Trusted Reputation Management Service for Peer-to-Peer Collaboration*

Lingli Deng, Yeping He, and Ziyao Xu

Institute of Software, Chinese Academy of Sciences
No.4 NanSi Street, ZhongGuanCun, Beijing, 100190, P.R. China
{denglingli,yphe,ccxu}@ercist.iscas.ac.cn

Abstract. The open and autonomous nature of peer-to-peer (P2P) systems invites the phenomenon of widespread decoys and free-riding. Reputation systems are constructed to ensure file authenticity and stimulate collaboration. We identify the authenticity, availability and privacy issues concerning the previous reputation management schemes. We propose to add integrity control for the reputation storage/computation processing in order to enhance the authenticity of the resultant reputation values; and present an integrity model to articulate necessary mechanisms and rules for integrity protection in a P2P reputation system. We design a fully-distributed and secure reputation management scheme, Trusted Reputation Management Service (TRMS). Employing Trusted Computing and Virtual Machine Technologies, a peer's reputation values and specific transaction records can be stored, accessed and updated in a tamper-proof way by the Trusted Reputation Agent (TRA) on the same platform, which guarantees the authenticity of reputation values. Transaction partners exchange directly with each other for reputation values, services and transaction comments with no reliance on a remote third party, ensuring the availability of reputation and peers' privacy.

Keywords: P2P, reputation management, data integrity, trusted computing, virtual machine.

1 Introduction

The open and autonomous nature of peer-to-peer systems invites the phenomenon of inauthentic files and free-riding. Since anyone can freely join and leave the system, it is easy to inject undesirable data, ranging from decoy files (that are tampered with or do not work)[1] to malware[2], without the fear of being punished. The prevalence of free-riders, peers who attempt to use the resources of others without sharing with them their own resources, has been reported to degrade system performance in popular P2P networks [3][4][5][6]. Reputation systems are constructed in P2P systems to prevent the spread of malicious data and to stimulate collaboration of selfish peers. A reputation system collects, distributes, and

* Supported by the National High Technology Research and Development Program ("863" Program) of China under grants No.2007AA010601 and No.2006AA010201.

R. Meersman and Z. Tari (Eds.): OTM 2008, Part II, LNCS 5332, pp. 1069–1086, 2008.

aggregates feedback about users' past behavior, encouraging reciprocal behavior and deterring dishonest participation[7].

Two major processes take place in a reputation-based P2P system: the query and response process and the service and comment exchange process. In a query and response process, a requestor sends a resource query request to locate potential providers.[1] Upon receiving a query request which hits with a local resource, a provider may query the requestor's reputation and base its decision whether to response or not on the requestor's reputation value. Upon receiving responses, the requestor chooses several provider candidates, issues reputation query requests for their reputation values, and chooses the most reputable one as the provider. In a typical service and comment exchange process, the provider maintains a list of current requestors in the order of descending reputation, and serves them in the same order. After consuming the service, the requestor submits a comment, to be used in the provider's reputation calculation.

By dividing a reputation system into a computation model and a reputation management scheme, we identify the authenticity, availability and privacy issues concerning the design of a secure management scheme. Reputation servers in centralized management schemes are prone to Denial of Service (DoS) attacks and have no guarantee for reputation availability, while distributed schemes lack effective evaluation and control over reputation agents to guarantee reputation authenticity. We propose Trusted Reputation Management Service (TRMS), a distributed scheme for reputation aggregation and distribution, to provide security guarantees while maintaining the system's overall efficiency and scalability.

Given a calculation model, the authenticity of a peer's reputation value depends on the authenticity of the involved data and calculating processes, which are prone to malicious modification in a distributed scheme. Therefore, a variation of Clark-Wilson model[8], Rep-CW, is proposed to address the integrity requirements and to guide the mechanism design of TRMS so that reputation authenticity in an open-natured P2P reputation system is assured. In particular, based on the available Trusted Computing (TC) and Virtual Machine (VM) technologies, each participating platform is equipped with a Trusted Reputation Agent (TRA), so that peers' reputation values and related transaction records are stored, accessed and updated by the local TRA in a trusted manner.

The availability of reputation is determined by the availability of the reputation server or the agent peers, and the efficiency of reputation aggregation and distribution processes. By combining a peer with its unique reputation agent into a single platform, TRMS ensures the agent's availability to an honest peer in the following way: first, the traffic overhead for reputation aggregation/distribution is minimized; second, the unfair impact on a peer's reputation due to its agent's resource limit or peer dynamics is eliminated since they represent the same platform owner. Moreover, a peer's private transaction record is kept within its own

[1] According to the resource routing mechanism used in the system, the requestor submits its request to (1) an index server (in a centralized unstructured P2P system); or (2) the whole system (in a distributed unstructured P2P system); or (3) specific index peer (in a structured P2P system).

platform. Finally, separation of running environments and access constraints are introduced to protect the agent against selfish or malicious local peers.

Our contributions include: (i) a discussion of the authenticity, availability and privacy issues in P2P reputation management; (ii) the proposal of the Rep-CW integrity model to enhance reputation authenticity; (iii) the design of TRMS, a distributed implementation of Rep-CW to solve these issues.

The paper is constructed as follows: Section 2 presents our motivation. Section 3 reviews related work. Section 4 describes Rep-CW. TRMS's design and analysis appear in Section 5 and 6. Section 7 concludes.

2 Motivation

A P2P reputation system addresses two concerns: (1) how to calculate a peer's reputation value based on its past transaction history; and (2) where to store and how to distribute peers' reputation values. Hence it can be divided into two layers accordingly: a management scheme on the bottom handling reputation storage and distribution and a calculation model on the top that aggregates the information, provided by the bottom layer, into a meaningful reputation value. We focus on the security and efficiency issues of a management scheme.

