CTRUST: A Dynamic Trust Model for Collaborative Applications in the Internet of Things

Anuoluwapo A. Adewuyi, *Student Member, IEEE*, Hui Cheng, *Member, IEEE*, Qi Shi, Jiannong Cao, *Fellow, IEEE*, Áine MacDermott, *Member, IEEE*, and Xingwei Wang

***Abstract*—Security through trust presents a viable solution for threat management in the Internet of Things (IoT). Currently, a well-defined trust management framework for collaborative applications on the IoT platform does not exist. In order to estimate reliably the trust values of nodes within a system, the trust should be measured by suitable parameters that are based on the nodes' functional properties in the application context. Existing models do not clearly outline the parametrization of trust. Also, trust decay is inadequately modeled in most current models. In addition, trust recommendations are usually inaccurately weighted with respect to previous trust, thereby increasing the effect of bad recommendations. A new model, CTRUST, is proposed to resolve these shortcomings. In CTRUST, trust is accurately parametrized while recommendations are evaluated through belief functions. The effects of trust decay and maturity on the trust evaluation process were studied. Each trust component is neatly modeled by appropriate mathematical functions. CTRUST was implemented in a collaborative download application and its performance was evaluated based on the utility derived and its trust accuracy, convergence, and resiliency. The results indicate that IoT collaborative applications based on CTRUST gain a significant improvement in performance, in terms of efficiency and security.**

***Index Terms*—Collaborative computing, distributed applications, Internet of Things (IoT), security and privacy protection, trust management.**

I. Introduction

THE INTERNET of Things (IoT) vision is fast becoming a reality and extensive research work is being carried out across the entire IoT spectrum to resolve any key challenges that remain [1]–[3]. IoT aims at enabling objects in the physical world to be able to communicate with the digital or cyber world, thus bridging the gap between the two. For this to occur, the objects must be "smart." Each smart object has a unique identifier and some basic computing capabilities, such as processing, networking, and service discovery [4]. These objects are then able to "talk" to one another through some communication protocols and connect to another device on the Internet. The IoT vision is built upon other research areas, such as cyber-physical systems, wireless sensor networks (WSNs), big data, machine learning, adhoc networks, mobile computing, and ubiquitous computing.

The realization of the IoT vision implicitly requires collaborations and interoperability between various devices and networks on a massive scale. Information security management is, therefore, of critical importance and more complex to manage. The heterogeneous and pervasive nature of IoT creates a larger threat landscape than ever before [5]. Also, due to the different IoT technologies being used, and the fact that most IoT devices have limited computing power, employ a distributed architecture and use less conventional networking methods, traditional security management used in the current Internet cannot be directly applied here [5], [6]. New security countermeasures are required that are lightweight, intelligent, and operable in real-time [5]–[7]. While this remains an ongoing research challenge, an interesting candidate solution is security through trust.

Trust is an important component of computer security [8]. Given that the IoT consists of services and devices provided by different actors who may be unknown to end users, the necessity of trust evaluation is higher than it is in the traditional Internet [5], [9]. Users need to be sure that the services offered to them will provide the highest utility. Collaborating peers need to be sure of the identities of the peers they interact with. There is also a need to avoid or mitigate the myriad of security risks inherent in the IoT paradigm. The difficulty in finding universal security solutions for the IoT means that trust becomes, perhaps, the most important security metric and a measure of how secure the interaction of an IoT entity with another is. In fact, it has been posited that trust management is wider in scope than traditional security management [7]. While traditional security aims to actively keep the realization of risks at a minimum, trust management in IoT strives to identify nodes that are reliable and are perceived to be unlikely to pose any unacceptable risks and restrict relationships to include only such nodes.

The need for reliable trust management is appreciated more in collaborative applications. In collaborative IoT applications, several users, or devices come together and pool their

Manuscript received November 28, 2018; revised February 7, 2019; accepted February 18, 2019. Date of publication February 27, 2019; date of current version June 19, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61872073 and in part by the Program for Liaoning Innovative Research Term in University under Grant LT2016007. *(Corresponding author: Hui Cheng.)*

A. A. Adewuyi, Q. Shi, and Á. MacDermott are with the Department of Computer Science, Liverpool John Moores University, Liverpool L3 3AF, U.K. (e-mail: a.a.adewuyi@2015.ljmu.ac.uk; q.shi@ljmu.ac.uk; a.m.macdermott@ljmu.ac.uk).

H. Cheng is with the School of Computer Science, University of Hertfordshire, Hatfield AL10 9AB, U.K. (e-mail: h.cheng2@herts.ac.uk).

J. Cao is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail: csjcao@comp.polyu.edu.hk).

X. Wang is with the School of Computer Science and Engineering, Northeastern University, Shenyang 110004, China (e-mail: wangxw@mail.neu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2019.2902022

resources to execute a task or provide a service. The resources could be bandwidth, network routes and access, processing power, or storage space. The motive behind the collaboration could be an increase in the speed of the task execution, as is the case in the collaborative download of a file. It could also serve to achieve redundancy and therefore increase reliability, as is the case in collaborative storage, routing, and streaming applications. These are a few examples of use cases where a collaboration is beneficial.

Most of these collaborations will be formed "on the go"; that is, the collaborating peers will be unknown to each other. This introduces even more security risks to the collaborating peers, such as privacy or data loss and creates entry points for other security attacks. The introduction of the notion of trust among peers is one way to minimize these threats [5], [7], [10].

Trust is an essential and common social concept but difficult to define. This is so because it is abstract and multifaceted. It is also either purely or mostly subjective, and its meaning depends on the context in which it is used [11], [12]. Several definitions of trust exist in the literature. Even though there is no agreed definition in literature, a large volume of research on trust shows it is a very important concept [5]. For example, a "trusting intention" is given in [11] as "the extent to which one party is willing to depend on the other party in a given situation with a feeling of relative security, even though negative consequences are possible." This definition correctly identifies trust as a decision (TaaD) taken by the trustor. Another issue is that several definitions of trust do not provide measurable indices that may be used to evaluate trust. The concept of computational trust is, thus, introduced. Computational trust is the adaptation of the social notion of trust to the digital world, so that it can be represented and evaluated by mathematical models [9].

