A Mobile Cloud Hierarchical Trust Management Protocol for IoT Systems

Jia Guo, Ing-Ray Chen
Department of Computer Science
Virginia Tech
{jiaguo, irchen}@vt.edu

Jeffrey J.P. Tsai
Department of Bioinformatics and Biomedical Engineering
Asia University
jjptsai@gmail.com

Abstract — **We propose and analyze a 3-tier cloud-cloudlet-device hierarchical trust management protocol called IoT-HiTrust for large-scale IoT systems. Our mobile cloud hierarchical trust management protocol allows an IoT device to report its service experiences and query the trustworthiness of another IoT device for service composition and selection following a simple localized report-and-query paradigm. We verify IoT-HiTrust's convergence, accuracy, and resiliency properties against self-promotion, discriminatory, bad-mouthing, ballot-stuffing, and opportunistic service attacks despite intermittent network disconnection to the cloud.**

Keywords— Internet of things; trust management; mobile cloud computing; service composition; performance analysis.

I. Introduction

In recent years we have witnessed a proliferation of Internet of things (IoT) devices such as RFID tags, sensors, smartphones, smart appliances, environmental monitoring devices, etc. each capable of providing services upon request. It is anticipated that service-oriented IoT applications will have great social impacts to our everyday life [9].

In this paper, we propose and analyze a mobile cloud hierarchical trust management protocol called IoT-HiTrust with the goal to support trustworthy service management in large-scale IoT applications. Trust management is needed because not all IoT devices will be trustworthy and some IoT devices may behave maliciously to disrupt the cloud service (e.g., for an adversary) or just for their own gain (e.g., for increasing their chances to be selected to provide requested services). Furthermore, users of IoT devices are likely to be socially connected via social networks. Therefore, misbehaving nodes with close social ties can collude and monopoly a class of services. For service-oriented IoT systems [1] it is important for an SR to know the service trustworthiness of SPs bidding to provide a requested service. For participatory sensing based applications [5], it is critical to assess source trustworthiness of IoT devices which report sensing results so untrustworthy data can be filtered out before data analysis is taken.

Our paper has the following unique contributions:

1. Unlike existing distributed IoT trust management protocols [1, 3], IoT-HiTrust is scalable as it leverages a 3-tier cloud-cloudlet-device hierarchy which allows an IoT device to report its service experiences and query the service trustworthiness of another IoT device by a simple localized *report-and-query* paradigm. The response to an IoT device's query retains the concept of subjective trust evaluation, i.e., it incorporates the IoT device's own observations and other IoT devices' recommendations weighted by the trust of this IoT device toward them.
2. We demonstrate that IoT-HiTrust can achieve trust accuracy, convergence, and resiliency against self-promotion, discriminatory, bad-mouthing, ballot-stuffing, and opportunistic service attacks while achieving scalability, because it can leverage cloud services to aggregate broad service evidence from all IoT devices in the system. Furthermore we demonstrate that the desirable trust accuracy, convergence, and resiliency properties can still be achieved despite intermittent network disconnection.

The rest of the paper is organized as follows. Section II discusses the system model. Section III describes IoT-HiTrust in detail and explains our efficient and effective hierarchical trust protocol design for managing a huge number of IoT devices. In Section IV we conduct a performance analysis of IoT-HiTrust and demonstrate its trust resiliency, convergence and accuracy properties. Finally in Section V we conclude the paper and outline some future research areas.

