Trust Management for SOA-based IoT and Its Application to Service Composition

Ing-Ray Chen, Jia Guo, and Fenye Bao

Abstract— A future Internet of Things (IoT) system will connect the physical world into cyberspace everywhere and everything via billions of smart objects. On the one hand, IoT devices are physically connected via communication networks. The service oriented architecture (SOA) can provide interoperability among heterogeneous IoT devices in physical networks. On the other hand, IoT devices are virtually connected via social networks. In this paper we propose adaptive and scalable trust management to support service composition applications in SOA-based IoT systems. We develop a technique based on distributed collaborative filtering to select feedback using similarity rating of friendship, social contact, and community of interest relationships as the filter. Further we develop a novel adaptive filtering technique to determine the best way to combine direct trust and indirect trust dynamically to minimize convergence time and trust estimation bias in the presence of malicious nodes performing opportunistic service and collusion attacks. For scalability, we consider a design by which a capacity-limited node only keeps trust information of a subset of nodes of interest and performs minimum computation to update trust. We demonstrate the effectiveness of our proposed trust management through service composition application scenarios with a comparative performance analysis against EigenTrust and PeerTrust.

Index Terms— Trust management; Internet of things; social networks; service composition; SOA; performance analysis.

1 Introduction

A future Internet of Things (IoT) system connects the physical world into cyberspace via radio frequency identification (RFID) tags, sensors, and smart objects owned by human beings [1, 10]. Physical objects can be equipped with RFID tags and electronically identifiable and tractable. Devices with sensing capability provide environmental information, body conditions, etc., which are remotely accessible. Smart objects like smart phones and consumer electronics with ample computing resources share information and provide billions of new services connecting everyone with everything.

In Service-Oriented Architecture (SOA) based IoT systems [11], each device is a service consumer and if desirable can be a service provider offering services or share resources and interacts with service consumers via compatible service APIs. SOA technologies (such as WS-*, REST, and CoRE) enable publishing, discovery, selection, and composition of services offered by IoT devices. The important application scenarios proposed for SOA-based IoT systems include e-health (continuous care) [6, 13], smart product management, smart events for emergency management [22], etc.

The motivation of providing a trust system for an SOA-based IoT system is easy to see. There are misbehaving owners and consequently misbehaving devices that for self-interest may perform "discriminatory" attacks to ruin the reputation of other IoT devices which provide similar services. Furthermore, users of IoT devices are likely to be socially connected via social networks like Facebook, Twitter, Google+, etc. Therefore, misbehaving nodes with close social ties can collude and monopoly a class of services.

SOA-based IoT systems challenge trust management in the following aspects. First, an IoT system has a huge amount of heterogeneous entities with limited capacity. Existing trust management protocols do not scale well to accommodate this requirement because of the limited storage space and computation resources. Second, a SOA-based IoT system evolves with new nodes joining and existing nodes leaving. A trust management protocol must address this issue to allow newly joining nodes to build up trust quickly with a reasonable degree of accuracy [19]. Third, IoT devices are mostly human carried or human operated [2]. Trust management must take into account social relationships among device owners in order to maximize protocol performance. Lastly and arguably most importantly, a SOA-based IoT system essentially consists of a large number of heterogeneous IoT devices providing a wide variety of services. Many of them (the owners) will be malicious for their own gain so they will perform attacks for self-interest. Many of them with close social ties will collude to ruin the reputation of other devices which provide similar services via bad-mouthing attacks, and conversely boost the reputation of each other via ballot-stuffing attacks. A trust management protocol for SOA-based IoT must be resilient to such attacks to be sustainable.

Despite the abundance of trust protocols for P2P and ad hoc sensor networks [9, 31, 32, 39, 40], there is little work on trust management for IoT systems [3, 4, 5, 7, 37]. We will survey related work in Section II and compare as well as contrast our approach with existing work. The problem we aim to solve is design and validation of a scalable and adaptive trust management protocol for

• *Ing-Ray Chen, Jia Guo, and Fenye Bao are with the Department of Computer Science, Virginia Tech, Falls Church, VA 22043; Email: {irchen, jiaguo, baofenye}@vt.edu.*

SOA-based social IoT systems capable of answering the challenges discussed above. Our trust management protocol, called *Adaptive IoT Trust*, is executed autonomously by IoT devices with little human intervention. The main idea is to combine peer evaluation with trust evaluation in SOA-based IoT systems. The goals are two-fold: (a) trust bias minimization; (b) application performance optimization. This is achieved by adaptive trust management, i.e., adjusting trust protocol settings in response to environment changes dynamically.

The contributions of this paper are as follows:

1. We propose an adaptive IoT trust protocol for SoA-based IoT systems with applications in service composition. The novelty lies in the use of distributed collaborating filtering [12] to select trust feedback from owners of IoT nodes sharing similar social interests.
2. We develop a novel adaptive filtering technique to adjust trust protocol parameters dynamically to minimize trust estimation bias and maximize application performance.
3. Our adaptive IoT trust protocol is scalable to large IoT systems in terms of storage and computational costs. We perform a comparative analysis of our adaptive IoT trust protocol against two prevalent trust protocols, namely, EigenTrust [39] and PeerTrust [40], in trust convergence, accuracy and resiliency properties achieved.
4. We demonstrate the effectiveness of our adaptive IoT trust protocol against EigenTrust and PeerTrust through service composition application scenarios in SOA-based IoT environments in the presence of malicious nodes performing opportunistic service and false recommendation attacks.

The rest of this paper is organized as follows. Section 2 discusses related work in trust management for IoT systems. Section 3 describes the system model. Section 4 details our trust management protocol. Section 5 assesses the performance of our trust protocol in terms of its desirable properties including convergence behavior, trust assessment accuracy, resiliency against malicious attacks, and scalability. In Section 6, we demonstrate the effectiveness of our trust management through trust-based service composition application scenarios, comparing its performance with two baseline schemes. Finally, Section 7 concludes the paper and discusses future work.

2 Related Work

One of the major challenges to IoT system design is device heterogeneity. Devices could be low-end with little to no storage/computational power (i.e., RFID tags), middle-end with restricted resources (i.e. sensors), to high-end (i.e., smart phones and laptops). Further, devices could connect to the network through various methods, like cables, Wi-Fi, Bluetooth, 3G, near field communication (NFC), etc. SOA technologies provide great opportunities to resolve the issue. Guinard *et al*. [11] proposed SOA-based IoT architecture where devices offer their functionalities via SOAP-based web services (WS-*) or RESTful APIs. This architecture supports the discovery, query, selection, and on-demand provisioning of web services. In order to realize web service on resource-constrained embedded devices, the IETF Constrained RESTful Environments (CoRE) working group has defined Constrained Application Protocol (CoAP) that realizes a minimum subset of REST [26]. One practical example is a web-based smart space framework [21] which applies REST to support pervasive applications, like resource sharing, in various devices.

The social relationships in IoT systems have attracted research attentions [2, 8, 16]. Doddy *et al*. [8] provided the vision of applying reality mining techniques developed to understand human relationships to IoT systems. Kranz *et al*. [16] investigated on the potential of combining social and technical networks to collaboratively provide services to both human users and technical systems in IoT systems. Atzori *et al*. [2] proposed the concept of social IoT (SIoT) and analyzed social relationships among things, such as *parental object relationship*, *social contact object relationship*, *co-work object relationship*, and *ownership relationship*. However, their work focuses on the relationship among things rather than users.

In the literature, Roman *et al*. [25] pointed out that traditional approaches for security, trust, and privacy management face difficulties when applying to IoT systems due to scalability and a high variety of relationship among IoT entities. Ren [24] proposed a key management scheme for heterogeneous wireless IoT systems. Zhou and Chao [28] proposed a media-aware traffic security architecture for IoT. The common drawback of their work is that they did not address the scalability issue.

Trust management for IoT is still in its infancy with limited work reported in the literature to date. Chen *et al*. [7] proposed a trust management model based on fuzzy reputation for IoT. However, their trust management model considers a specific IoT environment consisting of only wireless sensors with QoS trust metrics only such as packet forwarding/delivery ratio and energy consumption, and does not take into account the social relationship which is important in social IoT systems.

Bao and Chen [3, 4] proposed a trust management protocol considering both social trust and QoS trust metrics and using both direct observations and indirect recommendations to update trust. Their proposed trust management protocol considers a social IoT environment where environment conditions are dynamically changing, e.g., increasing misbehaving node population/activity, changeable behavior, rapid membership changes, and interaction pattern changes. To address the scalability issue, Bao and Chen further proposed a scalable trust management protocol [5] for large-scale IoT systems by utilizing a scalable storage management strategy. Relative to [3, 4, 5] this paper focuses on trust management for SOA-based IoT systems with the following specific contributions: (1) utilizing distributed collaborating filtering

[12, 38] to select trust feedback from nodes sharing similar social interests; (2) applying the proposed trust management to a SOA-based service composition application to demonstrate its effectiveness; (3) developing a novel adaptive filtering technique to dynamically adjust trust parameter settings so as to minimize trust estimation bias and maximize application performance; (4) validating the proposed trust management and its application to service composition through ns-3 simulation [41] based on real trace data, and (5) demonstrating the superiority of our adaptive IoT trust protocol design over EigenTrust [39] and PeerTrust [40] in trust convergence, accuracy and resiliency properties, as well as in service composition application performance.