As summarized in [9], there are six key issues to be addressed by a cost-effective P2P reputation system: (1) *High accuracy*: the system should calculate the reputation value for a peer as close to its real trustworthiness as possible. (2) *Fast convergence speed*: the reputation aggregation should converge fast enough to reflect the true changes of peer behaviors. (3) *Low overhead*: the system should only consume limited resources for peer reputation monitory and evaluation. (4) *Adaptiveness to peer dynamics*: since peers come and go in an ad hoc way, the system should adapt to peer dynamics instead of relying on predetermined peers. (5) *Robustness to malicious peers*: the system should be robust to various attacks by both independent and collective malicious peers. (6) *Scalability*: the system should scale to serve a large number of peers. These requirements fall into three groups. High accuracy and fast convergence speed are proposed for the calculation model, but also restrained by the quality of the reputation data provided by the management scheme, while adaptiveness and robustness are meant primarily for the management scheme. Low overhead and scalability are issues to be addressed by both layers. Unfortunately, most previous work make no clear distinction between the calculation model and the management scheme, and some tend to tackle security and efficiency issues on the model layer alone. Security assurance in reputation management has not gained enough attention.

Based on the architecture, existing management schemes are either centralized [10] or distributed [11][12][9]. By having a reputation server manage all peers' reputation, centralized schemes contradict the open and decentralized nature of P2P networking and provide poor scalability. Distributed management schemes aggregate peer transaction comments in a fully distributed manner. Neither of these schemes delivers satisfactory assurance for reputation authenticity, availability or privacy. The critical reputation sever(s) in a centralized scheme

are prone to DoS attacks targeting reputation availability. Existing distributed schemes delegate the reputation management of a peer to another randomly selected peer (agent), with no mechanism to regulate the latter's behavior to ensure reputation authenticity and privacy. The attack model below summarizes various attacks on a P2P reputation system by exploiting vulnerabilities of these schemes, and highlights our motivation for TRMS.

2.1 Attack Model

We assume attackers are motivated either by selfish or malicious intent. A selfish attacker (free-rider) seeks to acquire unfair gainings of its own, while malicious rivals try to damage the utility of others or the whole system.

Fraud Attacks. Attackers targeting reputation authenticity may subvert the reputation system with fake transactions or identities through fraud attacks, including: *Collusion*, of a malicious collective extolling each other in a large number of fake transactions, seeking for high reputation values; *Imputation*, of a malicious peer or collective unfairly degrading a victim's reputation by unfounded complaints against a large number of fake transactions; *Sybil*, of a single malicious attacker launching a collusion or imputation collective by assuming multiple fake peer identities; *Faker*, of a peer with low reputation seeking unfair gainings by impersonating another highly reputable peer, or of an unauthorized agent manipulating reputation data by impersonating an authorized agent; and *Tamper*, of a malicious agent distributing tampered reputation data.

Availability Attacks. We consider three potential attacks, including: *Denial*, by malicious or selfish reputation agents which refuse to provide proper reputation service; *Agent-DoS*, denial of service attack by blocking certain reputation agents from proper functioning; and *Network-DoS*, with repeated requests for network-intensive reputation aggregation to congest the whole network.

Privacy Attacks. Two kinds of attacks are considered in this paper: *Census*, where malicious peers collect private records of the victim by simply querying for its reputation; and *Leakage*, of private records by compromised agents.

3 Related Work

Early reputation systems[10] deploy centralized management schemes. However, centralized schemes are not scalable to accommodate large-scale P2P networks, and are therefore used almost exclusively by centralized unstructured P2P networks (e.g. Maze[5]), where servers are used also for service location.

Distributed schemes are proposed to enhance reputation availability and system scalability through fully distributed reputation aggregation and distribution, such as P2PREP[11], EigenTrust[12], and hiRep[13]. In P2PREP, a peer's reputation is locally computed by and stored in its transaction partners. A reputation querying peer submits its request by flooding the entire system, and randomly

audits some of the votes received by sending a vote confirmation message to the voter to verify the vote, resulting in prohibitive traffic overhead. The other two distributed schemes delegate a peer's reputation management responsibility to another peer (agent). By directing the transaction comments and reputation query messages for a specific peer to its agent peer, they yield greater availability and scalability than P2PREP. However, another concern arises in these agent-based schemes: how to choose an agent from the rest of population for a given peer. EigenTrust[12] uses a distributed hash table (e.g. Chord[14]) in agent assignment for a peer by hashing its unique ID. All peers in the system aware of the peer's ID can thus locate its reputation agent. It relies on the robustness of a well-designed DHT to cope with the network dynamics, and assigns multiple agents for a peer to provide resilience against malicious agents. On the other hand, hiRep[13] is proposed for managing reputation in unstructured P2P systems, where no DHT is present. Any peer can volunteer to function as a reputation agent. A peer maintains a list of acquainted agents, keeps updating their expertise values after every transaction, and chooses those with highest expertise as its trusted agents. A peer reports transaction comments only to its trusted agents, and checks only with them to fetch the reputation values of other peers.

Despite of better scalability and availability, the reputation authenticity in a distributed scheme is not as good as a centralized one, for it lacks an effective mechanism to ensure agents' proper behavior. Reputation accuracy is also degraded for incomplete reputation aggregation due to peer dynamics. As a distributed implementation of the Rep-CW model, TRMS ensures reputation authenticity by enforcing integrity, and yields high availability through distributed deployed TRAs. With neither reliance on central servers nor distributed storage structures, TRMS applies to both structured and unstructured P2P networks. Having its local TRA as a peer's only agent, TRMS improves reputation accuracy despite of peer dynamics, and blocks privacy leakage.

To protect reputation authenticity against imputation, TrustGuard[15] employs an electronic fair-exchange protocol to ensure that transaction proofs are exchanged before the actual transaction begins. But it still can not filter out inauthentic transactions between two collusive peers, who give good ratings with exchanged transaction proofs. Instead, collusion is handled at the calculation model layer by maintaining the submitter's credibility and using it as its comment's weight in reputation aggregation. However, without enhancement at the management layer, there is always unfair gaining for collusion, while TRMS is effective in identifying fake transactions coined by either collusion or imputation.