A trust management system (TMS) provides methods and mechanisms to evaluate the trustworthiness of interacting peers, based on a trust model. The trust model determines how trust is computed in a specific context. Several TMSs have been proposed and widely studied in literature. However, there is little work done on the management of trust in IoT contexts [7], [13]–[15]. Many of these models focus mainly or only on trust recommendations and aggregation. Little or no account is taken of other aspects of trust management, such as trust decay, trust parameter selection, and the weighting of trust parameters. In addition, they mainly consider the social interaction among nodes. Hence, trust is derived almost entirely from recommendations. However, given that collaborative applications are task-based systems, the trust score of a trustee node should indicate the degree to which a trustor believes that the trustee is both competent and willing to execute required tasks reliably.

Trust parameters should be based on the contextual functional properties that determine whether a service provider is reliable and provides good service. This notion of a functional trust model that is weighted by subjective beliefs and recommendations of the trustor guides the approach to trust modeling in this paper and agrees with the trust definition given in [16]. The temporal characteristic of trust should be recognized: past or stored trust values must decay with time and be replaced by newer assessments; this process should be modeled by an appropriate decay function. Recommendations from other nodes are necessary as they can be aggregated to make an informed trust decision on a node with which there have been no previous interactions. However, they may be abused to perform good-mouthing or bad-mouthing attacks. Therefore, the trust model should also include a belief function that guides the acceptance and usage of trust recommendations from other nodes.

In summary, there is a need for a well-defined trust model for such IoT applications, where the trust score is a performance metric based on functional properties relevant to the collaboration context. This paper aims to address these research gaps and provide a holistic approach to trust. To achieve this, we consider and model each of the individual components of a TMS required to reliably estimate trust. Based on these, a trust model is designed and evaluated in a collaboration context. Our model makes the following contributions.

1) We use weighted trust parameters (criteria) that can be specified at runtime to adapt the model to different contexts. This means that trust parameters, in contrast to recommendations, form the basic building block for trust computation. In most trust models in literature, the trust computation is built almost entirely on recommendations. This approach does not consider trust as a performance metric, and thus weakens the trustor's decision to trust. In our model, nodes decide the functional parameters required to assess trust satisfactorily, along with their relative importance.
2) We consider trust degradation as a distinct component of the trust computation that is solely based on time. Previous trust models use parameters to weight past trust to current experience. This is done recursively with every new interaction and therefore does not consider the time that has elapsed since a previous trust assessment was made. To resolve this, our model includes a trust decay function with a dynamic component to accommodate different degrees of nodes' willingness to trust. This ensures trust degrades in a consistent manner.
3) We implement an improved recommendation function with the addition of a belief degree. Other trust models only consider the recommender's trust scores in accepting recommendation. We identified other criteria that determine the degree to which a recommendation is accepted in social contexts and combined them to model the belief degree, which we then use to weight recommendations.
4) We introduce a parameter to model trust maturity or equilibrium between two nodes, the point at which trust can be computed using direct interactions alone. This implies that it is possible to determine trusted nodes solely by empirical methods, which is a novel contribution.

The rest of this paper is organized as follows. Section II provides a comprehensive overview of the state-of-the-art regarding trust modeling in IoT, including research gaps and a justification for a well-defined model, that we call

CTRUST. Section III specifies and analyses the CTRUST model design. In Section IV, we implement the model to guide node selection in a chosen collaborative context, present the simulation results and evaluate the performance of the model. In Section V, we compare our model to others in literature to show its distinction and significance in the field of study. Finally, Section VI concludes this paper and highlights possible future work.

II. Background

The concept of trust modeling and management in IoT is a rapidly evolving research area. It is therefore important to enumerate the desirable properties of an ideal TMS for IoT, which we do in Section II-A. These properties were elicited primarily from survey papers that review existing work and discuss challenges with current trust models in IoT. In Section II-B, we show that the current IoT trust models do not satisfy all the trust properties enumerated in Section II-A. We show that the same holds true for traditional trust models in Section II-C, and hence establish their unsuitability for use in IoT. As a result, we prove the necessity of a new trust model and provide a justification for the model presented in this paper.

A. Ideal IoT-Centric Trust Model

A system model and a holistic trust management framework are given in [7]. The overall objective is to ensure that trust models provide an acceptable level of and balanced approach to the security, functionality, and usability of the IoT applications wherein they are utilized. The appropriate design of a TMS is critical, therefore, and is the focus of this paper. The set of desirable properties for a suitable TMS for collaborative IoT applications must now be enumerated, based on work done in [7], [14], [17], and [18].

1) *Platform Consideration:* Devices on the IoT platform would typically have low computing power. Also, they would be distributed and operate in a decentralized network architecture. Therefore, the mechanisms used for the TMS must be lightweight, scalable, and essentially decentralized.
2) *TaaD:* Trust should be modeled as a decision-making process with the objective and subjective trust properties of the trustor taken into consideration. Each trustor should be able to decide the importance of a trust criterion or recommendation. Different trust values may be computed by different trustors for the same peer to reflect the trustors' different subjective trust dispositions.
3) *Trust Composition:* The TMS should incorporate the necessary objective [quality of service (QoS)] and subjective (social) properties of the trustee as trust parameters, so that the trustor can make a well-informed trust decision.
4) *Trust Persistence:* Trust is persistent and cumulative in nature. Hence, the TMS must provide a means of storing trust values effectively, taking into consideration the low storage capability of IoT devices.

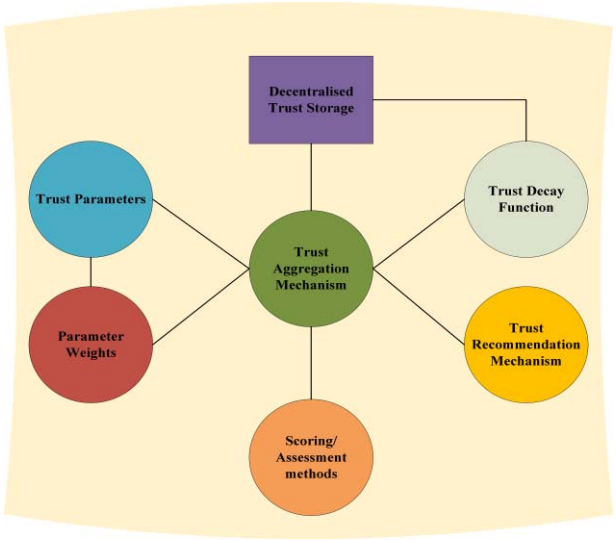

Fig. 1. Core components of a TMS and their interactions. This representation is based on the definition given in Section II-A.