EigenTrust [39] is a reputation scheme for P2P systems. Its basic idea is to aggregate trust recommendations towards a trustee node weighted by the trustor's opinion toward the recommenders. It is assumed that in a P2P network, there are pre-trusted peers that can provide trusted recommendations so as to guarantee trust convergence and break up malicious collectives.

PeerTrust [40] is also a reputation system for P2P systems. Its basic idea is also to aggregate feedbacks weighted by the recommender's trustworthiness. It considers more factors that affect a recommender's trustworthiness, including transaction context, community context, and credibility in terms of the trust and personalized similarity between the trustor and the recommender in order to filter out distrusted feedbacks.

This paper extends from [33] by adding extensive simulation validation, surveying state-of-the art related work, considering more sophisticated attacker model and analyzing the resiliency against these attacks, devising a smart storage management strategy for capacity-limited IoT devices for scalability with extensive analysis, addressing the best way to combine social similarity metrics to evaluate raters for application performance maximization, and adding a comparative performance analysis with EigenTrust [39] and PeerTrust [40] in trust convergence, accuracy and resiliency properties and in the application performance of the service composition application running on top of our adaptive IoT trust protocol in SOA-based IoT systems.

3 System Model

3.1 Social IoT Network Model

We consider a user-centric social IoT [2] environment where nodes are physically connected via communication networks and socially connected via users' social networks (Figure 1). Each node has a unique address to identify (i.e., URI). There is no centralized trusted authority. There are two types of nodes: devices and users (or owners). The user-device relationship is a one-to-multiple relationship. In our trust management, the trustor is a user and the trustee is a device (owned by another user). For each user, the trust evaluation information is computed and stored in a designated high-end device owned by the user.

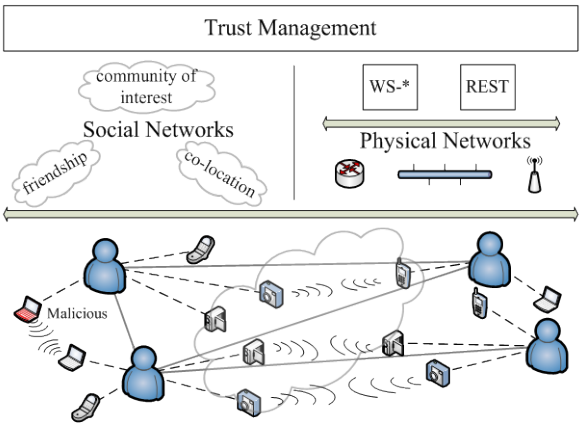


Figure 1: User-Centric Internet of Things Systems.

Trust is evaluated based on both direct user satisfaction experiences of past interaction experiences and recommendations from others.

In particular, for recommendations from others, we utilize the design concept of distributed collaborating filtering [12, 38] to select trust feedback from nodes sharing similar social interests. We consider the following three social relationships: *friendship*, *social contact*, and *community of interest* (*CoI*). More specifically, we use the social relationships between the trustor and the recommender for the trustor to weigh the recommendation provided by the recommender toward a trustee. The reason is that two users sharing similar social relationships including friendship (representing intimacy), social contact (representing closeness) and CoI (representing knowledge and standard on the subject matter) are likely to have similar subjective trust view towards services provided by a trustee IoT device. A similar concept to the *social contact* relationship is proposed in [20], where *familiar strangers* are identified based on colocation information in urban transport environments for media sharing.

These social relationships are represented by three lists: a *friend list* with current friends, a *location list* with locations frequently visited for social contact, and a *CoI list* with devices (services) directly interacted with. Each user has at least one designated high-end device (i.e., smart phone and laptop) storing these lists in the user's profile (see Figure 2). Other devices of the same user have the privilege to access the profile. By delegating the storage and computation of social networks to a high-end device for each user, many low-end devices (i.e., sensors) are able to share and utilize the same social information to maximize its performance. Energy spent for maintaining the lists and executing matching operations is negligible because energy spent for computation is very small compared with that for communication, and matching operations to identity a friend, social contact, or a CoI member are performed only when there is a change to the lists.

In the physical networks, devices provide and/or con-

sume services utilizing SOAP-based techniques or RESTful APIs (see Section 2). Each time when device $d1$ requests a service from device $d2$, $d1$ updates the user satisfaction experience record (in the *user satisfaction experience list* in Figure 2) towards $d2$ stored in the designated device of $d1$'s user. Similarly, $d1$ can query the trust information (in the *trust list* in Figure 2) towards $d2$ from the designated device of $d1$'s user. Note that elements in the user interaction experience list correspond to devices in the *CoI list.*

We consider a large IoT system in which a device with limited storage space cannot accommodate the full set of trust values towards all other devices. We address this scalability issue with a storage management design.

In the context of SOA, an owner provides services via its IoT devices. An IoT device providing a service will have to compete with other IoT devices which provide a similar type of service.

3.2 Attack Model

A malicious node in general can perform communication protocol attacks to disrupt network operations. We assume such attack is handled by intrusion detection techniques [18, 29, 34, 35] and is not addressed in this paper. In the context of SOA, we are concerned with trust-related attacks that can disrupt the trust system. Bad-mouthing and ballot-stuffing attacks are the most common forms of reputation attacks. Self-promoting and opportunistic service attacks are the most common forms of attacks based on self-interest [44-46]. Thus, a malicious IoT device (because its owner is malicious) can perform the following trust-related attacks:

1. *Self-promoting attacks*: it can promote its importance (by providing good recommendations for itself) so as to be selected as a SP, but then can provide bad or malfunctioned service.
2. *Bad-mouthing attacks*: it can ruin the reputation of a well-behaved device (by providing bad recommendations against it) so as to decrease the chance of that good device being selected as a SP. This is a form of collusion attacks, i.e., it can collaborate with other bad nodes to ruin the reputation of a good node.
3. *Ballot-stuffing attacks*: it can boost the reputation of a malicious node (by providing good recommendations) so as to increase the chance of that bad device being selected as a SP. This is a form of collusion attacks, i.e., it can collaborate with other bad nodes to boost the reputation of each other.
4. *Opportunistic service attacks*: it can provide good service to gain high reputation opportunistically especially when it senses its reputation is dropping because of providing bad service. With good reputation, it can effectively collude with other bad node to perform bad-mouthing and ballot-stuffing attacks.

A collaborative attack means that the malicious nodes in the system boost their allies and focus on particular victims in the system to victimize. Bad-mouthing and ballot-stuffing attacks are a form of collaborative attacks to the trust system to ruin the reputation of (and thus to victimize) good nodes and to boost the reputation of malicious nodes.

Table 1: Behavior of a Malicious Rater.

Trustor	Trustee	Bad-Mouthing	Ballot-Stuffing
malicious	*malicious*		
malicious	*non-malicious*		
non-malicious	*malicious*		√
non-malicious	*non-malicious*	√	

Table 2: Behavior of a Malicious Service Provider.

Service Requester	Self-Promoting	Opportunistic Service
malicious		
non-malicious	√	√

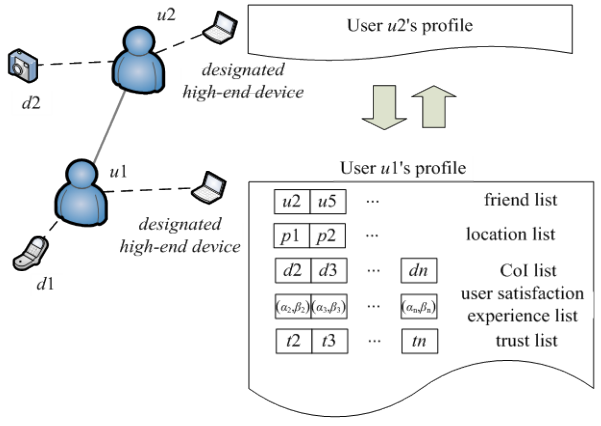


Figure 2: User Profile.

Table 1 summarizes the attack behavior of a malicious node as a rater, depending on the nature of the trustor and trustee nodes. If the trustor is non-malicious and the trustee is malicious, a malicious rater will perform ballot-stuffing attacks. If the trustor is non-malicious and the trustee is also non-malicious, a malicious rater will perform bad-mouthing attacks.

Table 2 summarizes the attack behavior of a malicious node as a SP, depending on the nature of the service requester. If the service requester is non-malicious, a malicious SP will perform both self-promoting and opportunistic service attacks. In particular, opportunistic service attacks are to be performed depending on the current reputation standing of the malicious SP itself.

4 Trust Management Protocol

Our adaptive IoT trust management protocol is distributed. Each user maintains its own trust assessment towards devices. For scalability, a user just keeps its trust evaluation results towards a limited set of devices of its interests. Each user stores its profile in a designated high-end device (Figure 2). The profile of user u_x includes:

(1) A "friend" list including all friends of u_x, denoted by

a set $F_x = \{u_a, u_b, \dots\}$;

(2) Locations that u_x frequently visited for social contact, denoted by a set $P_x = \{p_{x,1}, p_{x,2}, \dots\}$;

(3) List of devices that u_x has directly interacted with and the corresponding user satisfaction experience values, denoted by set $D_x = \{d_i, d_j, \dots\}$ and set $B_x = \{(\alpha_{x,i}, \beta_{x,i}), (\alpha_{x,j}, \beta_{x,j}), \dots\}$, where $\alpha_{x,i}$ and $\beta_{x,i}$ are the accumulated positive and negative user satisfaction experiences of user u_x towards device d_i;

(4) Trust values of user u_x towards IoT devices, denoted by a set $T_x = \{t_{x,i}, t_{x,j}, \dots\}$.

