Providing Witness Anonymity in Peer-to-Peer Systems

Bo Zhu
Center for Secure Information Systems
George Mason University
4400 University Drive
Fairfax, VA 22030-4444
bzhu@gmu.edu

Sanjeev Setia
Department of Computer Science
George Mason University
4400 University Drive
Fairfax, VA 22030-4444
setia@cs.gmu.edu

Sushil Jajodia
Center for Secure Information Systems
George Mason University
4400 University Drive
Fairfax, VA 22030-4444
jajodia@gmu.edu

ABSTRACT

In this paper, we introduce the concept of *witness anonymity* for peer-to-peer systems. Witness anonymity combines the seemingly conflicting requirements of anonymity (for honest peers who report on the misbehavior of other peers) and accountability (for malicious peers that attempt to misuse the anonymity feature to slander honest peers). We propose the *Secure Deep Throat* (**SDT**) protocol to provide anonymity for witnesses of malicious or selfish behavior to enable such peers to report on this behavior without fear of retaliation. On the other hand, in SDT the misuse of anonymity is restrained in such a way that any malicious peer that attempts to send multiple claims against the same innocent peer for the same reason (i.e., the same misbehavior type) can be identified. We also describe how SDT can be used in two modes. The active mode can be used in scenarios with real-time requirements, e.g., detecting and preventing the propagation of peer-to-peer worms, whereas the passive mode is suitable for scenarios without strict real-time requirements, e.g., query-based reputation systems. We analyze the security and overhead of SDT and present countermeasures that can be used to mitigate various attacks on the protocol. Our analysis shows that the communication, storage, and computation overheads of SDT are acceptable in peer-to-peer systems.

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*; C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*

General Terms

Security

Keywords

Privacy, Peer-to-Peer Systems, Witness Anonymity, k-Times Anonymous Authentication

CCS'06, October 30–November 3, 2006, Alexandria, Virginia, USA.

1. INTRODUCTION

One of the fundamental challenges in peer-to-peer systems is how to build trust relationships between peers. In large-scale peer-to-peer systems, the chance that a given pair of peers will have repeated interactions with each other is small. Hence, two interacting peers may not have prior experience and knowledge of each other, and need a way to evaluate the risk involved with a transaction. To address this issue, several research studies [12, 13, 19, 33] have proposed mechanisms for building and using reputation-based trust models in peer-to-peer systems.

In these systems, a peer is assigned a trust value or reputation based on a trust metric. Although various systems differ in how this metric is defined, in general, the trust value associated with a peer is calculated based on the feedback provided by other peers. Peers rate the performance or behavior of another peer based on their previous interactions. When a peer encounters a new peer, it can query the network for trust ratings of that peer, and then based on the received feedback it can decide whether to proceed with the transaction.

There are three important requirements for these trust management systems:

- **Reliability:** A peer that issues a query for the trust ratings of another peer should be able to compute the *true* trust value despite the presence of malicious peers.
- **Anonymity:** It should not be possible to identify the peers who provide feedback in the form of their trust ratings for another peer. This is especially important when the feedback is negative in nature, since it could lead to retaliation.
- **Accountability:** It should be possible to identify malicious peers who attempt to misuse the anonymity property to manipulate the trust value computed for a peer. For example, without accountability, a malicious peer may submit multiple negative claims anonymously about the trustworthiness of another peer.

Most previous work on trust management in peer to peer systems has focused on the first requirement above, i.e., how to reliably compute trust values in the presence of malicious peers. However, the issues of anonymity and accountability have not received much attention. Although some of these

systems [13, 29] make some provisions for peer anonymity, these provisions are easily circumvented as discussed in Section 9.

We observe that the issues of anonymity and accountability are closely coupled. Anonymity without accountability can be easily abused by malicious peers, so any system that enables peers to anonymously provide feedback on another peer must also include a mechanism for being able to identify peers that misuse the anonymity feature. We introduce the term *witness anonymity* to refer to this combination of seemingly conflicting requirements, i.e., identity anonymity for honest peers and accountability for misbehaving peers.

The major goal of our work is to show how peer-to-peer trust management systems can be extended to provide *witness anonymity*. The primary motivation for witness anonymity in peer-to-peer systems is similar to the need for whistleblower anonymity in real life. Without witness anonymity, peers that report on the misbehavior (e.g. false transactions) of other peers by submitting low trust ratings in response to a query, can become targets for retaliation [3]. This could take the form of tit-for-tat behavior, in which malicious peers intentionally lower their own trust rating for an honest peer. In extreme scenarios, peers giving a low rating to another peer may become targets for electronic attacks, e.g., denial of service, and even physical attacks.

Another important motivation for witness anonymity is simply to preserve the privacy of peers participating in the peer-to-peer trust management system. As a specific example, a peer A that responds to a query for the trust ratings of another peer B may not want to make public the fact that it has had previous interactions with B.

The third motivation for witness anonymity is to hide the trust topology of the peer-to-peer system from malicious parties. In particular, when the reputation system uses transitive trust (e.g. [19]) without witness anonymity the trust topology of the peer-to-peer system becomes public knowledge and can be exploited by malicious parties. For example, an adversary that wishes to launch an attack on a peer A may choose to compromise another peer B, if it knows that A has a high degree of trust in B. It can then exploit this trust by using B to launch an attack on A.

In this paper, we present a protocol called the *Secure Deep Throat* (**SDT**)[1] for providing witness anonymity in peer-to-peer systems. To the best of our knowledge, SDT is the first protocol that can support both aspects of witness anonymity, i.e., identity anonymity for honest peers, and accountability for peers that attempt to misuse the anonymity feature. SDT ensures the anonymity of a peer as long as she sends out only one feedback message per peer per malicious or selfish operation type. However, if a peer sends multiple claims against the same peer for the same reason, SDT includes a tracing mechanism to identify the peer.

The SDT protocol is based on the k-times anonymity authentication protocol proposed by Nguyen and Safavi-Naini [24]. We adapt their protocol to match the distributed and decentralized nature of peer-to-peer systems. In addition, we describe how SDT can be used in two modes: active mode and passive mode. The active mode is used in scenarios with real-time requirements for detecting malicious behavior, e.g. for detecting and preventing the propagation of peer-to-peer worms [34]. The passive mode is suitable for scenarios that do not have strict real-time requirements, e.g. query-based reputation systems [12].

The rest of the paper is organized as follows. In Section 2, we define the goals of SDT and analyze possible solutions based on available cryptographic techniques. In Section 3, we present the SDT framework including the system, adversary, and network models assumed in its design. In Section 4, we present the four procedures of the SDT protocol, i.e. *setup*, *registration*, *claim broadcasting*, and *public tracing*, under the active mode. In Section 5, we discuss two approaches for trading security for efficiency. We analyze the security and anonymity-related properties achieved in SDT and present the countermeasures to the collusion, Sybil and denial-of-service attacks in Section 6 and Section 7 respectively. In Section 8, we analyze the storage, communication, and computation costs of the SDT protocol. Related work is presented in Section 9. Finally, Section 10 contains our conclusions.