To encourage victims to report misbehavior honestly without fear of retaliation in a polling-based scheme, [16] proposes to provide anonymity for honest claims (i.e. negative comments) while preventing imputation by discarding repeated claims. Although a comment is not anonymous in TRMS, its content is encrypted with the recipient TRA's public key, hence a claim submitter's identity is hidden from its referenced peer; and as an authentic transfer is strictly tied to a single valid comment, imputation by replaying claims is also prevented.

A TC-based agent, protected by its local Trusted Platform (TP)[17] against unauthorized modification, is independent and may be trusted by remote entities as well as the owner of the TP, and has been used by some security proposals for P2P systems [18][19][20], as a virtual trusted third party in establishing mutual trust across platforms. A privacy-enhancing P2P reputation system is developed in [18] for B2C e-commerce, where a trusted agent within the recommender's TP is introduced for forming and collecting sensitive recommendations. [19] proposes a TC-based architecture to enforce access control policies in P2P environments. A trusted reference monitor (TRM) is introduced to monitor and verify the integrity and properties of running applications in a platform, and enforce policies on behalf of object owners. Moreover, a TRM can monitor and verify the information a peer provides to ensure data authenticity (e.g. check the response message to ensure that the responder's peer ID contained is authentic)[20].

4 P2P Reputation Integrity Model

We propose to improve reputation authenticity through effective fraud control at the reputation management layer by enforcing integrity. We introduce Rep-CW, a specialized Clark-Wilson model, to address the integrity policy in a P2P reputation system, demonstrate its effectiveness in preventing fraud attacks, and use it to analyze the vulnerabilities of previous management schemes.

4.1 Rep-CW Integrity Model

A security model characterizes a particular policy in a simple, abstract and unambiguous way and provides guidance in mechanism design for policy implementation. The Clark-Wilson (CW) integrity model [8] is celebrated as the origin for the goals, policies and mechanisms for integrity protection within computer systems. It is based on the well-established commercial practices, which have long served the goal to control error and fraud by enforcing integrity (regulating authorized data modifications as well as preventing unauthorized ones).

There are two kinds of data items in the CW model: Constrained Data Items (*CDI*s), to which the integrity model must be applied; and Unconstrained Data Items (*UDI*s), not covered by the integrity policy and may be manipulated arbitrarily. *UDI*s represent the way new information is fed into the system. A *CDI* is *valid*, if it meets the systems's integrity requirements; and the system is *in a valid state*, if all the *CDI*s are valid. The particular integrity policy desired is defined by two classes of procedures: Integrity Verification Procedures (*IVP*s), and Transformation Procedures (*TP*s). The purpose of an *IVP* is to confirm that all of the *CDI*s in the system conform to the integrity specification at the time the *IVP* is executed, while *TP*s are used to change the set of *CDI*s from one valid state to another. Data integrity assurance is achieved through the well-formed transaction, and separation of duty mechanisms. The former is meant to ensure *internal consistency* of data, so that a user should not manipulate data arbitrarily, but only in constrained ways that preserve or ensure the integrity of

the data, while the latter attempts to ensure the *external consistency* of the data objects, i.e. the correspondence between the data object and the real world object it represents, by dictating that at least two people are involved to cause a critical change. Assume that at some time in the past an *IVP* was executed to verify the system was in a valid state. By requiring the subsequent state transitions (*CDI*s change) are performed only by *TP*s, and each *TP* is certified to preserve the validity of system state, it is ensured that at any point after a sequence of *TP*s, the system is still valid. By enforcing 5 certification rules and 4 enforcement rules, CW assures data integrity in a two-part process: certification (C1-C5), which is done by the security officer, system owner, and system custodian with respect to an integrity policy; and enforcement (E1-E4) by the system.

Dealing with P2P networks of a highly dynamic and open nature, Rep-CW encounters several new issues that are not addressed by the CW model, which assumes a closed system environment. First, with the ever changing user (peer) group, it is not feasible to enforce certification rules by traditional enterprise regulations. Hence, in Rep-CW these rules are enforced by the program logic of related procedures (*TP*s and *IVP*s), and verified by integrity verifications based on programs' hash fingers. Second, as network transfers are needed for reputation aggregation and distribution, their integrity of both origin and content must be ensured by verification. Consequently, Rep-CW uses a digital certificate to bind a value to its origin, and employs integrity verification on a software entity to establish the trust on the integrity of its output data.

Table 1 presents the elements in Rep-CW: peers are the users of the reputation system; unverified transaction comments submitted by peers are new data

Table 1. Rep-CW Integrity Model for P2P Reputation

Element	**Description**
User	peer (representing the interest of its owner)
UDI	Transaction Comments (*TC*s) submitted by peers.
CDI	Valid transaction comments, in the form of Transaction Records (*TR*s); and valid reputation values as Reputation Certificates (*RC*s).
TP	Transaction Logging Agent (*TLA*): verifies the validity of received *TC*s, records valid ones into *TR*s; Reputation Calculation Agent (*RCA*): calculates and updates peers' *RC*s using *TR*s.
IVP	Reputation Verification Agent (*RVA*): verifies the validity of *RC*s. Transaction Verification Agent (*TVA*): verifies the validity of *TR*s.
Rule	**Content**
C1	A peer's *RC* (*TR*) is accredited only after verification by *RVA* (*TVA*).
C2	All *TLA*s and *RCA*s must be certified by attestation to preserve validity.
E1	A peer's *TR* is updated only by its authorized *TLA* and *RCA*; and its *RC* is calculated and issued only by the authorized *RCA*, accordingly.
E2	Reputation is updated only by valid *TC* submissions to authorized *TLA*.
C3	A valid transaction involves at least 2 authentic peers.
C5	All *TLA*s must be certified by integrity attestation to accept valid *TC*s into *TR*s and discard invalid ones.

fed to the system as *UDI*s and are accepted into transaction records (*CDI*s) after successful validity verification; peers' reputation certificates are also *CDI*s subject to system's integrity protection. The data validity is defined as follows.