5) *Trust Decay:* Trust is temporal in nature. Stored or previous trust values degrade gracefully over time. The TMS should include a trust decay function to guide this process.
6) *Risk Mitigation:* It should provide effective mitigation of self-promotion, good-mouthing, ballot-stuffing, opportunistic service, and on–off attacks, as described in [19]. The inclusion of a robust recommendation and belief function in the TMS would make it difficult for nodes to profit from malicious activities.
7) *Trust Accuracy:* This is a measure of how close the trust value computed for a node is to the ground trust. The ground trust for a node is the trust value that we would compute if we had perfect knowledge of its behavior. The TMS must have a high degree of trust accuracy.
8) *Trust Convergence:* This is a measure of how long it takes for the trust value computed for a node to reach its ground trust and maintain it, given a consistency in the behavior of the node. The TMS should ensure trust converges quickly.
9) *Trust Resilience:* This is a measure of the ability of the TMS to adapt to changes in the trust community, such as an increase in the ratio of malicious peers to good peers. The trust values computed by the TMS should remain accurate and quickly converge to new ground truth values in these situations.

These properties, along with their importance, will be discussed in Section III. The basic components of a TMS are illustrated in Fig. 1. Each component and its interaction with others will also be discussed in Section III.

B. Existing IoT-Centric Trust Models

The peer-to-peer (P2P) nature of collaborative IoT applications means that there is no central authority. Modeling and evaluating trust in such contexts is usually difficult [10]. Collaborating peers will often be strangers to one another,

having no shared history between them. Centralized TMSs should therefore not be considered in a collaborative IoT context. While a centralized approach to trust management is chosen in [13], the use of multiple trust management servers in different geographical locations is assumed. This means that all the nodes must be registered under one of these servers and that the servers themselves are owned by a single entity. It does not take into consideration of collaborative applications that may be performed without access to the Internet. Indeed, this paper focuses on service provisioning rather than collaborative situations, and only proposes a framework for different services without specifying solutions for individual contexts.

The concept of social IoT trust is utilized in [14], [15], and [19], where it is argued that existing social relationships between owners must be taken into account in trust management. This usually involves sharing some private information, such as user identities, locations, and other relationships. This opens the door to personal, malicious attacks from bad peers. While trust is a human concept, it also depends on the context in which it is used. In the IoT context, the trust is between the interacting nodes, which may sometimes by required to exchange some information for identification and trust computation. However, this should be done in a transparent and nonintrusive manner that maintains the privacy of nonrelevant information [7].

In [20], a trust management model for IoT based on fuzzy reputation is proposed. However, the model is specific to WSNs and only evaluates objective properties of packet forwarding/delivery ratios and energy consumptions [13], [19]. Thus, the model cannot be applied to collaborative IoT scenarios without some extensions. Furthermore, it neither properly models TaaD of the trustor nor considers the subjective properties of the trustee. The trust model in [21] is designed specifically for health IoT systems and cannot be applied to collaborative IoT contexts.

Boa and Chen proposed a dynamic TMS, and extended it to trust-based service composition in IoT [19] and [22]. This paper considers three parameters to derive a trust value: 1) honesty; 2) cooperativeness as a service provider; and 3) the community-interest of the nodes. The model includes a weighting factor to determine the relative importance of recommendations based on the trust level of node providing the recommendation. This factor can be dynamically increased to improve the resilience of the system with respect to the proportion of good peers and malicious peers. This however subjects the system to opportunistic service and on–off attacks [18], where a malicious node can provide good service but bad recommendations about other nodes. This is a consequence of evaluating a node's trust score entirely on the subjective opinions of some other nodes, as will be shown in the next section.

There is also the need to consider the temporal nature of trust. It is usually useful to store trust scores from past interactions and utilize them in making trust decisions in the future, thereby building a trust history. This idea is widely employed in trust models to aggregate trust values over time. However, as is the case with recommendations, this notion may be abused by malicious nodes. If trust is to be a reliable assessment of the performance of nodes on functional properties, then it is necessary to track the behavior of the nodes with respect to such properties and to detect and respond to changes over time. There is, therefore, the need for a trust decay function such that previous trust values degrade gracefully over time. This is also required to prevent on–off and opportunistic service attacks. In most existing trust models, however, trust decay is not considered.

Boa and Chen extended their previous work to service oriented architecture-based IoT systems and service management in social IoT in [14] and [15], respectively. The new model focuses on social trust based on the parameters of friendship, social contact, and community of interests. This model is not feasible for use in collaborative IoT contexts, as previously argued. Moreover, as it is based on [19] and [22], it inherits the limitations of subjective opinions discussed above. The nomadic, adhoc nature of IoT collaborations implies that collaborators may have no previous transactions with one another. In these cases, the use of a TMS solely based on reputation, such as EigenTrust [23] or PeerTrust [24], is not a good solution for several reasons, as will be discussed in the next section.

C. Traditional Reputation Models in IoT Contexts

While reputation is an integral part of trust, the two are not equivalent. The reputation of a person or device usually depends on the subjective views of others. In a largely decentralized architecture such as collaborative IoT, there is no standard way to determine whether the present reputation score of the device was not bought or given by a group of malicious peers. Since there may be no central database to keep track of reputation ratings, it is not always feasible to find out which peer contributed a ranking to the present overall score. A reputation-based system works in large P2P networks and e-commerce applications such as eBay because there is a centralized trust authority and database [10]. This makes it possible to track the consistency of the rankings of every peer in the system.

It can also be seen that reputation-based trust systems, such as [25], are entirely based on the trustors' subjective opinions which tend to be reinforced through an inherent feedback mechanism. This works well in large networks due to the "wisdom of the crowd" [26]. It is highly likely that the opinions of 1000 people about a seller on eBay will be a true reflection of the seller's activities. In a collaborative IoT scenario, however, the number of peers involved is small. The opinions of such a small number of rankers can be easily influenced and may not truly represent a trustee's trustworthiness. The feedback mechanism can cause multiple counting of the same behavior, leading to aggravated rewards or punishments. Therefore, reputation should not be used alone for trust computation in such contexts.

The use of entirely subjective opinions of others to determine trust scores presents yet another problem in a collaborative IoT scenario. Take a collaborative download application as an example. Each possible helper peer may advertise the price charged per bandwidth used. A peer may receive a low

ranking solely based on a higher price. This does not consider (and may not be a true reflection of) the helper peer's objective qualities in estimating its trustworthiness. The higher price may be a consequence of faster and better service offered. When this level of service is needed, the previous ranking will affect the trust score of the helper peer and may prevent another peer from patronizing its service. Traditional reputation systems mitigate this issue by providing some feedback on rankings. This is achieved by eliciting reviews or by providing categorical scores alongside an overall trust score. This provides additional insights to the trustor and leads to a better trust decision. However, such a level of detail may not be feasible in IoT environments due to the limited computing requirements.