2. THE GOALS OF OUR WORK AND POSSIBLE SOLUTIONS

2.1 Design Goals

To achieve both anonymity and security in sending feedback (e.g., about the reputation of a peer) and event reports (e.g., about the misbehavior detected), a protocol that supports witness anonymity in peer-to-peer networks is expected to provide the following security and anonymity-related properties. In this paper, the terms *claim*, *feedback message*, and *report* are synonyms and used interchangeably. Similarly, the terms *user* and *peer* are synonyms and used interchangeably.

- **Identity Anonymity** Even if all the adversaries collude with each other, they will not be able to identify the source of an anonymous claim, i.e., the witness, as long as she sends the claim only once per adversary per type of malicious or selfish operation.

- **Backward Anonymity** The anonymity of a user that has acted as a witness is maintained even if other members in the network are compromised at a later time.

  Backward anonymity addresses the situation where users other than a witness are compromised. In this situation, adversaries can obtain the secrets known to the compromised users, e.g., the secret keys of the compromised users and the claims that were sent by witnesses and are stored by the compromised users. Due to backward anonymity, however, adversaries cannot use this information to deduce the identity of the user that has acted as a witness.

  Clearly, if a witness herself is compromised, her anonymity cannot be maintained. Using the secret key of the witness and the claims sent by the witness, adversaries can easily determine whether the compromised user has acted as a witness in the past.

- **Traceability** If a malicious or selfish user sends multiple claims against the same user for the same type of malicious or selfish operation, she will be identified.

[1]It is well known that the information from the anonymous source dubbed "Deep Throat" played an important role in helping unravel the Watergate scandals in the early 1970s.

- **Non-Slanderability** A good user can never be framed by adversaries, even if all of them collude with each other. This property is optional, and it is a must only when we assume there is a distributed trust mechanism in place so that the number of adversaries in the peer-to-peer system is lower than a threshold at any given time.

In addition, it is desirable that the new protocol can provide other properties such as ***efficiency*** (i.e., the storage, communication, and computation cost should be acceptable) and ***decentralization*** (i.e., an online central party is not necessary for the protocol).

2.2 Available Cryptographic Techniques for Providing Witness Anonymity

2.2.1 Blind Signature and Untraceable Electronic Cash

Blind signature and its applications to untraceable electronic cash cannot be used for our purpose. In untraceable electronic cash schemes [11, 7], the problem of detecting double-spending is similar to the problem of detecting multiple claims regarding the same peer in reputation systems. However, these schemes require the use of an online trusted party such as a bank. However, we cannot assume the existence of an online trusted party in peer-to-peer systems. Even if we assume that there is such a trusted party and it would be online periodically, a few problems still remain. One problem is that during the periods that the trusted party is off-line a group of innocent peers P might have been deleted from the friend lists of another group of peers O. This can occur if a single malicious peer has sent numerous claims to slander the members of P and these claim are accepted by the peers in O. Communication and computation overhead is another issue. The online trusted party could become the bottleneck of the verification. Most importantly, the content of the claim in a reputation system such as "the peer with the identity A is a malicious user because it has misbehavior of type I" cannot be predetermined. However, untraceable electronic cash schemes are not flexible enough to handle claims that are not predetermined.

2.2.2 Group and Ring Signatures

A typical group signature scheme must satisfy *unforgeability*, *anonymity*, *unlinkability*, *exculpability*, *traceability*[2], and *coalition-resistance* [2]. In group signature schemes [10, 2, 6], each group member can sign documents on behalf of the whole group. The receiver of a signed document can verify the signature to ensure that the document is signed by a group member. However, no one except the trusted group manager can recover the exact identity of the signer. The major weakness of group signature schemes is that in order to solve possible disputes at a later time, they give the group manager the unnecessary ability to trace any user even if there are no disputes. In other words, group signature schemes provide only partial anonymity to the signer, and are not suitable for scenarios where users have privacy concerns, e.g., peer-to-peer networks.

[2]Note that traceability as provided by group signatures and ring signatures is different from the one defined in Section 2.1.

Ring signatures[21, 8] can be viewed as a variant of group signatures without the traceability property. In ring signature, the group consists of only users without a group manager. Consequently, there is no way to revoke the anonymity of the signer, even if there is a dispute. That is to say, ring signatures can provide full anonymity, but fail to ensure accountability.

2.2.3 k-Times Anonymity Authentication

k-times anonymous authentication was first proposed by Teranishi, Furukawa, and Sako [31]. The entities in the k-times anonymous authentication scheme include a *group manager*, *users*, and *application providers*. This scheme provides stronger anonymity to a user in the sense that even the group manager cannot trace the identity of the user, as long as she follows the rule, i.e. authenticating herself and using the service provided by application providers (e.g., trial browsing of content) fewer than k times, where k is a predetermined number. In [31], the group membership is decided by the group manager, and application providers have no control over giving users access to their services. Nguyen and Safavi-Naini [24] proposed dynamic k-times anonymous authentication to enhance the privileges of application providers. In [24], application providers can select their user groups and grant or revoke access to users independently.

Compared to blind signature, group signature, and ring signature, k-times anonymous authentication is more suitable for achieving our design goals. It can provide *identity anonymity*, *backward anonymity*, and *traceability* at the same time.

3. THE FRAMEWORK OF THE SECURE DEEP THROAT PROTOCOL

In this section, we describe the system, adversary, and network models assumed in the design of the SDT protocol. We then provide an outline of the SDT protocol before presenting a detailed description in Section 4.

3.1 System Model

Our work assumes a peer-to-peer system where there is no online centralized *Trusted Third Party* (**TTP**). There are two types of entities that participate in SDT, namely the *Offline*[3] *Group Manager* (**OGM**) and *users*. We assume that there are a large number of users in the network and that a small fraction of the users are adversaries. The number of all the users and adversaries are denoted as n and t_a, respectively.

Unless explicitly specified, we assume the existence of a distributed trust mechanism, e.g., one of the protocols proposed in [22, 20, 35]. In these protocols, the number of adversaries in the system during a given time period, e.g., the key refresh period, is less than a threshold denoted as t. In addition, any group of t or more good users can cooperate together to provide certification services, e.g., assigning a certificate to a new user or revoking the certificate of an existing user when there are t or more claims against this user.

[3]The OGM is involved only in the first two procedures, which are assumed to be executed offline in SDT.

3.2 Adversary Model

We consider two types of adversaries – malicious users and selfish users. Malicious users attempt to disable the normal functionalities of the network. They may sniff, modify, or replay network communication messages. Selfish users take advantage of services provided by other peers without contributing back. For example, a selfish user may refuse to forward packets for other users.

For both types of adversaries, we assume that they will collude together to maximize the effectiveness of their attacks. We assume that it is possible to detect malicious and selfish operations, e.g., via worm detection software or by reviewing previous transactions with a specific peer. The details of how to detect such operations are beyond the scope of this paper.