Definition 1 (External Consistency). *A comment* $tc = \langle description, comment \rangle$ *submitted by peer* P *is valid, if* $tc.description$ *corresponds to a unique cross-platform file transfer from another peer* Q*.*

The validity of a comment contains two meanings: (1) *authenticity*, that there exists a cross-platform transfer from Q to P, which coheres with tc's description; and (2) *uniqueness*, that tc is the first valid comment submitted for the transfer. In all, given an authentic file transfer, there is but one valid comment.

Definition 2 (Internal Consistency). P*'s reputation certificate* $rc = \langle P, value, valid\ period \rangle$ *is valid, if* rc *has not expired and is issued by a certified RCA authorized for managing* P*'s reputation.*

In a P2P reputation system, reputation value updates indicate transitions of system state, and correspond to the transformation procedure *RCA* in Rep-CW. To ensure external consistency, another transformation procedure *TLA* verifies the validity of each received transaction comment according to Definition 1, records only valid ones into transaction records. To ensure internal consistency, the integrity verification procedure *RVA* verifies the validity of reputation certificates according to Definition 2; while integrity verification procedure *TVA* verifies the validity of transaction records maintained by *TLA*.

Figure 1 shows how the rules (Table 1) of Rep-CW (C-Certification rules; E-Enforcement rules) control the system operation. An *TLA* checks newly submitted *TC*s and accepts valid ones into the system by updating *TR*s. An *RCA* takes *TR*s and *RC*s as input and produces new versions as output. These two sets of both *TR*s and *RC*s represent two successive valid states of the system, while an *RVA* (or *TVA*) verifies the validity of *RC*s (or *TR*s). Associated with each system part is the rule that governs it to ensure integrity.

Rep-CW prevents unauthorized modifications and regulates authorized modifications to reputation data in the following way: First, execution of *RVA* verifies the *RC*s' external and internal consistency (rule C1). Second, for any subsequent state transition (i.e. reputation update), the related *RCA* is certified by means of integrity verification based on program hash finger to ensure internal consistency of the changed *TR*s and *RC*s (rules C2 and C5). Third, the authorization lists specified in rules E1 and E2 are used to prevent unauthorized modifications on *TR*s and *RC*s. Fourth, by dictating that any valid transaction involve at least two authentic peers (rule C3), Rep-CW prevents a single attacker from manipulating reputation data. We preserve the numbering for rules in CW to clarify the correspondence between the two models.[2]

[2] Rep-CW contains no counterparts for the rules E3, E4, and C4 of the CW model, because: (1) in an open P2P collaboration environment, no prior authentication and authorization are imposed on participating peers' identities (E3 and E4); and (2) Rep-CW can not protect reputation certificates from manipulation, instead it blocks manipulated ones from being used in the system, so no log *CDI* is used (C4).

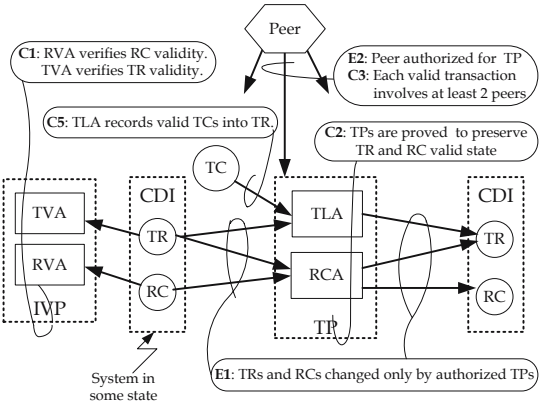


Fig. 1. Rep-CW Integrity Model

4.2 Integrity Analysis Based on Rep-CW

We demonstrate the effectiveness of Rep-CW in enhancing reputation authenticity by preventing fraud attacks. For each fraud attack in Section 2.1, Table 2 summarizes the corresponding Rep-CW integrity rules the attacker must break to launch a successful assault. First of all, to perform a collusion or imputation, comments on inauthentic transactions from attackers must be accepted by the system, which breaks rule C5. Second, by assuming multiple fake identities for a given physical platform and coining inauthentic transactions among them, a Sybil attacker virtually violates rules C3 and C5. Third, the act of a faker, who tries to use another peer's reputation, is against rules C1 and E2. Finally, a malicious or compromised agent, exploited by a tamper attack, cannot pass the integrity verification to be an authorized *RCA*, as required by C1, C2 and E1.

Table 2. Fraud Attacks v.s. Integrity Rules

Attack	C1	C2	E1	E2	C3	C5
Collusion						×
Imputation						×
Sybil					×	×
Faker	×			×		
Tamper	×	×	×			

Table 3 presents a comparison of typical P2P reputation management schemes from the Rep-CW's point of view. We make the following observations.

Previous distributed schemes are vulnerable to fraud attacks, because: first, they are prone to collusion and imputation attacks, as reputation agents cannot identify inauthentic transaction comments for the lack of *TLA* (in EigenTrust[12]

Table 3. A Comparison of Management Schemes

Scheme	TLA	RCA	TVA	RVA	C1	C2	E1	E2	C3	C5
eBay	×	√	×	√	√	√	√	√	×	×
P2PREP	√	√	×	×	×	×	×	×	×	×
EigenTrust	×	√	×	√	×	×	√	√	×	×
TrustGuard	√	√	×	√	×	×	√	√	×	×
hiRep	×	√	×	√	×	×	√	√	×	×

and hiRep[13]) or the strict binding of accepted comments to authentic file transfer events (in P2PREP[11] and TrustGuard[15]); second, the (PKI-based) *RVA* procedure provided is not capable of filtering out tampered or inauthentic reputations issued by compromised agents; third, no mechanism for separation of duty is provided to suppress Sybil attacks.

In a centralized scheme (e.g. eBay[10]), the globally trusted reputation server acts as the unique authorized *RCA* in issuing reputation certificates for all peers. A reputation certificate is verified using the server's public key by the querying peer (corresponding to *RVA*). All comments must be submitted to the server to be used in reputation update, while being immune to faker and tamper attacks, the system with a centralized scheme is still vulnerable to attacks using inauthentic transactions (e.g. collusion, imputation and Sybil), as no mechanisms for *TLA* or separation of duty is implemented. Moreover, the reputation server is prone to DoS attacks, and becomes the bottleneck for system scalability.