II. System Model

A. *Cloud-Cloudlet-Device Architecture*

As illustrated in Figure 1, we leverage a 3-tier mobile cloud hierarchy [8] for hierarchical IoT trust management. Cloudlets (each occupying a physical region) are sitting in the middle layer, bringing IoT devices closer to the cloud. We assume that a cloudlet is formed by a set of heavyweight IoT devices (e.g., PCs, servers, etc.) with moderate computational and storage capability to offload lightweight IoT devices (e.g., smart phones, sensors, PDAs, etc.) sitting at the bottom layer. Heavyweight IoT devices typically are well connected to the Internet; they may be mobile but only move within a cloudlet region. Lightweight IoT devices typically are intermittently connected to the Internet; they are carried by their owners and can move from one cloudlet to another due to mobility. When an IoT device is disconnected from the Internet, it connects to a regional cloudlet through wireless communication for service continuity. To save energy and bandwidth, an IoT device will always communicate with the cloud through its regional cloudlet due to physical proximity. The cloud at the top level is a logical entity consisting of many cloud servers and is assumed infallible with tight security and reliability protection. Using IoT-HiTrust, the cloud periodically evaluates trustworthiness of all IoT devices in a cloudlet region and selects a group of heavyweight IoT devices to form the region's cloudlet. In return for the surrogate services provided by these heavyweight IoT devices to bring IoT devices closer to the cloud, the cloud grants access privileges to cloud resources.

Henceforth, we will call heavyweight IoT devices selected to govern a region's cloudlet as "cloudlet devices," with the understanding that cloudlet devices are just heavyweight IoT devices. Figure 1 shows two cloudlets, CL_1 and CL_2, each with three heavyweight IoT devices serving as cloudlet devices. We use the term "*node"* to loosely refer to a node in the hierarchy. A node at the bottom layer of the hierarchy is an IoT device, at the middle layer is a cloudlet, and at the top layer is the cloud.

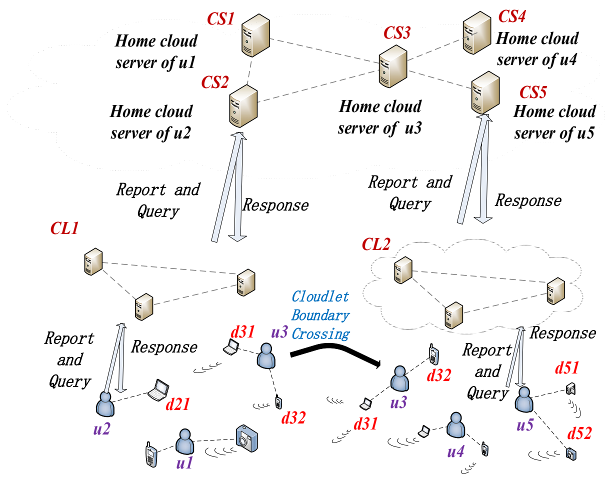

Figure 1: Cloud-Cloudlet-Device Architecture.

Figure 1illustrates a cloud with 5 cloud servers. Each user (or an IoT device) has its unique identity. A user's unique id is at the cloud service level. An IoT device's unique id is at the device level. Each user with its unique id is internally assigned to a "home" cloud server based on distributed hash table (DHT) techniques for load balancing. The home server of a user will manage the user's data internally. While a user's VM may migrate from one place to another, the user's home cloud remains the same. So if the user's VM is asking for the user's data, the request will be routed to the user's home cloud. In Figure 1, CS_2 is the home cloud server of user u_2 and CS_3 is the home cloud server of user u_3. For the case in which a user owns several IoT devices, all IoT devices also map to the owner's home cloud server as their home cloud server. For example in Figure 1, user u_3 owns two devices, d_{31} and d_{32}, for which the home cloud sever is also CS_3. Each cloudlet only relays requests/responses from/to IoT devices under its region. In addition, each cloudlet caches trust information. In case of Internet disconnection, a cloudlet can operate in disconnection mode [8] to answer user queries issued from IoT devices in its region. One can reduce the regional size to the radio range to ensure that mobility and instability of radio environments will not be a major factor to prevent nodes of a cloudlet region from communicating directly with the cloudlet. When an IoT device in one cloudlet region moves to another cloudlet region, it performs a registration/deregistration action to the two involving cloudlets. For example in Figure 1, user u_3 moves across the cloudlet boundary, causing deregistration with the old cloudlet CL_1 and registration with the new cloudlet CL_2. A pointer is recorded by CL_1 so that a response for u_3 can be redirected to CL_2.