3.3 Network Model

We assume the existence of a Mixnet-based anonymous communication system [27, 28, 15] so that an adversary cannot discover the identity-related information (e.g., the IP address) of the sender of a claim through traffic analysis. In this paper, we assume the existence of a mechanism for monitoring the number of claims sent by a given peer, irrespective of whether they are generated or forwarded by her.

3.4 Outline of The SDT Protocol

In both active and passive modes, the SDT protocol uses the following four procedures: *setup*, *registration*, *claim broadcasting*, and *public tracing*. We now present an outline of the operation of SDT in the active mode. (The passive mode of operation is described in Section 5.1. In this mode, the *claim broadcasting* and *public tracing* procedures operate differently.)

During the setup phase, the OGM generates a network-wide public/secret key pair, and publishes the public key. In addition, the OGM needs to publish the method of generating the tag bases, which correspond to the messages (or the meaningful content of the claim such as *user x executed a malicious behavior of type y*) to be anonymously authenticated in the *claim broadcasting* procedure, and other public information, e.g. the security parameters chosen.

Registration is done between the OGM and the user who wants to join the network. After this step, the user obtains a member public/secret key pair, and the OGM adds the user's identification and public key to an *identification list* (**LIST**). A user who has completed the *registration* procedure is called a member of the network.

Once a user detects any malicious or selfish behavior, and would like to act as a witness, i.e., broadcast a claim bearing witness to the misbehavior, she first calculates a tag base using the method published in the *setup* phase. Then she generates an anonymous claim using this tag base, and broadcasts the claim through a Mixnet-based anonymous communication system. The claim will be accepted by other users only if the sender (i.e., the witness) is a member of the network and the claim is generated by her secret key assigned in the *registration* procedure. Users that receive this claim store it into a local claim log, if they have not seen the same claim before. Otherwise, they simply drop the claim. Then the claim is forwarded again using the Mixnet-based anonymous communication system. Note that "the same claim" is different from "the claim against the same user for the same reason". In SDT, due to a random parameter, a witness can generate different claims against the same user for the same reason, and thus these claims are considered as distinct claims. (See Section 4.2 for more details.)

Using only the public information (i.e. LIST) and the claim log, anyone can do public tracing. This procedure outputs a user ID i or **NO-ONE**, which respectively mean "the user i has tried to misuse witness anonymity by sending multiple claims against the same user for the same reason" and "the public tracing procedure cannot find malicious entities misusing witness anonymity".

4. THE SECURE DEEP THROAT PROTOCOL

In this section, we present the details of the four procedures of the SDT protocol in the active mode of operation.

4.1 Preliminaries

4.1.1 Notation and Terminology

Let $\mathbb{N}$ and $\mathbb{Z}_p$ denote the set of natural integers and the set of natural integers in the range from 0 to $p-1$, respectively. A function $f : \mathbb{N} \to \mathbb{R}^+$ is called *negligible*, if for every positive number α, there exists a positive integer κ_0 such that for every integer $\kappa > \kappa_0$, it holds that $f(\kappa) < \kappa^{-\alpha}$. Let PT denote polynomial-time, and PPT denote probabilistic PT. For a set X, "$x \in_U X$" denotes that x is an element randomly and uniformly chosen from X. Let $\mathcal{H}_X$ denote a one-way hash function from the set of all finite binary strings $\{0,\ 1\}^*$ onto the set X.

4.1.2 Bilinear Groups

Let $\mathbb{G}_1$, $\mathbb{G}_2$ be additive cyclic groups generated by P_1 and P_2, respectively, whose orders are a prime p, and $\mathbb{G}_T$ be a cyclic multiplicative group with the same order p. Suppose there is an isomorphism $\Psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\Psi(P_2) = P_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing with the following properties:

- **Bilinearity:** $e(aP,\ bQ) = e(P,\ Q)^{ab}$ for all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, $a,\ b \in \mathbb{Z}_p$
- **Non-degeneracy:** $e(P_1,\ P_2) \neq 1$
- **Computability:** There is an efficient algorithm to compute $e(P,\ Q)$ for all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$

For simplicity, hereafter we set $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$, but the proposed schemes can be easily modified for the general case when $\mathbb{G}_1 \neq \mathbb{G}_2$. In the rest of this paper, for a group $\mathbb{G}$ of prime order, we denote the set $\mathbb{G}^* = \mathbb{G} \backslash \{\mathcal{O}\}$, where $\mathcal{O}$ is the identity element of the group. We define a Bilinear Pairing Instance Generator as a PPT algorithm $\mathcal{G}$ that takes as input a security parameter 1^κ and returns a uniformly random tuple $t = (p,\ \mathbb{G}_1,\ \mathbb{G}_T,\ e,\ P)$ of bilinear pairing parameters, including a prime number p of size κ, a cyclic additive group $\mathbb{G}_1$ of order p, a multiplicative group $\mathbb{G}_T$ of order p, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ and a generator P of $\mathbb{G}_1$.

4.2 Procedures of The Protocol

4.2.1 Setup

Given as input a security parameter 1^κ, the Bilinear Pairing Instance Generator generates a tuple $(p,\ \mathbb{G}_1,\ \mathbb{G}_T,\ e,\ P)$ as in Section 4.1.2. The OGM selects $P_0,\ H \in_U \mathbb{G}_1$, $\gamma \in_U \mathbb{Z}_p^*$, and sets $P_{pub} = \gamma P$ and $\Delta = e(P,\ P)$. The group public and secret keys are $gpk = (P,\ P_{pub},\ P_0,\ H,\ \Delta)$ and $gsk = \gamma$, respectively. The identification list of group members denoted as LIST is initially empty.

An important task of the *setup* procedure is to define the way of generating the tag bases, which are used in the *claim broadcasting* procedure to create anonymous claims. Various methods can be employed to generate the tag bases. In this paper, we use the method defined in Equation (1).

$$(T_j,\ \check{T}_j) = \mathcal{H}_{\mathbb{G}_T \times \mathbb{G}_T}(Type_{ad},\ ID_{ad},\ MAX_{Claim},\ j), \quad (1)$$

for $j = 1, \ldots,\ MAX_{Claim}$, where

$Type_{ad}$ — denotes the type of event reported in the claim, e.g., accusing a user of refusing to forward a packet. $Type_{ad} \in TYPE$, where $TYPE$ is the set of all the event or claim types supported in SDT.

ID_{ad} — denotes the identity of the user that the claim is accusing.

MAX_{Claim} — the maximum number of claims that a user can send to accuse the same user for the same type of operations.

j — The number of claims, including this one, that have been sent against a user with the identity ID_{ad} because of misbehavior of type $Type_{ad}$.

In this paper, we set $MAX_{Claim} = 1$. Namely, no one should send multiple claims against the same user for the same reason. Otherwise, she will be detected and identified. As such, there is only one tag base (i.e., $j = 1$) per type of misbehavior per user.

