Wireless Pers Commun (2017) 95:3785–3812
DOI 10.1007/s11277-017-4026-5

P-Accountability: A Quantitative Study of Accountability in Networked Systems

Zhifeng Xiao[1] · Yang Xiao[2,3] · Jie Wu[4]

Published online: 3 February 2017

Abstract Accountability in computing implies that an entity should be held responsible for its behaviors with verifiable evidence. In order to study accountability, quantitative methods would be very helpful. Even though there are some researches in accountability, there are no other works which study quantitative accountability in practical settings, while quantitative accountability is defined as using quantities or metrics to measure accountability. In this paper, we propose P-Accountability, which is a quantitative approach to assess the degree of accountability for practical systems. P-Accountability is defined with two versions, a flat model and a hierarchical one, which can be chosen to use depending on how complex the system is. We then provide a complete case study that applies P-Accountability to PeerReview, which provides Byzantine fault detection for distributed systems. In addition, we propose Traceable PeerReview, which is our effort to apply PeerReview to wireless multi-hop environments. In addition, through the system evaluation we can show that the simulation outcomes are aligned with the numeric results.

Keywords Accountability · Quantification · Wireless networks · Distributed system · Performance metric

✉ Yang Xiao
yangxiao@ieee.org

Zhifeng Xiao
zux2@psu.edu

Jie Wu
jiewu@temple.edu

1 Department of Computer Science and Software Engineering, Penn State Erie, The Behrend College, Erie, PA 16509, USA

2 School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China

3 Department of Computer Science, The University of Alabama, Tuscaloosa, AL 35487-0290, USA

4 Department of Computer and Information Science, Temple University, Philadelphia, PA 119122, USA

1 Introduction

Recent advances have witnessed the development and application of accountability, which has become a core requirement of building trustworthy computer systems [1] and dependable networked systems [2, 3]. Generally, accountability in computing implies that an entity should be held responsible for its own activities with verifiable evidence [4]. Prior researches have discussed the design of accountable systems in different contexts, including accountable logging using flow-net [4, 5] and virtual flow-net [6], a multi-resolution flow-net accountable logging [7, 8], a quantitative accountable logging method [9, 10], an accountability system called PeerReview in distributed systems with deterministic protocols in terms of ensuring detecting Byzantine (i.e., arbitrary) faults [11], a design for ID accountability in terms of using self-certifying network layer addresses for future Internet [12, 13], a design of source signature and verification for packets in a future Internet architecture [14], an extension of PeerReview called Cryptographically Strong, Accountable Randomness (CSAR) with an effort to achieve accountability for distributed systems that use randomized protocols [15], an accountable method of detecting and preventing from malicious software modification and violations in virtual networks [16], a layered trust management to support accountability in email systems [17], an accountability interface called AudIt for ISPs for handling packet loss and delay [18], an accountable network storage service called Certified Accountable Tamper-evident Storage service (CATS) for evidence of read and write responses [19], an effort to add accountable congestion for Transmission Control Protocol (TCP) and the Internet Protocol (IP) [20, 21], an accountable operating system (OS) in terms of providing accountable administrators [22–24], an approach for temporal accountability for medical sensor networks [25], an accountable method for household appliances in terms of power usage [26, 27], an accountable framework for sensing-oriented mobile cloud computing [28], a mutual verifiable provable data possession scheme for public cloud storage [29, 30], etc.

Through the previous studies that we have investigated, we observe that accountability was interpreted differently for each specific system. However, the common core of accountability is honored. In summary, an accountable computing system is by design (1) for responsibility assignment with irrefutable evidence, and (2) for applying the reward and punishment to the responsible party, if necessary. It is challenging to achieve these two goals, because in a complex networked system, any entity could be responsible for an event (e.g., a cloud server, a Personal Computer (PC), a mobile device, a router/switch, a smart phone App). When a network incident occurs (e.g., external attacks or system misconfiguration), the network operator needs to identify the origin of related events and then apply fixies. However, to prevent a responsible entity from denying its behavior, an accountable system must present irrefutable evidence that can prove an entity's misbehavior, and this verification can be either conducted by other correct entities or a third party. Based on the evidence, every entity can be held responsible, and punishment or reward can be applied thereafter.

The motivations of this paper are as follows. First, we regard accountability as a system feature which is constrained by certain requirements. In this case, it is beneficial to have a practical and unified metric for accountability assessment, as this quantified information will supply valuable guidance for system improvement. Second, prior research shows that it is usually unaffordable to implement a system with perfect accountability due to various touch conditions and uncertainties in the real world such as strong identification for every computing device [11, 13, 14], per-hop/message verification [11], and a powerful repository/logging system [4, 8, 19], which generate tremendous computational and storage overhead that contribute to performance degradation for production network. In addition, it is difficult to achieve perfect accountability for a networked system due to numerous network dynamics such as packet delay, packet loss, and node failure.

Even though there are some researches in accountability, there are no other works which study quantitative accountability in practical settings, while quantitative accountability is defined as using quantities or metrics to measure accountability. In this paper, we propose a practical guidance that instead of pursuing perfect accountability, system designers only need to provide satisfactory accountability so that system resources will not be overconsumed. To this end, it is essential to study the balance of accountability and system overhead. A quantitative study of accountability is the first step to achieve this goal.

Our goal is to find out the capability of a system in terms of accountability. In this paper, we develop P-Accountability, a generic model for accountability assessment for networked systems. P-Accountability models an accountable network system by abstracting the notions of entities, events, and the mapping relation between them. P-Accountability should be customized to suit the needs for a practical system. First, we need to identify the event space and entity space for a system, and then find out the factors that may affect the degree of accountability in order to give formal analysis and an empirical study.

The contributions of this paper are explained as follows:

- We propose P-Accountability, which is a quantitative model for accountability assessment. The model includes two parts: a flat model and a hierarchical model. The flat model is applicable to accountable systems with flat structures, meaning that the entities are on the same logical level. We then extend the flat model to a hierarchical model, which considers a multi-level environment where each entity in a certain level may be further composed of fine-grained entities in a lower level. This indicates that blame may be assigned to a more concrete entity in some circumstances. P-Accountability derives from practical needs, and we also propose an analytical method to study a system to approach the experimental results.
- We provide a complete case study to apply P-Accountability to PeerReview, which studies the Byzantine fault detection problem (i.e., the problem to detect Byzantine faults) for distributed systems. Note that Byzantine faults are important since they exhibit arbitrary behaviors. We discover that message loss may be a key factor to affect accountability provided by PeerReview. We demonstrate that P-Accountability can be adopted to assess PeerReview given that message loss is inevitable.
- With the feedback we obtain from the case study, we propose Traceable PeerReview which extends PeerReview to a wireless multi-hop network environment.
- Our evaluation results show that P-Accountability is effective in the assessment of accountability.

We structure the rest of the paper as follows: we review the prior studies in Sect. 2. We then define two models of P-Accountability in Sects. 3 and 4, respectively. In Sect. 5, we conduct a case study that applies P-Accountability to PeerReview. Section 6 proposes Traceable PeerReview, which extends PeerReview to a wireless multi-hop environment. Section 7 discusses the evaluation outcomes. The paper is concluded in Sect. 8.

2 Related Work

2.1 Existing Accountable Systems

Accountability has been applied to a variety of networked systems for security enhancement. Certified Accountable Tamper-evident Storage service (CATS) [19] is an application-level storage service providing verifiable misbehavior detection. AudIt [18] is described as an accountability interface that facilitates ISPs to determine the Administrative Domains (ADs) that are responsible for dropping or delaying the traffic. The authors in [13, 14] present Accountable Internet Protocol (AIP), which employs a hierarchy of self-certifying addresses to enable network-layer accountability to ensure that a forged IP address can be detected with verifiable evidence. However, AIP requires modifying the current IP protocol, which is unlikely to be immediately deployed. As an alternative to AIP, the authors in [14] design a perfect accountability scheme by binding an unforgeable signature to each packet and having the closest router verify it for fake IP detection. PeerReview in [11, 31] offers accountability support for distributed systems that suffer Byzantine faults in a deterministic environment via a tamper-evident logging scheme to record node's actions and a witness scheme to conduct periodical status checking for every node in the system. Cryptographically Strong, Accountable Randomness (CSAR) [15] extends PeerReview's work to ensure that a Random Number Generator is accountable for all the numbers it generates with verifiable proof; this property fulfills the need of accountability for a randomized system. The authors in [25] study temporal accountability for medical sensor networks, also adopting some of the techniques of PeerReview. The authors in [26, 27] study accountable home appliances in smart grids and some of the methods of PeerReview are also adopted.

The authors in [16] propose two approaches to enable accountability in hosted virtual networks. The first approach leverages network measurement techniques to detect violations of Service Level Agreements (SLAs), and the second approach re-architects a router to prevent SLA violation beforehand. The authors in [17] have proposed a layered trust management framework to help email receivers eliminate their unwitting trust and provide them with accountability support. Re-Explicit-Congestion-Notification (Re-ECN) [20, 21] is able to locate congestion points in a network such that upstream parties causing congestions can be identified; Re-ECN requires changing the current TCP specification. The authors in [22–24] have proposed an accountable administration model for operating systems where all system administrators can be accounted for even if they are untrustworthy. The authors in have proposed Flow-Net [4–6] which is a logging mechanism that captures events and the relations among them for system accountability. The authors in [8, 9] extends Flow-Net to have multiple resolutions. The authors in [9, 10] study accountable logging and logging overhead. Flow-Net can be implemented as an OS kernel service [22, 23] to capture, log, and audit events with multiple granularities, and thus become a system-level tool for evidence generation in the design of accountable systems. Based on the systems we have surveyed, we conclude that one needs to specify three core elements for an accountability scheme, i.e., entity, event, and evidence, in which an entity can generate various events, while evidence is used to link an event to the responsible entities.