B. *Recommendation Filtering based on Social Similarity*

Our trust model is based on social relationships among human owners of IoT devices. A user upon receiving a recommendation from an IoT device, will measure the trustworthiness of the recommender (or rater) so as to apply "recommendation filtering" based on its social relationships with the recommender (or rater). We consider three core social metrics for measuring social relationships which are multifaceted: friendship (representing intimacy), social contact (representing closeness), and community of interest (representing knowledge and standard on the subject matter). The idea is that two users sharing similar social relationships are likely to have similar views towards services provided by a trustee IoT device. Social relationships between owners are translated into social relationships between IoT devices as follows:

1. *Friendship*: Each owner has a list of friends (i.e., other owners), representing its social relationships. This friendship list varies dynamically as an owner makes or denies other owners as friends. If the owners of two IoT devices are friends, then it is likely they will be cooperative with each other. For ease of discussion, we do not differentiate friends from acquaintances. We can easily extend IoT-HiTrust to take this finer granularity into consideration.
2. *Social Contact*: A device may be carried or operated by its owner in certain environments (e.g., work vs. home or a social club). Two devices have high social contact opportunities when their owners have similar mobility patterns.
3. *Community of Interest* (CoI): Each owner has a list of communities of interest such as health, sport, travel, etc. Nodes belonging to a similar set of communities likely share similar interests or capabilities [3].

To facility measuring social similarity with other owners, an IoT device belonging to user u_x maintains three lists in its profile (as illustrated in Figure 2):

1. Friends of u_x, denoted by a set $F_x = \{u_1, u_2, \dots\}$;
2. Locations that u_x frequently visited for social contact, denoted by a set $S_x = \{Loc_1, Loc_2, \dots\}$;
3. Communities of interest that u_x is a member of, denoted by a set $C_x = \{CoI_1, CoI_2, \dots\}$.

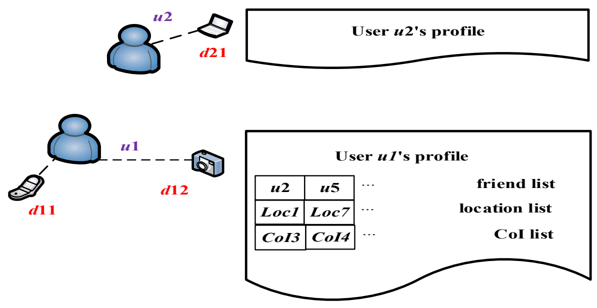

Figure 2: Each User Stores Its Friend, Location, and CoI Lists for Detecting Social Similarity with other Users.

A user may designate one of its IoT devices to update such information and share the information with other IoT devices it owns. Although social similarity is between two users, it is propagated to IoT devices owned by the two users, so social similarity is also between two devices.

C. 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 [7] and is not addressed in this paper. We are concerned with trust-related attacks that can disrupt the trust system. By a malicious node, we refer to a node having only self-interest. A smart malicious node can choose to provide good or bad service depending on whether it would benefit itself and its allies (other malicious nodes or friends) which altogether can collude to monopoly service. In this paper we consider a malicious IoT device (because its owner is malicious) capable of performing the following trust-related attacks and our IoT-HiTrust protocol design must maintain desirable accuracy, convergence, and resiliency properties against these attacks:

1. *Self-promoting attacks*: a malicious node can promote its importance (by providing good recommendations for itself) for it to be selected as an SP, but then can provide bad or malfunctioned service.
2. *Bad-mouthing attacks*: a malicious node 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 an SP.
3. *Ballot-stuffing attacks*: a malicious node 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 an SP.
3. *Discriminatory attacks* (or conflicting behavior attacks): a malicious node can discriminatively attack non-friends or nodes without strong social ties (without many common friends) because of human nature or propensity towards friends in social IoT systems. Serving as a recommender, if the target node is a friend, it will always provide a good service recommendation even if the target node does not provide good service. On the other hand, if the target node is a non-friend, it will perform bad-mouthing attack if the target node is a good node and ballot-stuffing attack if the target node is a malicious node.
4. *Opportunistic service attacks*: a malicious node 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. We assume that a malicious node has a low trust threshold below which it will behave (providing good service) in order to raise its reputation, and a high trust threshold above which it will misbehave (providing bad service) to take the advantage of its high reputation for self-gain.