In [27], a recommender system is enhanced by adding a trust layer. However, the method assumes prior friendship between nodes and therefore only considers social trust parameters. Lastly, because reputation-based TMSs use recommendations, they make the collaborative sphere more vulnerable to bad-mouthing and good-mouthing attacks [19]. This introduces an unnecessary bias into the trust model. It also corroborates the authors' argument against the assessment of trust solely on subjective opinions in a collaborative IoT context.

In summary, most existing trust models in IoT are primarily based on recommendations, with the inherent risks as highlighted above. In contrast, our proposed trust model emphasizes the parametrization of trust. The trust parameters are based on the functional properties of nodes which are relevant to the application context. Recommendations are only used initially to augment the trustor's assessment. In addition, the process of trust decay is clearly and adequately modeled, which is an improvement upon existing models. Our trust model is discussed in detail in the next section.

III. CTRUST Model Design and Analysis

We now propose CTRUST as a suitable trust model to evaluate and manage trust between nodes in collaborative applications in IoT. The trust a node has in another is based on its assessment of their current and past direct interactions and the recommendations it accepts from other nodes. Trust criteria form the basis on which assessments are made, and a trustor determines the weights of each criterion. A node can then compute trust scores which it uses to choose which other nodes to collaborate with. Trust scores are stored and are used to guide future interactions, although their importance declines over time. The model consists of trust assessment, decay recommendation and aggregation functions, all of which are discussed in detail in subsequent sections. A high-level workflow of the model is illustrated in Fig. 2. The following gives an overview of CTRUST.

1) Trust would be composed of one or more trust criteria (parameters) relevant to a collaborating context. Each trust parameter could be objective (QoS) or subjective (social) in terms of assessment. An assessment of a node on a parameter is called a partial trust score.
2) A trustor would be able to assign weights to each trust parameter. The weights indicate the relative importance of each parameter to the trustor. Therefore, partial trust scores are weighted before trust aggregation.
3) Trust would be propagated in a distributed manner, with no intervening central authority. Each node stores its previously computed trust values and may accept trust recommendations on partial trust scored from other nodes.
4) A recommendation function is implemented to guide the degree of acceptance of trust recommendations on partial trust scores. We call this degree of acceptance *belief change*, and it is modeled based on social characteristics.
5) Trust scores decay over time based on a mathematically modeled trust decay function. We also define the points at which previous trust is taken to have decayed completely and when current trust has reached maturity.
6) A trust aggregation function determines how partial trusts are aggregated to compute an overall trust score for a node. The aggregation function chosen depends on the collaboration context. In this paper, we use *a dynamic weighted sum* method. Trust updates (on partial trust scores) are *event-driven* and occur whenever nodes interact with one another.

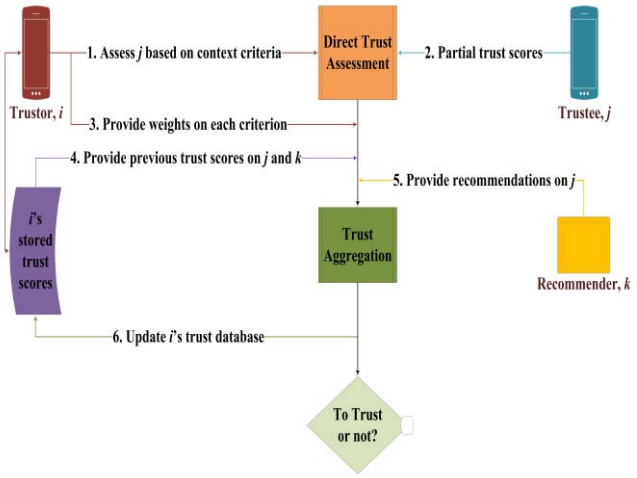

Fig. 2. Flow diagram illustrating the basic steps involved in trust computation in the CTRUST model.

We now proceed to define the trust model in detail. Let **C** be the set of all possible collaborating nodes under the current application or collaboration context. **T[C]**, the trust space over C, is then a sextuple expressed by the following notation:

$$T[C] \equiv \left[T_{ij}, P, W_i, V_{ij}, F, t_{\frac{1}{2}}(i)\right] \quad \forall i, j \in C \tag{1}$$

where T_{ij} is the trust score of node j as computed by node i; $P = \{p_i, p_2, p_3, \ldots, p_n\}$ is the set of trust parameters or properties on which each node in C is assessed by other nodes; $W_i = \{w_i(p_1), w_i(p_2), w_i(p_3), \ldots, w_i(p_n)\}$ is the set of weights on each parameter in P, as assigned by i; $V_{ij} = \{V_{ij}(p_1), V_{ij}(p_2), V_{ij}(p_3), \ldots, V_{ij}(p_n)\}$ is the set of values denoting node i's assessment (partial trust score) of j on each parameter $p \in P$; $F = f(W, V) \equiv T_{ij}$ is the trust aggregation function; and $t_{(1/2)}(i)$ is the half-life of any partial trust score computed by i.

A. Trust Parameters

Our design follows a multicriteria approach toward trust computation. A trustor makes a trust decision based on multiple criteria. Each criterion is a trust parameter. Parameters could either be objective or subjective. Parameters are considered objective if they are verifiably measured. Such properties include the speed of a transaction, reliability, rate of work, proximity, cost of a service, stake in the collaboration, etc. If p is an objective parameter, then for a node $j \in C$, $(V_{ij}(p))_t$ is approximately the same if measured by any $i \in C$ at instant t. Subjective parameters such as honesty, cooperativeness or friendliness are assessed as perceived by the trustor. Therefore, $(V_{ij}(p))_t$ does not have the same value from all $i \in C$, even if they all assessed j at the same instant, t. Therefore, the trust model supports both QoS and social trust parameters, which means a greater robustness and latitude of applications.

We do not explicitly specify the trust parameters in our model. This is because the choice of which parameters to use depends on the collaboration context and should be decided when C is set up. Suppose a service composition with n collaboration contexts, $C_1, \ldots, C_n$ is set up, one approach will be to populate the set P with all possible parameters for all the contexts wherein the trust model will be implemented. Then the weights of parameters which are not relevant to the current context can be set to 0.