4.2.2 Registration

A user U_i can join the network as follows.

a. User U_i interacts with the OGM to determine her identity in the network. This can be done in various ways. In one possible approach, U_i selects an identity and forwards it to the OGM, who checks the availability of this identity. If it has been chosen by others, U_i will have to select another identity and repeat this process until the identity picked is available. Let the identity of U_i be denoted by i.

b. User U_i selects $x',\ r \in_U \mathbb{Z}_p^*$, and sends a commitment $C' = x'P + rH$ of x' to the OGM.

c. The OGM sends $y,\ y' \in_U \mathbb{Z}_p^*$ to U_i.

d. User U_i computes $x = y + x'y'$ and $(C, \beta) = (xP, \Delta^x)$. Next, U_i sends $(C,\ \beta)$ to the OGM with a standard proof $Proof_1$ (please refer to [9] for this proof) to show that C is correctly computed from $C',\ y,\ y'$, and that U_i knows the value of x satisfying $C = xP$.

e. The OGM verifies that the proof is valid, and that $e(C,\ P) = \beta$ is satisfied. If the verification succeeds, the OGM adds a new entry $(i,\ \beta)$ to the LIST. The OGM then generates $a \in_U \mathbb{Z}_p^*$ different from all corresponding previously generated values, computes $S = \frac{1}{\gamma + a}(C + P_0)$, and sends $(S,\ a)$ to user U_i. In addition, the OGM selects s items from the LIST (not including $(i,\ \beta)$), and forwards them to U_i. All the LIST items are signed using the OGM's secret key, and thus can be verified by any user.

f. User U_i confirms that $e(S,\ aP + P_{pub}) = e(C + P_0,\ P)$ is satisfied. The new member U_i's secret key is $usk = x$, and her public key is $upk = (a,\ S,\ C,\ \beta)$. U_i also verifies the s LIST items from the OGM.

4.2.3 Claim Broadcasting

Each user maintains two claim databases locally. One is a private claim database denoted as DB_{PC}. DB_{PC} records the claims that have been sent by the user herself. The other database is a common claim database denoted as DB_{CC}. DB_{CC} records the claims that she receives from other users.

Once a user detects any malicious or selfish operation that is a member of the set $TYPE$, she first examines DB_{PC} to check whether she has sent a claim against the malicious or selfish user for the same reason in the past. If the witness cannot find any entry with the same $(ID_{ad},\ Type_{ad})$ pair, she generates a new anonymous claim and stores it in DB_{PC}. Otherwise, the witness will not generate another claim.

The anonymous claim is generated via the following steps. The witness first selects a random number $l \in_U \mathbb{Z}_p^*$. Then she computes the tag base $(T_1,\ \check{T}_1)$ following the method published in the *setup* procedure. Next, she calculates the tag $(\Gamma,\ \check{\Gamma}) = (T_1^x,\ (\Delta^l \check{T}_1)^x)$ using the tag base. Finally, the witness broadcasts $(\Gamma,\ \check{\Gamma})$ with a proof $Proof_2$ (please refer to [25] for this proof) using a Mixnet-based anonymous communication system. The format of the anonymous claim is $[Type_{ad},\ ID_{ad},\ \Gamma,\ \check{\Gamma},\ l,\ Proof_2]$.

When a user receives the claim, she first checks whether there is an entry corresponding to the same claim in her DB_{PC} and DB_{CC}. If so, she simply drops the claim. Otherwise, the receiver computes T_1 according to equation (1), and checks whether $Proof_2$ is valid. If the proof is invalid, the user ignores the claim. Otherwise, she records the claim in DB_{CC}, and forwards the claim using the Mixnet-based anonymous communication system.

Once any user U_i finds t claims against the same user U_a for the same type of malicious or selfish operation (i.e., with the same ID_{ad} and $Type_{ad}$), she first checks whether all these claims are from distinct sources. Let the set of these t claims be denoted by V. Users can easily judge whether the claims in V are from distinct sources by comparing their Γ's. If any pair of claims in V has the same Γ, it means that a malicious user sent multiple claims against the same user for the same reason. She can be traced using the method presented in Section 4.2.4, and these claims are removed from V. If there are still t claims after this check, U_i can assert that U_a is an adversary who has executed the type of malicious or selfish operation indicated in all these claims. A message including the t claims collected is generated by U_i and broadcast to the network. Any user receiving the message can verify the claims included and agree with U_i's judgment on U_a if all the t claims are valid.

4.2.4 Public Tracing

Each member in the network can trace the identity of an adversary, if the adversary sends multiple claims against the same user for the same type of misbehavior. The *public tracing* procedure is as follows.

a. Look for two entries $[Type_{ad},\ ID_{ad},\ \Gamma,\ \check{\Gamma},\ l,\ Proof_2]$ and $[Type'_{ad},\ ID'_{ad},\ \Gamma',\ \check{\Gamma}',\ l',\ Proof'_2]$ in the DB_{PC} and DB_{CC}, such that $Type_{ad} = Type'_{ad}$, $ID_{ad} = ID'_{ad}$, $\Gamma = \Gamma'$ and $l \neq l'$, and that both $Proof_2$ and $Proof'_2$ are valid. If no such entry can be found, output **NO-ONE**.

b. If such a pair of entries is found, compute β as $\beta = (\frac{\check{\Gamma}}{\check{\Gamma}'})^{\frac{1}{l-l'}} = [\frac{(\Delta^l \check{T}_1)^x}{(\Delta^{l'} \check{T}'_1)^x}]^{\frac{1}{l-l'}} = \Delta^x$.

c. Search for a pair $(i,\ \beta)$ in the part of LIST stored locally. If such a pair is found, broadcast a message including this pair together with the two entries as proofs to disclose the malicious user's identity.

d. If such a pair cannot be found locally, generate a message including β and the two entries and broadcast it to the network. If any member receiving this message finds a pair $(i,\ \beta)$ in her local LIST, she repeats step a-c to verify the proofs and to disclose the malicious user's identity.

5. TRADEOFF BETWEEN SECURITY AND EFFICIENCY

In this section, we discuss how the security requirements of SDT can be relaxed in return for higher protocol efficiency, i.e., lower communication, computation, and storage costs.

5.1 Passive Mode Operation of SDT

The active mode of SDT is designed to support real-time detection of misbehavior by adversaries. In other words, a malicious or selfish peer is disclosed and expelled from the network as soon as t good peers detect her misbehavior, e.g. proliferating worms [34] or Trojans [23]. However, in situations where the real-time requirement is not critical, e.g. query-based reputation systems, SDT can operate in the passive mode to achieve better efficiency. More specifically, to reduce the communication, computation, and storage overheads, peers accept a delay in the disclosure and banishment of malicious or selfish peers.

To support the passive operation mode, modifications are required to the *claim broadcasting* and the *public tracing* procedures discussed in Section 4, as described below.