In summary, previous distributed schemes hardly provide any integrity protection against fraud attacks, while centralized schemes deliver limited integrity protection at the cost of system scalability and performance. To enhance reputation authenticity while maintaining system scalability and availability, we propose TRMS, a distributed implementation of Rep-CW, whose protection mechanisms are deployed on participating platforms, certified through TCG integrity attestation [21], and protected by Xen virtual machine environment [22].

5 TRMS: Trusted Reputation Management Service

5.1 Background: Trusted Computing and Virtual Machines

A Trusted Platform (TP) is a normal open computer platform equipped with a tamper-resistant hardware Trusted Platform Module (TPM) as the root of trust, providing three basic functionalities to propagate trust to application software and/or across platforms [21]. (1) Through *Integrity Attestation* mechanism, the integrity of a TP, including the integrity of many components of the platform, is recorded by the Platform Configuration Registers (PCRs) in TPM and can be checked by both local users and remote entities to deduce their trust in the platform [17]. (2) *Sealed storage* provides protection against theft and misuse of sensitive data held on the platform so that it is available only when the platform is in a particular integrity state, i.e. when the correct programs are running. (3)

An *Attestation Identification Key* (AIK) is created by the TPM and used in an attestation protocol to provide a signature over PCRs to prove authenticity.

The VM technology [23] allows multiple operating systems to simultaneously run on one machine. A Virtual Machine Monitor (VMM) is a software layer underneath the operating system that provides (1) a VM abstraction that models and emulates a physical machine and (2) isolation between VMs such that each VM runs in its own isolated sandbox. We use Xen [22], an open-source VMM. In Xen-speak, each VM is referred to as a *domain*, and Xen itself the *hypervisor*. The hypervisor remains in full control over the resources given to a domain. Domain0 (*Dom*0) is the first instance of an OS that is started during system boot as a management system for starting further domains. All other domains are user domains that receive access to the hardware under *Dom*0's mediation.

5.2 Overview

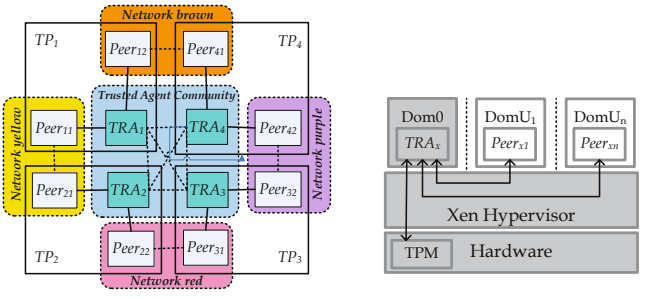
TRMS equips each participating platform with a Trusted Reputation Agent (TRA) to manage local peers' reputation. Peers' reputation values and related transaction records are stored, accessed, and updated by the TRA on the local trusted platform. TRMS uses the TC mechanism for integrity measurement, storage and reporting to verify that a remote TRA is running in an expected manner. For a given transaction, the reputation values, service and comment are exchanged between the two directly involved platforms with no reliance on a remote third party, which guarantees reputation availability and eliminates the traffic overhead for network-wide reputation aggregation and distribution. To protect locally stored reputation data against manipulation by its selfish owners, TRA runs in a protected VM separated from peers to avoid run-time manipulation, and stores reputation data in sealed storage (encrypted and protected by TPM) against unauthorized access other than TRA.

In terms of integrity protection, TRMS realizes the *TP*s and *IVP* of the Rep-CW model through the TRA, by requiring that: (1) A peer's reputation certificate should be verified to be valid by the querying peer's TRA according to Definition 2 (corresponding to *RVA*). (2) The transaction comment, submitted by the consuming peer, is verified according to Definition 1 by the serving peer's local TRA (acting as *TLA*) before accepted into transaction records. (3) The TRA (functioning like *RCA*) on a participating platform is responsible for maintaining the verified transaction records and updating local peers' reputation certificates, according to the calculation model.

5.3 Architecture

There are two kinds of symmetric collaboration in a P2P network under TRMS: a Trusted Agent Community (TAC) formed by TRAs from participating platforms and the network of regular peers. A peer's reputation related information is managed by the TRA on its local platform, ensuring rule E2 in Rep-CW. In TRMS, for a peer to join a P2P network and interact with another peer, their local TRAs have to join TAC and establish mutual trust first. Through TRA

attestation (described later), a querying peer trusts another peer's reputation certificate only if the integrity of the latter's platform and TRA is verified. Rule C2 is ensured by denying reputation from unverified TRAs. Consider the illustrative scenario in Figure 2(a). $TRA_1 - TRA_4$ on $TP_1 - TP_4$ join TAC, and the peers on $TP_1 - TP_4$ join corresponding P2P networks. E.g., both $Peer_{11}$ on TP_1 and $Peer_{21}$ on TP_2 join a P2P network *yellow*, while TP_1's another peer $Peer_{12}$ collaborates with $Peer_{41}$ on TP_4 in another P2P network *brown*.



(a) P2P networks under TRMS. (b) A Peer Platform with TRA.

Fig. 2. TRMS Architecture

As the local reputation agent, TRA should be verified and trusted by remote querying peers. Selfish or malicious local peers (or even the platform's owner) have the motive to subvert, replace, or manipulate the TRA and/or its data. Therefore, Xen virtual machines are used to separate TRA from local peers. Figure 2(b) depicts the architecture of a participating platform with TRA, based on Xen. The trusted components (the shaded areas) includes the trusted hardware (TPM), the security kernel (Xen Hypervisor and Dom0), and TRA running in Dom0. The hardware and security kernel provide TRA with necessary security services, including basic cryptographic functions, platform and program attestation, sealed storage, and protected running environments. TRA's sensitive data includes local peers' reputation values and transaction records. The isolation of VMs protects data and processing integrity from being tampered locally.