Despite the abundant research efforts, accountability has not been widely deployed in real world settings. Researchers have designed and tested several systems [11, 15, 19, 22, 23] for experimental purposes. In terms of performance evaluation, prior studies mainly focus on general performance metrics such as system response time, communication overhead, and throughput, while the core property, i.e., accountability, has not been well evaluated. To this end, a generic model is desired to assess accountability for the aforementioned systems. P-Accountability is an attempt to fulfill this demand. Even though the papers [9, 10] consider accountable metrics, they focus on accountable logging while this paper focuses on root-cause of accountability, which is the core aspect of accountability and is more important. Note that a short and preliminary version of this work was presented in the conference [32].

The authors in [33–35] provides verifiable proofs for cloud out-source data including image data and image data retrievals. The authors in [36, 37] studies provable digital evidence for networks.

2.2 Theoretical Definitions of Accountability

Security quantification [38–40] has been studied in recent years. There are a few prior works attempting to formally define accountability. However, in most definitions, the degree of accountability decreases due to internal problems such as cryptographic design flaws. Our model focuses on external and practical factors (e.g., network dynamics) that affect the degree of accountability. These factors may not be considered when the protocol is designed in the first place, but they may become significant when the system is running in real world.

The authors in [41] develop a mathematical model that employs an inductive method [42] to verify accountability protocols. Two factors including validity of evidence and fairness are used to verify the accuracy of accountability protocols. The model has been applied to protocol analysis for a certified email protocol and a non-repudiation protocol.

The authors in [43] propose an accountability model based on I/O automata, Communicating Sequential Processes, and discrete timed process algebra. The model enables auditors to identify dishonest party in a protocol with pre-defined specification. The issue of the proposed model is its inability to handle cryptography.

The authors in [44] present a general definition of accountability based on π - calculus [45] and IO/automata, as well as with interpretations in both symbolic and computational models. Informally, the proposed model highlights two features of accountability: (1) fairness implies that honest parties will not be falsely blamed; (2) completeness, on the other hand, implies that dishonest parties will be blamed. In addition, the authors have applied the proposed model to three protocols as case studies.

The authors in [46] propose an accountability model that firstly takes anonymity into account, as some applications requires keeping parties anonymous unless someone breaks the security policy. From this point of view, the proposed model is more versatile.

3 A Flat Model for P-Accountability

A typical notation style is employed throughout the paper: capitalized letters like V and E, represent set, and lowercase letters like v and e, denote members of corresponding sets.

3.1 A Flat Model

The major concern of prior studies [11–13, 15, 17, 18] is to generate verifiable evidence that is used to hold each entity accountable for its actions (i.e., events). However, the entities and events in different contexts differ. Therefore, a networked system can be modeled as $Q = (V, E)$, in which V and E are the entity set and the event set, respectively. Precisely, $V = \{v|v$ is an entity in the system$\}$, and $E = \{e|e$ is an event in the system$\}$. Typically, an event is caused by one or multiple entities. Let V_e denote a set of entities that cause an event e. A typical accountable system should be able to handle blame assignment, which can be modeled as a mapping function: $\alpha{:}E \rightarrow \{V_e|V_e \subseteq V\}$ that takes as input an event e and returns an entity (or entities) generating (or causing) the event. Ideally, the mapping function will always output the correct results. The ideal situation is named as a *perfect mapping* (*PM*). Formally, a mapping becomes a PM if and only if $\alpha(e) = V_{e|PM} \subseteq V$, where $V_{e|PM}$ is the complete set of responsible entities. In real world systems, however, perfect mapping is difficult to achieve due to many external factors such as network dynamics. In that case, we have $\alpha(e) = V_e \subset V_{e|PM} \subseteq V$, which indeed makes the system less accountable, as not all responsible parties can be identified. However, we still regard it as a *correct* mapping, because none of returned results are incorrect. For instance, an Internet packet is delayed by multiple admin domains including AD1, AD2, and AD3, while only AD1 is picked up by the accountable system responsible for the delay event. The mapping in this case is correct, but not perfect.

Definition 1 *Accountability* in a networked system $Q = (V, E)$, for $\forall e \in E$, if $\alpha(e)$ is a PM, then the system is accountable; otherwise the system is non-accountable. In other words, accountability $A(Q)$ is defined as

$$A(Q) = \prod_{e \in E} I(\alpha(e) \text{ is a PM}) \tag{1}$$

where $I(x)$ is an indication function, which returns 1 is if x is true and 0 otherwise.

Definition 1 implies that accountability is binary. Value 1 indicates perfect accountability for the system, while value 0 indicates that the system is completely non-accountable. As mentioned in Sect. 1, perfect accountability is usually not feasible to achieve in practical settings due to various uncertainties and rigorous conditions. In another word, the mapping could be correct but not perfect. Apparently, a binary value is not sufficient to describe the degree of accountability. To fill this gap, we propose P-Accountability, in which the prefix "P" represents both performance and probability. P-Accountability intends to be a performance metric for empirical study, as well as a probabilistic analysis approach.

Definition 2 *P-Accountability (flat model)* in a network system $Q = (V, E)$, where $|E|$ denotes the number of total events in E, P-Accountability $A_F(Q)$ is defined as follows:

$$A_F(Q) = \frac{1}{|E|} \cdot \sum_{e \in E} \left(\frac{|V_e|}{|V_{e|PM}|} \cdot I(\alpha(e) \text{ is correct}) \right). \tag{2}$$

Definition (2) is a fine-grained model, compared to definition (1). Meanwhile, when the system is perfectly accountable, i.e., for $\forall e \in E$, $I(\alpha(e)$ is a PM$) \equiv 1$, (1) and (2) are consistent, since $A_F(Q)$ equals one in this particular case. In this paper, once P-Accountability is defined in (2), the definition of accountability in (1) does not need to be used.

Model (2) gives an empirical definition which is applicable to practical systems, while it is not convenient for analysis. Let $P(e)$ be the probability that $\alpha(e)$ is a PM. We can then estimate P-Accountability using a probabilistic approach:

$$A_F(Q) \approx \tilde{A}(Q) \equiv P(e)$$

3.2 Usage of the Flat Model

The flat model can be generally applied to any flat networked system in which entities and events can represent appropriate system elements. For instance, AIP [12] is able to detect Internet hosts with forged IPs. Therefore, a network device with an IP can be a basic entity. Meanwhile, a basic event could be "host A 10.0.0.6 sends a message to host B 10.0.0.5". AIP is able to determine whether host A and host B are the machines they claimed to be. The flat mode can also be applied to Audit [18], which is able to hold each admin domain accountable for the packets passing through it. In this case, an admin domain is a basic entity, while an event could be that how a packet is handled within a domain, i.e., relayed or dropped. It is convenient to use the flat model to assess system accountability once we can determine the entity space and event space. We conduct a complete case study that applies P-Accountability to PeerReview in Sect. 5.

4 A Hierarchical Model for P-Accountability

P-Accountability defined in the previous section is only applicable to a flat network model, which drives us to dive into a more mixed network setting consisting of multiple hierarchies. To enhance the applicability of P-Accountability, we extend the flat model to a hierarchical model, which is able to handle such circumstances.

4.1 A Hierarchical Definition of Accountability

A networked system is a collection of a variety of network devices with different software running on the platform [47]. From a logical point of view, a hierarchical structure can be used to describe a network [48]. Each hierarchy consists of one kind of entities. Table 1 describes an example network with five hierarchies, in which the top one H_1 represents the entire network; depending on the context, H_1 could be as large as the Internet, or as small

Table 1 An example network with five hierarchies

Hierarchy	Description	Entities
H_1	Network	Internet, WAN, LAN, PAN, etc.
H_2	Sub-networks	Federal departments, universities, enterprises, other administrative domains, etc.
H_3	Devices	Routers, PCs, smart phones, etc.
H_4	Applications	Computer programs
H_5	Conversational Elements	Messages, packets, traffic flows, etc.

as a Personal Area Network (PAN) [49, 50]. H_2 is comprised of domains/sub-networks. In this paper, we loosely define H_2 which contains all sub-networks regardless of the internal relations among them. H_3 consists of all kinds of devices with network interface cards, while H_4 is a layer of software applications that run on top of devices of H_3. Apparently, a physical device will host multiple applications. H_5, the bottom layer, is made up of conversational elements which are essentially messages generated by H_4 applications and passed among H_3 devices. It is worth noting that the above structure can be customized depending on the particular system being considered. For example, when a system lies within a PAN, H_2 is not needed. Also, when we model a networked system with embedded devices connected together, a possible case is that each device may only host one application; e.g., the only mission of a temperature sensor node is to capture the environment temperature and report to the control node. In this case, H_3 and H_4 are essentially the same so it is meaningless to distinguish them.

This hierarchical structure provides more flexibility in the assessment of accountability. For instance, one of the challenges to defend against Denial of Service attacks is to identify the source of attacking traffic, which is possibly generated by some H_4 malware on some H_3 zombie machines from multiple H_2 organization networks across the world. Obviously, the level of the identified attack source will determine the degree of accountability. For instance, consider an event "a zero-day virus x running on a Bat machine y in company z launched an attack toward an Internet target", we call the virus program x the root cause of this event. If a system is able to identify the root cause for most events, its accountability level should be valued higher, compared to a system that can only pinpoint a causal entity from upper hierarchies such as the machine y or the company network z in this case.