In the *claim broadcasting* procedure, when a peer detects a malicious or selfish operation performed by a peer A, instead of generating and sending an anonymous claim immediately, she keeps silent until she receives a query for the trust ratings of A. For example, another peer B who wants to know the trust ratings of A will broadcast a query to collect feedback from other peers. On receiving this query, each good peer who has witnessed A's misbehavior generates an anonymous claim in the same manner as in the active mode, and sends it through a Mixnet-based anonymous communication system. After collecting a sufficient number of feedback messages (e.g., t negative claims), B calculates the reputation of A based on the feedback received according to the trust metric defined. If B receives t or more claims from distinct peers accusing of A for the same reason, she knows that A is a malicious or selfish peer. As a result, B will refuse to participate in transactions with A, and inform other peers about this fact by broadcasting a message containing all these claims.

To prevent an adversary from sending multiple claims to slander or boost other peers, B needs to perform the *public tracing* procedure over all the claims received using the procedure described in Section 4.2.4.

In a peer-to-peer system, it is likely that there will be multiple peers who are interested in the trust ratings of A. Thus, when the witness receives a query on A for the second time, she should not generate a new claim. Otherwise, her anonymity will be compromised because of sending multiple claims against A. Instead, she can locate the claim regarding A in her DB_{PC}, which was generated in response to the first query process, and use it as the reply. Note that the same claim is counted only once even if any querying peer receives multiple copies of it.

The discussion so far has mainly focused on negative feedback. However, SDT can be used for positive feedback as well. For example, let N_p and N_n denote the numbers of positive and negative claims collected, respectively. A simple trust metric can be defined as $\frac{N_p}{N_p+N_n}$. However, we argue that positive feedback should not be considered in the active mode. Generally, the larger the number of positive claims received for a given peer, the higher the trust rating of the peer. However, a major difference between positive and negative feedback is that there is no threshold such that a peer is fully trusted if the number of positive claims regarding the peer is larger than this threshold. Suppose that such a threshold exists and is denoted as t'. An adversary A can cheat the system by first conducting t' transactions with distinct peers honestly. Thereafter A will be fully trusted and can cheat other peers in its subsequent transactions. This attack is a type of strategic oscillation attack [30]. Since we assume that most of peers in the network are benign and the claims are flooded in the active mode, counting positive feedback will result in large overheads. Fortunately, the goal of the active mode of operation is to detect malicious behavior in real-time, and thus it is sufficient to consider only negative feedback. In contrast, the goal of the passive mode of operation is to obtain an accurate trust value about the peer queried. Thus, we should consider both positive and negative feedback in the passive mode.

5.2 Probabilistic Forwarding

An approach for improving the efficiency of SDT in the active mode is for each peer receiving a claim to forward the claim with a probability p_f, instead of flooding claims to the whole network and storing them on each peer. Intuitively, the lower the probability p_f, the smaller the average number of peers storing a claim (denoted as n_s), and the lower the probability that at least one good peer stores t or more claims against the same adversary for the same type of misbehavior (denoted as p_r).

In this approach, the tradeoffs in security include: (i) a larger number of witnesses are needed before the adversary is disclosed (ii) given an upper bound on the number of witnesses needed, there is a non-zero probability that an adversary will escape disclosure. Let t_r denote the number of witnesses detecting an adversary's misbehavior and generating claims against her. Let p_d denote the security

requirement, i.e., the lower bound of p_r. We first analyze the relationship between n_s and t_r while ensuring a high p_d, and then discuss how to select an appropriate p_f.

Assuming that an adversary engages in malicious or selfish behavior while interacting with peers that are uniformly distributed in the network, the claims against this adversary are generated, forwarded, and stored with a uniform probability. Therefore, the probability that a peer stores a given claim is n_s/n. Thus, we have $p_r = (n - t_a) \cdot (\frac{n_s}{n})^t \cdot C_{t_r}^t \geq (n - t + 1) \cdot (\frac{n_s}{n})^t \cdot C_{t_r}^t \geq p_d$.

Figure 1 shows the average number of peers required to store a claim (i.e. n_s) so that a high-level security (i.e. $p_d = 0.9999$) is achieved under different t_r's. We notice that n_s declines very fast when t_r increases. For example, when $t_r = 1.1 \cdot t$, i.e., requiring 10% more witnesses, n_s is around 24% to 27% lower than the case when $t_r = t$. Therefore depending on the level of security required and potential number of adversaries in the network, we can find an optimal n_s that achieves a good balance between security and efficiency. Let this optimal n_s be denoted by n_s^o.

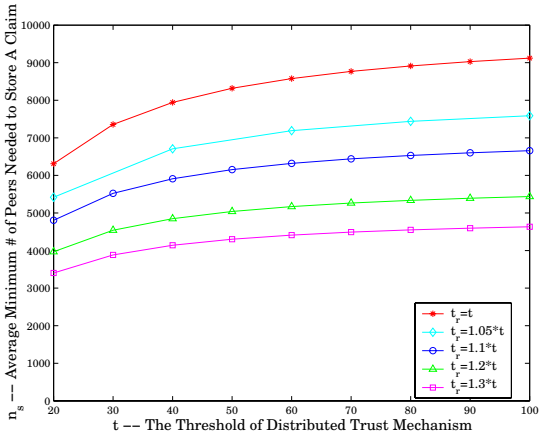

Figure 1: n_s under different t_r's ($n = 10000$, $p_d = 0.9999$)

After determining n_s^o, we need to compute an appropriate probability of forwarding such that if every peer forwards the claims received with a probability p_f, the number of peers storing a claim is approximately equal to n_s^o. Let d denote the average degree/number of neighbors of a peer. When $d \cdot p_f < 1$ we have:

$$n_s < d + d^2 \cdot p_f + \cdots + d^{i+1} \cdot p_f^i + \cdots \approx \frac{d}{1 - d \cdot p_f} \quad (2)$$

Finally, we can compute the range of p_f as follows:

$$\frac{n_s^o - d}{n_s^o \cdot d} < p_f < \frac{1}{d} \quad (3)$$

In peer-to-peer systems, peers can join and leave the system at will. For both the active and passive modes of SDT, the trust value of a peer A is calculated based on the trust ratings collected from all the online good peers who have interactions with her. Therefore as long as there are at least t good peers online that detect A's misbehavior, A will be detected and expelled from the network. However, when we use the probabilistic forwarding approach, the dynamic nature of peer-to-peer networks does have an impact on the parameter setting. More specifically, let p_{on} denote the probability that a peer is online. Equations (2) and (3) should be modified by replacing d with $d \cdot p_{on}$.

6. SECURITY AND ANONYMITY-RELATED PROPERTIES ACHIEVED IN SDT

In this section, we examine how the security and anonymity-related goals defined in Section 2.1 are achieved in SDT.

6.1 Identity Anonymity

Generally, the possible methods of breaking identity anonymity as defined in Section 2.1 can be divided into two categories: traffic-based analysis and protocol-based analysis.

The idea behind traffic-based analysis is to detect common information among sniffed network traffic, assuming that any two packets are transferred along the same route if they have information in common. The "common information" could be either identical content (e.g., the same sequence number) in sniffed packets, or identical time consumed in handling sniffed packets (i.e., time analysis). In SDT, to prevent traffic analysis [26], the witness sends her claim through a Mixnet-based anonymous communication system [27, 28, 15] so that adversaries cannot discover the identity of the sender. Previous research [14, 16] shows that reputation systems can improve the performance of Mixnets. As such, we expect that the combination of SDT-based reputation systems and Mixnet-based anonymous communication systems is mutually beneficial and can enhance the effectiveness of both systems.

