Efficient Privacy Preserving Reputation Protocols Inspired by Secure Sum

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*—The secure sum protocol is a well-known protocol for computing the sum of private inputs from distributed entities such that the inputs remain private. In this paper we present protocols for computing reputation in a privacy preserving manner that are inspired by the secure sum protocol. We provide a protocol that is secure under the semi-honest adversarial model as well as one that is secure under the stronger non-disruptive malicious model. Although the protocols are inspired by secure sum, they do not suffer from the issues that plague secure sum. Our protocols are resilient against colluding entities, which secure sum is not. The protocols that we develop are also efficient. We require an exchange of $O(n)$ messages under the semi-honest model, where n is the number of feedback providers in the protocol. This is the same complexity offered by secure sum. Our protocol for the non-disruptive malicious model exchanges $O(N \log n)$ messages, where N is the number of entities in the system.**

I. Introduction

Reputation systems represent a key technology for securing distributed applications from misuse by dishonest entities. A reputation system computes the reputation score of an entity as the aggregate of the feedback provided by the other entities in the system. Reputation scores help identify the entities that are exhibiting undesirable 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 [11] 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 [14] for spam filtering.

The reputation score of a target entity is a function of the feedback values provided by other entities. Thus an accurate reputation score is possible only if the feedback is accurate. However, it has been observed that the users of a reputation system may avoid providing honest feedback [16]. 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. To counter the issue of retaliatory feedback, eBay 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 article, we focus on privacy preserving reputation protocols for decentralized environments. Examples of decentralized environments include peer-to-peer file sharing networks, decentralized social networks (such as FOAF), MANETs, etc. The well-known secure sum protocol is a natural fit for computing reputation in a decentralized manner while preserving privacy. However, the basic secure sum protocol suffers from substantial problems. Most significantly, it is not secure under the semi-honest adversarial model when entities are allowed to collude. We develop reputation protocols that are inspired by secure sum. However, unlike secure sum, our protocols are resilient against colluding entities. Moreover, one of our protocols is secure under the stronger non-disruptive malicious adversarial model.

The protocols that we present are also efficient in terms of the number of messages exchanged and the bandwidth utilized. Our protocol that is resilient against semi-honest malicious adversaries has linear communication complexity ($O(n)$, where n is the number of feedback providers in the protocol). Comparable protocols with similar strengths by Pavlov et al. [15] and Gudes et al. [8] are quadratic. The protocol that is secure against non-disruptive malicious adversaries requires an exchange of $O(N \log n)$ messages, which is still more efficient than comparable protocols (N is the number of entities in the system).

II. Preliminaries

A. Preserving Privacy

Definition 1. ***Preserving Privacy.*** *Let Π be a protocol for computing reputation. Let a be an agent that contributes l_{at} (its local private feedback about an agent t) as input to protocol Π for computing the reputation of agent t. The privacy of agent a is said to be preserved if the execution* *of protocol Π does not result in the disclosure of agent a's private feedback to any other agent in the system.*

B. Adversarial Models

In this article, we present protocols that preserve privacy under the following two adversarial models:

Semi-Honest. Semi-honest agents always correctly follow the protocol for computing the reputation of an agent. However, they use the intermediate information received during the protocol to derive the private feedback values of other agents. Semi-honest agents may mount the following types of attacks:

1) *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 feedback of agent a about agent t from those intermediate values.
2) *Type 2 Attack.* An adversary observes the reputation of agent t immediately before and after agent a updates its feedback about agent t. Since the reputation is computed in an additive manner, the adversary can learn about agent a's private feedback as: $\delta l = r'_t - r_t$, where δl is the difference between agent a's previous and current 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 that addresses the Type 2 attack in a decentralized additive reputation system.

Non-Disruptive Malicious. In addition to the capabilities of semi-honest agents, non-malicious disruptive agents may also deviate from the protocol. However, non-disruptive malicious agents have a single objective: to learn private feedback values of other agents. They do not intend to disrupt the normal function of the protocol other than to achieve this objective. This is a practical model when the output of the protocol is received by all participants and we consider that all participants are interested in learning the correct output.

C. 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):

Definition 2. ***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], [18], 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. However, the random numbers may be selected on a distribution (such as Gaussian) that renders the probability of such occurrences low.

III. The Basic Framework of the 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 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. Summation of local feedback values 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.

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.

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 [14] and EigenTrust [11]. 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. For a seed agent d, $P(d \in Z_a) = 1 - 0.99 = 0.01$.

IV. Protocol 1: Resiliency against Semi-Honest Adversaries

We now present a reputation protocol that preserves privacy against both types of attacks under the semi-honest adversarial model. The protocol is inspired by the well-known secure sum protocol [4], [18], in which each agent adds its feedback value to a running total and the last agent sends the sum to the querying agent. However, in contrast to the secure sum protocol, our protocol is secure against colluding agents. Additionally, our protocol provides security against the type 2 attack. The communication complexity of our protocol is the same as the secure sum protocol, that is, $O(n)$.