B. Parameter Weights

The weights determine how important a parameter is relative to the overall trust score. Each node determines the weight of each parameter, based on their current subjective opinions. As a result, two nodes may have an equivalent value set, V, at a given instant, yet compute different trust scores, T, for a third node. The weights are assigned such that

$$w_i(p) \in [0, 1] \quad \forall i \in C, p \in P \text{ and } \sum W_i = 1. \tag{2}$$

Even though set P is the same for every node in C, a node can eliminate parameters that it does not want to consider by assigning them a weight of 0. The weights can be dynamically adjusted by the trustor at any time during a session of interactions. Nodes can update their record of set W at any time, according to changes in their perceptions of relative importance of each parameter. The dynamic weighting allows for more accurate modeling of human trust. The relative importance of the factors that determine the extent to which a person trusts another can vary greatly over time. Similarly, the relative importance of collaboration criteria can vary for each node from one session to another.

C. Partial Trust Scores and Aggregation

The set V_{ij} represents the normalized assessment of node j, by node i, on each of the trust parameters in P. The collaboration context defines the parameters, i.e., how they are measured and on what scale. Objective parameters such as rate of work or network speed are well defined and will be measured uniformly across C, while the measurement scale for subjective ones such as friendliness may differ from node to node. Each member of V_{ij} is a partial trust score as they determine the overall trust score, T_{ij}. The values are normalized to [0, 1] so that T_{ij} is also within the same range, and the normalization method must be defined in C.

Three factors account for any $V_{ij}(p)$ at the current time: its previous value based on past interactions; the current, direct assessment of j by i; and indirect assessment of j by some other node k. These will be discussed in detail later.

The trust aggregation function, F, specifies how the partial trust scores are aggregated to compute T_{ij}. In this paper, we use a weighted sum function, $F = W \times V$. Therefore,

$$T_{ij} = \sum_{x=1}^{n} w_i(p_x) \times V_{ij}(p_x) \quad \forall i, j \in C, p_x \in P. \tag{3}$$

It can be observed that the derivation of this function incorporates objective and subjective properties of both the trustor and trustee. Furthermore, our trust model allows for the aggregation function to be left unspecified until the collaboration context is set up, which is expressed as $F = f(W, V)$. This is important because the context should determine the method by which partial trust scores are aggregated. Some contexts may require a product of weighted scores, or a more complex function such as Bayesian inference or regression analysis, to compute a trust value [28], [29]. Therefore, it is best to leave the aggregation function unspecified until the context is set up, as this makes the model robust and applicable to more contexts.

D. Trust Decay

It is necessary to model the impact, over time, of previous trust scores on the current trust values; this is achieved by introducing the concept of trust decay. The trust score, T_{ij}, gradually degrades over time when there is no interaction between i and j. In the social world, the longer we are further away from a person, the easier it is to distrust that person. This is so because we are not sure whether they still retain the values for which we admired them. Interactions help to re-evaluate our opinions of them on these values, and serve to reinforce the trust relationship, or score. Accurately modeling trust decay is very difficult, and there is limited existing literature on the subject. The following assumptions seem to hold true in the usual social context, and form the basis for the trust decay function of our model.

1) As the trust formation depends on partial trust values on each trust component, trust decay applies to these values and not the overall trust score, which may not correlate with the trust decay rate. The reason for this is that in the interval between interactions, the trustor's perception of the relative importance of some of the trust parameters may have changed.
2) The rate of trust decay is almost entirely subjective. It depends on the trustor's willingness to trust and the length of time or number of interactions the trustor requires to establish a node's behavior in the present.
3) It is reasonable to assume that trust decays at an exponential rate in the absence of interactions [30]. The longer the period of inactivity between the peers, the greater the rate of decay.

4) When a new session of interactions is made in the present time, previous trust decays with every new interaction. This is so because the new interactions tend to form the trustor's new opinions and therefore, trust score of the trustee. After a certain number of new interactions, past trust values may no longer be relevant to trust computation in the present.

The above assumptions provide the basis by which trust decay is incorporated into our model. Let $t_{(1/2)}(i)$ be the duration required for a partial trust score assigned by i to decay to half of its initial value, i.e., its half-life. The decay function follows an exponential trend and is represented by the following mathematical equations:

$$\begin{aligned}\left(V_{ij}(p)\right)_{0\to t} &= \left(V_{ij}(p)\right)_0 \times \frac{1}{2}^{\frac{t}{t_{\frac{1}{2}}(i)}} \\ &\equiv \left(V_{ij}(p)\right)_0 \times e^{-\lambda_i t} \qquad (4)\\ &\equiv (\phi_i)_t \times \left(V_{ij}(p)\right)_0 \quad \forall i,j \in C, p \in P\end{aligned}$$

$$\lambda_i = \frac{\ln 2}{t_{\frac{1}{2}}(i)} \approx \frac{0.693}{t_{\frac{1}{2}}(i)} \qquad (5)$$

$$(\phi_i)_t = e^{-\lambda_i t} \qquad (6)$$

where $(V_{ij}(p))_0$ is a partial trust score at the end of the last session of interactions between i and j; $(V_{ij}(p))_{0\to t}$ is the current value of $(V_{ij}(p))_0$ after time t of no interactions between i and j; λ_i is the decay constant for partial trust scores from i; and $(\phi_i)_t$ is the trust decay multiplier for node i.

The multiplier is the proportion of the partial trust score that has not decayed after time, t, of no interaction. Equation (4) can then be simplified and rewritten as

$$\left(V_{ij}(p)\right)_{0\to t} = (\phi_i)_t \times \left(V_{ij}(p)\right)_0 \quad \forall i,j \in C, p \in P.$$

After an adequate number of interactions in a new session, the effective proportion of $(V_{ij}(p))_0$ that determines $V_{ij}(p)$ in the current session becomes 0, according to Assumption 4) above. At this point, the trust has attained maturity in that session. Trust maturity is discussed further in Section III-F.

E. Trust Recommendations and the Belief Function

There may be times when node i is about to start a new session of interactions with node j, and it is currently in a session with another node k, which has some assessment on j. Node i may make use of this assessment to make an initial update of j's partial trust scores prior to initiating interactions. This is called an indirect assessment, or a recommendation, on j by i, through k. The drawbacks of trust systems solely based on reputation or recommendations (see Section II-C), must be avoided. A recommendation belief function is therefore required to determine the degree to which any node i accepts k's recommendations on j. The following premises should be taken into consideration in order to model the belief function accurately.