4.1 Direct Interaction Experiences

We adopt Bayesian framework [14] as the underlying model for evaluating *direct trust* from direct user satisfaction experiences. The reason we choose Bayesian because it is well-established and because of its popularity in trust/reputation systems. In service computing, a service requester could rate a service provider after direct interaction based on nonfunctional characteristics. The nonfunctional characteristics include user-observed response time, failure probability, prices, etc. The current user satisfaction experience of user u_x toward device d_i is represented by a value, $f_{x,i}$. We consider the simple case in which the direct user satisfaction experience $f_{x,i}$ is a binary value, with 1 indicating satisfied and 0 not satisfied. Then, we can consider $f_{x,i}$ as an outcome of a Bernoulli trial with the probability of success parameter $\theta_{x,i}$ following a Beta distribution (a conjugate prior for the Bernoulli distribution), i.e., Beta($\alpha_{x,i}$, $\beta_{x,i}$). Then, the posterior $p(\theta_{x,i} \mid f_{x,i})$ has a Beta distribution as well. Equation 1 shows how the hyper parameters $\alpha_{x,i}$ and $\beta_{x,i}$ are updated considering trust decay.

$$\begin{aligned} \alpha_{x,i} &= e^{-\varphi \Delta t} \cdot \alpha_{x,i}^{(old)} + f_{x,i} \\ \beta_{x,i} &= e^{-\varphi \Delta t} \cdot \beta_{x,i}^{(old)} + 1 - f_{x,i} \end{aligned} \quad (1)$$

In Equation 1, $f_{x,i}$ contributes to positive observations and $1 - f_{x,i}$ contributes to negative observations. When updating $\alpha_{x,i}$ and $\beta_{x,i}$, we consider an exponential decay, $e^{-\varphi \Delta t}$, on $\alpha_{x,i}^{(old)}$ and $\beta_{x,i}^{(old)}$, where φ is the decay factor which is normally is a small number to model small trust decay over time, and Δt is the trust update interval.

The direct trust of user u_x to device d_i, $t_{x,i}^d$, is calculated as the expected value of $\theta_{x,i}$, i.e.,

$$t_{x,i}^d = E[\theta_{x,i}] = \frac{\alpha_{x,i}}{\alpha_{x,i} + \beta_{x,i}} \quad (2)$$

In the literature, $\alpha_{x,i}$ and $\beta_{x,i}$ are often set to 1 initially since no prior knowledge available. In this paper, we consider the social relationships (if available) between u_x and the user of d_i (say u_y) as the prior knowledge and set the initial values of $\alpha_{x,i}$ and $\beta_{x,i}$ to $sim(u_x, u_y)$ and $1 - sim(u_x, u_y)$, respectively, where $sim(u_x, u_y)$ is the similarity between u_x and u_y characterizing their social connection. This is discussed below.

4.2 Recommendations

When the devices of two users have direct interactions, they can exchange their profiles and provide trust recommendations. In addition, a device can also aggressively request trust recommendations from another device belonging to a friend when necessary. To preserve privacy, one can use a hash function (with session key) to prevent the identities of uncommon friends/devices from being revealed.

We utilize the design concept of distributed collaborating filtering [12, 38] to select trust feedback from nodes sharing similar social interests. A node will first measure its "social similarity" with a recommender in friendship, social contact (representing physical proximity) and CoI (representing knowledge on the subject matter) and then decide if the recommendation is trustable. The reason we consider these metrics is that these metrics are core social metrics for measuring social relationships which are multifaceted [43]. We adopt cosine similarity to measure the distance of two social relationship lists (see Figure 2), with 1 representing complete similarity and 0 representing no similarity. Computational efficiency is the main reason why we choose cosine similarity to measure the similarity of two vectors in high-dimensional positive spaces because of limited computational capacity of IoT devices. In this paper we further introduce a new design concept called *application performance maximization* by which the best weights assigned to the three similarity metrics are identified to optimize application performance, when given a node population characterized by friendship, social connection, and community of interest relationships as input. Later in Section 6 we will deal with the subject of the effect of social similarity in friendship, social connection, and community of interest on application performance and identify the best way of combining these metrics to maximize the service composition application performance.

We describe how these social similarity measures may be estimated dynamically as follows:

- **Friendship Similarity** (sim_f): The friendship similarity is a powerful social relationship (intimacy) for screening recommendations. After two users u_x and u_y exchange their friend lists, F_x and F_y, they could compute two binary vectors, $\overrightarrow{VF_x}$ and $\overrightarrow{VF_y}$, each with size $|F_x \cup F_y|$. An element in $\overrightarrow{VF_x}$ (or $\overrightarrow{VF_y}$) will be 1 if the corresponding user is in F_x (or F_y), otherwise 0. Let $\|\vec{A}\|$ be the norm of vector $\vec{A}$ and $|B|$ be the cardinality of set B. Then, we could use the "cosine similarity" of $\overrightarrow{VF_x}$ and $\overrightarrow{VF_y}$ (giving the cosine of the angle between them) to compute sim_f as follows:

$$sim_f(u_x, u_y) = \frac{\overrightarrow{VF_x} \cdot \overrightarrow{VF_y}}{\|\overrightarrow{VF_x}\| \|\overrightarrow{VF_y}\|} = \frac{|F_x \cap F_y|}{\sqrt{|F_x| \cdot |F_y|}} \quad (3)$$

- **Social Contact Similarity** (sim_l): The social contact similarity presents closeness and is an indication if two nodes have the same physical contacts and thus

the same sentiment towards devices which provide the same service. The operational area could be partitioned into sub-grids. User u_x records the IDs of sub-grids it has visited in its *location list* P_x for social contact. After two users u_x and u_y exchange their location lists, P_x and P_y, they could compute sim_l in the same way of computing sim_f as follows:

$$sim_l(u_x, u_y) = \frac{|P_x \cap P_y|}{\sqrt{|P_x| \cdot |P_y|}} \quad (4)$$

- **Community of Interest Similarity** (sim_c): Two users in the same COI share similar social interests and most likely have common knowledge and standard toward a service provided by the same device. Also very likely two users who have used services provided by the same IoT device can form a CoI (or are in the same CoI). After two users u_x and u_y exchange their device lists, D_x and D_y, they could compute sim_c in the same way of computing sim_f as follows:

$$sim_c(u_x, u_y) = \frac{|D_x \cap D_y|}{\sqrt{|D_x| \cdot |D_y|}} \quad (5)$$

The social similarity between two users can be a weighted combination of all social similarity metrics, i.e., friendship, social contact, and community of interest, considered in this paper:

$$sim(u_x, u_y) = \sum_{v \in \{f,l,c\}} w_v \cdot sim_v(u_x, u_y) \quad (6)$$

where $w_f + w_l + w_c = 1$ and $0 \le w_f, w_l, w_c \le 1$. Each user can send trust recommendations request to its friends periodically (in every Δt interval) or before requesting a service. Upon receiving recommendations, user u_x selects top-k recommendations from k users with the highest similarity values with u_x and calculates the indirect trust ($t_{x,i}^r$) towards device d_i as follows:

$$t_{x,i}^r = \sum_{u_y \in U} \frac{sim(u_x, u_y)}{\sum_{u_y \in U} sim(u_x, u_y)} \cdot t_{y,i}^d \quad (7)$$

Here, U is a set of up to k users whose $sim(u_x, u_y)$ values are the highest, and $t_{y,i}^d$ is the direct trust of user u_y toward device d_i serving as $u_y's$ recommendation toward d_i provided to u_x. Each recommendation is weighted by the ratio of the similarity score of the recommender to the sum of the similarity scores of all recommenders. We also note that if u_y is malicious, then it can provide $t_{y,i}^d$=0 against a good device for bad-mouthing attacks, and $t_{y,i}^d$=1 for a bad node for ballot-stuffing attacks.