To generalize the model, a network consists of n hierarchies. Let H_i $(1 \leq i \leq n)$ represent the ith hierarchy. To be consistent, we still let V and E denote a set of entities set and a set of events, respectively. However, in this model, an event is caused by entities at different hierarchies, meaning that the event cause has granularities. The definition of the mapping function α does not change, but the result entity set V_e should locate at a certain hierarchy. In addition, if different granularities are considered, $\alpha(e)$ might output multiple responsible entities in different hierarchies. For instance, if an event e is caused by v, which is a H_4 program running on a H_3 device u, then we regard both v and u are correct results of $\alpha(e)$. Apparently, v has a finer granularity than u; as such, v is a better candidate to describe the event cause. Essentially, a more accountable system should have a higher chance to identify responsible entities with finer granularity. Therefore, the notion of perfect mapping is redefined as "for all $e \in E$, the mapping result V_e is correct, complete, and is located at the deepest hierarchical level". A result being the deepest means that the system is able to find a correct result with the finest granularity.

Definition 3 *P-Accountability (hierarchical model)* in a networked system, P-Accountability $A_H(H)$ is defined as follows:

$$A_H(H) = \frac{1}{|E|} \cdot \sum_{e \in E} \left(\frac{d_e}{d_{e|PM}} \cdot \frac{|V_e|}{|V_{e|C}|} \cdot I(\alpha(e) \text{ is correct}) \right) \tag{3}$$

In the above definition, d_e and $d_{e|PM}$ denote the indices of the hierarchies where V_e and $V_{e|PM}$ are located at, respectively. $V_{e|C}$ denotes the complete and correct entity set for event e at hierarchy d_e. Obviously, if V_e lies in a higher hierarchy than $V_{e|PM}$, then event e is less accountable, because existing evidence does not generate a PM that points to the root causal entities.

4.2 Example: P-Accountability for AudIt

AudIt enables Internet Service Providers (ISPs) to proactively send feedback to the traffic source regarding link quality. The proposed hierarchical model can be applied to AudIt, for which we have H_2 admin domains, H_3 PCs and routers, as well as H_4 applications. The main event for an AD is relaying traffic, i.e., a packet could be relayed or dropped. Our model assigns a higher value of accountability to the system, if a H_3 router or a network interface card can be identified being responsible for an event of packet loss. Also, the corresponding version of the probabilistic model can be given as the probability that a responsible router or network interface card can be identified for a packet drop/delay event.

5 Applying P-Accountability to PeerReview

5.1 PeerReview Overview

PeeReview [11] implements a set of protocols that apply to generic distributed systems. PeerReview offers robust Byzantine fault detection such that nodes with misbehaviors can be witnessed by honest nodes, which produce evidence to irrefutably link a faculty or malicious node with bad actions. The following describes the assumptions of PeerReview:

- Each node owns a deterministic application, indicating that a certain input will yield the same output regardless of the current state of the system. PeerReivew relies on a reference mechanism to examine the behavior of nodes. In another word, output produced by the reference will be compared to the one produced by a node, and if they are not matched, the node should be flagged.
- Each message will be eventually transmitted to the destination, if sufficient retransmissions are applied.
- Identify of a node cannot be forged due to the security of public key infrastructure.
- A node can be indicated as either ‘trusted’, ‘suspected’, or ‘exposed’.
- A set of witness $\omega(k)$ is used to closely monitor the behavior of node k; if k is detected faulty, other nodes in the system will be notified by the witnesses.

There are six corner stones for PeerReview, including commitment protocol, tamper-evident logs, audit protocol, consistency protocol, evidence transfer protocol, and challenge/response protocol. In addition, PeerReview uses α_k^j to denote the evidence, which is essentially a signed statement by node j with its private key. With α_k^j, another node i can verify that node j has correctly logged event e_k, which specifies j’s action at the moment, as well as all events before e_k. PeerReview ensures that node j is unable to falsify e_k without being detected, because all evidence will be available to the entire network, and the witness scheme makes sure that a node will be examined periodically.

Results in [31] show that PeerReview cannot achieve a high degree of accountability in a complex network environment (e.g., the Internet) due to the end-to-end packet dynamics [51]; in addition, false accusations may happen because messages could be eventually lost. Eventual message loss means that both the direct and the witness assisted deliveries fail, thus the message will never reach the destination.

5.2 Network Model

The flat model suits PeerReview very well because there is merely one hierarchy which contains all nodes (i.e., hosts) in a distributed system. Before further analysis is given, we introduce some notations:

$Q(V, E)$ A networked system supported by PeerReview. V and E denote the entity set and the event set, respectively. More concrete definitions of V and E will be given later in this section

V_H A set of honest nodes in the system such that $V_H \subseteq V$

φ The fraction of faulty nodes. Note that φ is not unchanged. It could be low when the system is started, but it will increase as some honest nodes may be compromised and become faulty or malicious. Later on, φ may shrink due to the eviction of compromised nodes in an accountable system

w The size of witnesses. In this paper, we assume that the all nodes have the same witness size, which is a constant value w, i.e., $\forall v \in V$, $|\omega(v)| = w$. In the simulation, the w can be adjusted as a parameter

5.3 P-Accountability of PeerReview

In order to evaluate the system accountability for PeerReview, we first give analysis that how a node can judge another one, as well as the underlying reasons. Nodes in a networked system can have three possible statuses defined in set $O = \{$Correct (C), Faulty (F), Ignorant (I)$\}$, and we let o_v be the node v's status. An ignorant node refers to a node that is unresponsive to any incoming message. A faulty node refers to a node whose behavior can be arbitrary. Rigorously, an ignorant status is a special case of the faulty status. A correct node refers to an honest node whose actions comply with the protocol specification. When a node is compromised, it becomes be either faulty or ignorant.

An indication of a node refers to the way it is judged by other nodes. PeerReview has given three kinds of indications from a set $U = \{$Trusted (T), Exposed (X), Suspected (S)$\}$. Each node v keeps a table of indications for all nodes in the network. Let $\Gamma_v = \{ < i, u_{i,v} > | i \in V, u_{i,v} \in U\}$ denote the table kept at node v. $u_{i,v}$ is an indication of node i from v's viewpoint. As such, a node can be trusted, exposed, or suspected by any other node, and the indication will be recorded. When the system just starts, each node v's indication table will be initialized as $\Gamma_v = \{ < i, u_{i,v} > | \forall i \in V, u_{i,v} = T\}$; in other words, node v trusts every other node in the initial state. However, Γ_v will be dynamically updated as the system is running.

Figure 1 shows three kinds of indications along with the corresponding three node statuses. Formally, let $u_{i,v}|o_i$ denote the result of i indicated by v, given that the real status

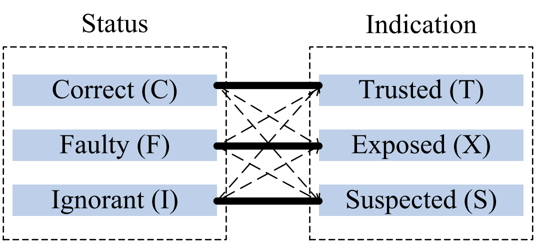


Fig. 1 Statuses and indications

Table 2 Nine indication possibilities of $u_{I,v}|o_I$

Accurate indication	Error
$T|C$, $X|F$, $S|I$	$X|C$, $S|C$, $T|F$, $S|F$, $T|I$, $X|I$

of node i is o_i. For instance, $(u_{i,v} = T)|(o_i = C) = T|C$ means that node i is 'Trusted' by node v when node i is a correct node. From a combination point of view, apparently, there are nine possibilities, listed in Table 2.

Table 2 shows that there are three accurate indications and six false indications, i.e., errors. In this paper, we regard message loss as the only cause of errors. As mentioned, PeerReview enables a temporarily lost message to eventually reach the destination, which essentially turns off message loss. In theory, a PeerReview-supported system is error free and perfectly accountable, while in reality this assumption is too strong.

The PeerReview primitives determines that event $X|I$ will never occur, because no evidence can be provided to expose an ignorant node. Even under the new assumption that a message may never reach the destination, the fact of absence for event $X|I$ will not change.

Definition 4 *False positives* cover two cases: (1) a faulty node is Trusted, i.e., $T|F$; (2) an ignorant node is Trusted, i.e., $T|I$.

Definition 5 *False negative* cover two cases: (1) a correct node is Exposed, i.e., $X|C$; (2) a correct node is Suspected, i.e., $S|C$.

For PeerReview, the event space is defined as $E = \{(i,j)|\forall i \in V, \forall j \in V_H$, node i is indicated by node $j\}$. The size of event set is thus $|E| = |V| \cdot |V_H|$. We also define a PM as a node i being correctly indicated by node j. Therefore, P-Accountability for PeerReview can be defined:

$$A_F(Q) = \frac{\sum_{v \in V} (\text{Number of PMs at node } v)}{|V| \cdot |V_H|} \tag{4}$$

For $\forall v \in V$, let P_C be the probability of node v correctly indicating the status of any other node in V, and P_{FP} and P_{FN} denote the probabilities of false positive and false negative, respectively. We obtain the probabilistic model of P-Accountability $\tilde{A}(Q)$ as follows.

$$\tilde{A}(Q) = P_C = 1 - P_{FP} - P_{FN} \tag{5}$$

Equation (5) is a probabilistic approach to measure accountability. In order to calculate P_C we will first calculate the End-to-end (E2E) Message Loss Probability (MLP) and eventual MLP, and then obtain P_{FP} and P_{FN}, respectively.