The following constraints are enforced by each participating platform's trusted components (the first by secure kernel; and the other two by TRA):

1. There is always a unique TRA running on the local platform.
2. There is always at most one local peer joined in a given P2P network.
3. Transactions between local peers are excluded for reputation calculation.

Constraint 1 is intended to prevent malicious local peers or platform owners from subverting TRA's monitoring (rules E1 and E2). Constraints 2 and 3 enforce the Separation of Duty principle (rule C3) by dictating a valid transaction involve two physical platforms. The certification for their proper enforcement is achieved by the integrity attestation of TRA and its platform.

5.4 TRA: Trusted Reputation Agent

Our design is based on three basic assumptions: (1) TC hardware is tamper-resistant; (2) the isolation of VMs is flawless; (3) each participating platform is a trusted platform, with necessary TC hardware and VMM software.

By local/remote attestation, local/remote peers verify the integrity of the issuing TRA and its running environment (both hardware TPM and security kernel software) before accrediting a reputation certificate. We assume the following credentials are bestowed on a trusted platform when performing attestation.

- TPM's AIK, (PK_{AIK}, SK_{AIK}), is produced by TPM to be used to prove the authenticity of PCR values or programs' public key certificates to a remote challenger. Its private part is protected by TPM, and its public part is issued by a private CA or through Direct Anonymous Authentication protocol.
- TRA's Asymmetric Key, (PK_{TRA}, SK_{TRA}), used to sign or encrypt data exchanged between TRAs. Its private part is protected by the TPM, and the public key certificate is issued by TPM using AIK.

Employing TPM, TRA has the following primitives:

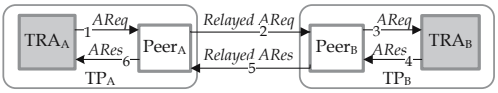
- $TRA.Seal(H(TRA), x)$, used by TRA to seal x with its own integrity measurement $H(TRA)$. The sealed x is unsealed by TPM only to the same TRA whose integrity measurement equals $H(TRA)$.
- $TRA.Unseal(H(TRA), x)$, unseals x, if $H(TRA)$ was used in sealing x.
- $TRA.GenNonce(n)$, generates a random number n.
- $TRA.Attes(H(TRA), PK_{TRA})$, responds to a remote attestation challenge, by returning local TRA's public key certificate, bound to $H(TRA)$ and signed by local TPM's AIK: $(H(TRA), PK_{TRA})_{SK_{AIK}}$.

5.5 Cross-Platform Interactions in TRMS

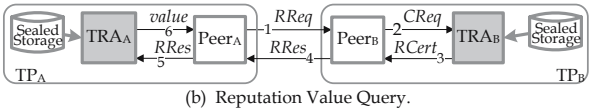
According to Rep-CW, TRMS provides three cross-platform interactions: (1) the mutual attestation between TRAs to ensure they (the corresponding *RCA*s) preserve the validity of reputation data; (2) the reputation query process, in which the querying peer verifies (by calling *RVA* implemented by TRA) the reputation certificate's validity according to Definition 2; and (3) the modified service and comment exchange process, where the serving peer's TRA (the corresponding *TLA*) verifies received transaction comment's validity according to Definition 1.

Mutual Attestation between TRAs. Once two peers are to establish mutual trust for a reputation query process, they use TC-based remote attestation to verify the integrity of the other's TRA and platform, before accrediting the other's reputation certificate. Suppose TRA_A on platform TP_A wishes to verify the integrity of TRA_B on platform TP_B. They interact as follows: (Figure 3(a))

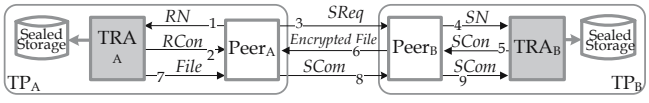
1. TRA_A sends TRA_B a *Attestation Request* (*AReq*) via $Peer_A$ and $Peer_B$.
2. TRA_B calls $TRA_B.Attest$ to get its integrity certificate, and wrap it and AIK's certificate with a *Attestation Response* (*ARes*): $(PK_{AIK_B})_{SK_{CA}} \,||\, (H(TRA_B),\, PK_{TRA_B})_{SK_{AIK_B}}$.



(a) Mutual Attestation between TRAs.



(b) Reputation Value Query.



(c) Service and Comment Exchange.

Fig. 3. Cross-Platform Interactions in TRMS

3. *ARes* sent by TRA_B is handed over to TRA_A via $Peer_B$ and $Peer_A$.
4. TRA_A verifies the validity and integrity of the certificates in *ARes*.[3]

Reputation Value Query. Suppose peer $Peer_A$ wishes to query the reputation of peer $Peer_B$, and the involved TRAs have exchanged public keys (PK_{TRA_A} and PK_{TRA_B}) during a prior successful mutual attestation. According to Rep-CW's rule C1, $Peer_A$ queries for $Peer_B$'s reputation value and verifies its validity as follows: (Figure 3(b))

1. $Peer_A$ sends a *Reputation Request* (*RReq*) to $Peer_B$.
2. $Peer_B$ checks its *Reputation Certificate* (*RCert*) issued by TRA_B on TP_B. If it is still valid, $Peer_B$ skips to Step 4; otherwise, it sends a *Certification Request* (*CReq*) message to TRA_B, asking for TRA_B a new one.
3. To respond to *CReq*, TRA_B first calls $TRA_B.Unseal$ to read $Peer_B$'s reputation data from sealed storage, recalculates $Peer_B$'s reputation value; then calls $TRA_B.Seal$ to seal the new reputation data, and issues a new $RCert(Peer_B) = (Peer_B, value, T_{issue}, T_{expiry}, H(TRA_B))_{SK_{TRA_B}}$.
4. $Peer_B$ wraps its valid *RCert* with a *Reputation Response* (*RRes*) to $Peer_A$.
5. TRA_A verifies *RCert*'s validity, and returns the reputation value to $Peer_A$.