1) The purpose of recommendations is usually to guide i as it attempts to initiate or reinitiate interactions with j. They either make the trustor initially more sceptical or more open to trusting j. After interactions are made, k's indirect recommendations are quickly discarded, as i's direct assessment forms the basis of the trust score.
2) Recommendations do not necessarily affect i's trust relations with k, even if they are proven to be incorrect. K might have been misinformed. It follows then, that the best way to prevent good and bad-mouthing attacks is to minimize the effect of indirect recommendations on a partial trust score, and hence, T_{ij}.
3) Let the change between k's current recommendation and i's previous partial trust score of j on some parameter be ΔV. The smaller the absolute proportion of change, $|(\Delta V/(V_{ij}(p))_0)|$, the easier it is for i to accept. This stems from the observation that in a social context, we are less likely to receive a recommendation about a person if the recommendation represents a significant difference from previously observed behavior.
4) The longer the time t that has passed since the last session of interactions between i and j, the more open i will be in accepting k's recommendation. This is because of trust decay; the longer the time t, the smaller the proportion, ϕ_i, of previous trust left. The value of trust is a function of time; the extent to which we would believe a value that implies a significant change in a person's behavior depends on the time that has elapsed since our last interaction with them.
5) The greater the value of T_{ik}, or more specifically $V_{ik}(p)$, the more likely we are to receive the recommendation of k on j on some trust parameter, p. In a social context, we more readily believe recommendations on a subject from someone who we rate high on the same subject.

The belief function can be then derived mathematically from premises (3)–(5) above

$$\begin{aligned}\text{Belief, } \beta_{ij\leftarrow k} &\propto \left(1 - \left|\frac{V_{kj}(p) - \left(V_{ij}(p)\right)_0}{\left(V_{ij}(p)\right)_0}\right|\right)\\ \beta_{ij\leftarrow k} &\propto (1-\phi_i)\\ \beta_{ij\leftarrow k} &\propto V_{ik}(p) \qquad (7)\\ \Rightarrow \beta_{ij\leftarrow k} &= K\left(1 - \left|\frac{V_{kj}(p) - \left(V_{ij}(p)\right)_0}{\left(V_{ij}(p)\right)_0}\right|\right)\\ &\quad \times (1-\phi_i) \times V_{ik}(p) \qquad (8)\end{aligned}$$

where K is a constant. The value of K does not need to be verified. Since the change belief is a relative indicator of how much a recommendation is to be accepted, the exact value of K need not be known. Therefore, we set $K = 1$ so that at the beginning of a new session of interactions between i and j

$$\beta_{ij\leftarrow k} = \left(1 - \left|\frac{V_{kj}(p) - \left(V_{ij}(p)\right)_0}{\left(V_{ij}(p)\right)_0}\right|\right) \times (1-\phi_i) \times V_{ik}(p). \qquad (9)$$

This belief function indicates how much node i is willing to accept a recommendation on j from k, i.e., the weight i assigns to that recommendation. It therefore determines i's indirect assessment of j through k, on parameter p, $\Psi_{ij\leftarrow k}(p)$, which is given by

$$\Psi_{ij\leftarrow k}(p) = \beta_{ij\leftarrow k} \times V_{kj}(p), \quad j \neq k. \qquad (10)$$

A node cannot provide recommendations on itself. This comprehensively defends against self-promotion attacks. Once new interactions begin between i and j, then according to premise (1) above, $\beta_{ij\leftarrow k} = 0$. This renders any good-mouthing or ballot-stuffing attacks by k useless. Together with the trust decay function, it also provides an effective defense against opportunistic or on–off attacks by k. This is so because i does not accept recommendations on nodes it is currently interacting with. Also, a node performing random attacks simply degrades the partial trust score it receives from i. Thus, a malicious node stands very little chance to gain by providing a bad recommendation or service to i, and its ability to impact $V_{ij}(p)$ is severely limited.

F. Trust Update and Maturity

Let $D_{ij}(p)$ be i's direct assessment of j on trust parameter p in the current session of interactions. The method of assessing $D_{ij}(p)$ depends on the parameter. It could be a rate of work done, which can be computed simply based on observation. It could also be a co-location score, for which a formula needs to be applied. Generally, the method for evaluating direct assessments must be specified for each parameter when designing the collaboration context.

To update trust reliably, the concept of trust maturity must now be introduced. In addition to direct assessments, previous trust scores, and recommendations impact the current value of any partial trust score, and we have described trust decay and recommendation belief functions for these. There should be a point in time at which direct assessments are sufficient to compute trust scores. Let Γ be the number of interactions required to reliably measure $V_{ij}(p)$ based on $D_{ij}(p)$ only. After Γ interactions between two nodes in any session, the trust score computed by one on the other attains maturity or equilibrium. In other words, trust maturity is a state that is attained in a collaborative session when direct assessments of the interactions between any two nodes are sufficient for either node to accurately assess the other's trust scores. At this point, past trust between the nodes is assumed to have decayed completely and recommendations are not considered in computing the trust scores of either node.

The value of Γ depends on the collaboration context, C, and must be determined by initial experiments. Once this value has been determined, it can be used in future sessions to weight previous trust scores. For example, after Z interactions between i and j in a new session, the effective proportion of a previous trust score, $(V_{ij}(p))_0$, is given by

$$\mu_i = \max\left(\left(1 - \frac{Z}{\Gamma}\right) \times (\phi_i)_t, 0\right). \tag{11}$$

In other words, once $Z \geq \Gamma, \mu_i = 0$ in accordance with Assumption 4) in Section III-D. At the start of a new session of interactions between i and j, the initial value of $D_{ij}(p) = 0.5$. This is the midpoint between complete distrust (0) and perfect trust (1). It is taken as the neutral trust value in the absence of any information. It is also the default assessment value that is used for computing trust scores for a node with which i has had no previous interactions.

