1}^{n} x_i = x$. The seed then initiates the *backwards* round.

tuple (FORWARDS, q, t, p, r, S, S_t) **received from agent** $a_{(f,in)}$
1 **if** $a \in S \wedge |S_t| \geq 2$
2 **then** $r_{(f,in)} \leftarrow r$
3 $y_{(q,t,p)} \leftarrow$ random($-Y, Y$)
4 $r_{(f,out)} \leftarrow r_{(f,in)} + l_{at} + y_{(q,t,p)}$
5 $S_{(f,in)} \leftarrow S$
6 $S_{(f,out)} \leftarrow S_{(f,in)} - a$
7 **if** $|S_{(f,out)}| > 0$
8 **then** $a_{(f,out)} \leftarrow$ trustworthy($a, S_{(f,out)}$)
9 $a_{(q,t,p)} \leftarrow a_{(f,out)}$
10 send tuple (FORWARDS, $q, t, p, r_{(f,out)}$, $S_{(f,out)}, S_t$) to $a_{(f,out)}$
11 **else**
12 $a_{(f,out)} \leftarrow$ random_element(D)
13 $a_{(q,t,p)} \leftarrow$ NIL
14 send tuple (SEED, $q, t, p, r_{(f,out)}$, $S_{(f,out)}, S_t$) to $a_{(f,out)}$
15 store $y_{(q,t,p)}$ and $a_{(q,t,p)}$ in $\overrightarrow{\mathcal{Y}}$ and $\overrightarrow{\mathcal{A}}$ respectively

tuple (SEED, q, t, p, r, S, S_t) **received from agent** $a_{(f,in)}$
1 **if** $a \in D \wedge S = \phi$
2 **then** $n \leftarrow |S_t|$
3 $x \leftarrow$ random($-Y, Y$)
4 select $x_1, x_2, \ldots, x_n$ uniformly from $[-Y, Y]$ such that $\sum_{i=1}^{n} x_i = x$
5 $S_{temp} \leftarrow S_t$
6 **for** $i \leftarrow 1$ **to** n
7 **do** $s_i \leftarrow$ random_element(S_{temp})
8 $S_{temp} \leftarrow S_{temp} - s_i$
9 send tuple (PARTX, q, t, p, x_i) to s_i
10 $a_{(b,out)} \leftarrow$ random_element(S_t)
11 send tuple (BACKWARDS, q, t, p, r, S_t) to $a_{(b,out)}$

tuple (PARTX, q, t, p, x) **received from agent** d
1 **if** $d \in D \wedge y_{(q,t,p)}$ and $a_{(q,t,p)}$ exist in $\overrightarrow{\mathcal{Y}}$ and $\overrightarrow{\mathcal{A}}$ respectively
2 **then** $x_{(q,t,p)} \leftarrow x$
3 store $x_{(q,t,p)}$ in $\overrightarrow{\mathcal{X}}$

tuple (BACKWARDS, q, t, p, r, S) **received from agent** $a_{(b,in)}$
1 **if** $a \in S \wedge y_{(q,t,p)}$, $a_{(q,t,p)}$, and $x_{(q,t,p)}$ exist in $\overrightarrow{\mathcal{Y}}$, $\overrightarrow{\mathcal{A}}$, and $\overrightarrow{\mathcal{X}}$ respectively
2 **then** $r_{(b,in)} \leftarrow r$
3 $r_{(b,out)} \leftarrow r_{(b,in)} - y_{(q,t,p)} + x_{(q,t,p)}$
4 $S_{(b,in)} \leftarrow S$
5 $S_{(b,out)} \leftarrow S_{(b,in)} - a$
6 **if** $|S_{(b,out)} - a_{(q,t,p)}| > 0$
7 **then** $a_{(b,out)} \leftarrow$ trustworthy(a, $S_{(b,out)} - a_{(q,t,p)}$)
8 send tuple (BACKWARDS, q, t, p, $r_{(b,out)}, S_{(b,out)}$) to $a_{(b,out)}$
9 **else if** $|S_{(b,out)}| > 0$
10 $a_{(b,out)} \leftarrow$ trustworthy($a, S_{(b,out)}$)
11 send tuple (BACKWARDS, q, t, p, $r_{(b,out)}, S_{(b,out)}$) to $a_{(b,out)}$
12 **else** $a_{(b,out)} \leftarrow q$
13 send tuple (RESULT, q, t, p, $r_{(b,out)}, S_{(b,out)}$) to $a_{(b,out)}$
14 discard $y_{(q,t,p)}, a_{(q,t,p)}$, and $x_{(q,t,p)}$

tuple (RESULT, q, t, p, r, S) **received from agent** $a_{(b,in)}$
1 **if** $a = q$
2 **then** $r_t \leftarrow r$ ▷ query complete

Figure 5: Protocol 2 (contd.).