A. Protocol Outline

- 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, where $|S_t| \geq 3$. 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 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 seed agent.
- The seed agent generates a random number $x \in [-Y, Y]$ and then selects $n - 1$ random numbers and one chosen number such that their sum is equal to x, where $n = |S_t|$. 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, if possible, 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 random number x and the feedback values of all agents in S_t. This value of r is considered the reputation value of agent t.

B. Privacy

*1) **Type 1 Attack:*** Each agent exchanges information with five agents during the protocol. All five of those agents must collude to learn the 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.

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

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

Please see the appendix for proofs.

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 VI-B 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 [11] and Advogato [14], 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.

*2) **Type 2 Attack:*** The true sum of the 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 x added by the seed agent. Thus simply observing a reputation value before and after an update, does not reveal the 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.

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

Please see the appendix for proof.

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

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 feedback 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.

C. Correctness

Theorem 4. *If all agents properly follow Protocol 1, then at the completion of a query, $r_t = \sum_{a \in S_t} l_{at} + x$.*

Please see the appendix for proof.

The addition of x implies that the result of the query deviates from the true sum by a random value on the interval $[-Y, Y]$. Absolute difference is given as: *absolute difference* $=$ $|$*actual reputation* $-$ *perturbed reputation*$|$. Since $x \in [-Y, Y]$, $\Rightarrow$ *absolute difference* $\leq Y$. Relative difference is expressed as: *relative difference* $= |$*actual reputation* $-$ *perturbed reputation*$|$ / *actual reputation*, where *actual reputation* $\neq 0$. The bound on relative difference is inversely proportional to the actual reputation.

D. 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 [15].

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

```
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 $k$

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
```

Fig. 1. Protocol 1 – The Semi-Honest Model.

given in [15] 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 [15]).

E. Discussion

One of the key innovations in this protocol is that an agent himself selects partners whom he wants to share intermediate information with. This selection is based on the existing trust relationships that the agent has with others. The advantages of this approach are twofold.

Firstly, since the agent himself selects the partners whom to trust, he can maximize the probability that his privacy will be preserved. Choosing the agents whom to trust also allows an agent to quantify the value of that probability. This is in contrast to the secure sum protocol, in which the partners are pre-determined by the querying agent for each source agent. In the case that the probability does not meet an agent's threshold for privacy, the agent is also able to abstain from providing feedback. Our protocol provides security against colluding agents, which the secure sum protocol does not.

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. [15] for the same adversarial model, which requires each agent to exchange messages with all other agents in the protocol resulting in quadratic communication complexity. Additionally, unlike in [15], an agent in our protocol contributes its private feedback only if the probability that its privacy will be preserved is sufficient enough.

Another innovation in our protocol is the presence of seed agents, which help in preventing the type 2 attack.

We evaluate our protocol on data from a real and large web of trust in section VI.

F. Protocol Specification

All agents in the system are driven by a common protocol. The protocol for an agent a in the system is given in figure 1 as a collection of events and associated actions. The protocol assumes that $|S_t| \geq 3$.

Description of the functions used in Protocol 1: **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.

V. Protocol 2: Resiliency against Non-Disruptive Malicious Adversaries

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

Protocol 1 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 feedback of other agents. This is a practical model when the output of the protocol is received by all participants and we consider that all participants are interested in learning the correct output.

We anticipate the following actions that non-disruptive malicious agents could take to sabotage protocol 1.

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.

A. Extensions to Protocol 1

Protocol 2 adds the following extensions to Protocol 1 to make it resistant to the malicious actions described above.

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 SSL or IPSec.

2) Source Managers: The set S_a is no longer maintained by agent a. In Protocol 2, 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. The idea of source managers is inspired by score managers in EigenTrust [11].

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 2, 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 1 is replaced by the vector $\overrightarrow{\mathcal{P}}$ in Protocol 2. 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.

To assign and locate source managers, a Distributed Hash Table (DHT) may be used. An agent's source managers would be located by hashing the unique ID of the agent.

3) Verifiable Participation: To prevent an agent from maliciously dropping other agents from the set S, Protocol 2 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 1. 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.

B. 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 [17], which requires $O(log\ N)$ messages for a lookup, the number of additional messages required by protocol 2 is $(n + 1) \cdot O(log\ N)$, or $O(n\ log\ N)$. The communication complexity of protocol 2 is thus: $O(n) + O(n\ log\ N)$, or $O(n\ log\ N)$.