In protocol-based analysis, adversaries try to deduce the identity of the sender by analyzing the semantic context of messages. For example, to ensure that the receiver can verify the signature, the sender may include her public key in the packet. If that is the case, adversaries can easily discover the identity of the sender. In SDT, there is no public key or identity-related information in a claim, and the verification process is based on the zero knowledge proof. Given that the *Discrete Logarithm Problem* (**DLP**) is hard, SDT is robust against brute force attacks. More specifically, given T_1 and Γ, there is no PPT function which adversaries can use to find the secret of the witness (i.e., x) and thus deduce her identity.

6.2 Backward Anonymity

Most peer-to-peer systems can be viewed as overlay networks on top of another network, e.g. the Internet. The security of peer-to-peer systems will certainly be affected by the security of the underlying network. In addition, due to its open nature a peer-to-peer system is also vulnerable to attacks within the system, e.g. worms [34]. Therefore, our protocol should be resistant to possible compromises of peers. In SDT, a claim itself does not provide information disclosing the identity of the sender. Even if adversaries compromise multiple peers and collect additional claims, they cannot differentiate the claims sent by a peer from those sent by others. Thus, compromising more peers does not increase the probability of deducing the identity of the sender, i.e. the witness. Backward anonymity is ensured even when the OGM cooperates with adversaries, because in the *registration* procedure the OGM is convinced that a peer is holding the secret key corresponding to a given public key through zero knowledge proof and has no information about the secret key of that peer.

6.3 Traceability

In SDT, if an adversary sends multiple claims against a user for the same reason, she is identified by the *public tracing* procedure. To successfully disclose an adversary misusing witness anonymity, good peers need to find a valid record of this adversary, i.e., the LIST item containing the identity and public key of the adversary. In SDT, each item in the LIST has s copies distributed within the peer-to-peer system. As shown in Section 8.1, the probability that all the copies of a given item are controlled by the adversary group is tiny, even when s is very small. If we assume that such a probability is negligible, any adversary sending multiple claims against the same user for the same reason will be traced, and her identity will be disclosed.

6.4 Non-Slanderability

In this paper, we assume the existence of a distributed trust mechanism, e.g., one of the protocols proposed in [22, 20, 35]. Together with the *traceability* property provided by SDT, the maximum number of claims that the adversary group can send against the same good user for the same reason is the number of adversaries in that group, which is assumed to be less than t. In other words, adversaries cannot collect enough number of claims, i.e. t or more claims, to convince others that an innocent user is malicious or selfish.

7. COUNTERMEASURES TO VARIOUS ATTACKS

In this section, we consider several attacks on the SDT protocol and present possible countermeasures.

7.1 Collusion Attacks

The effectiveness of collusion attacks is limited by the combination of the *traceability* property provided by SDT and the distributed trust mechanism. The former ensures that any adversary sending multiple claims against the same user for the same reason will be detected and traced. As a result, an adversary can only send one claim per user per type of malicious or selfish behavior. Moreover, the existence of the latter guarantees that the number of adversaries in the peer-to-peer system is less than t. Consequently, even if all the adversaries collude with each other, they cannot generate t or more claims against an innocent user for the same reason.

7.2 Sybil Attacks

In a Sybil attack [17], malicious peers can assume multiple identities, and thus can control a substantial part of the system. If such an attack is possible, it would undermine the basis of trust schemes for peer-to-peer systems, i.e., most peers are honest and each peer can vote only once when assessing a given peer. According to [17], without a logically centralized authority, Sybil attacks are always possible except under certain assumptions regarding the resources possessed by the attacker.

A straightforward solution for SDT is to let the OGM assume the responsibility of the centralized authority. More specifically, during the *registration* procedure, the OGM is responsible for binding the user registering as a member to a real-world identity, and for limiting the number of the registrations per identity to one.

Assume that in a Sybil attack there is single real-world user assuming multiple identities. We refer to this real-world user as the *Sybil source* and the users corresponding to the assumed identities as *Sybil users*. To prevent Sybil attacks when there is no centralized authority, Douceur [17] proposed the use of a scheme in which the potential Sybil users are challenged to solve some resource-intensive task that can only be accomplished by multiple real-world users but will be impractical for a Sybil source. For example, suppose that a peer A receives t claims against another peer B. When employing the direct identity validation method proposed in [17] against Sybil attacks, A should first challenge all the t witnesses who generated the claims to perform some task that can only be completed by t real-world users before asserting that B is malicious or selfish. However, in SDT, the identities of the witnesses are unknown due to the requirement of witness anonymity. Thus the only way to deliver the task to the witnesses is via a system-wide broadcast. Although this method is relatively expensive, it may not be impractical given certain assumptions regarding resource parity and coordination among peers [17].

7.3 Denial-of-Service Attacks

In the SDT protocol, *Denial-of-Service* (**DoS**) attacks can be launched against the *claim broadcasting* procedure and the *public tracing* procedure.

The DoS attacks launched in the *claim broadcasting* procedure are similar to those against reputation systems, e.g., sending numerous fake claims [30]. In SDT, such attacks are restrained by the traceability property. However, a smart adversary may choose to send one claim per innocent peer per misbehavior type. Given that the size of the network is large, she can still flood the network with a number of fake claims, although it does not help her successfully slander any innocent peer.

When SDT is in the passive mode, this type of attack can be prevented by the following method. When a peer A receives an anonymous claim against another peer B, she first checks whether it has received a query regarding B within the previous T seconds. If there is no such query, A will drop the claim. If we set a small upper bound on the number of queries that each peer can launch every T seconds, the effectiveness of DoS attacks can be reduced.

In this paper, we assume the existence of a mechanism for monitoring the number of claims sent by a peer, irrespective of whether they are generated or forwarded by her. In the active mode, each valid claim is generated after an interaction between the sender and the peer accused. Suppose that an interaction between two peers takes time t_i. Assuming that each peer accused can have only one interaction at a time, since we know that only negative feedback is considered in the active mode and the number of adversaries is assumed to be less than t, any peer that sends out claims at a rate higher than $\frac{t}{t_i}$ per unit time is suspected of launching a DoS attack. For example, given that $t = 100$ and $t_i = 300$ seconds, a peer sends claims at a rate of one claim per second can be suspected to be launching a DoS attack.

Given that we can limit the number of claims that a peer can send within a certain time period as shown above, and since the computation requirements of SDT are relatively small (even on a low-powered Pentium III platform as shown in Section 8.4), we argue that most users in today's peer-to-peer systems have sufficient computational power to handle such DoS attacks. Similarly, considering the amount of

storage available on computers [32] and the small size of each claim in SDT (as shown in Section 8.3), DoS attacks launched with the goal of exhausting the storage of a given peer are also ineffective.