In the *backwards* round, the tuple (BACKWARDS, q, t, p, r, S) arrives once at each agent. Each of those n

Table 2: Description of the functions used in Protocol 2.

Function	Description
random_element(S)	Returns a random element from the set S
timestamp()	Returns current time. For any given target, an agent can only initiate one query per the smallest unit of time in the timestamp.
random(x,y)	Returns a random number uniformly distributed on the interval $[x, y]$
trustworthy(a, S)	Returns an agent k from the set S such that $l_{ak} \geq 0 \wedge \forall s \in S - k, l_{ak} \geq l_{as}$. If two or more agents meet this criteria, then one of the agents is selected at random. If none of the agents meet this criteria, then an agent is selected at random from S.

agents, $a_i \in S_t$ subtracts $y_{(q,t,p)}^{a_i}$ from r and adds x_i to it. When $(\text{RESULT}, q, t, p, r, S)$ arrives at q, all agents $a_i \in S_t$ have removed $y_{(q,t,p)}^{a_i}$ and added x_i to r, thus $r = \sum_{i=1}^{n} l_{a_i t} + \sum_{i=1}^{n} y_{(q,t,p)}^{a_i} - \sum_{i=1}^{n} y_{(q,t,p)}^{a_i} + \sum_{i=1}^{n} x_i$, or $r_t = r = \sum_{a \in S_t} l_{at} + x$. □

Theorem 2. *If the agents who participate in Protocol 2 are semi-honest, then at the completion of a query, the probability that a type 1 attack will reveal the local feedback value of an agent $a \in S_t$, who is not the last agent in the* forwards *or the* backwards *round, is: $P(a_{(f,out)} \in Z_a) \times P(a_{(b,out)} \in Z_a) \times P(d \in Z_a)$.*

Proof. An agent $a \in S_t$, who is not the last agent in the *forwards* round, exchanges information with five agents from the start to the end of a query. Those agents are identified in the protocol as $a_{(f,in)}$, $a_{(f,out)}$, $a_{(b,in)}$, $a_{(b,out)}$, and d. In a type 1 attack, agents may act individually or they may collude. Let's first see what each of these agents learns individually.

$a_{(f,in)}$ does not receive anything from a thus it does not learn anything. $a_{(f,in)}$ knows:

$$r_{(f,in)} = c_1 \tag{1}$$

$c_1, c_2, \ldots$ are constants.

$a_{(f,out)}$ receives $r_{(f,out)} = r_{(f,in)} + l_{at} + y_{(q,t,p)}$ from a. Since $r_{(f,in)} + y_{(q,t,p)}$ is added to l_{at}, $a_{(f,out)}$ does not learn l_{at} (data perturbation). It does not learn $y_{(q,t,p)}$ due to the same assumption. $a_{(f,out)}$ knows:

$$r_{(f,in)} + l_{at} + y_{(q,t,p)} = c_2 \tag{2}$$

$a_{(b,in)}$ does not receive anything from a thus it does not learn anything. $a_{(b,in)}$ knows:

$$r_{(b,in)} = c_3 \tag{3}$$

$a_{(b,out)}$ receives $r_{(b,out)} = r_{(b,in)} - y_{(q,t,p)} + x_{(q,t,p)}$ from a. Since $r_{(b,in)} - l_{at} + x_{(q,t,p)}$ is still added to l_{at}, $a_{(b,out)}$ does not learn l_{at} (data perturbation). $a_{(b,out)}$ knows:

$$r_{(b,in)} - y_{(q,t,p)} + x_{(q,t,p)} = c_4 \tag{4}$$

d does not receive anything from a thus it does not learn anything. d knows:

$$x_{(q,t,p)} = c_5 \tag{5}$$

Now let's see what the agents learn if they collude. The set $\{a_{(f,in)}, a_{(f,out)}, a_{(b,in)}, a_{(b,out)}, d\}$ allows 32 possible subsets of colluding agents. The colluding agents are able to determine l_{at} only with the subset $\{a_{(f,in)}, a_{(f,out)}, a_{(b,in)}, a_{(b,out)}, d\}$, that is if it contains all five agents.