TABLE I
LIST OF MODEL PROPERTIES

Symbol	Description	Type
T_{ij}	Trust value of j as computed by i, at the current instance and context	Derived
p	A trust metric or parameter by which trust is assessed in the current context	Design
$w_i(p)$	The importance of p as determined by i	Input
$D_{ij}(p)$	The direct assessment score of j, as measured or perceived by i, on parameter p	Derived
$\left(V_{ij}(p)\right)_t$	The trust score of j, as determined by i, on parameter p, at time t	Derived
$\left(V_{ij}(p)\right)_0$	The trust score of j, as determined by i, on parameter p, at the end of the last session	Derived
$(\phi_i)_t$	weight of $\left(V_{ij}(p)\right)_0$ in next session of interactions after time t of no interactions between i and j	Input
$\left(V_{ij}(p)\right)_{0.}$	The real value of $\left(V_{ij}(p)\right)_0$ that determines $\left(V_{ij}(p)\right)_t$ at time t, in the current session	Derived
μ_i	Best defined as $\left(V_{ij}(p)\right)_{0\rightarrow t} / \left(V_{ij}(p)\right)_0$	Derived
$\beta_{ij\leftarrow k}$	The proportion of a recommendation on j, from k, that i is willing to accept	Derived
$\Psi_{ij\leftarrow k}(p)$	The indirect assessment score of j on parameter p, as received by i from k, based on $\beta_{ij\leftarrow k}$	Derived
Γ	Number of interactions in a session required to reliably measure trust by direct assessment only	Design
N[C]	Number of all nodes in the collaboration context and community, C	Input
N[G]	Number of nodes in C that are actively in collaboration with i at the current instance	Input

We have now discussed all the three factors needed to compute $V_{ij}(p)$ at current time, $(V_{ij}(p))_t$. Let $G \subseteq C$ be the set of nodes currently in a collaboration session with node i. Suppose there are s number of nodes in G, $G(1), \ldots, G(s)$, that have a recommendation on j, then

$$\left(V_{ij}(p)\right)_t = \begin{cases} \frac{D_{ij}(p) + (\phi_i \times (V_{ij}(p))_0) + \sum_{x=1}^{s} \Psi_{ij\leftarrow G(x)}(p)}{1 + \phi_i + \sum_{x=1}^{s} \beta_{ij\leftarrow G(x)}(p)}, & Z = 0 \\ \frac{D_{ij}(p) + (\mu_i \times (V_{ij}(p))_0)}{1 + \mu_i}, & Z > 0 \end{cases}$$
$$\forall p \in P, G(x) \in G, j \notin G. \tag{12}$$

We have defined all the properties required to setup the CTRUST model. A brief description of each property is given in Table I for easy reference. Fig. 2, as noted earlier, is a basic illustration of the trust computation process in CTRUST which has been discussed in the preceding sections.

We have seen that CTRUST supports multiple QoS and social parameters. While CTRUST allows the aggregation function to be specified according to the context, we used a dynamic weighted sum in this paper. Also, trust recommendations in CTRUST are propagated in a distributed and decentralized manner. In this section, we have shown that the trust update mode is event-based; i.e., trust scores are updated after every interaction. In [18] and [28], trust computation models are classified according to their trust composition, propagation, aggregation, update, and formation. Using that classification scheme and notation, CTRUST is a QoS + Social/Distributed/Dynamic weighted sum/Event/Multitrust with dynamic weighted sum. We evaluate the performance of the model in the next section.

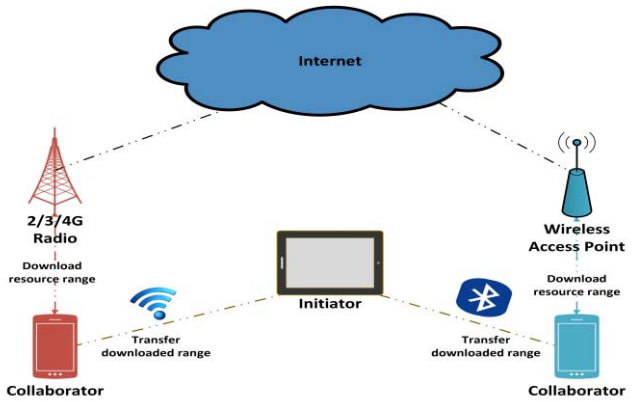

Fig. 3. Illustration of a collaborative download session. Collaborators download byte ranges of the requested resource and transfer back to the initiator using a WLAN connection, e.g., Wi-Fi or Bluetooth.

IV. Model Performance and Evaluation

For evaluation, the trust model was implemented in a collaborative download application. The aim here is to measure the performance of the model in a real-word collaborative context, in terms of the trust properties of trust accuracy, convergence, and resilience. In turn, this will prove the effectiveness of the trust composition, persistence, decay, and risk mitigation methods applied in the model, as discussed in the previous section. The trust properties of platform consideration (IoT) and TaaD are implicit in the model's design. Thus, we prove that CTRUST satisfies all the trust properties of a suitable TMS for IoT as enumerated in Section II-A.

In the following Section, the experimental collaboration setup is introduced and explained. In Section IV-B, the trust parameters are defined, and initialization values are provided for other model parameters as required. The actual evaluation is discussed in Sections IV-C–IV-E.

A. Context Overview: Collaborative Downloading

The concept of collaborative downloading (CD) has been addressed in previous works [31]–[33]. CD is a P2P paradigm where the bandwidth of multiple devices is pooled to download a resource. Usually, a peer requiring a resource requests that the other collaborating peers assist to download the resource using their Internet connection and bandwidth, as illustrated in Fig. 3. This is especially useful where the peers or nodes are nomadic, and individual mobile bandwidth is small relative to the resource to be downloaded. One scenario is where the content to be downloaded is commonly requested by the collaborating peers. Rather than downloading the resource individually, they can collaborate so that each peer downloads partitioned data ranges of the resource. These partitions can then be aggregated and delivered to each peer.

Another scenario is in places where there is limited or no Wi-Fi, ADSL, or other broadband, and the only available Internet connectivity is the more expensive and/or slower mobile 2/3/4 G network. Peers may collaborate to save money and time in such cases. Access to a resource server or WSN sink is also optimized using this technique. Rather than making multiple connections to the same server (or sink in WSN) for the same data ranges, each peer downloads a different data range at a time, thus optimizing the server or sink uplink.

In a CD system, nodes available to help with a download send out broadcasts of their availability. These broadcasts are seen by all other nodes in the same geographic vicinity. These nodes form the set C. A node wishing to initiate a download (we call this node the initiator) will pick collaborators from this set of nodes using some selection algorithm. The selected nodes form the set G. The initiator sends out the URL of the resource (workload) to be downloaded to these nodes. The workload is divided into blocks. Each block is a range of bytes of the workload. The blocks are distributed among each node in G using some work schedule algorithm. The nodes transfer the completed blocks by uploading the byte range downloaded as a file object to initiating device over the wireless communication channel. Therefore, CD application could be thought of as a distributed download manager. When all the blocks have been downloaded, the file objects are combined to retrieve the original resource.