4.3 Adaptive Control of the Weight Parameter

The trust value of user u_x toward d_i is denoted as $t_{x,i}$ and is obtained by combining direct trust and indirect recommendations (if available) as follows,

$$t_{x,i} = \mu \cdot t_{x,i}^d + (1 - \mu) \cdot t_{x,i}^r \quad (8)$$

Here, μ is a weight parameter ($0 \le \mu \le 1$) to weigh the importance of direct trust relative to indirect trust feedback. The selection of μ is critical to trust evaluation. A contribution of the paper is that we propose a method based on adaptive filtering [12] to adjust μ dynamically in order to effectively cope with malicious attacks including self-promoting, bad-mouthing, ballot-stuffing, and opportunistic attacks and to improve trust evaluation performance. The basic design principle is that a successful trust management protocol should provide high trust toward devices who have more positive user satisfaction experiences and, conversely, low trust toward those with more negative user satisfaction experiences. Specifically, the current trust evaluation (i.e., $t_{x,i}(\mu)$ as a function of μ) should be as close to the average user satisfaction experiences observed over the last trust update window Δt as possible. Therefore, we formulate the selection of μ as an optimization problem as follows:

$$\begin{aligned} &\underline{\text{Find}}: \mu, 0 \le \mu \le 1 \\ &\underline{\text{Minimize}}: \text{MSE}(\mu) = \textstyle\sum_i \left(t_{x,i}(\mu) - \overline{f_{x,\iota}^{(new)}}\right)^2 \end{aligned} \quad (9)$$

Here, $t_{x,i}(\mu)$ is obtained from Equation 8 using past direct user satisfaction experiences and indirect trust feedback, and $\overline{f_{x,\iota}^{(new)}}$ is the most recent direct user satisfaction experiences observed by user u_x within the last trust update interval Δt. The objective can be achieved by minimizing the mean square error (MSE) of trust evaluations against actual user satisfaction experiences towards all applicable devices, such that the trust value could be a good indicator or predictor for quality of service (with direct user satisfaction experiences considered as ground truth). After user u_x obtains new user satisfaction experiences over Δt, it can compute the average user satisfaction experience value $\overline{f_{x,\iota}^{(new)}}$ and update μ by minimizing MSE in Equation 9. The optimization problem in Equation 9 can be solved by plugging $t_{x,i}(\mu)$ in Equation 8 into Equation 9 and minimizing $\text{MSE}(\mu)$ as follows:

$$\text{MSE}(\mu) = \sum_i \left(\mu \cdot t_{x,i}^d + (1 - \mu) \cdot t_{x,i}^r - \overline{f_{x,\iota}^{(new)}}\right)^2 \quad (10)$$

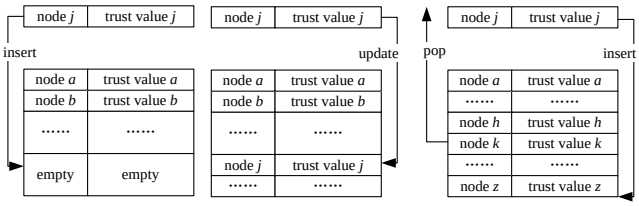
The minimum value of $\text{MSE}(\mu)$ is obtained at the point where the derivative is zero, i.e., $\text{MSE}'(\tilde{\mu}) = 0$. Thus, $\tilde{\mu}$ is obtained as follows,

$$\tilde{\mu} = \frac{\sum_i \left(\overline{f_{x,\iota}^{(new)}} - t_{x,i}^r\right)\left(t_{x,i}^d - t_{x,i}^r\right)}{\sum_i \left(t_{x,i}^d - t_{x,i}^r\right)^2} \quad (11)$$

The optimal value of μ (i.e., $\hat{\mu}$) should be in the range of [0, 1] because it is a weight parameter. Therefore,

$$\hat{\mu} = \begin{cases} 0 & \tilde{\mu} < 0 \\ \tilde{\mu} & 0 \le \tilde{\mu} \le 1 \\ 1 & \tilde{\mu} > 1 \end{cases} \quad (12)$$

Each user computes its own optimal value of μ (i.e., $\hat{\mu}$) and updates it dynamically in every trust update time interval Δt, based on Equations 11 and 12, using the historical data collected in its storage, so there is essentially no extra overhead. This adaptive design is applicable to



(a) Storage space is not full. (b) Node *j*'s trust value exists. (c) Otherwise (pop *k* and insert *j*).

Figure 3: Storage Management for Small IoT Devices.

other trust parameters (i.e., φ and (w_f, w_l, w_c)) as well. However, introducing these trust parameters in Equation 9 leads to a more complex optimization problem and may not be feasible for IoT devices with limited resources. In this paper we focus on adaptive control of μ and leave adaptive control of other trust parameters as future work.

Here we note that our dynamic weight adjustment scheme is driven by minimizing the difference between the subjective trust $t_{x,i}(\mu)$ as a result of following the trust aggregation protocol in Equation 8, and the new user satisfaction experience $\overline{f_{x,\iota}^{(new)}}$ obtained in the last trust update interval Δt. If d_i is a malicious node and it retains high reputation either because it performs opportunistic service attacks to gain high reputation, or because other nodes provide ballot-stuffing attacks to boost its reputation, then our trust system will be temporarily deceived of its true status because the difference between these two quantities will be small. However, the moment d_i performs self-promoting attacks and provides bad service to user u_x, this bad user experience will be immediately observed by user u_x and, as a result, the difference between these two quantities will be large enough to drive the change of μ to minimize $\mathrm{MSE}(\mu)$ in Equation 10. It is noteworthy that μ is dynamically adjusted based on minimizing the sum of the differences of all devices observed by user u_x over Δt, so adjusting μ to minimize $\mathrm{MSE}(\mu)$ moves toward the right direction of minimizing the difference between "the subjective trust" vs. "what service quality is actually provided" for all devices with which user u_x has interaction experiences over Δt.

4.4 Storage Management for Small IoT Devices

Considering a large-scale IoT system in which each node has limited storage space to keep direct user satisfaction experiences and trust values of a small set of nodes with which it shares interests. A node has to decide which trust values to keep. In general, nodes are more interested in others with higher trust values. However, simply saving the trust values towards the most trustworthy nodes cannot make the trust evaluation process converge and is not adaptive to dynamic environments since there is little chance to accumulate trust towards newly joining nodes. Our storage management strategy considers nodes with the highest trust values and recent interacting nodes as these nodes are most likely to share common interests.

Figure 3 illustrates how our approach works conceptualizing the storage size of each node as n (meaning that there is space to save trust values of up to n nodes). When a slot is needed, for a node's trust value to be kept it must be in the top Ω of the n trust values, or this node is one of the most recent interacting nodes. We consider Ω = 50% in this paper and the selection of optimal Ω value in dynamic IoT systems can be solved using the same adaptive control in Section 4.3.

When node i obtains the trust value towards node j, if the storage space is not full or node i does have the trust information of node j in its storage space, then node i will simply save the trust value towards node j. If the storage space is full and node i does not have the trust information of node j in its storage space, node i will put the trust value towards node j and pop out the trust value towards the earliest interacting node among those with trust values below the median (Ω = 50%). By using a max-min-median heap, find medium, maximum or minimum operations can be performed in O(1) constant time, while all others operations (find, insert and delete) can be performed in O(log n) logarithmic time.

Table 3: Parameter List and Default Values Used.

parameter	value	parameter	value	parameter	value
N_T	400	$m \times m$	16×16	T	200hrs
N	40	P_M	30%	φ	0.001
Ω	50%	σ_ϵ	0.01	λ	1/day
Δt	2hrs				

5 Trust Protocol Performance

In this section, we report simulation results obtained as a result of executing our proposed autonomous trust management protocol by IoT devices. We choose ns-3 [41, 42] as the simulator as it emerges as the de facto standard open simulation platform for networking research; it is a discrete-event network simulator, targeted primarily for research and educational use.

The focus in this Section is to demonstrate our protocol's desirable convergence and accuracy properties, as well as its resiliency property against malicious attacks. In Section 6, we will apply it to service composition and compare its performance against the baseline trust management schemes.

Our simulation results have three parts. First, we demonstrate trust convergence, accuracy and resiliency properties of our adaptive IoT trust protocol design against malicious attacks. We then demonstrate the effectiveness of our storage management protocol design for IoT devices with limited storage space. Lastly, we perform a comparative analysis of our adaptive IoT trust protocol against two baseline schemes: EigenTrust [39] and PeerTrust [40].

Table 3 lists the default parameter values. We consider an IoT environment with N_T = 400 heterogeneous smart objects/devices. These IoT devices are randomly assigned to N = 40 users. Users are connected in a social network represented by a friendship matrix [17]. We consider these users moving according to the SWIM mobility

model [15] modeling human social behaviors in an $m \times m$=16×16 operational region for the purpose of assessing the social contact similarity metric between any pair of users. Direct trust of node i toward node j is assessed upon completion of a service request from node i to node j. Each node requests services from a selected device with a time interval following an exponential distribution with parameter λ, with 1/day being the default unless otherwise specified. The trust update interval Δt is 2 hours at which time if there is no direct trust update due to service request and completion, direct trust will be decayed according to Equation 1. Indirect trust is always updated in every Δt interval according to Equation 7. The system runs continuously although we often can observe trust convergence in less than 200 hours, given that bad nodes follow the attack behaviors specified in Section 3.2.

The user satisfaction levels of service invocations are generated based on a real dataset [27] and are used as "ground truth" based on which the accuracy of our trust protocol is assessed. As the direct trust of user u_x toward device/service provider d_i (i.e., $t_{x,i}^d$) is calculated by Equation 1 with "ground truth" user satisfaction experiences as input, $t_{x,i}^d$ essentially is equal to ground truth. However, we account for the presence of noise in the IoT environment (i.e., error of assessing user satisfaction level received) by considering a standard deviation parameter σ_c (set to 1% as default) to reflect the deviation of the actual user satisfaction level as recorded in the database from the direct trust evaluation outcome in terms of $t_{x,i}^d$.

Initially, $t_{x,i}$ is set to 0.5 (ignorance) by user u_x for all i's. Then, trust is updated dynamically as nodes encounter each other, as services are requested and rendered, and as trust feedback are acquired. We consider $w_f = w_l = w_c = 1/3$ (in Equation 6) as we assess the convergence and accuracy properties of our trust protocol in this section. Later in Section 6 we will identify the best weight assignment (w_f, w_l, w_c) for social similarity computation for the service composition application.