5.3.1 E2E MLP and Eventual MLP

Let P_0 be the E2E MLP between any two end nodes. In PeerReview, however, a temporary message loss event does not imply an eventual message loss, as the challenge/response protocol will retransmit the lost message later on using a challenge message. Figure 2 describes this situation: node j sends a message to i, while the message is lost due to network issues. As there is no response from i, node j decides to initiate the

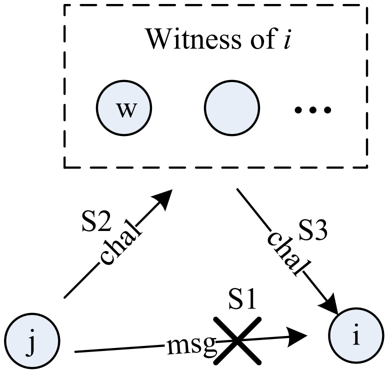


Fig. 2 Communication in PeerReview

challenge/response protocol by creating a challenge, and send it to i's witnesses, i.e., $\omega(i)$, which will forward the challenge to i. Let P_e denote the probability that a message is eventually lost, we can conclude that a message is eventually lost if and only if all challenges forwarded by witnesses are lost. The statement yields the following calculation.

$$\begin{aligned} P_e &= P_0 \cdot \sum_{r=0}^{w} \left(\begin{pmatrix} w \\ r \end{pmatrix} \cdot (1 - P_0)^r \cdot P_0^{w-r} \cdot P_0^r \right) \\ &= P_0^{w+1} \cdot \sum_{r=0}^{w} \left(\begin{pmatrix} w \\ r \end{pmatrix} \cdot (1 - P_0)^r \cdot 1^{w-r} \right) \\ &= P_0^{w+1} (2 - P_0)^w \end{aligned} \tag{6}$$

5.3.2 *False Positive*

To indicate the status of a node i, node j needs to constantly fetch a set of evidence from i's witnesses. Once there is any accusation from the evidence set pointing to i, node i will not be trusted by j anymore. Nevertheless, we cannot assume that all witnesses are trustworthy as some of them may become compromised and controlled by hackers. A compromised witness is capable of accusing a correct node or tolerating a faulty node with fabricated evidence. Luckily, PeerReview is able to prevent a piece of evidence from being forged. Therefore, as long as at least one witness is honest (which is also assumed by PeerReview), the only cause of false positive is that all messages containing evidence are lost. Therefore, after one operation of evidence transfer protocol, we have

$$P_{FP} = P_e^{w \cdot (1 - \phi)} \tag{7}$$

The system will run the evidence transfer protocol when it is needed, and each run is independent. In other words, if we take a snapshot of the system at a certain moment, we will discover all status indications only depends on the latest run of the evidence transfer. The issue is that as φ keeps growing, resulting $\lim_{\varphi \to 1} P_{FP} = \lim_{\varphi \to 1} P_e^{w(1-\varphi)} = 1$. This means that if all witnesses of i turns faulty, i will always be trusted.

5.3.3 False Negative

False negative consists of $X|C$ and $S|C$. The cause of $X|C$ is twofold: (1) if i is a witness of j, $X|C$ will be caused by message loss in auditing; (2) if i is not a witness of j, $X|C$ will occur because some witnesses of j, which also suffer $X|C$, forward wrong evidence to node i. For the second cause, let r be the number of honest witnesses that suffer error $X|C$, so r pieces of faulty evidence will be generated and distributed. If by any chance any of the r evidence messages arrives at node i, then I will also be suffering X|C. For convenience of derivation, we assume that for each time of audit one log segment will be sliced into $\bar{\mu}$ small pieces and loaded into $\bar{\mu}$ messages. The probability of false negative is given below:

$$P_{FN} = \Pr(X|C) + \Pr(S|C) \tag{8}$$

in which

$$\begin{aligned}\Pr(X|C) = {} & \frac{w}{N} \cdot \left(1 - (1 - P_e)^{\bar{\mu}}\right) \\ & + \left(1 - \frac{w}{N}\right) \cdot \sum_{r=0}^{w(1-\varphi)} \left(\binom{w(1-\varphi)}{r} \cdot \left(1 - (1 - P_e)^{\bar{\mu}}\right)^r \cdot \left(1 - P_e^r\right) \right)\end{aligned} \tag{9}$$

$$\Pr(S|C) = P_e + (1 - P_e) \cdot P_0 \tag{10}$$

A proof sketch of (9) is given as follows: Consider a correct node x. There are two cases in which node x will be exposed by another correct node y. Case 1: node y is one of node x's witnesses (with probability w/N), and at least 1 msg of $\bar{\mu}$ messages are eventually lost (with prob. $\left(1 - (1 - P_e)^{\bar{\mu}}\right)$). Case 2: node y is not node x's witness (with prob. $(1 - w/N)$); if there are r witnesses of node x that have already suffered $X|C$, they will pass the fault evidence (r in total) to node y. Therefore, node y will suffer $X|C$ if and only if at least one evidence of r reaches node y.

Based on Eqs. (5)–(10), we obtain $\tilde{A}(Q)$ for PeerReview.

Theorem 1 *In the PeerReview context, if eventual message loss exists during the system lifetime, the entire system will ultimately become entirely non-accountable.*

Proof In a practical system, it is likely that the eventual message loss rate is not equal to 0 all the time, thus $A_F(Q)$ will keep decreasing and ultimately reach 0, because that eventual message loss results in errors. If more and more messages are lost, more errors will show up and accumulate; in the end, the percentage of correct indications will drop to zero. In other words, P-Accountability reaches 0. Therefore, the system becomes non-accountable at all. □

6 Accountable Wireless Multi-hop Networks

PeerReview-supported systems mainly consider end-to-end communication, which suits generic distribute systems very well. However, for a multi-hop network, we argue that although the fundamental idea of PeerReview still applies, there are some critical challenges that require additional efforts. For example, relay nodes play an important role in a multi-hop network. When a relay is compromised, new vulnerabilities will emerge. PeerReview is unable to deal with a malicious relay as its function differs from an end node. This concern leads us to design Traceable PeerReview (TPR), which is an extension of the original version in the multi-hop network environment. In particular, a new protocol called Message Tracing protocol is designed and integrated into the original protocol set.

A wireless channel suffers various network dynamics such as interference, congestion, and packet loss/delay [52]. If a message is lost, any hop could be the cause of it. The default number of retransmissions for IEEE 802.11 is seven [53, 54], which opens the door of message loss given a bad channel. In addition, nodes in a system may be down due to misconfiguration, program crash, external attack, and so on, leading to an unstable wireless connection. Therefore, a message may never be able to reach the destination, i.e., message eventual loss. In this section, message loss is considered as a factor for the analysis of P-Accountability for TPR.

6.1 TPR Environment

A wireless multi-hop network presents some characteristics that are different from a generic distributed system:

1. In additions to being source and destination, a node also plays a third role of relay.
2. We define a path between nodes S and D as $Z_{S,D} = \langle \mathrm{S}, R_1, R_2, \ldots, R_l, \mathrm{D}\rangle$, where $R_1, R_2, \ldots, R_l$ are the relays. If a message chooses path $Z_{S,D}$, it takes $(l + 1)$ hops to reach the destination.
3. In addition to the E2E ACK, a new type of ACK called Hop ACK is enabled in the multi-hop environment. A Hop ACK is used to indicate successful message delivery between hops. In particular, if a Hop ACK is not received by the sender if after q-1 retransmissions, the sender will pick another route.

6.2 Problem Description

Potential vulnerabilities will emerge when simply apply PeerReview to the wireless multi-hop network. As shown in Fig. 3, a message travels through multiple relays to reach the destination. Let $Z_{S,D} = \langle \mathrm{S}, R_1, R_2, \ldots, R_l, \mathrm{D}\rangle$ be the path of the message, then the delivery will be successful if and only if (1) all relays in the path are honest, and (2) the message is not lost along the route. Consider the case that a relay, say R_k, is faulty, then there might be a few possible consequences:

1. R_k is inactive to messages coming from source S, meaning that the connection from S to D will never be established.

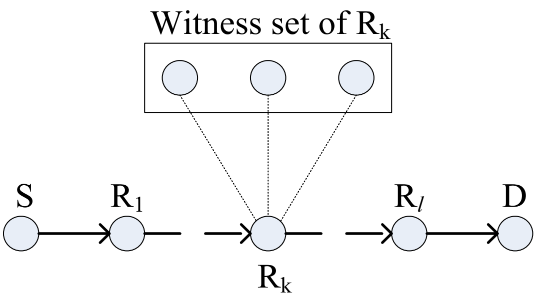


Fig. 3 Relays in a multi-hop network

2. R_k selectively or randomly relays the traffic from S; as such, recipient D is unable to obtain a complete message from the source.
3. R_k performs a replay attack by re-sending a message to D, which may cause a denial of service on D.
4. R_k may intentionally or mistakenly relay a message to a different recipient other than D, leading to a data leakage incident.

PeerReview is capable of dealing with the above problems only if the relay R_k keeps its log faithfully, because logs will be replayed by R_k's witnesses, who inform other nodes to expose R_k if there is any action that deviates from the reference. However, if R_k takes a more intelligent strategy to not leave any evidence about message relay in its logs, then PeerReview will not be able to expose it. For instance, a malicious relay R_k receives a message m_x, but instead of delivering it to the next hop, R_k discards m_x and does not update its log. Later on, when R_k's log is checked by the witnesses, m_x will never appear in the log entries, meaning that PeerReview is not able to expose a malicious node as such. However, it does not mean R_k can always be at large. Consider R_{k-1}, which might be honest, the predecessor of R_k within the route, has delivered m_x to R_k, and all log entries related to this message delivery exist in R_{k-1}'s log. Based on these info, the behavior of R_k can be inferred: if R_{k-1} received a hop ACK of m_x, R_k must have problem as it is certain that R_k has received m_x while failed to update the log. This example shows the basic idea of TPR: to expose a faulty relay, it is not sufficient to examine a single suspect; instead, a cooperative inspection approach that involves other nodes in the path is needed for evidence generation.