Service and Comment Exchange. To enable validity verification for transaction comments (Rep-CW's rule C5), TRMS makes several modifications to the regular service and comment exchange process: First, to filter out fake transactions used by collusion or imputation attackers, the two peer's TRAs are actively involved as witnesses for each cross-platform file transfer event through the request notification/registration. Second, random nonce is used to uniquely identify each authentic transfer, preventing a dishonest peer from replaying comment

[3] To realize mutual attestation in one process, we can modify the above protocol in the following way: TRA_A generates its own integrity certificate, and sends it to TRA_B along with TP_A's certificate, via the *AReq* message in Step 1; and TRA_B verifies their validity before executing Step 2.

duplicates. Third, to deal with potential conceal of negative comments by dishonest local peers, on one hand, the remote reporting peer encrypts its comment with the public key of the serving peer's TRA, hence the serving peer cannot perform content-based filtration against negative comments; on the other hand, a timeout mechanism is included so that a negative comment is automatically generated for the serving peer by its TRA if no valid comment is received in time, hence the serving peer cannot perform source-based filtration either.

Suppose $Peer_A$ on TP_A wishes to download a file from $Peer_B$ on TP_B. It is assumed that the exchange for public keys of TRA_A and TRA_B has been exchanged by a prior mutual attestation. The process is depicted in Figure 3(c).

1. $Peer_A$ sends a *Request Notification* (RN) to TRA_A, including the service description, ($Peer_B$, agent TRA_B, and the requested file's description[4]) expecting a *Request Confirmation* ($RCon$) in reply.
2. TRA_A registers the received RN and generates a $RCon$ message, containing the description from RN and a $nonce_A$ returned by $TRA_A.GenNonce$. $RCon(Peer_A, desc) = ((nonce_A)_{PK_{TRA_B}}, desc, Peer_A)_{SK_{TRA_A}}$.
3. $Peer_A$ wraps the $RCon$ with a *Service Request* ($SReq$) to $Peer_B$.
4. If $Peer_B$ consents to serve $Peer_A$ with the described file, it sends a *Service Notification* (SN) to TRA_B, carrying the $RCon$ from TRA_A.
5. TRA_B registers the file description, the $nonce_A$ in SN and another $nonce_B$ returned by $TRA_B.GenNonce$, and encrypts them with TRA_A's public key as a *Service Confirmation* ($SCon$) in response to SN from $Peer_B$. $SCon(Peer_B, SN) = (desc, nonce_A, nonce_B)_{PK_{TRA_A}}$.
6. $Peer_B$ encrypts the file with TRA_A's public key PK_{TRA_A}, and sends it along with the $SCon$ from TRA_B, to TRA_A via $Peer_A$.
7. TRA_A verifies the received $SCon$ and the $SReq$ contained, makes sure that the $nonce_A$ in $SReq$ is used in Step 1, decrypts the file and $nonce_B$, checks the file against $SReq$'s description,[5] and gives file and $nonce_B$ to $Peer_A$.
8. $Peer_A$ generates a *Service Comment* ($SCom$), $SCom(nonce_B) = (desc, comment, nonce_B)_{PK_{TRA_B}}$, and sends it to $Peer_B$.
9. $Peer_B$ forwards received $SCom$ to TRA_B, who updates $Peer_B$'s transaction record after a successful verification of $SCom$'s $desc$ and $nonce_B$ against its registration records. For each item registered in Step 5, a negative comment is generated automatically by TRA_B, if no valid $SCom$ is received in time.

Summary. In a reputation-based P2P network using TRMS, the query and response process is similar to the one described in Section 1, and a reputation query process is also contained: a service request also serves as a reputation request, and a reputation response is combined with a service response. The service querying peer selects the most reputable responder to submit its service request, triggering a service and comment exchange described above.

[4] E.g., the file's hash finger, whose authenticity is verifiable by a third party.

[5] The objective description (such as file's hash finger) is verified by TRA; while the subjective satisfaction is evaluated by the consuming peer later in a comment.

6 Analysis of TRMS

Traffic Overhead. As a peer's TRA resides on its local platform, the network traffic in TRMS occurs exclusively between actual or potential transaction partners. All the network messages can be piggied back to the regular service query and exchange messages, except the *SCom* message carrying the transaction comment. As each valid *SCom* is sent from the consumer to the provider for an authentic cross-platform transfer, the traffic overhead is minimal, compared with other distributed schemes performing costly network-scale aggregation periodically.

Table 4. Rep-CW Rules as Implemented by TRMS

Rule	Implementation in TRMS
C1	Before accrediting a reputation certificate, the querying peer calls its TRA to verify the integrity of both the certificate and the issuing TRA. Since transaction records are the private data of TRA and are protected by sealed storage, TRMS provides no further verification of the records.
C2	All TRAs are certified by integrity attestation to preserve reputation validity, otherwise reputation certificates issued by them would be discarded.
E1	Only the certificates issued by certified TRAs are used by honest peers, and a peer's reputation certificate is issued only by its local TRA.
E2	A peer must submit valid transaction comments to its local TRA to have its reputation updated, since: TRA's signature on a certificate ensures the identification and exclusion of tampered reputation; and transaction records, protected by sealed storage, are only accessible to the local TRA.
C3	Since there is at most one local peer participating in a P2P network at any time on the same platform, the generation (by *TLA* and *RCA*), verification (by *RVA*) and usage of a given peer's reputation must involve two platforms.
C5	TRA verifies received comments' validity and discards invalid ones.

Reputation Authenticity. As demonstrated by Table 4, all the integrity rules in Rep-CW are effectively implemented by TRMS. By implementing these rules, TRMS eliminates the weaknesses exploited by various fraud attackers (Table 2). Specifically, in TRMS: (1) invalid comments about fake transactions minted by collusion or imputation attackers are discarded; (2) Sybil attacker's cost for a fake identity is enhanced greatly since each authentic peer has to be on a single physical platform to form an effective collective; (3) the trust chain from a TP via TRA to a peer's reputation certificate must be in place for its value to be accredited, which guarantees the identification and exclusion of tamper attacks; (4) since TRA is the only authorized *RCA* for local peers and there is at most one local peer in a given P2P network, a faker can use neither a remote peer's reputation certificate (not issued by the local TRA) nor another local peer's certificate (not in the same P2P network).