To launch DoS attacks during the execution of the *public tracing* procedure, adversaries may generate fake broadcasting messages to exhaust the computational resources of other peers and the communication resources of the network. However, the effectiveness of these attacks is limited. In each message, the attacker needs to include the proofs that could be verified by other peers. Otherwise, the receiver will drop the message so that it cannot be propagated. The proofs in the pair of entries can be verified only if they are generated using the same secret key of a member in the network. As a result, the identity of the holder of this secret key will be disclosed in the *public tracing* procedure. In other words, in order to launch this attack successfully, the adversary has to expose herself or a peer that has been compromised by her.

8. THE COMMUNICATION, STORAGE, AND COMPUTATION COSTS OF SDT

8.1 Distributed Storage of the LIST

In the SDT protocol, the OGM maintains a complete identification list LIST offline. However, in most cases, e.g., to disclose the identity of malicious or selfish users during the *public tracing* procedure, users may want to access the LIST immediately instead of waiting for the OGM to be online. In addition, due the large size of the network, it would be too onerous to let each peer maintain a full copy of the LIST. Therefore, in SDT the LIST is stored in a distributed form. As discussed in Section 4.2.2, during the *registration* procedure the OGM assigns each member a portion of the LIST, which is signed by her secret key. The assignment is executed in such a way that each peer has no knowledge about the items of the LIST stored on another peer, and each item in the LIST has multiple copies stored on different members.

When there is no central party storing the LIST, it is possible that all the information about an adversary in the LIST is stored only by the adversary group. They can remove the public information of this adversary from the LIST. As a result, this adversary will not be identified even if submits multiple claims against the same peer for the same reason because the output of the *public tracing* procedure will always be always **NO-ONE**. However, we argue that, given that there are s copies of the public information for each member, and they are uniformly distributed within the network, the probability (denoted as p_{ad}) that the entry for an adversary in the LIST is stored only by the adversary group is vert small. The probability can be calculated as $p_{ad} = C^s_{t_a}/C^s_n$. In Table 1, we show the values of p_{ad} for different settings, when the network size is 10000. For example, when each member stores only around three items of the LIST, the probability that the public information of an adversary is stored only by the adversary group is less than 9.70×10^{-7}.

In fact, even if this low probability event does occur, this attack can be utilized by good peers to detect more adversaries with the help from the OGM, as illustrated by the following example. Assume that there are a group of adversaries denoted as $M = \{M_1,\ M_2, \ldots,\ M_6\}$, and s is set to 5. M_1 sends multiple claims against a good peer denoted as Q_1. Another good peer denoted as Q_2 receives these claims, and thus detects the misuse of witness anonymity according to the *public tracing* procedure. Then Q_2 broadcasts a query, and asks for helps to disclose the identity of the senders (i.e. M_1) corresponding to β calculated in the *public tracing* procedure. Unfortunately all the five copies of M_1's LIST information are stored by $M_2,\ M_3, \ldots,\ M_6$, and they refuse to respond to the query. In such a case, Q_1 may ask for the help from the OGM, who will check the LIST stored, and then broadcast a signed message claiming that members of M are malicious and at the same time disclosing their public information.

Table 1: p_{ad} under Different t_a and s ($n = 10000$)

	$t_a = 50$	$t_a = 75$	$t_a = 100$
$s = 2$	2.45×10^{-5}	5.55×10^{-5}	9.90×10^{-5}
$s = 3$	1.18×10^{-7}	4.05×10^{-7}	9.70×10^{-7}
$s = 4$	5.53×10^{-10}	2.92×10^{-9}	9.41×10^{-9}
$s = 5$	2.55×10^{-12}	2.07×10^{-11}	9.04×10^{-11}

8.2 Communication Costs

Assume that the SDT protocol is implemented by an elliptic curve or hyperelliptic curve over a finite field. To ensure the security of the protocol, the prime p should be at least 160 bits long, $\mathbb{G}_1$ should be a subgroup of an elliptic curve or a Jacobian of a hyperelliptic curve over a finite field of order p, and $\mathbb{G}_T$ should be a subgroup of a finite field of size at least approximately 2^{1024}. This will ensure that the *Decisional Bilinear Diffie-Hellman* (**DBDH**) problem [24] is sufficiently hard.

The major communication cost of the SDT protocol is for forwarding claims. By employing techniques in [18], elements of $\mathbb{G}_T$ can be compressed by a factor of three. As a result, the size of each claim is 405 bytes, excluding $Type_{ad}$ and ID_{ad}. Usually, it is sufficient to set the length of $Type_{ad}$ to one byte. As to the length of ID_{ad}, it should be set to at least $\lceil \log n \rceil$ bits. For a network with at most 10 million peers, we can set the length of ID_{ad} to three bytes, and thus the length of a claim is 409 bytes in total. Due to the relatively small size of the claim, given that we can limit the number of queries that a peer can launch and the number of claims that a peer can send within a certain time period as shown in Section 7.3, the communication overhead of SDT is acceptable in peer-to-peer systems.

If we store the LIST in a distributed manner, it leads to additional additional communication costs. Specifically, in the *public tracing* procedure, a peer may need to broadcast a message to search for pair (i, β) corresponding to a claim against a user. Intuitively, the smaller the number of LIST items that are stored on a peer, the higher is the probability that she may broadcast a message to ask for help to identify the adversary in the *public tracing* procedure. Fortunately, assuming that adversaries are rational in the sense that they may not launch attacks in which they are doomed to be caught, we argue that such cases may happen infrequently.

8.3 Storage Requirements

The storage requirements of SDT are due to (i) the cryptographic keys (ii) the items of the LIST, and (iii) the local claim databases (i.e. DB_{PC} and DB_{CC}). A user in the SDT

protocol only needs to store her public/secret key pair and the public key of the OGM. With respect to the items of the LIST, as shown in Section 8.1, they can be stored in a distributed form, and the cost is only s LIST items per user. Each LIST item is a (i, β) pair, and its size is 46 bytes after compression, given that the identity of a peer is represented with three bytes.

With respect to the storage required for the local claim database, in the passive mode, the claims stored by each peer are actually the claims that were generated by her. She does not store the claims that are forwarded. In the active mode, however, to avoid the looping of the claim message during its broadcast, each peer needs to store the claims forwarded. As shown in Section 5.2, the probabilistic forwarding approach can help reduce the storage cost. Moreover, for both the active and passive modes, the storage cost can be reduced by deleting claims after a suitable time interval.

8.4 Computation Costs

The most expensive computational operation used in the SDT protocol is the calculation of bilinear pairing. If the bilinear mapping used is the well-known Tate Pairing, using MIRACL libraries [1] (optimized using Comba method for modular multiplication) on a Pentium III 1GHz desktop, it takes 20 milliseconds to calculate a 512-bit Tate pairing (for effective 1024-bit security) without pre-computation [4]. More recent results [5] show that the Tate pairing can be evaluated up to 10 times faster than in previously reported implementations by the use of various optimizations.