We test the resiliency of our trust protocol against malicious node behavior (i.e., performing self-promotion, bad-mouthing and ballot-stuffing attacks) by randomly selecting a percentage P_M out of all as dishonest malicious nodes with P_M=30% as the default. A normal or good node follows the execution of our trust management protocol faithfully, while a malicious node provides false trust feedback by means of ballot-stuffing, bad-mouthing, and self-promoting attacks to gain advantage.

5.1 Trust Convergence, Accuracy and Resiliency against Malicious Attacks

In this section, we examine the trust convergence, accuracy and resiliency properties of our adaptive IoT trust protocol design. We first compare static control (i.e., μ is fixed at a constant) vs. adaptive control (i.e., μ is changed dynamically based on Equation 12).

Figure 4 shows trust evaluation results for a trustor node toward a "good" trustee node randomly picked. We

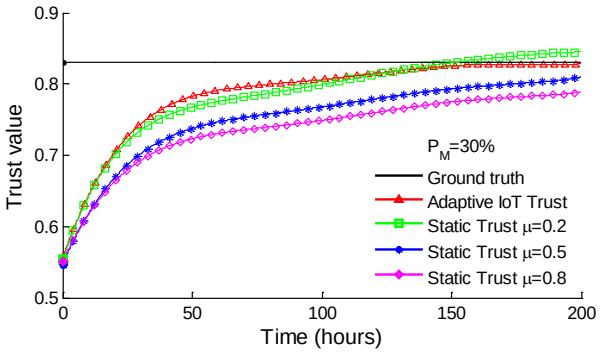


Figure 4: Trust Value of a Good Node with P_M = 30%.

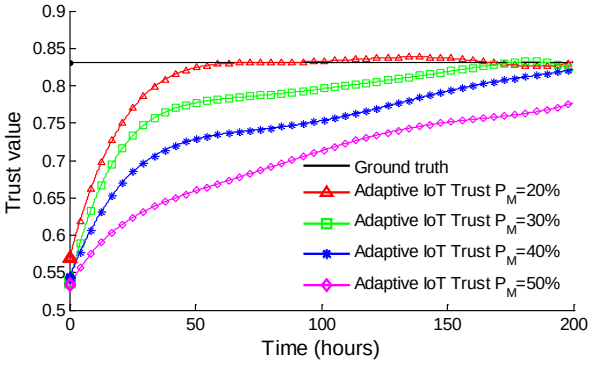


Figure 5: Trust Value of a Good Node under Adaptive Control with P_M ranging from 20% to 50%.

see that trust convergence behavior is observed for either fixed or adaptive control. There is a tradeoff between convergence time vs. trust bias. With static control, when a higher μ value is used, the trust convergence time is longer, but the trust bias is smaller, i.e., the trust value is closer to ground truth after convergence. With adaptive control, on the other hand, the trustor node is able to adjust μ dynamically to minimize both the convergence time and the trust bias after convergence. Here we note that the trust value of a "good" trustee is not 1 because we use the user satisfaction levels of service invocations based on a real dataset [27] with a standard deviation parameter σ_c (set to 1% as default) reflecting the deviation of the actual user satisfaction level recorded in the database from the direct trust evaluation outcome.

An interesting observation in Figure 4 is that if μ is too small (e.g., 0.2) the trust value is over-estimated upon convergence, which is not a desirable outcome as trust overshoot is considered a bad property detrimental to the stability of a trust system [36]. Our adaptive protocol dynamically adjusts μ for fast convergence without incurring trust overshoot.

Figure 4 is for the case in which the percentage of malicious nodes P_M = 30%. We conduct experiments to test the residency of our trust protocol against increasing malicious node population. Figure 5 shows that as the population of malicious nodes increases, both the convergence time and trust bias increase. However, the system is found to be resilient to malicious attacks for P_M as high as 40%, with proper convergence and accuracy behaviors

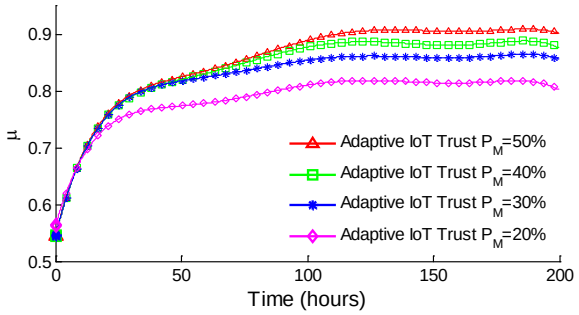


Figure 6: Adjustment of μ against Increasing Malicious Node Population.

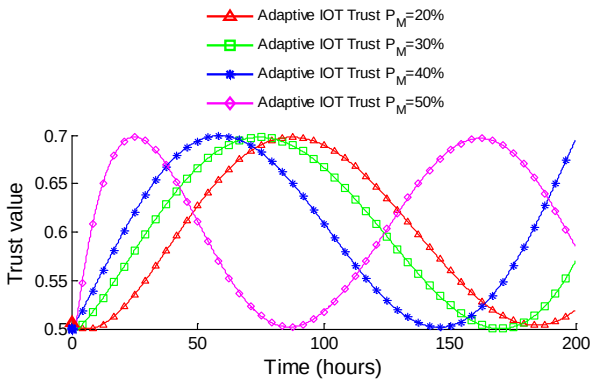


Figure 7: Trust Value of a Bad Node under Adaptive Control with P_M ranging from 20% to 50%.

exhibited. In general we observe that the trust bias is minimum, e.g., < 5% when $P_M \leq 40\%$ and the trust bias becomes more significant, e.g., > 10% when $P_M \geq 50\%$. This demonstrates the resiliency property of our trust protocol against malicious attacks.

Correspondingly, Figure 6 shows how our trust-based adaptive control protocol adjusts μ in Equation 12 in response to increasing malicious node population.

We observe that as the malicious node population increases, the system will have to rely more on direct trust by increasing μ and conversely rely less on indirect trust by decreasing $1-\mu$ so as to mitigate the effect of bad-mouthing and ballot-stuffing attacks by malicious nodes. Figure 6 shows that when P_M = 20%, the optimal converged μ value is 0.78, while when P_M = 50%, the optimal converged μ value is 0.90. This follows the design principle of "go up slowly, reduce quickly," that is, when a node acts maliciously, its trust value should reduce quickly, and when a node acts cooperatively, its trust should just go up slowly. When a node is being observed maliciously, its trust value will be reduced quickly because in this case a high μ value will be used by our trust protocol and a high μ value means that the trust value of the malicious node will be very close to direct trust which is low as the node is being observed maliciously. Conversely, when a node is being observed cooperatively, its trust value will just go up slowly because in this case a low μ value will be used by our adaptive protocol and a low μ value means that both direct trust and indirect trust will contribute to the overall trust based on Equation 8. Although in this case, the direct trust observed is high as the node is being observed cooperatively, it will only increase the overall trust value slowly by a weight of μ, with the indirect trust contributing to the overall trust by a weight of $1-\mu$. The system cannot rely on direct trust 100% because malicious nodes can perform opportunistic service attacks and there is an error of assessing direct trust due to noise in the environment. Figure 6 demonstrates that our adaptive control mechanism is effective in terms of convergence of μ to its optimal value under which trust bias is minimized.

Figure 7 shows trust evaluation results for a trustor node toward a "bad" trustee node. Among all attacks, the bad node performs *opportunistic service attacks* with the high trust threshold being 0.7 and the low trust threshold being 0.5. Specifically, the bad node provides good service to gain high reputation opportunistically when it senses its reputation drops below 0.5. Once it reputation rises to 0.7, it provides bad service again. We see from Figure 7 that our adaptive trust protocol is able to accurately track the trust fluctuation of the bad node performing opportunistic service attacks. We observe that the rate of trust fluctuation is higher when P_M is higher because more malicious nodes can collude to quickly bring the trust level of the bad node to 0.7.

The effect of the decay parameter φ is analyzed in Figure 8. A smaller φ means a slower trust decay rate with φ=0 meaning no trust decay. We choose φ=0.001 to achieve the desirable convergence behavior. We see that as φ increases, it takes longer to achieve trust convergence. This is because a good node remains good for its lifetime so a larger trust decay rate requires a good node to become more socially and service active over time in order to regain its trust status. In this case, we see that φ=0 produces the fastest convergence rate. This is not necessarily true for cases in which a good node may be compromised dynamically for which φ>0 may become the best setting. The determination of the optimal φ to trade convergence with accuracy as dictated by environment conditions is a future research area.

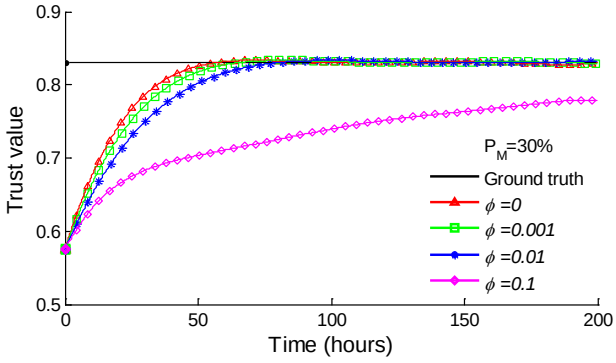


Figure 8: Effect of Decay Parameter on Trust Convergence.