Reputation Availability. It is clear that there is no motive for Denial attacks, since an agent (TRA) and the peers it serves reside on a single platform and represent the same owner. TRMS also provides resilience against DoS attacks:

On one hand, interacting through local peers, TRAs do not handle network traffic directly, therefore are immune to remote Agent-DoS attackers; on the other hand, DoS attacks, which block TRA from functioning by repeated requests, are not in the interest of local peers. Moreover, each platform has its own dedicated TRA, so Agent-DoS attacks targeting one or a few TRAs can hardly affect the whole system. Finally, as any reputation-related process involves at most two platforms, with minimal traffic overhead, there is little chance for a Network-DoS attack.

Transaction Privacy. TRMS confines the exposure of a peer's transaction record to the verified local TRA. First, TRMS does not support transaction history query for a specific peer and the *RRes* message contains only the reputation value not record, which prevents census attacks. Second, a peer's transaction record is protected by sealed storage and accessible only to its local TRA.

Summary. Compared with previous P2P reputation management schemes, TRMS enhances reputation authenticity by the trusted and verifiable data management and a strong binding of a transaction comment with an authentic cross-platform file transfer. Given a calculation model, the system yields higher reputation accuracy as a peer's transaction history is exclusively and more completely collected by its local TRA in the presence of peer dynamics. As a general reputation management scheme, it can be used to support various reputation calculation models. It provides stronger robustness against reputation attacks and delivers great network scalability as a totally distributed scheme, with minimal traffic overhead.

7 Conclusion

We identify the authenticity, availability and privacy issues in P2P reputation management; propose Rep-CW integrity model to address the reputation authenticity requirements; and design TRMS, a fully-distributed reputation management scheme, using available TC and VM technologies, with enhanced reputation authenticity, availability and privacy guarantees.

References

1. Liang, J., Kumar, R., Ross, K.W.: Pollution in P2P File sharing systems. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 1174–1185. IEEE Press, New York (2005)
2. Kalafut, A., Acharya, A., Gupta, M.: A Study of Malware in Peer-to-Peer Networks. In: 6th ACM SIGCOMM conference on Internet measurement, pp. 327–332. ACM Press, New York (2006)
3. Adar, E., Huberman, B.A.: Free riding on Gnutella. First Monday 5(10), 2 (2000)
4. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A Measurement Study of Peer-to-Peer File Sharing Systems. In: Kienzle, M.G., Shenoy, P.J. (eds.) MMCN 2002. SPIE Press (2002)
5. Yang, M., Zhang, Z., Li, X., Dai, Y.: An Empirical Study of Free-Riding Behavior in the Maze P2P File-Sharing System. In: Castro, M., van Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640, pp. 182–192. Springer, Heidelberg (2005)

6. Pouwelse, J.A., Garbacki, P., Epema, D.H.J., Sips, H.J.: The Bittorrent P2P File-Sharing System: Measurements and Analysis. In: Castro, M., van Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640, pp. 205–216. Springer, Heidelberg (2005)
7. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation Systems. Commun. ACM 43(12), 45–48 (2000)
8. Clark, D.D., Wilson, D.R.: A Comparison of Commercial and Military Computer Security Policies. In: IEEE Symposium on Security and Privacy, pp. 184–194. IEEE Press, New York (1987)
9. Zhou, R., Hwang, K.: PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. IEEE Trans. Parallel Distrib. Syst. 18(4), 460–473 (2007)
10. Houser, D., Wooders, J.: Reputation in Auctions: Theory and Evidence from eBay. Journal of Economics & Management Strategy 15(2), 353–369 (2006)
11. Damiani, E., di Vimercati, D.C., Paraboschi, S., Samarati, P., Violante, F.: A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In: 9th ACM conference on Computer and Communications Security, pp. 207–216. ACM Press, New York (2002)
12. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The Eigentrust Algorithm for Reputation Management in P2P Networks. In: 12th International Conference on World Wide Web, pp. 640–651. ACM Press, New York (2003)
13. Liu, X., Xiao, L.: hiREP: Hierarchical Reputation Management for Peer-to-Peer Systems. In: International Conference on Parallel Processing, pp. 289–296. IEEE Press, Washington (2006)
14. Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 149–160. ACM Press, New York (2001)
15. Srivatsa, M., Xiong, L., Liu, L.: TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In: 14th International Conference on World Wide Web, pp. 422–431. ACM Press, New York (2005)
16. Zhu, B., Setia, S., Jajodia, S.: Providing Witness Anonymity in Peer-to-Peer Systems. In: 13th ACM Conference on Computer and Communications Security, pp. 6–16. ACM Press, New York (2006)
17. Maruyama, H., Seliger, F., Nagaratnam, N.: Trusted Platform on Demand. IBM Research Report, RT0564 (2004)
18. Kinateder, M., Pearson, S.: A Privacy-Enhanced Peer-to-Peer Reputation System. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) EC-Web 2003. LNCS, vol. 2738, pp. 206–215. Springer, Heidelberg (2003)
19. Sandhu, R., Zhang, X.: Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. In: 10th ACM Symposium on Access Control Models and Technologies, pp. 147–158. ACM Press, New York (2005)
20. Zhang, X., Chen, S., Sandhu, R.: Enhancing Data Authenticity and Integrity in P2P Systems. IEEE Internet Computing 9(6), 42–49 (2005)
21. Trusted Computing Group, `http://www.trustedcomputinggroup.org`
22. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the Art of Virtualization. In: ACM Symposium on Operating Systems Principles, pp. 164–177. ACM Press, New York (2003)
23. Goldberg, R.P.: Survey of Virtual Machine Research. IEEE Computer 7(6), 34–45 (1974)