III. IoT-HiTrust Protocol Design

A. Reporting

Whenever a service is received, a user (using its primary IoT device) reports whether it is satisfied with the service provided by an IoT device to the user's home cloud server via a service rating report. Let the current user satisfaction experience of user u_x toward device d_i be represented by a value, $f_{x,i}$ which can be a real number in the range of 0 to 1 indicating the user satisfaction level, or simply a binary value, with 1 indicating satisfied and 0 not satisfied. Here $f_{x,i}$ is the first piece of information sent from u_x to its home cloud server. A timestamp is also sent in the report to indicate the time at which this service rating happens. This allows cloud servers to know the event occurrence times of reports for regression analysis if necessary.

Each user maintains its (F, S, C) profile separately. When user u_x encounters user u_y, they exchange their (F_x, S_x, C_x) and (F_y, S_y, C_y) profiles so as to measure their mutual social similarity. To preserve energy, they can exchange the profile information the very first time they encounter or periodically. This is especially so if user profile information does not change much over time. To preserve privacy, they only want to reveal common elements in the *F, S,* and *C* lists (If any) but do not want to let the other party know their entire (F, S, C). To achieve this, users u_x and u_y can first authenticate each other using standard PKI. User u_x can then use a cryptographic hash function in combination with a secret session key *K* (established via PKI during user authentication) to generate a hash-based message authentication code HMAC(*K, p*) for $p \in (F_x, S_x, C_x)$ and then transmit HMAC(*K, p*) along with HMAC(*K*, HMAC(*K, p*)) to u_y. When u_y receives the message, it can unilaterally generate HMAC(*K*, HMAC(*K, p*)) using HMAC(*K, p*) sent by u_x. If this matches with HMAC(*K*, HMAC(*K, p*)) sent by u_x, then u_y verifies the message received is indeed sent by u_x. Then u_y can compare HMAC(*K, p*) with HMAC(*K, q*) for $q \in (F_y, S_y, C_y)$. If HMAC(*K, p*)=HMAC(*K, q*) then *p*=*q* and a common friend, location, or CoI (corresponding to *F, S,* or *C*) is identified. If HMAC(*K, p*)≠HMAC(*K, q*), it prevents the identities of uncommon friends/locations/CoIs from being revealed.

User u_x and user u_y then apply cosine similarity [3] to compute the social similarity between u_x and u_y in friendship, social contact, and community of interest, denoted by $sim_i(u_x, u_y)$, $i \in \{f, s, c\}$, and report them through the local cloudlet to the home cloud servers of user u_x and u_y. When the home cloud server of u_x receives $sim_i(u_x, u_y)$, $i \in \{f, s, c\}$, from user u_x which just encounters user u_y, it computes the social similarity between users u_x and u_y (who now serves as a rater or recommender) as a weighted combination of all social similarity metrics, i.e., friendship, social contact, and community of interest, as follows:

$$sim(u_x, u_y) = \sum_{i \in \{f,s,c\}} w_i \cdot sim_i(u_x, u_y) \tag{1}$$

where $0 \le w_f, w_s, w_c \le 1$, with $w_f + w_s + w_c = 1$, are social similarity weight parameters to be dynamically adjusted by IoT-HiTrust to maximize trust protocol performance.

B. *Querying/Replying and Trust Computation*

Whenever a user wants to know the trust value of an IoT device, it simply sends a query to its home cloud server. For example, in Figure 1, u_2 will send a query to its home cloud server CS_2 to know its "subjective" trust toward d_{31} which belongs to u_3.

Let the "subjective" trust value of user u_x toward d_i be denoted by $t_{x,i}$. The home cloud server of u_x computes $t_{x,i}$ by combining $u_x's$ direct trust toward d_i ($t^d_{x,i}$) based on self-observation reports, and $u_x's$ indirect trust toward d_i ($t^r_{x,i}$) based on other users' ratings, as follows:

$$t_{x,i} = \mu_{x,i} \cdot t^d_{x,i} + (1 - \mu_{x,i}) \cdot t^r_{x,i} \qquad (2)$$

Here, $\mu_{x,i}$ is a weight parameter ($0 \le \mu \le 1$) to weigh the importance of direct trust relative to indirect trust. The selection of $\mu_{x,i}$ is critical to trust evaluation. We apply adaptive filtering developed in [3] to adjust $\mu_{x,i}$ dynamically to effectively cope with malicious attacks and to improve trust accuracy.