5.2 Trust Evaluation with Limited Storage Space

The results presented in Section 5.1 are based on the assumption that each node has sufficient storage to save trust values of all nodes. In this section, we consider a more realistic scenario in which many small IoT devices

only have a limited storage space. A trustor node in this case would run the trust storage management strategy described in Section 4.4 to store trust values considered important to the node.

Figure 9 compares the trust value obtained by a trustor node toward a good trustee node randomly picked, when P_M = 30% and each node has 10%, 50% or 100% space to accommodate all trust values. We first note that the curve labeled with "adaptive trust-based 100% storage" in Figure 9 is the same as the curve labeled with "adaptive trust-based" in Figure 4. We observe that the convergence time and trust bias after convergence are comparable for the 10% and 50% storage cases and they don't deviate much from those for the 100% storage case. This demonstrates the effectiveness of our management strategy for limited storage. We attribute this to its ability to function like a filter, thus excluding highly deviated trust feedback coming from untrustworthy nodes to shield the system from false recommendation attacks.

Lastly, we examine the effect of our management strategy for limited storage on hit ratio. We define the "top-m hit ratio" as the percentage of the top-m most trustworthy nodes having their trust values stored in the limited n slots. Figure 10 shows the top-20 hit ratio as a function of time for a randomly selected node. We can see that initially the hit ratio is zero because there is no trust information stored for any node. As the trust value converges, the hit ratio quickly increases and approaches its peak. We see that the maximum achievable hit ratios are 90%, 85%, 75% and 50% under 100%, 50%, 10% and 5% storage spaces, respectively. Even with as little storage space as 10%, the hit ratio only deteriorates from 90% to 75%. This again demonstrates the effectiveness and high space utilization of our management strategy for limited storage.

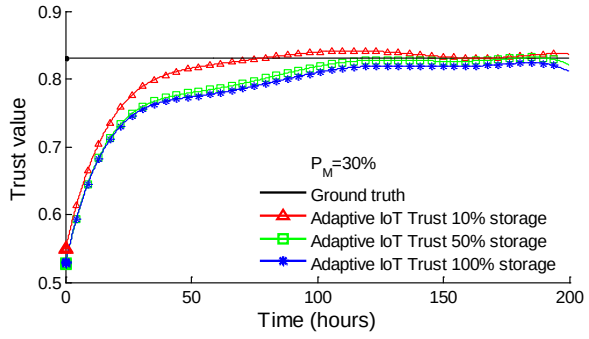


Figure 9: Adaptive Control with Limited Storage.

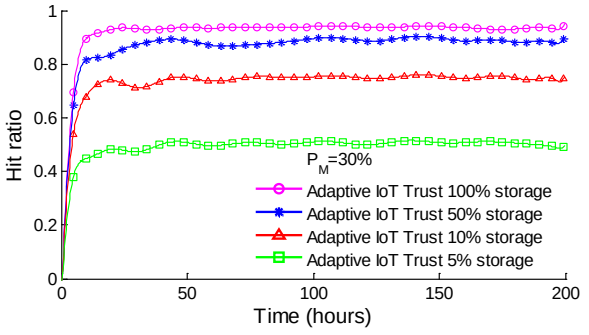


Figure 10: Hit Ratio with Limited Storage.

5.3 Comparative Analysis

Figure 11 shows head-to-head performance comparison data of our adaptive IoT trust protocol against two baseline schemes, EigenTrust [39] and PeerTrust [40], for the trust evaluation of a good node randomly selected. The environment conditions are setup the same way as in Figure 4 with P_M=30%. We see that while all protocols converge at about the same rate, our protocol achieves accuracy but EigenTrust and PeerTrust both suffer inaccuracy. Figure 12 shows the corresponding 3-dimensional view with P_M varying in the range of 20% to 40%. We see that the trust bias gap (difference to ground truth) for EigenTrust and PeerTrust widens as P_M increases, while it remains minimum for our adaptive IoT trust protocol against increasing malicious node population. This demonstrates the resiliency property of our trust protocol against malicious attacks. We attribute the superiority of our adaptive IoT trust protocol over EigenTrust and PeerTrust to our protocol's adaptability to adjust the best trust parameter (μ) dynamically to achieve trust accuracy despite the presence of a high percentage of malicious nodes performing opportunistic service attacks to boost their own reputation scores opportunistically and colluding (via bad-mouthing attacks) to ruin the reputation of this good node.

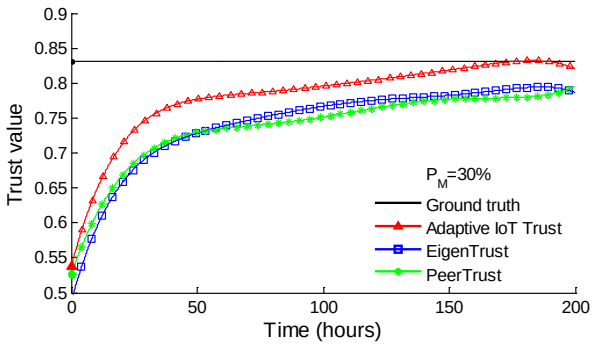


Figure 11: Performance Comparison of Trust Convergence, Accuracy and Resiliency when P_M=30%.

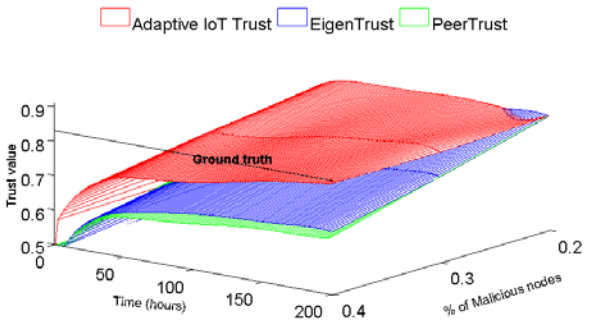


Figure 12: Performance Comparison of Trust Convergence, Accuracy and Resiliency in 3-D View with P_M ranging from 20% to 40%.

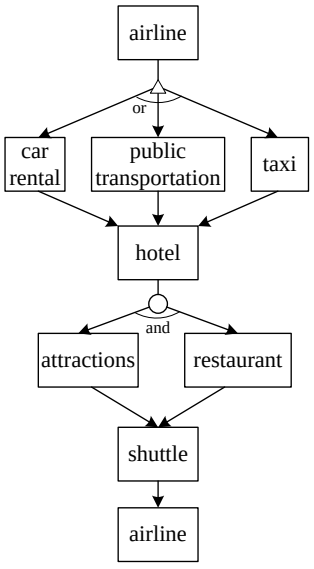


Figure 13: A Service Composition Example (Travel Planning).

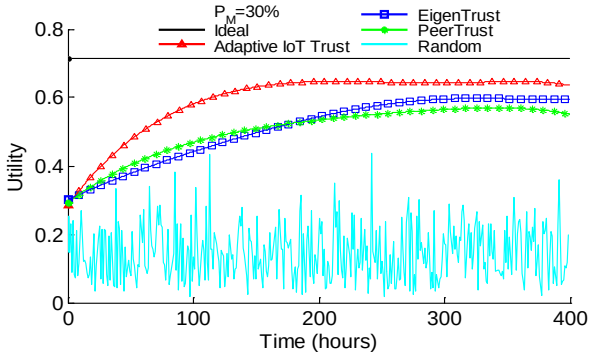


Figure 14: Utility of Service Composition without Constraints.

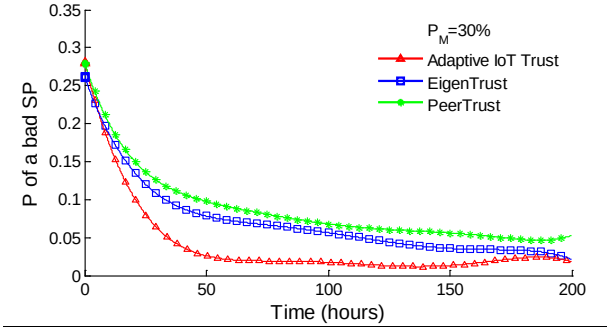


Figure 15: Probability of a bad SP being selected for Service Composition without Constraints.

6 Trust-Based Service Composition

In this section, we apply our trust management to a trust-based service composition application in SOA-based IoT systems. In SOA, service composition can be classified as static, semi-automatic, and automatic. Service composition methods include workflow composition, AI planning, etc. [23]. Dynamic service composition could become a complex planning problem. In this paper, we consider a template-based semi-automatic service composition application for which a template (or a workflow) describes the data flow and logic of a composite service.

Figure 13 shows an example for travel planning. There are 9 atomic services connected by three types of workflow structures in this example, namely, *sequential*, *parallel* (AND), and *selection* (OR). Each service would have multiple SP candidates.

We use the "true" user satisfaction levels received from the SPs selected for the service composition application to derive the overall user satisfaction level, called the *utility* score, to evaluate the performance of service composition. The utility score of a candidate service composition is calculated recursively. Specifically, the utility score of a composite service (whose utility score is us_s) comprising two subservices (whose utility scores are us_1 and us_2) depends on the structure connecting the two subservices as follows:

- Sequential Structure: $us_s = us_1 \times us_2$;
- Selection Structure: $us_s = \max(us_1, us_2)$;
- Parallel Structure: $us_s = 1 - (1 - us_1) \times (1 - us_2)$.