From equations 1 and 2 we have:

$$c_1 + l_{at} + y_{(q,t,p)} = c_2$$
$$l_{at} + y_{(q,t,p)} = c_6 \tag{6}$$

From equations 3, 4, and 5 we have:

$$c_3 - y_{(q,t,p)} + c_5 = c_4$$
$$y_{(q,t,p)} = c_7 \tag{7}$$

Subtracting equation 7 from equation 6 we get:

$$l_{at} + y_{(q,t,p)} - y_{(q,t,p)} = c_6 - c_7$$
$$l_{at} = c_8$$

As observed, l_{at} can be revealed only if $a_{(f,in)}$, $a_{(f,out)}$, $a_{(b,in)}$, $a_{(b,out)}$, and d are all in Z_a. The probability that these five agents are in Z_a is: $P(a_{(f,in)} \in Z_a) \times P(a_{(f,out)} \in Z_a) \times P(a_{(b,in)} \in Z_a) \times P(a_{(b,out)} \in Z_a) \times P(d \in Z_a)$.

Since a has no control over who $a_{(f,in)}$ and $a_{(b,in)}$ are, we assume that they are in Z_a and thus $P(a_{(f,in)} \in Z_a) = 1$ and $P(a_{(b,in)} \in Z_a) = 1$.

Thus the probability that a type 1 attack will reveal the local feedback value of an agent $a \in S_t$, who is not the last agent in the *forwards* round, is: $P(a_{(f,out)} \in Z_a) \times P(a_{(b,out)} \in Z_a) \times P(d \in Z_a)$. □

Theorem 3. *If the agents who participate in Protocol 2 are semi-honest, then at the completion of a query, the probability that a type 1 attack will reveal the local feedback value of an agent $a \in S_t$ is at most $P(d \in Z_a)$.*

Proof. In the *forwards* round, the privacy of an agent a is preserved due to the addition of y to its local feedback value l_{at}. The value of y is a secret known only by a itself, thus the probability that y or l_{at} will be revealed is 0. In the *backwards* round, a's privacy is preserved by the addition of x to l_{at}. Other than a, the value of x is known only by the seed agent d. The probability that x will be revealed is $P(d \in Z_a)$. Any attacker or collective of attackers must know x to learn l_{at}. Thus, the probability that l_{at} would be revealed is at most $P(d \in Z_a)$. □

Theorem 4. *If the agents who participate in Protocol 2 are semi-honest, then the probability that a type 2 attack will reveal the local feedback value of an agent $a \in S_t$ is at most $P(d \in Z_a)$.*

Proof. Let's say that a querying agent q mounts an attack of type 2 on agent a. Immediately before agent a updates l_{at}, q queries for the reputation of agent t and receives r_t which is given as:

$$\sum_{k \in A-a} l_{kt} + l_{at} + x_d = c_1 \tag{8}$$

where d is the seed agent in the query and x_d is the random value that it adds. $c_1, c_2, \ldots$ are constants.

Then immediately after agent a updates l_{at}, q queries again for the reputation of agent t and receives r'_t which is given as:

$$\sum_{k \in A-a} l_{kt} + l'_{at} + x_{d'} = c_2 \tag{9}$$

where r'_t and l'_{at} are the updated values of r_t and l_{at} respectively, d' is the seed agent in the query and $x_{d'}$ is the random value that it adds.

From equations 8 and 9 we have:

$$c_1 - l_{at} - x_d = c_2 - l'_{at} - x_{d'}$$

$$l_{at} - l'_{at} = c_3 - x_d + x_{d'} \tag{10}$$

Equation 10 shows that to learn $\delta l = l_{at} - l'_{at}$, the seed agents d and d' would have to be in Z_a. The probability that both d and d' are in Z_a is: $P(d \in Z_a) \times P(d' \in Z_a)$.

The probability for q to learn δl is at its highest if d and d' are the same agents. In that case the probability would be $P(d \in Z_a)$.

Thus the probability that a type 2 attack will reveal the local feedback value of an agent $a \in S_t$ is at most $P(d \in Z_a)$. $\square$