Traceable PeerReview (TPR) is highlighted by being able to: (1) detect the exact hop when a message disappears or manipulated, and (2) generate a piece of verifiable evidence for exposing the first malicious node in the route.

6.3 Traceable PeerReview

This subsection provides the technical details of Traceable PeerReview, the core of which is the Message Tracing protocol.

6.3.1 Modifications on PeerReview

Tamper-evident logs (Sect. 4.4 in [11]): we include four additional addresses needed to generate evidence: source, destination, last hop, and next hop, in the log entry for each individual message the passes through a relay.

6.3.2 Message Tracing Protocol

On top of the slight modification on logs, we add a Message Tracing protocol which is core of TPR. Equipped with message tracing, a source is able to initiate the challenge/response protocol if the E2E ACK is not received, and start to suspect D unless the challenge is properly addressed. If D is honest, then there are two possible reasons that D is suspected: (1) either the message or the ACK is lost, or (2) one of the relays along the path is faulty. We describe the Message Tracing below:

- *Step 1* Consider a particular message m_x that is sent from the source S to the destination D, if the E2E ACK is not received by S, then S will kick off the tracing process to examine the path from S to D. To start, source S creates a *Tracing* tag $\sigma_S(seq_x)$, which essentially is a signed message of the sequence number of m_x. The tracing tag is then sent to the relay, say R_1, right next to the source. R_1 will verify S's identity by decrypting the tag with S's public key, and then, if the tag is valid, R_1 retrieves all log entries that involve m_x and send them to S, which will replay the log entries to check if R_1 behaves honestly. Since the setting is deterministic, R_1's reaction involving m_x should be deterministic as well. As soon as R_1's behavior is validated, S delivers a *Pass* tag to R_1 which can be trusted by now. Next, R_1 will create a *Tracing* tag using its own private key for signature and start checking the validity of the next hop, say R_2. If, however, S determines that R_1 is faulty, a *Reject* tag is produced and R_1 won't be trusted.

- *Step 2* Consider a relay in the middle of the path, say R_i receives a tracing tag from the its last hop R_{i-1}, R_i is asked to send all log entries relevant to m_x to R_{i-1}. In particular, R_{i-1} checks the following about m_x in R_i: the source ID, the destination ID, the last hop node ID, and the next hop ID. If all of these identifiers are correct, then R_i is verified to have forwarded m_x honestly, or R_i is proved faulty. All log entries R_i sent to R_{i-1} become evidence that can be verified by a third party. If R_i happens to be D, the tracing for m_x is finished. Note that the log is tamper-evident, any malicious modification to the log will be detected by the witnesses.
- *Step 3* If all nodes from S to D are honest, the protocol will start tracing the acknowledgement, denoted by ACK(m_x), which should be sent from D. As such, D kicks off another round of tracing along the path through which ACK(m_x) travels.
- *Step 4* If a faulty node is detected. Related evidence will be distributed to the rest of the network through the evidence transfer protocol (Sect. 4.9 in [11]).

The design of Message Tracing can detect faulty relay nodes in most cases. However, when a recently compromised node which is also in the path blocks the tracing, it is difficult to detect the actual faulty node. Consider path $Z_{S,D}(m_x)$ where a message m_x is lost somewhere in the middle. Assume that node k is the faulty relay node which causes the loss of m_x, while node h turns into a faulty node just before the tracing procedure.

- *Case I* If node h is located before k in the path, then h can be identified by Message Tracing, and k will be detected if it causes message loss in the future. This case can be handled by the current protocol.
- *Case II* if node k is located before h in the path, Message Tracing will stop working since k becomes faulty and does not comply with the protocol any more. To fix it, we add a backward tracing scheme: if a faulty node blocks the tracing process in path from S to D, we will launch a reverse tracing of m_x starting from the destination. The difference is that a backward tracing will stop only if m_x appears in a node's event logs, and we can determine that this node is responsible for the loss of m_x since all subsequent nodes never received m_x.
- *Case III* if another node *h*' also turns faulty just before tracing, it is likely that the actual faulty node k is located in between h and h'. Therefore, both forward and backward tracing will not work, because the tracing cannot reach node k at this moment. In this case, k is not detected. The protocol will choose another path to redeliver m_x. Although k remains undetected, there is a high probability that k can be eventually detected as long as it keeps being malicious.

6.4 Traceable PeerReview Analysis

Let P_h be the probability that a message is lost in a hop, and let P_C denote the probability that the process of message tracing is finished; in other words, a message will never be eventually lost during tracing, meaning that the scheme either identifies a faulty relay, or the hop that causes message loss. Let $\tau_{i,j}$ be the hop count between nodes i and j. and let $\mathrm{R_k}$ be the first faulty node along the tracing path. We have

$$P_C = \left(1 - P_h^q\right)^{3\cdot\tau_{s,k}} \tag{11}$$

Proof of (11) The tracing will be successfully finished if and only if all relays can be reached. As mentioned in the tracing description, a hop checking involves three kinds of tracing messages including a tracing tag, a log segment, and a message indicating pass/reject. In addition, a message can be re-transmitted by up to q-1 times. Therefore, the probability of a successful one-hop checking is $(1 - P_h^q)^3$. There are two cases to be discussed according to $\mathrm{R_k}$'s location:

Case 1 R_k is located in the path before m_x arrives at D. Obviously, the tracing can only be finished if all hop checkings within the $\tau_{s,k}$ hops are successful, which has a probability of $\left(1 - P_h^q\right)^{3\cdot\tau_{s,k}}$.

Case 2 R_k lies in the path of ACK(m_x) from D to S. Similar to Case 1, we have

$$P_C = \left(1 - P_h^q\right)^{3\cdot\left(\tau_{s,d}+\tau_{d,k}\right)} = \left(1 - P_h^q\right)^{3\cdot\tau_{s,k}}$$

Theorem 1 *The message complexity of message tracing is O($\tau_{sd} + \tau_{ds}$), which is proportional to the hop count of the tracing path.*

Proof Assume that the hop level ACK/retransmission is enabled; as such a message can be retransmitted up to $q - 1$ times, the message tracing protocol requires that any relay to exchange three different messages (see Fig. 4) with its next hop node. If message loss and retransmission are taken into account, the maximum messages for a single hop is $3q$, and the maximum number of hops is $\tau_{sd} + \tau_{ds} - 1$; thus, the message overhead is computed as $3q \cdot (\tau_{sd} + \tau_{ds} - 1) = O(\tau_{sd} + \tau_{ds})$.□

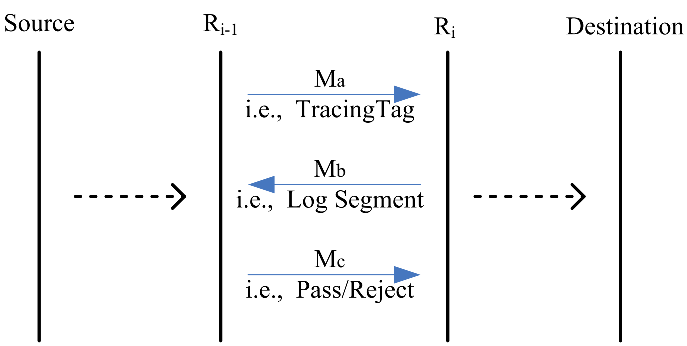


Fig. 4 Tracing in path $Z_{S,D}(m_x)$

6.5 P-Accountability on TPR

Since events $T|C$, $X|C$, and $S|C$ are mutually exclusive, then $\Pr(T|C) = 1 - \Pr(X|C) - \Pr(S|C)$. Similarly, $\Pr(X|F)$ and $\Pr(S|I)$ can be computed. Therefore, for TPR, P-Accountability $\tilde{A}(Q)$ can be defined below:

For any node $v \in V$,

$$\begin{aligned}\tilde{A}(Q) &= \Pr(\text{node } v \text{ makes correct indications})\\ &= \Pr(T|C) + \Pr(X|F) + \Pr(S|I)\\ &= 1 - (\Pr(X|C) + \Pr(S|C))\\ &\quad + 1 - (\Pr(T|F) + \Pr(S|F))\\ &\quad + 1 - (\Pr(T|I) + \Pr(X|I))\\ &= 3 - \Pr(X|C) - \Pr(S|C) - \Pr(T|F)\\ &\quad - \Pr(S|F) - \Pr(T|I) - \Pr(X|I)\end{aligned} \tag{12}$$

6.5.1 Eventual E2E Message Loss

Given end nodes i and j, the probability of the E2E message (sent from i to j) loss is

$$P_{t(i,j)} = 1 - (1 - P_h^q)^{\tau_{i,j}} \tag{13}$$

and the probability of *eventual* E2E message loss is

$$P_{E(i,j)} = P_{t(i,j)} \times \left(\sum_{k=0}^{w\cdot(1-\varphi)} \left(\binom{w\cdot(1-\varphi)}{k} \cdot \prod_{r=1}^{k} \left(1 - P_{t(i,r)}\right) \cdot \prod_{l=1}^{w\cdot(1-\varphi)-k} \left(P_{t(i,l)}\right) \cdot \prod_{r=1}^{k} \left(P_{t(r,j)}\right) \right) \right) \tag{14}$$

Equation (14) is another version of Eq. (6) in the multi-hop scenario. The basic idea of calculating (6) still applies: if a message m_x is lost (with probability $P_{t(i,j)}$), source i will transfer challenges to its witnesses. Message m_x is eventually lost if and only if all challenges forwarded by the witnesses are lost.