The direct trust $t^d_{x,i}$ in Equation 2 is computed by Beta Reputation [4] under which the trust value is modeled as a random variable in the range of [0, 1] following the Beta (α, β) distribution. The numbers of positive and negative experiences are modeled as binomial random variables. Since the beta-binomial is a conjugate pair, this leads to a posterior beta distribution with updated parameters. Specifically, we can calculate $t^d_{x,i} = \alpha/(\alpha + \beta)$ is the mean "direct" trust where α is the number of positive service experiences and is updated by $\alpha = \alpha + f_{x,i}$, and β is the number of negative service experiences and is updated by $\beta = \beta + (1 - f_{x,i})$ upon receiving an assessment $f_{x,i}$ in the range of [0, 1] (a real number) from user u_x about d_i's service quality. Here $f_{x,i}$ contributes to positive service experience and $1 - f_{x,i}$ contributes to negative experience.

The indirect trust $t^r_{x,i}$ in Equation 2 is computed by the home cloud server of u_x by locating social similarity records $sim(u_x, u_y)'s$ in its local storage, and selecting top-R raters from R users with the highest similarity scores with u_x as follows:

$$t^r_{x,i} = \sum_{u_y \in U} \frac{sim(u_x, u_y)}{\sum_{u_z \in U} sim(u_x, u_z)} \cdot t^d_{y,i} \qquad (3)$$

Here, U is a set of up to R raters whose $sim(u_x, u_y)$ scores are the highest, $u_y \in U$ is a rater selected, and $t^d_{y,i}$ is the service rating provided by u_y toward device d_i. Note that $t^d_{y,i}$ is stored in the home cloud server of u_y but it is obtainable after the home cloud server of u_x communicates with the home cloud server of u_y. In Equation 3, the service rating provided from u_y toward d_i $(\text{i.e.}, t^d_{y,i})$ is weighted by the ratio of the similarity score of u_x toward u_y to the sum of the similarity scores toward all raters. That is, if the similarity score of u_x toward u_y is high relative to that of u_x toward other raters, then the home cloud server of u_x will put a relatively high weight on the rating $t^d_{y,i}$ provided by u_y to compute $t^r_{x,i}$.

C. *Dealing with Network Disconnection*

A cloudlet may lose connectivity with the cloud if all heavyweight IoT devices selected as the cloudlet devices for the cloudlet experience network disconnection in emergency situations such as a network service disruption. A cloudlet must maintain service continuity to IoT devices during network disconnection. In case the cloudlet can still communicate with a neighbor cloudlet which still has connectivity with the cloud, then all reports, queries, and responses can route through the neighbor cloudlet. Otherwise, it will have to operate in disconnected mode [8] using cached data. For a heavy IoT device that is typically not mobile or only moves within the region, the home regional cloudlet has all the data it needs for answering a user query regarding the trustworthiness of that IoT device, since it caches all reports, responses, and social similarity reports with that IoT device pass through it. So all it has to do is to follow the computational procedure described earlier to assess the trustworthiness of that IoT device, as if the computation is performed by the cloud itself. For a lightweight IoT device, the home region cloudlet may lose some precision in estimating the trustworthiness of the IoT device because it does not have all the data needed. For example it may miss some reports of that IoT device when that IoT device moves away from its region. The trust accuracy is largely affected by the mobility of the user carrying the IoT device. In Section IV we will assess the extent to which the trust accuracy is impacted under various mobility scenarios.

Table I: Parameters and Default Values.

parameter	value	parameter	value	parameter	value
N_T	2000	$m \times m$	16×16	T	200 hrs
N_U	500	P_M	30%	R	3
N_C	10	σ_c	1%	λ	1/day

IV. IoT-HiTrust Protocol Performance

In this section, we analyze IoT-HiTrust performance using ns3 simulation. Table I lists the parameters and their default values.