We also use the percentage of malicious nodes selected as SPs for providing the travel service as an additional performance metric. For *trust-based service composition*, the goal is to select service providers based on trust evaluation such that the composite service utility score is the best. We compare the performance of *trust-based service composition* with two baseline approaches:

1. *Ideal service composition* which returns the maximum achievable utility score derived from ground truth or global knowledge.
2. *Random service composition* which randomly selects service providers for service composition without regard to trust.

We differentiate two types of service composition applications: without constraints and with constraints, i.e., a budget limit for travel planning. In both scenarios, we compare the performance of *trust-based service composition* running on top of our adaptive IoT trust protocol against that running on top of EigenTrust and PeerTrust.

6.1 Service Composition without Constraints

In *trust-based service composition without constraints*, the SR selects the SP with the highest trust value for each required service.

Figure 14 shows the ns-3 simulation results with P_M=30%. We observe that trust-based service composition with our adaptive IoT trust protocol significantly outperforms random service composition and upon convergence approaches the performance of ideal service composition based on ground truth. Further, our adaptive IoT trust protocol outperforms EigenTrust and PeerTrust as the underlying trust protocol for trust-based service composition. In addition, we also observe that the performance gap widens as P_M increases.

Figure 15 shows the percentage of bad nodes selected for service composition without service constraints. Our adaptive IoT trust protocol again outperforms both EigenTrust and PeerTrust with EigenTrust slightly performing better than PeerTrust.

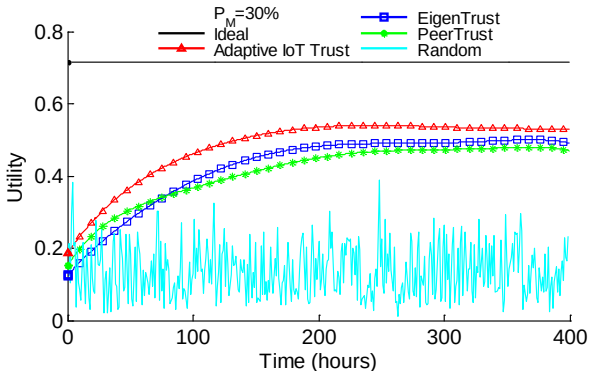


Figure 16: Utility of Service Composition with Constraints.

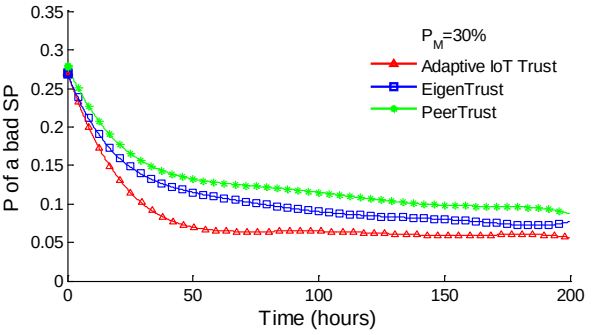


Figure 17: Probability of a bad SP being selected for Service Composition with Constraints.

We attribute the superiority of Adaptive IoT Trust over EigenTrust and PeerTrust to our protocol's adaptability to adjust the best trust parameter (μ) dynamically to minimize trust bias, and, consequently, maximize the performance of the service composition application.

6.2 Service Composition with Constraints

One example of service constraints is budget limit. Simply selecting the most trustworthy SPs may lead to infeasible solutions. Suppose that each SP announces its price when publishing the service and the SR has a budget limit for service composition. In *trust-based service composition with constraints*, the SR calculates the overall utility score and the overall price for each candidate configuration, using the trust value it has toward a SP to predict the utility score for that SP, and selects the configuration with the highest utility score among those with the overall price below the budget limit.

Figure 16 shows the ns-3 simulation results with P_M=30%. We first observe that the utility scores are lower than those without budget constraints since good service providers may post high price, thus preventing them from being included. We again observe that the trend is similar to Figures 14 in terms of performance ranking, with *trust-based service composition* with our adaptive IoT trust protocol outperforming that with either EigenTrust or PeerTrust. Figure 17 shows the percentage of bad nodes selected for service composition with budget limit constraints. Our adaptive IoT trust protocol again outperforms both EigenTrust and PeerTrust by a significant margin nearly cut in half in the percentage of bad nodes selected for service composition. We again attribute the superiority of our protocol over EigenTrust and PeerTrust to our protocol's adaptability in response to a high percentage of nodes performing malicious attacks.

6.3 Effects of Social Similarity on Trust Feedback

So far we have assumed $w_f = w_l = w_c = 1/3$ (in Equation 6) for computing social similarity, considering there is an equal contribution from friendship, social contact, and CoI. However, in some application environments (say remote travel agent service) in which nodes that are friends or in the same CoI may be more credible than nodes that are co-located in providing trust feedback, while in another environment (say local restaurant service), it is the other way around. So there is an optimal weight assignment (w_f, w_l, w_c) that can provide the most credible trust feedback. In this section, we examine the effect of (w_f, w_l, w_c) on protocol performance with the service composition application with constraints as our test case.

Figure 18 shows the simulation results of the MSE of the difference between the utility obtainable under trust-based service composition and the ideally achievable utility for the service composition application vs. (w_f, w_l, w_c). Note that w_c=$1 - w_f - w_l$ and is not shown in the 3-D diagram. One can see clearly from Figure 18 that there exists an optimal weight assignment (w_f, w_l, w_c) = (0.9, 0.0, 0.1) under which MSE is minimized, i.e., the utility obtainable via trust-based service composition is closest to the ideally achievable utility with perfect global

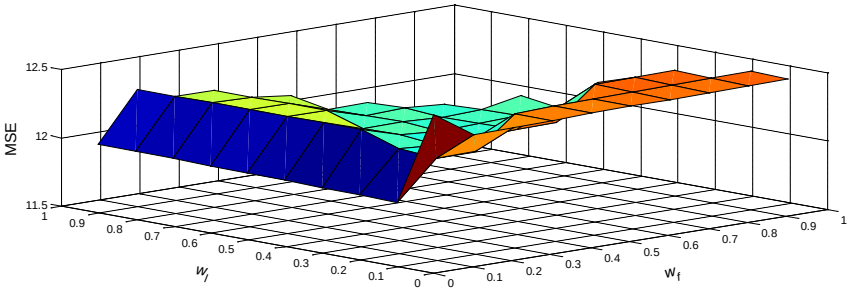


Figure 18: Mean Square Error of Utility Difference vs. (w_f, w_l, w_c).

knowledge of node status. Here it is worth noting that the social contact similarity metric is not a factor in this application scenario for trust feedback because all services except one (restaurant in Figure 13) do not require social contact similarity. However, this is not universally true should another service composition flowchart be given as input. The methodology developed in the paper will allow each service requester to dynamically decide and apply the optimal weight combination (w_f, w_l, w_c) that will lead to the most credible trust feedback to minimize trust bias and as a result maximize the utility or the user satisfaction level of the application.

7 Conclusion

In this paper, we designed and analyzed an adaptive and scalable trust management protocol for SOA-based IoT systems. We developed a distributed collaborating filtering technique to select trust feedback from owners of IoT nodes sharing similar social interests. We considered three social relationships, i.e., friendship, social contact, and community of interest, for measuring social similarity and filtering trust feedback based on social similarity. Further, we developed an adaptive filtering technique by which each node adaptively adjusts its best weight parameters for combining direct trust and indirect trust into the overall trust to minimize convergence time and trust bias of trust evaluation. We demonstrated via simulation the superiority of our adaptive IoT trust protocol over EigenTrust and PeerTrust in trust convergence, accuracy and resiliency against malicious nodes performing self-promoting, bad-mouthing, ballot-stuffing, and opportunistic service attacks.

For scalability we proposed a storage management strategy for small IoT devices to effectively utilize limited storage space. By using the proposed method, our trust protocol with limited storage space is able to achieve a similar performance level as that with unlimited storage space. To demonstrate the applicability, we applied our trust management protocol to a service composition application, with or without service constraints in SOA-based IoT systems. Our simulation results demonstrated that with our adaptive trust protocol design, the application running on top of the trust protocol is able to approach the ideal performance upon convergence and can significantly outperform the counterpart non-trust-based random selection service composition, as well as service composition running on top of EigenTrust and PeerTrust. We also demonstrated that our technique is effective in deciding and applying the best weight combination (w_f, w_l, w_c) for combining social similarities that will lead to the most credible trust feedback to minimize trust bias and maximize the utility of the application.

In the paper we only considered persistent attackers [30], i.e., attackers that perform self-promoting, opportunistic service, bad-mouthing, and ballot-stuffing attacks with probability one, or wherever there is a chance. In the future, we plan to consider other attacker behavior models including opportunistic collusion attacks (where malicious nodes collude only opportunistically depending on the situation given), random attacks (where malicious nodes perform attack on and off randomly to elude detection) and insidious attacks (where malicious nodes hide until a critical mass is gathered so as to launch more effective collusion attacks) to further test the resiliency property of our adaptive and scalable trust protocol design. Also, the incentives considered in this paper are self-interest (based on which a node performs self-promoting and opportunistic service attacks) and social relationships (based on which a node performs bad-mouthing and ballot-stuffing attacks). The use of participant incentives for collusion attacks is an interesting extension out of this paper.