6.5.2 Error Analysis

The six error types are in line with the ones introduced in Sect. 5. Also, we treat normal faulty (NF) nodes and relay faulty (RF) nodes differently. Let $V_{NF} \cup V_{RF} = V_F$ and $V_{NF} \cap V_{RF} = \emptyset$; let p_{nf} be the probability of a node being a normal faulty node, and let p_{rf} be the probability of a node being relay faulty node. Consider nodes i and j, given that i is honest, then we can provide analysis for the errors.

- *Error* $(u_{j,i} = X)|(o_j = C) = X|C$. TPR does not introduce additional error causes for $X|C$, because an honest node can always provide a message of verifiable log entries to its previous hop during tracing. Thus, Eq. (9) in the multi-hop setting can be given as

$$\Pr(X|C) = \frac{w}{|V|} \cdot \left(1 - \left(1 - P_{E(i,j)}\right)^{\overline{\mu}}\right) + \left(1 - \frac{w}{|V|}\right) \cdot \sum_{r=0}^{w(1-\varphi)} \left(\binom{w(1-\varphi)}{r} \cdot \left(1 - \left(1 - P_{E(i,j)}\right)^{\overline{\mu}}\right)^{r} \cdot \left(1 - P_{E(i,j)}^{r}\right)\right) \tag{15}$$

- Error $S|C$ and $S|F$:

  - Case C1: node i and j are source and destination, respectively. Node i will suspect another node j, regardless of its status, is caused by eventual message loss, which may either occur (1) in the route from i to j for the original message, or (2) in the route from j to i for the ACK message.

$$\Pr(C1) = P_{E(i,j)} + \left(1 - P_{E(i,j)}\right) \cdot P_{t(i,j)} \tag{16}$$

  - Case C2: Nodes i and j are next to each other during tracing. If j is the successor of i, and that j's status is unknown.

$$\Pr(C2) = P_h^{2q} \cdot P_{E(i,j)} \tag{17}$$

  - Because of the mutual exclusivity between C1 and C2, the following equation holds:

$$\Pr(S|C) + \Pr(S|F) = \Pr(C1) + \Pr(C2) \tag{18}$$

- *Error $T|F$ and $T|I$.* TPR introduces another cause of T|F: if the evidence for exposing a faulty relay node is lost, then the rest network will keep trusting the faulty node. We then have

$$\Pr(T|F) + \Pr(T|I) = \frac{w}{|V_C|} \cdot p_{rf} \cdot P_{E(t,i)} + \left(1 - \frac{w}{|V_C|}\right) \cdot p_{nf} \cdot \prod_{r=1}^{w \cdot p_c} P_{E(r,i)} \tag{19}$$

- *Error $X|I$.* The adoption of TPR will also not expose an ignorant node as it is always unresponsive. Therefore, Pr $(X|I) = 0$

Combining Eqs. (12)–(19), we obtain $\tilde{A}(Q)$ for TPR.

7 Evaluation

This section presents numerical results in Sect. 2.1 and simulation outcomes in Sect. 2.2 to validate the proposed quantitative model. Numerical results are those obtained via mathematical models and simulation results are those obtained via simulating the methods. The simulated system is a wireless multi-hop ad hoc network; however, the proposed method can be used in any networks including wired and wireless networks.

7.1 Numerical Results

7.1.1 PeerReview

$\tilde{A}(Q)$ for PeerReview can be numerically plotted in Fig. 5. In this figure, P-Accountability is a probability affected by E2E message loss probability and witness size. The reason that a larger witness set increases P-Accountability is that more witnesses will help forward the challenge once a message is lost.

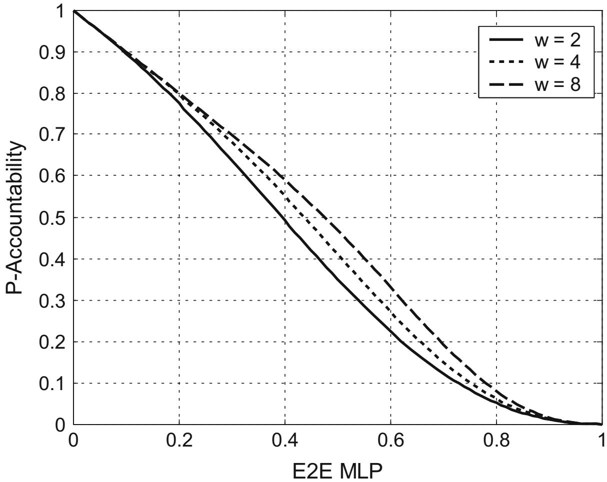
7.1.2 Traceable PeerReview

If no tracing messages are eventually lost in the tracing route, we call the tracing a successful one. The probability of successful tracing is determined by two factors including hop re-transmissions, i.e., $(q - 1)$, and the average number of hops τ_{avg} between the source and the first faulty relay. Figure 6 is obtained via Eq. (11) and shows the effects of other parameters on P_C, and we assume that Byzantine faults are directly related to the message loss. Figure 6 shows that P_C increases as q increases, indicating a positive impact on P_C, while τ_{avg} causes a negative impact on P_C. To make sense of the observation, an increased number of retransmissions reduces the chance of message loss, while a higher number of hops does the opposite. This claim is consistent with the analysis result in Fig. 7, which shows how P-Accountability is affected by the message loss probability P_h. The higher P_h is, the lower P-Accountability will be.

7.2 Simulation Results

TPR is an extension of PeerReview, and our simulation program is also extended from the original PeerReview software library [55]. Figure 8 describes the network stack of a computing device with TPR enabled. TPR is independent of applications, meaning that TPR can be installed as a plugin so that general Byzantine faults can be detected with verifiable evidence. If we follow the example in Table 1, we can identify two hierarchies in

Fig. 5 The E2E MLP and P-Accountability for PeerReview when $\varphi = 0.1$



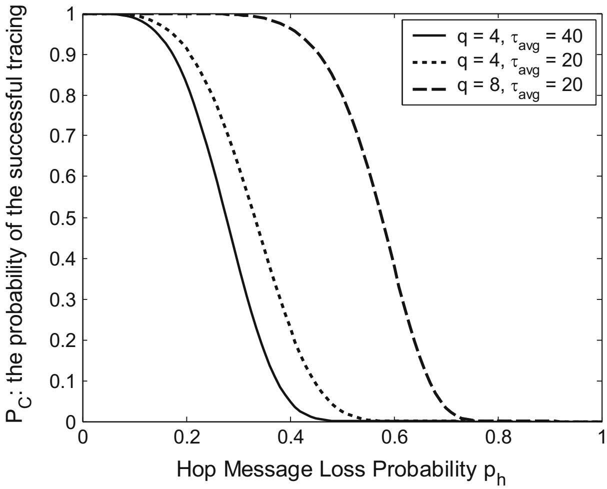


Fig. 6 Hop MLP and P_C for TPR ($\varphi = 0.1$)

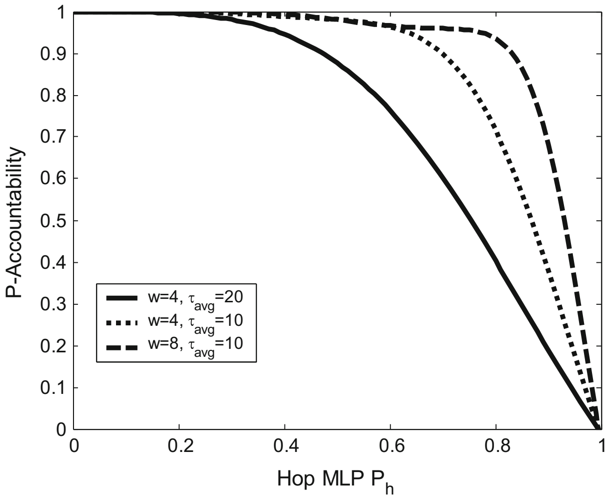


Fig. 7 Hop MLP and P-Accountability for TPR ($\varphi = 0.1$)

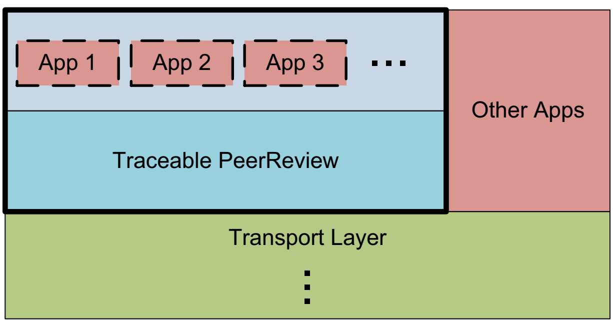


Fig. 8 Framework of TPR enabled system

any TPR-supported systems, including H_3 that contains all nodes, and H_4 that consists of multiple applications.

TRP will create a log file for each application, and each node is associated with one key pair as its identity. As shown in Fig. 8, all applications are running independently. In addition, witnesses are in H_3, meaning that the same set of witnesses are shared across applications on a node.

As a single device has multiple applications running on it, we can then define a metric that is aligned with definition (3). The event space can be given as "the status of application x on node y is indicated by every other correct node in the system". The entity space then consists of all H_3 nodes and all H_4 applications. Let M be the number of applications running on a node, the size of event space is $M \cdot |V| \cdot |V_H|$.

$$A_H(Q) = \frac{\sum_{m \in M} \sum_{v \in V_H} \sum_{k \in V} \left(I(u_{k,v} \text{ is correct}) \right)}{M \cdot |V| \cdot |V_H|} \tag{20}$$

The numerator in the right hand side of the above equation is the total number of correct indications for node v. Our metric in (20) is in line with the one given in (3). However, the notion of perfect mapping in this setting means an ability to identify a responsible application in H_4, which has a finer granularity than pinpointing a responsible node in H_3. In addition, when judging an ignorant node, there is no way to determine the honesty of the applications running on it, because the node itself is totally unresponsive.