The experiment setting is explained as follows. We consider a large IoT 16x16 area with $N_T = 2000$ IoT devices. These IoT devices are randomly assigned to $N_U = 500$ users. The number of cloud servers is $N_C = 10$ such that each cloud server can approximately handle $N_T/N_C = 200$ IoT devices. Users are connected in a social network represented by a friendship matrix. We consider these users moving according to the small world in motion (SWIM) mobility model [6] modeling human social behaviors for the purpose of assessing the social contact similarity metric between any pair of users. Three IoT devices in a region are selected as cloudlet devices periodically responsible for caching and relaying reports, queries, and responses for IoT devices in the region. 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. The system runs continuously although trust convergence is achieved in less than 200 hours. The number of recommendations (or ratings) is *R*=3.

The user satisfaction levels of service invocations, i.e., $f_{x,i}$ in the range of [0, 1] from user u_x about d_i's service quality, are from a real dataset [11] 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., $t^d_{x,i}$) is calculated by Equation 2 with "ground truth" user satisfaction experiences as input, $t^d_{x,i}$ 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 recorded in the database from the direct trust evaluation outcome $t^d_{x,i}$. 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 $w_f = w_l = w_c = 1/3$ considering friendship, social contact, and community of interest are equally important. The % of malicious users (P_M) is set to 30%. Malicious users are chosen randomly from 500 users. A malicious user will perform self-promoting, bad-mouthing, ballot-stuffing, discriminatory, and opportunistic service attacks as described in Section II.C. In particular, a malicious user u_y can provide a bad recommendation $t^d_{y,i}$=0 (see Equation 3) against a good device *i* for bad-mouthing attacks, and conversely a good recommendation $t^d_{y,i}$=1 for a malicious device *i* for ballot-stuffing attacks. Our protocol handles ballot-stuffing and bad-mouthing attacks by recommendation filtering (see Section II.B) and indirect trust $t^r_{x,i}$ computation (see Equation 3).

Figure 3 shows IoT-HiTrust performance in terms of trust accuracy, convergence, and resiliency against attacks. It shows the progressive trust value of a "good" IoT device as assessed by the cloudlet in the home region where the IoT device initially resides vs. time for the case in which P_M = 30%. The "ground truth" trust value of this good device is depicted by the solid line. Note that the "ground truth" trust value of this good node is not 1 because $f_{x,i}$ (service experience of user u_x toward d_i) retrieved from the trace dataset [11] is not necessarily 1. The progressive trust value measured by IoT-HiTrust is depicted by dashed lines. We consider the scenario in which the "good" IoT device's cloudlet is disconnected due to network disconnection at times (marked by zig-zag patterns) during which it can only perform disconnected trust assessment. The label "no region roaming" means that the target IoT device stays in the same region all the time. The label "neighbor region roaming" means that the target IoT device stays in the home region but roams to neighbor regions from time to time. The label "hopper" means that the target IoT device moves across region boundary frequently.

We see that trust accuracy decreases as the target "good" IoT device moves across the regional boundary more frequently. However, given time, all cases eventually converge after sufficient data are collected to allow accurate trust assessment. Correspondingly, Figure 4 shows IoT-HiTrust performance in terms of the trust value of a "bad" IoT device vs. time when P_M = 30%. This bad node's trust value is up and down because of opportunistic service attacks performed by the bad node. We again confirm that trust accuracy decreases as the target "bad" IoT device moves across the regional boundary more frequently. However, trust accuracy is restored as soon as the home regional cloudlet is reconnected. Figures 3 and 4 verify that our IoT-HiTrust protocol is effective to deal with intermittent disconnection for providing service continuity, especially for IoT devices that do not move much such as heavyweight IoT devices.