9. RELATED WORK

In [14, 16], Dingledine *et al* propose to use reputation systems to increase the reliability and efficiency of various Mixnet-based applications, e.g., remailer networks and anonymous publishing. Their protocols do not provide anonymity for the witnesses in reputation systems.

XRep [13] supports weak anonymity for the peer. The reputation is bound to a pseudonym or an opaque identifier, i.e. the digest associated with a servent. However, the real IP address of the peer is still required when it replies to a voting query.

In TrustMe [29], when a peer i joins the network, a bootstrap server randomly assigns a set of peers to be its *Trust-Holding Agent* (**THA**) peers. Subsequently, any peer having interactions with peer i will send out an encrypted report which can only be decrypted by the THA peers for peer i. As such, each THA peer stores all the reports related to peer i and can thus reply to queries from other peers regarding peer i. A major weakness of TrustMe is that if any THA peer randomly chosen at the bootstrapping stage is an adversary, the identities of the reporting peers are disclosed. In other words, TrustMe does not support, or at best partially supports witness anonymity.

10. CONCLUSIONS

This paper presents the *Secure Deep Throat* (**SDT**) protocol to provide witness anonymity to users that report on the misbehavior or trust ratings of other peers in peer-to-peer systems. SDT can be used in an active mode in scenarios with real-time requirements for detecting malicious behavior, e.g., for detecting and preventing the propagation of peer-to-peer worms [34], or it can be used in a passive mode in scenarios that do not have strict real-time requirements, e.g., query-based reputation systems [12]. We discuss the security and anonymity-related requirements of a protocol for providing witness anonymity, and show that these requirements are met by SDT. Further, we describe countermeasures that can be used to defend against collusion attacks, Sybil attacks, and denial-of-service attacks against SDT. Finally, we analyze the storage, communication, and computation costs of SDT and show that the overhead of the protocol is acceptable in peer-to-peer systems.

11. REFERENCES

[1] MIRACL library. http://indigo.ie/~mscott/.

[2] G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Proceedings of The Third International Conference on Financial Cryptography (FC'99), LNCS 1648*, pages 196–211, 1999.

[3] AuctionBytes. Online auction feedback survey. Retrieved from http://www.auctionbytes.com/cab/pages/feedbacksurvey1105 on May 5, 2006.

[4] P. S. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Proceedings of Advances in Cryptology – CRYPTO 2002, LNCS 2442*, pages 354–368, 2002.

[5] P. S. L. M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In *Proceedings of Annual International Workshop on Selected Areas in Cryptography (SAC'03), LNCS 3006*, pages 17–25, 2003.

[6] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Proceedings of Advances in Cryptology - EUROCRPYT 2003, LNCS 2656*, pages 614–629, 2003.

[7] S. Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, LNCS 773*, pages 302–318, 1993.

[8] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Proceedings of Advances in Cryptology - CRYPTO 2002, LNCS 2442*, pages 465–480, 2002.

[9] J. Camenisch and M. Michels. A group signature scheme with improved efficiency (extended abstract). In *Proceedings of Advances in Cryptology - ASIACRYPT'98, LNCS 1514*, pages 160–174, 1998.

[10] D. Chaum and E. V. Heyst. Group signatures. In *Proceedings of Advances in Cryptology - EUROCRYPT '91, LNCS 547*, pages 257–265, 1991.

[11] D. L. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO88, Lecture Notes in Computer Science 403*, pages 319–327, 1989.

[12] F. Cornelli, E. Damiani, S. D. C. D. Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servents in a P2P network. In *Proceedings of the 11th International Conference on World Wide Web*, pages 376–386, 2002.

[13] E. Damiani, S. D. C. D. Vimercati, S. Paraboschi,

P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–216, 2002.

[14] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar. A reputation system to increase MIX-net reliability. In *Proceedings of The 4th International Workshop on Information Hiding (IHW'01), LNCS 2137*, pages 126–141, 2001.

[15] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[16] R. Dingledine and P. Syverson. Reliable MIX cascade networks through reputation. In *Proceedings of Financial Cryptography (FC'02), LNCS 2357*, 2002.

[17] J. R. Douceur. The sybil attack. In *Proceedings of The First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, pages 251–260, 2002.

[18] R. Granger, D. Page, and M. Stam. A comparison of CEILIDH and XTR. In *Algorithmic Number Theory, 6th International Symposium, ANTS-VI*, pages 235–249, 2004.

[19] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International Conference on World Wide Web (WWW 2003)*, pages 640–651, 2003.

[20] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang. URSA: Ubiquitous and robust access control for mobile ad hoc networks. *IEEE/ACM Transactions on Networking*, 12(6):1049–1063, 2004.

[21] M. Naor. Deniable ring authentication. In *Proceedings of Advances in Cryptology - CRYPTO 2002, LNCS 2442*, pages 481–498, 2002.

[22] M. Narasimha, G. Tsudik, and J. H. Yi. On the utility of distributed cryptography in P2P and MANETs: The case of membership control. In *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)*, pages 336–345, Nov. 2003.

[23] I. B. S. News. New peer-to-peer trojan worm attacks enterprises, Mar. 2006. Retrieved from http://www.justloadit.com/pr/6169 on May 5, 2006.

[24] L. Nguyen and R. Safavi-Naini. Dynamic k-times anonymous authentication. In *Proceedings of The Third International Conference on Applied Cryptography and Network Security (ACNS 2005)*, pages 318–333, 2005.

[25] L. Nguyen and R. Safavi-Naini. Dynamic k-times anonymous authentication. Full version, 2005.

[26] J.-F. Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *DIAU00, Lecture Notes in Computer Science 2009*, pages 10–29, 2000.

[27] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[28] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In *ACM Conference on Computer and Communications Security (CCS 2000)*, pages 33–42, 2000.

[29] A. Singh and L. Liu. TrustMe: Anonymous management of trust relationships in decentralized P2P systems. In *Proceedings of The Third International Conference on Peer-to-Peer Computing (P2P 2003)*, pages 142–149, 2003.

[30] M. Srivatsa, L. Xiong, and L. Liu. TrustGuard: Countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th International Conference on World Wide Web*, pages 422–431, 2005.

[31] I. Teranishi, J. Furukawa, and K. Sako. K-times anonymous authentication (extended abstract). In *Proceedings of ASIACRYPT 2004, LNCS 3329*, pages 308–322, 2004.

[32] Www.Programmersheaven.Com. Poll archive – how much storage capacity does your computer have?, June 2004. Available at http://www.programmersheaven.com/c/userpoll/Poll_archive.htm?PollID=148.

[33] L. Xiong and L. Liu. PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, July 2004.

[34] W. Yu, C. Boyer, S. Chellappan, and D. Xuan. Peer-to-peer system-based active worm attacks: Modeling and analysis. In *Proceedings of IEEE International Conference on Communications (ICC '05)*, pages 295–300, 2005.

[35] B. Zhu, F. Bao, R. H. Deng, M. S. Kankanhalli, and G. Wang. Efficient and robust key management for large mobile ad-hoc networks. *Computer Networks*, 48(4):657–682, July 2005.