7.2.1 P-Accountability on Traceable PeerReview

PeerReview assumed that a message will be eventually received by the destination as long as the lost message is retransmitted sufficiently enough. Our simulation assigns a certain probability that a message will get lost in each hop; in addition, we set a limit on the number of retransmissions, meaning that there is a chance that a message is eventually lost. The goal of the simulation is to figure out how P-Accountability is influenced by message loss in a TPR-enabled environment. For that purpose, we have simulated a wireless multi-hop network deployed in a square area. There are two applications running on each node, and all nodes are homogeneous. Table 3 lists all parameters of the simulation.

The percentage of each indication type is shown in Table 4. We can observe that when p_h rises from 0.2 to 0.8, the percentage of correct indications has dropped by 52.6%, meaning that the worse the channel quality is, the less accountable the system would be. In contrast, the percentage for each error type increases. Another finding is that over the five error types, three of them, including $X|C$, $T|F$, and $T|I$, rarely occur, which is also expected. For $X|C$, it will be triggered if and only if a message is lost in the audit protocol so that a witness cannot generate output as expected. In that case, a temporary $X|C$ will occur, although it may be addressed later on by the challenge/response protocol. $T|F$ and $T|I$ are caused by the message loss during evidence transfer. $S|C$ and $S|F$, on the other hand, do

Table 3 Simulation parameters

Parameter	Values
P_h—hop message loss probability	0.0, 0.1, 0.2 up to 1.0
N—node # in total	[50, 100, 250, 500]
φ—initial faulty nodes rate	[0.0, 0.05, 0.1, 0.2]
w—witness size	[2, 4, 8]

Table 4 Error ratios for traceable PeerReview

	P_h		
	0.2 (%)	0.4 (%)	0.8 (%)
Indication			
Correct	73	55	37
$X\|C$	1.5	2.3	1.7
$S\|C$	12	19	28
$T\|F$	2.4	3.9	4.2
$S\|F$	9	15	21
$T\|I$	2.0	4.9	8.1

appear quite often, as they happen once a message or its ACK is lost. For errors, some of them will be resolved, while the rest contribute to make the system less accountable. According to the analysis in Sect. 4. E.2, $X|I$ will never exist, and our simulation result is aligned with this conclusion.

In our simulations, we change hop message probability in our programs under different simulations so that we can obtained Fig. 9, which shows the relationship between P-accountability and hop message loss probability. On the other hand, Byzantine faults can cause message loss and in the simulation, we assume that Byzantine faults are the only reason for the loss besides some network protocol limitation such as collisions and retransmissions. Figure 9 demonstrates how P-Accountability is affected by the hop message loss probability, i.e., p_{h}. The result is consistent with the numeric outcome in Fig. 7. It confirms two findings: (1) P-Accountability decreases as p_h increases; (2) when p_h is fixed, the more witnesses there are, the more accountable the system would be. It turns out the chance of errors will also be decreased by a larger size of witness set, as more witnesses will help forward the challenge, and a node being challenged have to justify itself to every member in the witness set. As such, it is more likely that the response will reach the other end. Moreover, an evidence message is less likely to be lost, leading to a reduced chance of the occurrence of $T|F$ and $T|I$.

In our simulation, some relay faulty nodes are inserted into the network. Figure 10 shows that comparison of a TRP-enabled system and a PeerReview-enabled system. We can discover that when the hop message loss probability is fixed, the TPR-enabled system

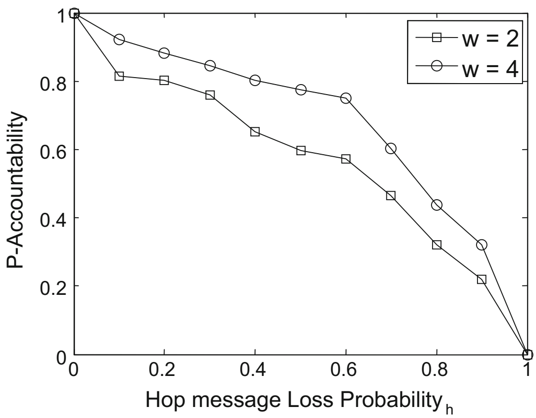


Fig. 9 Simulation result—P_h and P-Accountability

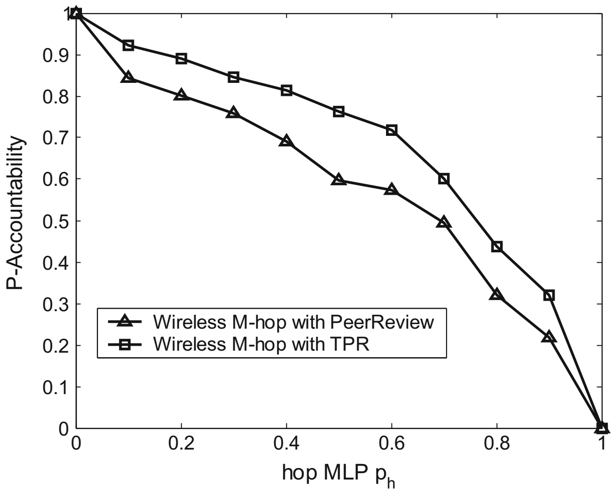


Fig. 10 Simulation result—hop MLP and P-Accountability for TPR

presents a higher degree of accountability, because TRP is able to expose the faulty relays, while the original PeerReview is unable to.

8 Conclusion

P-Accountability is our attempt to quantify and evaluate accountability for networked systems. To achieve this goal, we have developed a flat model and a hierarchical model. Both models are generic and widely applicable. The flat model suits systems with flat structure, meaning that all concerned entities are homogeneous. The hierarchical model is an advanced version that applies to more complex environments. P-Accountability should be customized before applied to a practical system.

To examine our model, we first conduct a case study that applies P-Accountability to PeerReview. In addition, we propose Traceable PeerReview which enables PeerReview to be used in wireless multi-hop network by making relay nodes accountable. We conduct extensive simulation to validate our analysis. We show that in the PeerReview case, message delivery plays a crucial role in determining accountability. Our simulation results are well approached by the numeric results in Sects. 5 and 6. We also discover that Traceable PeerReview increases P-Accountability in the simulation, compared to the original version. Both our analysis and simulation results show that P-Accountability makes accountability more flexible, adjustable, and fine-grained for a networked system. This research will also generate many potential applications; for example, it offers a practical approach to study the trade-off between accountability and system overhead, as the latter is usually easy to quantify, while P-Accountability explores a feasible way for the former. This will greatly benefit the design of distributed and networked systems that regard accountability as the first class requirement.

Acknowledgements This work was supported in part by the US National Science Foundation (NSF) under grants CNS-0716211, CNS-0737325, CCF-0829827, and CNS-1059265. The authors would like to thank the anonymous reviewers for their valuable comments as well as the authors of PeerReview for offering their program source code.

References

1. Department of Defense. (1985). *Trusted computer system evaluation criteria*. Technical Report 5200.28-STD, Department of Defense.
2. Yumerefendi, A. R., & Chase, J. S. (2005) The role of accountability in dependable distributed systems. In *Proceedings of HotDep*
3. Yumerefendi, A. R., & Chase, J. S. (2004) Trust but verify: accountability for network services. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop* (p. 37). Leuven: ACM.
4. Xiao, Y. (2009). Flow-net methodology for accountability in wireless networks. *IEEE Network, 23*(5), 30–37.
5. Xiao, Y., Meng, K., & Takahashi, D. (2012). Accountability using flow-net: Design, implementation, and performance evaluation. *Security and Communication Networks, 5*(1), 29–49. doi:10.1002/sec.348.
6. Takahashi, D., Xiao, Y., & Meng, K. (2014). Virtual flow-net for accountability and forensics of computer and network systems. *Security and Communication Networks, 7*(12), 2509–2526.
7. Fu, B., & Xiao, Y. (2014). A multi-resolution flow-net methodology for accountable logging and its application in TCP/IP networks. In Proceedings of the IEEE international conference on communications 2014 (IEEE ICC 2014).
8. Fu, B., & Xiao, Y. (2015). A multi-resolution accountable logging and its applications. *Computer Networks, 89*(4), 44–58.
9. Fu, B., & Xiao, Y. (2012). Q-Accountable: A overhead-based quantifiable accountability in wireless networks. In *Proceedings of IEEE consumer communications and networking conference (IEEE CCNC 2012)* (pp. 138–142).
10. Fu, B., & Xiao, Y. (2014). Accountability and Q-Accountable logging in wireless networks. *Wireless Personal Communications, 75*(3), 1715–1746.
11. Haeberlen, A., Kouznetsov, P., & Druschel, P. (2007). PeerReview: Practical accountability for distributed systems. In *Proceedings of twenty-first ACM SIGOPS symposium on operating systems principles* (pp. 175–188). New York, NY: ACM.
12. Andersen, D., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D., & Shenker, S. (2007). Holding the internet accountable. ACM HotNets-VI.
13. Andersen, D., Feamster, N., Koponen, T., Moon, D., & Shenker, S. (2008). Accountable internet protocol (AIP). In *Proceedings of the ACM SIGCOMM 2008 conference on data communication* (pp. 339–350). New York, NY: ACM.
14. Mirkovic, J., & Reiher, P. (2008). Building accountability into the future internet. In *4th workshop on secure network protocols, 2008* (pp. 45–51). NPSec 2008.
15. Backes, M., Druschel, P., Haeberlen, A., & Unruh, D. (2009). CSAR: A practical and provable technique to make randomized systems accountable. In *Proceedings of the 16th annual network and distributed system security symposium (NDSS'09), San Diego, CA*.
16. Keller, E., Lee, R., & Rexford, J. (2009). Accountability in hosted virtual networks. In *Proceedings of the ACM SIGCOMM workshop on virtualized infrastructure systems and architectures (VISA)*.
17. Liu, W., Aggarwal, S., & Duan, Z. (2009). Incorporating accountability into internet email. In *Proceedings of the 2009 ACM symposium on applied computing* (pp. 875–882).
18. Argyraki, K., Maniatis, P., Irzak, O., Ashish, S., Shenker, S., & Epfl, L. (2007). Loss and delay accountability for the internet. In *IEEE international conference on network protocols, 2007* (pp. 194–205). ICNP 2007.
19. Yumerefendi, A. R., & Chase J. S. (2007). Strong accountability for network storage. *ACM Transactions on Storage*, *3*(3), 33. doi:10.1145/1288783.1288786.
20. Briscoe, B., Jacquet, A., Moncaster, T., & Smith, A. (2009). Re-ECN: Adding accountability for causing congestion to TCP/IP. In *IETF internet-draft*.
21. Briscoe, B., Jacquet, A., Moncaster, T., & Smith, A. (2009). Re-ECN: The motivation for adding accountability for causing congestion to TCP/IP. In *IETF internet-draft.*
22. Zeng, L., Chen, H., & Xiao, Y. (2011). Accountable administration and implementation in operating systems. In *Proceedings of the IEEE GLOBECOM 2011*.
23. Zeng, L., Chen, H., & Xiao, Y. (in press). Accountable administration in operating systems. *International Journal of Information and Computer Security*. http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijics.
24. Xiao, Y. (2008). Accountability for wireless LANs, ad hoc networks, and wireless mesh networks. *IEEE Communication Magazine, 46*(4), 116–126.
25. Liu, J., & Xiao, Y. (2011). Temporal accountability and anonymity in medical sensor networks. *Mobile Networks and Applications, 16*(6), 695–712.