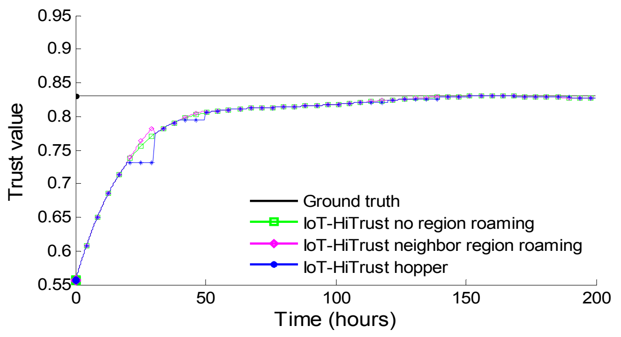

Figure 3: Trust Value of a Good Node under Intermittent Disconnection.

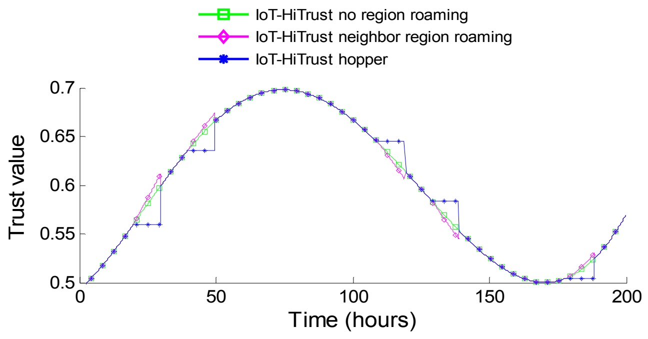

Figure 4: Trust Value of a Bad Node under Intermittent Disconnection.

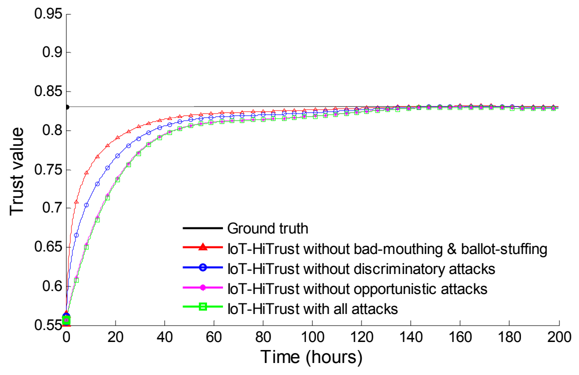

Figure 5: Trust Value of a Good Node under Various Attack Types.

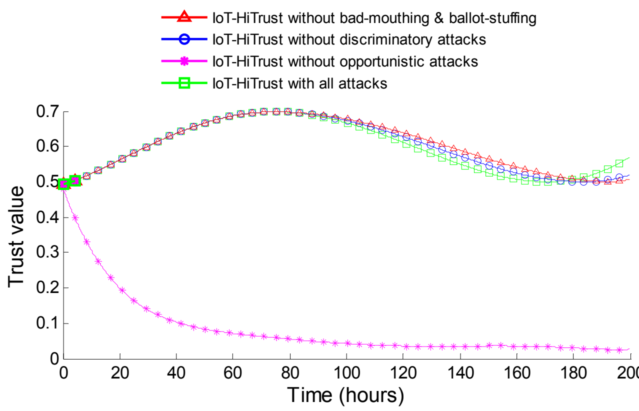

Figure 6: Trust Value of a Bad Node under Various Attack Types.

We test the sensitivity of the results w.r.t. the attack type in Figures 5 and 6.

Figure 5 demonstrates the effect of the attack type on trust accuracy, convergence, and resiliency of a "good" target node that roams between neighbor regions. The curve labeled by "with all attacks" (green curve) corresponds to the curve in Figure 3. There are three other curves, i.e., "without bad-mouthing and ballot-stuffing attacks," "without discriminatory attacks," and "without opportunistic attacks," each with a particular attack type being removed so as to see its absence on accuracy, convergence, and resiliency. We see from Figure 5 that recommendation attacks (i.e., bad-mouthing and ballot-stuffing attacks combined) have the biggest effect on accuracy, convergence, and resiliency because without them (red curve) the system can approach ground truth (black curve) in the shortest amount of time. Discriminatory attacks also have some effect on accuracy, convergence, and resiliency because without them (blue curve) the system can converge faster than with them (green curve), although the sensitivity is not as high. Lastly opportunistic attacks have the least effect on accuracy, convergence, and resiliency because without them (pink curve) the system does not converge any faster than with them (green curve). This is so because a bad node performing opportunistic attacks will only affect its own trust value, not the "good" target node's trust value. Therefore the most severe attack type appears to be the recommendation attack as it will ruin the reputation of the "good" target node.