Acknowledgment

This material is based upon work supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under W911NF-12-1-0445.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks,* vol. 54, pp. 2787-2805, Oct. 2010.

[2] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a Social Structure to the Internet of Things," *IEEE Communication Letters,* vol. 15, no. 11, pp. 1193-1195, Nov. 2011.

[3] F. Bao, and I. R. Chen, "Dynamic Trust Management for Internet of Things Applications," *2012 International Workshop on Self-Aware Internet of Things*, San Jose, California, USA, 2012.

[4] F. Bao, and I. R. Chen, "Trust Management for the Internet of Things and Its Application to Service Composition," *IEEE WoWMoM 2012 Workshop on the Internet of Things: Smart Objects and Services*, San Francisco, CA, USA, 2012.

[5] F. Bao, I. R. Chen, and J. Guo, "Scalable, Adaptive and Survivable Trust Management for Community of Interest Based Internet of Things Systems," *11th International Symposium on Autonomous Decentralized System, Mexico City,* Mexico, 2013.

[6] N. Bui, and M. Zorzi, "Health Care Applications: A Solution Based on The Internet of Things," *4th International Symposium on Applied Sciences in Biomedical and Communication Technologies,* Barcelona, Spain, 2011, pp. 1-5.

[7] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, and X. Wang, "TRM-IoT: A Trust Management Model Based on Fuzzy Reputation for Internet of Things," *Computer Science and Information Systems,* vol. 8, no. 4, pp. 1207-1228, Oct., 2011.

[8] P. Doody, and A. Shields, "Mining Network Relationships in the Internet of Things," *International Workshop on Self-Aware Internet of Things*, San Jose, California, USA, 2012, pp. 7-12.

[9] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-Based Framework for High Integrity Sensor Networks," *ACM Transactions on Sensor Networks,* vol. 4, no. 3, pp. 1-37, May 2008.

[10] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, "A Survey on Facilities for Experimental Internet of Things Research," *IEEE Communications Magazine,* vol. 49, no. 11, pp. 58-67, Nov. 2011.

[11] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services," *IEEE Transactions on Services Computing,* vol. 3, no. 3, pp. 223-235, 2010.

[12] Z. Huang, D. Zeng, and H. Chen, "A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce," *IEEE Intelligent Systems,* vol. 22, pp. 68-78, 2007.

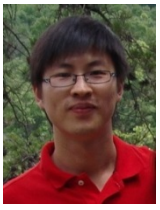
[13] A. J. Jara, M. A. Zamora, and A. F. G. Skarmeta, "An Internet of

Things-Based Personal Device for Diabetes Therapy Management in Ambient Assisted Living (AAL)," *Personal and Ubiquitous Computing*, vol. 15, no. 4, pp. 431-440, 2011.
[14] A. Jøsang, and R. Ismail, "The Beta Reputation System," *Bled Electronic Commerce Conference*, Bled, Slovenia, 2002, pp. 1-14.
[15] S. Kosta, A. Mei, and J. Stefa, "Small World in Motion (SWIM): Modeling Communities in Ad-Hoc Mobile Networking," *7th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Boston, MA, USA, 2010.
[16] M. Kranz, L. Roalter, and F. Michahelles, "Things That Twitter: Social Networks and the Internet of Things," *CIoT Workshop at the Eighth International Conference on Pervasive Computing*, Helsinki, Finland, 2010.
[17] Q. Li, S. Zhu, and G. Cao, "Routing in Socially Selfish Delay Tolerant Networks," *IEEE Conference on Computer Communications*, San Diego, CA, 2010, pp. 1-9.
[18] I. R. Chen, F. Bao, M. Chang, and J.H. Cho, "Trust-based intrusion detection in wireless sensor networks," *IEEE International Conference on Communications,* Kyoto, Japan, June 2011, pp. 1-6.
[19] P. Massa, and P. Avesani, "Trust-aware Recommender Systems," *ACM Recommender Systems Conference, Minneapolis, Minnesota*, USA, 2007.
[20] D. A. Menasce, "QoS Issues in Web Services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72-75, 2002.
[21] C. Prehofer, J. v. Gurp, V. Stirbu, S. Sathish, P. P. Liimatainen, C. d. Flora, and S. Tarkoma, "Practical Web-Based Smart Spaces," *IEEE Pervasive Computing*, vol. 9, no. 3, pp. 72-80, 2010.
[22] M. Presser, and S. Krco, *The Internet of Things Initiative D2.1: Initial report on IoT applications of strategic interest*, 2011.
[23] J. Rao, and X. Su, "A Survey of Automated Web Service Composition Methods," *1st Conf. on Semantic Web Services and Web Process Composition*, San Diego, CA, USA, 2004, pp. 43-54.
[24] W. Ren, "QoS-aware and compromise-resilient key management scheme for heterogeneous wireless Internet of Things," *International Journal of Network Management,* vol. 21, no. 4, pp. 284-299, July 2011.
[25] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," *Computer,* vol. 44, no. 9, pp. 51-58, 2011.
[26] Z. Shelby, "Emebedded Web Services," *IEEE Wireless Communications,* vol. 17, no. 6, pp. 52 - 57 Dec. 2010.
[27] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32-39, 2014.
[28] L. Zhou, and H.-C. Chao, "Multimedia Traffic Security Architecture for the Internet of Things," *IEEE Network,* vol. 25, no. 3, pp. 35-40, May-June, 2011.
[29] J.H. Cho, et al., "Effect of Intrusion Detection on Reliability of Mission-Oriented Mobile Group Systems in Mobile Ad Hoc Networks," *IEEE Trans. on Reliability*, vol. 59, 2010, pp. 231-241.
[30] R. Mitchell and I. R. Chen, "Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems," *IEEE Transactions on Reliability*, vol. 62, no. 1, 2013, pp. 199-210.
[31] I.R. Chen, F. Bao, M. Chang, and J.H. Cho, "Dynamic Trust Management for Delay Tolerant Networks and Its Application to Secure Routing," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 5, 2014, pp. 1200-1210.
[32] J.H. Cho, I.R. Chen, and A.Swami, "Modeling and analysis of trust management for cognitive mission-driven group communication systems in mobile ad hoc networks," *Inter. Conf. on Computational Science and Engineering, 2009,* pp. 641-650.
[33] I. R. Chen, J. Guo and F. Bao, "Trust Management for Service Composition in SOA-based Internet of Things Systems," *IEEE 2014 WCNC*, Istanbul, Turkey, April 2014, pp. 3486-3491.
[34] R. Mitchell and I. R. Chen, "Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems," *IEEE Transactions on Reliability*, vol. 62, no. 1, 2013, pp. 199-210.
[35] R. Mitchell and I.R. Chen, "A survey of intrusion detection in wireless network applications," *Computer Communications*, vol. 42, 2014, pp. 1-23.
[36] J.H. Cho, A. Swami, and I.R. Chen, "Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 1001-1012, May 2012.
[37] Z. Yan, P. Zhang, and A.V. Vasilakos, "A Survey on Trust Management for Internet of Things," *Journal of Network and Computer Applications*, vol. 42, pp. 120-134, 2014.
[38] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, 2014, pp. 1-10.
[39] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," *12th International Conference on World Wide Web*, Budapest, Hungary, May 2003.
[40] L. Xiong, and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities", *IEEE Trans. on Knowledge and Data Engineering*, v.16, pp. 843-857, July 2004.
[41] The ns-3 simulator, http://www.nsnam.org.
[42] ns-3 Tutorial, http://www.cpe.ku.ac.th/~anan/myhomepage/wp-content/uploads/2012/01/ns3-part1-introduction.pdf.
[43] W. Sherchan, S. Nepal, and C. Paris, "A Survey of Trust in Social Networks," *ACM Computing Survey*, Vol. 45, No. 4, Article 47, August 2013.
[44] Z. Malik and A. Bouguettaya,"RATEWeb: Reputation Assessment for Trust Establishment among Web Services," *The VLDB Journal*, vol. 18, 2009, pp. 885-911.
[45] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness Management in the Social Internet of Things," *IEEE Transactions on Knowledge and Data Management*, vol. 26, no. 5, 2014, pp. 1253-1266.
[46] Y. B. Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Trust management system design for the Internet of Things: A context-aware and multi-service approach," *Computers and Security*, vol. 39, Nov. 2013, pp. 351-365.

AUTHOR BIOGRAPHIES

Ing-Ray Chen received the BS degree from the National Taiwan University, and the MS and PhD degrees in computer science from the University of Houston. He is a professor in the Department of Computer Science at Virginia Tech. His research interests include mobile computing, wireless systems, security, trust management, and reliability and performance analysis. Dr. Chen currently serves as an editor for *IEEE Communications Letters*, *IEEE Transactions on Network and Service Management*, *The Computer Journal*, and *Security and Network Communications*.

Jia Guo received the B.S. degree in Computer Science from Jilin University, China in 2012. Currently he is pursuing his Ph.D. degree in the Computer Science Department at Virginia Tech. His research interests include trust management, Internet of things, mobile computing, secure and dependable computing, and performance analysis.



Fenye Bao received the B.S. degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2006 and the M.E. degree in software engineering from Tsinghua University, Beijing, China in 2009. He received his PhD degree in Computer Science from Virginia Tech in 2013. Currently he is a technical staff member of LinkedIn. His research interests include trust management, security, social networks, wireless sensor networks, and mobile computing.