26. Liu, J., & Xiao, Y. (2012). An accountable neighborhood area network in smart grids. In *Proceedings of 7th FTRA international conference on embedded and multimedia computing (EMC 2012), lecture notes in electrical engineering* (Vol. 181, pp. 171–178). Springer.
27. Liu, J., Xiao, Y., & Gao, J. (2014). Achieving accountability in smart grids. *IEEE Systems Journal, 8*(2), 493–508.
28. Xiao, Z., Xiao, Y., & Chen, H. (2014). An accountable framework for sensing-oriented mobile cloud computing. *Journal of Internet Technology, 15*(5), 813–822. doi:10.6138/JIT.2014.15.5.11.
29. Ren, Y., Shen, J., Wang, J., Han, J., & Lee, S. (2015). Mutual verifiable provable data auditing in public cloud storage. *Journal of Internet Technology, 16*(2), 317–323.
30. Li, J., Li, X., Yang, B., & Sun, X. (2015). Segmentation-based Image copy-move forgery detection scheme. *IEEE Transactions on Information Forensics and Security, 10*(3), 507–518.
31. Xiao, Z., & Xiao, Y. (2012). PeerReview re-evaluation for accountability in distributed systems or networks. *International Journal of Security and Networks, 7*(1), 40–58.
32. Xiao, Z., Xiao, Y., & Wu, J. (2010). A quantitative study of accountability in wireless multi-hop networks. In *2010 39th international conference on parallel processing (ICPP)* (pp. 198–207).
33. Zhou, Z., Wang, Y., Wu, Q. M. J., Yang, C., & Sun, X. (2016). Effective and efficient global context verification for image copy detection. *IEEE Transactions on Information Forensics and Security*. doi:10.1109/TIFS.2016.2601065.
34. Xia, Z., Wang, X., Zhang, L., Qin, Z., Sun, X., & Ren, K. (2016). A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. *IEEE Transactions on Information Forensics and Security*. doi:10.1109/TIFS.2016.2590944.
35. Fu, Z., Wu, X., Guan, C., Sun, X., & Ren, K. (2016). Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics and Security, 11*(12), 2706–2716.
36. Rekhis, S., & Boudriga, N. A. (2009). Visibility: A novel concept for characterising provable network digital evidences. *International Journal of Security and Networks, 4*(4), 234–245.
37. Ray, I., & Poolsappasit, N. (2008). Using mobile ad hoc networks to acquire digital evidence from remote autonomous agents. *International Journal of Security and Networks, 3*(2), 80–94.
38. Madan, B. B., Goeva-Popstojanova, K., Vaidyanathan, K., & Trivedi, K. S. (2004). A method for modeling and quantifying the security attributes of intrusion tolerant systems. *Performance Evaluation, 56,* 167–186.
39. Breu, R., Innerhofer-Oberperfler, F., Yautsiukhin, A. (2008). Quantitative assessment of enterprise security system. In *Third international conference on availability, reliability and security (ARES 08)* (pp. 921–928).
40. Sallhammar, K., Helvik, B., & Knapskog, S. (2006). A game-theoretic approach to stochastic security and dependability evaluation. In *2nd IEEE international symposium on dependable, autonomic and secure computing (pp. 61–68).*
41. Bella, G., & Paulson, L. C. (2006). Accountability protocols: Formalized and verified. *ACM Transactions on Information and System Security, 9*(2), 138–161.
42. Bella, G. Inductive verification of cryptographic protocols. *Ph.D. thesis*, Research Report 493, Computer Laboratory, University of Cambridge.
43. Jagadeesan, R., Jeffrey, A., Pitcher, C., & Riely, J. (2009). Towards a theory of accountability and audit. In *ESORICS'09, volume 5789 of LNCS* (pp. 152–167). Springer.
44. Küsters, R., Truderung, T., Vogt, A. (2010). Accountability: Definition and relationship to verifiability. In *Proceedings of the 17th ACM conference on computer and communications security, New York, NY, USA* (pp. 526–535).
45. Milner, R. (1999). *Communicating and mobile systems: the pi calculus*. Cambridge: Cambridge University Press.
46. Feigenbaum, J., Jaggard, A. D., & Wright, R. N. (2011). Towards a formal model of accountability. In *Proceedings of the 2011 workshop on new security paradigms workshop, New York, NY, USA* (pp. 45–56).
47. Ramazani, S., Kanno, J., Selmic, R. R., & Brust, M. R. (2016). Topological and combinatorial coverage hole detection in coordinate-free wireless sensor networks. *International Journal of Sensor Networks, 21*(1), 40–52.
48. Mu, J., Song, W., Wang, W., & Zhang, B. (2015). Self-healing hierarchical architecture for ZigBee network in smart grid application. *International Journal of Sensor Networks, 17*(2), 130–137.
49. Xiao, Y., Shen, X., & Jiang, H. (2006). Optimal ACK mechanisms of the IEEE 802.15.3 MAC for ultra-wideband systems. *IEEE Journal on Selected Areas in Communications, 24*(4), 836–842.
50. Fantacci, R., & Tarch, D. (2006). Efficient scheduling techniques for high data-rate wireless personal area networks. *International Journal of Sensor Networks, 2*(1/2), 128–134.

51. Paxson, V. (1997). End-to-end Internet packet dynamics. *SIGCOMM Computer Communication Review, 27,* 139–152.
52. Liu, X., Liu, X., Li, Z., & Wang, B. (2014). The portable distributed fusion algorithm between loss and lossless systems. *International Journal of Sensor Networks, 16*(1), 16–22.
53. Xiao, Y., & Rosdahl, J. (2002). Throughput and delay limits of IEEE 802.11. *IEEE Communications Letters, 6*(8), 355–357.
54. Xiao, Y., & Rosdahl, J. (2003). Performance analysis and enhancement for the current and future IEEE 802.11 MAC protocols. *ACM SIGMOBILE Mobile Computing and Communications Review, 7*(2), 6–19.
55. PeerReview Software. http://peerreview.mpi-sws.org/.

Zhifeng Xiao received his Bachelor of Engineering in Computer Science in 2008 from Shandong University, China. His Ph.D. in Computer Science was completed at the University of Alabama in 2013. Zhifeng Xiao joined Penn State Behrend College, in August 2013. His research interests are in the design and analysis of secure distributed and networked systems. His personal homepage is http://cs.bd.psu.edu/~zux2/.

Yang Xiao currently is a Professor of Department of Computer Science at the University of Alabama, Tuscaloosa, AL, USA. His current research interests include networking and computer/network security. He has published over 200 journal papers and over 200 conference papers. Dr. Xiao was a Voting Member of IEEE 802.11 Working Group from 2001 to 2004, involving IEEE 802.11 (WIFI) standardization work. He is a Fellow of IET. He currently serves as Editor-in-Chief for International Journal of Security and Networks, International Journal of Sensor Networks, and Journal of Communications. He had (s) been an Editorial Board or Associate Editor for 20 international journals. He served (s) as a Guest Editor for over 20 times for different international journals. Dr. Xiao has delivered over 30 keynote speeches at international conferences around the world and gave more than 60 invited talks at different international institutes.

Jie Wu is the Associate Vice Provost for International Affairs at Temple University. He also serves as Director of Center for Networked Computing and Laura H. Carnell professor. He served as Chair of Computer and Information Sciences from 2009 to 2016. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.