Figure 6 demonstrates the effect of the attack type on trust accuracy, convergence, and resiliency of a "bad" node that roams between neighbor regions. Similarly the curve labeled by "with all attacks" (green curve) corresponds to the curve in Figure 4. There are three other curves, i.e., "without bad-mouthing and ballot-stuffing attacks," "without discriminatory attacks," and "without opportunistic attacks," each with a particular attack type being removed so as to see its absence on accuracy, convergence, and resiliency. Since the target node is a "bad" node, we see from Figure 6 that opportunistic attacks have the most severe effect on accuracy, convergence, and resiliency because without them (pink curve) the system converges to ground truth (zero for a bad node) while with them the system converges to the up and down curve (green curve). This is so because the "bad" target node performing opportunistic service attacks will alternate between behave and misbehave to keep its trust value between the high threshold and low threshold. We see from Figure 6 that the other two attack types do not have a high impact on the trust value of this "bad" target node. In practice, a bad node will always perform opportunistic service attacks to disguise itself as a good node without being caught. The best the system can do is to accurately track a bad node's trust status to refrain it from providing bad service and to decrease the chance of selecting bad nodes for providing service.

V. Conclusion

In this paper, we designed and analyzed a scalable hierarchical trust management protocol called IoT-HiTrust for large IoT systems. We verified that IoT-HiTrust is effective and efficient for dealing with intermittent disconnection, while achieving desirable trust properties including accuracy, convergence, and resiliency against malicious attacks. In the future, we plan to further validate our hierarchical trust protocol with real-world mobile cloud applications [9] together with real-world IoT service quality and mobility traces [10, 11].

Acknowledgment

This work is supported in part by the U.S. Army Research Office under contract number W911NF-12-1-0445.

References

[1] I. R. Chen, F. Bao, and J. Guo, "Trust-based Service Management for Social Internet of Things Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, Nov-Dec 2016, pp. 684-696.

[2] I. R. Chen and F.B. Bastani, "Effect of Artificial-Intelligence Planning-Procedures on System Reliability," *IEEE Transactions on Reliability*, vol. 40, no. 3, 1991, pp. 364-369.

[3] I.R. Chen, J. Guo, and F. Bao, "Trust Management for SOA-based IoT and Its Application to Service Composition," *IEEE Transactions on Service Computing*, vol. 9, no. 3, 2016, pp. 482-495.

[4] A. Jøsang, and R. Ismail, "The Beta Reputation System," *Bled Electronic Commerce Conference*, Bled, Slovenia, 2002, pp. 1-14.

[5] W.Z. Khan, Y. Xiang, M.Y. Aalsalem, and Q. Arshad, "Mobile Phone Sensing Systems: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 402–427, 2013.

[6] S. Kosta, A. Mei, and J. Stefa, "Small World in Motion (SWIM): Modeling Communities in Ad-Hoc Mobile Networking," *7th IEEE Conf. Sensor, Mesh and Ad Hoc Communications and Networks*, Boston, MA, USA, 2010.

[7] R. Mitchell and I. R. Chen, "Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles using Behavior Rule Specifications," *IEEE Trans. Systems, Man and Cybernetics*, vol. 44, no. 5, 2014, pp. 593-604.

[8] M. Satyanarayanan, et al., "The role of cloudlets in hostile environments," *IEEE Pervasive Computing*, Oct. 2013, pp. 40-49.

[9] Y. Wang, I.R. Chen, and D.C. Wang, "A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges," *Wireless Personal Communications*, vol. 80, no. 4, 2015, pp. 1607-1623.

[10] I.R. Chen and N. Verma, "Simulation study of a class of autonomous host-centric mobility prediction algorithms for wireless cellular and ad hoc networks," *36th Annual Symposium on Simulation*, 2003, pp. 65-72.

[11] 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.