Compared to the protocol by Pavlov et al. [15] that has $O(n^2)$ complexity, our protocol performs better after $n = 13$ for $N = 11,558$ (Advogato.org) and after $n = 19$ for $N = 1,000,000$. Moreover, the protocol described by Pavlov et al. is secure only against semi-honest adversaries in contrast to our protocol 2 that provides security under the stronger non-disruptive malicious adversarial model.

VI. An Experiment

A. Data Set

The data set that we use for our experiment is the real and large web of trust of Advogato.org [14]. The members of Advogato.org 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 instance of the Advogato web of trust referenced in this paper comprises of $11,558$ users and $51,119$ trust ratings. 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 $apprentice = 0.33$.

B. Experiment: Probability that Privacy will be Preserved

The objective of the experiment is to observe the effectiveness of the protocols in preserving the privacy of agents in a real web of trust.

Algorithm: We query the reputation of every agent in the environment (a total of $11,557$ agents). Over the course of successful queries (where $|S_t| \geq 3$), 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: Over the course of successful queries, the number of instances of source agents is $45,109$. As discussed in theorem 2, the probability that a type 1 attack will reveal the private 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 private 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 I.

TABLE I
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%

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 ($100\% - 31.8\% = 68.2\%$) 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 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 IV-B, 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.

VII. Related Work

The work by Pavlov et al. [15] also focuses on decentralized additive reputation systems. However, their protocol that is resilient against semi-honest 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 of $O(n)$. Moreover, our protocol allows agents to quantify the probability that their privacy will be preserved and abstain in the case that probability is not sufficient enough. Additionally, we identify the type 2 attack and present a solution for it. We also provide experimental evaluation of our proposal.

A number of privacy preserving reputation systems are based on the premise that a trusted hardware module is present at every agent. The systems that fall under this category include [12], [19], [3]. A system by Kinateder et al [13] 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. These systems include [10], [9], [2]. However, these systems either rely on TTPs or centralized constructs, such as the "bank" in [2]. In contrast, our reputation protocols are decentralized.

VIII. 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 seed agents, and most importantly the ability of feedback providers to themselves select trustworthy agents whom they want to share intermediate information with. Being able to select trustworthy partners allows an agent to maximize the probability that its privacy will be preserved. Additionally, an agent is able to quantify this probability and abstain from contributing its private feedback if the privacy guarantee does not satisfy the desired threshold. Our protocol that is resilient against non-disruptive malicious adversaries has log-linear communication complexity. This makes the protocol more efficient than comparable protocols discussed in the literature. Moreover, our protocols are fully decentralized and do not require any specialized hardware. An experiment conducted on data from the real and large web of trust of Advogato.org demonstrates that the protocols preserve the privacy of agents with significant success.

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 Univ., 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] E. Gudes, N. Gal-Oz, and A. Grubshtein. Methods for computing trust and reputation while preserving privacy. In *Proc. of the IFIP Conf. on Data and Applications Security*, 2009.

[9] 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.

[10] 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.

[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)*, 2003.

[12] 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.

[13] M. Kinateder, R. Terdic, and K. Rothermel. Strong pseudonymous comm. for p2p reputation systems. In *Proc. of the 2005 ACM Symp. on Applied Computing*, 2005.

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

[15] 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.

[16] 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.

[17] 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.

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

[19] 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

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

Proof: In the *forwards* round, the tuple $(\textsc{forwards}, q, t, p, r, S, S_t)$ 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.

In the *backwards* round, the tuple $(\text{BACKWARDS}, q, t, p, r, S)$ arrives once at each agent. Each of those n agents, $a_i \in S_t$ subtracts $y^{a_i}_{(q,t,p)}$ 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 $y^{a_i}_{(q,t,p)}$ and added x_i to r, thus $r = \sum_{i=1}^{n} l_{a_i t} + \sum_{i=1}^{n} y^{a_i}_{(q,t,p)} - \sum_{i=1}^{n} y^{a_i}_{(q,t,p)} + \sum_{i=1}^{n} x_i$, or $r_t = r = \sum_{a \in S_t} l_{at} + x$. ■

Theorem 2. *If the agents who participate in Protocol 1 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:

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

From equations 3, 4, and 5 we have:

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

Subtracting equation 7 from equation 6 we get: $l_{at} + y_{(q,t,p)} - y_{(q,t,p)} = c_6 - c_7 \implies l_{at} = c_8$.

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 1 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 1 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: $\sum_{k \in A-a} l_{kt} + l_{at} + x_d = c_1$, 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: $\sum_{k \in A-a} l_{kt} + l'_{at} + x_{d'} = c_2$, 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.

We can conclude that: $c_1 - l_{at} - x_d = c_2 - l'_{at} - x_{d'} \implies l_{at} - l'_{at} = c_3 - x_d + x_{d'}$.

This 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 $P(d \in Z_a)$. ■