B. Collaboration Context Setup

We identified three criteria to judge the suitability of a node as a collaborator in the CD context described above: its download speed, reliability in successfully completing workloads, and the level of security risk it poses to the collaboration. Based on these criteria, we now define the following three trust parameters to evaluate the CTRUST model in this collaboration context.

1) *Successful Completion Rate (SCR):* This is a measure of the reliability of a collaborating node. It is based on the number of times, in the current session, that a node has successfully both downloaded and transferred a work queue block back to the initiating node. In a new session of collaboration, the direct assessment D_{ij} (SCR) begins at 0.5. The initiating node keeps a cumulative count of the total number of both assigned and successful blocks in the current session.
2) *Cumulative Bandwidth Average (CBA):* This is a measure of the work speed of the collaborating node. It is determined by the average bandwidth of a node, as measured by the initiator. It is cumulative over the current session of interactions. The initiating node keeps a running total of the total number of bytes and time taken, for each collaborating node in the current session. The CBA is then normalized. At the beginning of a new session of interactions the default value of $D_{ij}(\mathrm{CBA}_{\mathrm{norm}}) = 0.5$ is used.
3) *Inverse Risk Index (IRI):* A malicious collaborating node may modify blocks before sending them to the initiator. It may even just send a block of the expected size, with all bytes set to 0 or 1. It may also try to disrupt the CD session, by ignoring the work queue order, for example. The introduction of a parameter to assess the nodes' malicious intent is required to keep would-be malicious nodes in check. Whenever some tampering or fraud is discovered, either in a block marked as complete or in the work queue, the initiator marks that block (or

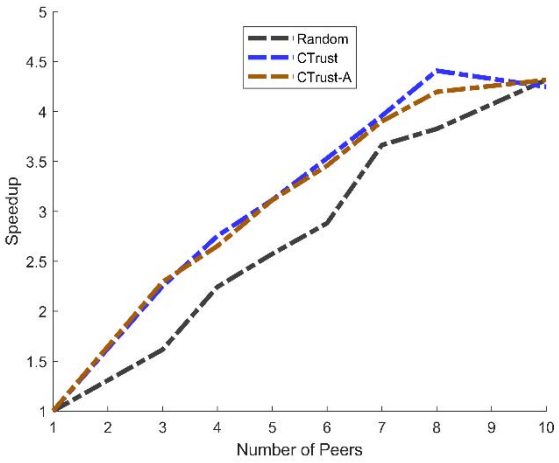

Fig. 4. Plot of speed-up against different group sizes, $N[G]$. The figure shows the download speed-up achieved using either modes of CTRUST for node selection compared to a random selection of nodes.

whatever block the malicious node is currently downloading) as a bad block and updates D_{ij} (SCR) to reflect the risk that the node poses. IRI is cumulative over a session. As usual, the initial value of D_{ij} (IRI) at the beginning of a new collaboration session is 0.5.

These parameters were considered by the researchers to be the most important functional parameters in a collaborative download context. For any context, a decision must be made to determine which functional requirements may serve as trust parameters. This is an initial step in setting up the collaboration context without which the trust model cannot be implemented.

We now proceed to compute $(V_{ij}(p))_t$ as follows. If there has been no previous interaction between i and j, and there are no recommendations on j from any of the other collaborating nodes, then $(V_{ij}(p))_t = (D_{ij}(p))_t$. At the end of a session of interactions (which is one download session), the final computed value for $(V_{ij}(p))_t$ becomes $(V_{ij}(p))_0$ for the next collaboration session. Indirect assessments are handled as described in Section III-E. Once the initiator has set the weights for each parameter, T_{ij} can be computed as described in Section III-C. The results of the simulation are discussed below.

We set up the collaboration community with size, $N[C] = 10$ per session. Also, the maximum number of nodes the initiator i collaborates with at any time, $N[G] = 5$, except where it is necessary to increase the group size to illustrate a point. An example of such case is the speedup illustration in Fig. 4. In this experiment, we assumed that the subjective utility function of the initiator is linear and that all parameters are of equal importance. Therefore, we used an equal weight for all the parameters, i.e., $w_1 = w_2 = w_3 = (1/3)$. In each simulation, there were 60 download sessions of interactions. The default trust value is 0.5. In simulating the behavior of nodes with respect to the parameters, nodes are randomly set up such that they tend to complete anywhere between 50% and 100% of the blocks assigned to them. The same goes for block tampering or risk. A random average bandwidth between 0.5 B and 1 B is assigned to each node. The performance of the model is discussed in the following sections.

TABLE II
TWO-SAMPLE T-TEST COMPARING SESSION SPEEDS OBTAINED USING CTRUST AND RANDOM MODES FOR NODE SELECTION

	Random vs. CTRUST	Random vs. CTRUST-A	CTRUST vs. CTRUST-A
Observations	60	60	60
P value at (α=0.05)	1.27E-17	3.77E-23	0.0516 (lowest value obtained)

C. Utility of the Model in Collaboration Context

The aim of the collaboration is to increase the download speed of a resource. Fig. 4 illustrates the speedup achieved. To understand the effect of the computing and time overhead expended on node selection, we run two different modes of CTRUST. In the first, the initiator utilizes the nodes selected at the beginning of a session until the end. In the second, after a trust update of a node, the initiator decides whether to continue with that node, or to check $[C]$ for another known node with a higher trust score. Trust update is triggered by the event of change to the status of a block; that is, whenever a block is returned to the initiator. This second mode is an adaptive mode, which we call CTRUST-A. For comparison, we run another simulation with the same group of nodes but selected randomly. In this experiment $N[C] = 10$, that is, the maximum number of nodes available for collaboration.

We observe that even when the utilization level is up to 70% of the nodes in such a small community (i.e., $N[G] = 7$), there is still a significant improvement in the speedup achieved when nodes are selected based on the trust model as opposed to randomly. Beyond this point, i.e., if $N[C]/N[G] > 0.7$, then the initiator has limited ability to discriminate based on its preferences, since it can reject only 30% or less of the available nodes regardless of their trust scores. As a result, the impact of the trust model on speedup rapidly decreases. When $N[G] = N[C]$, no selection takes place since all the nodes in the collaboration community are being utilized for downloading. Thus, there is no difference in speedup when $N[G] = 10$, as can be seen in Fig. 4.

The speedup achieved is comparable to the results that were obtained in [31], with the added advantage of trust. We observe that even with the extra computation involved, CTRUST and CTRUST-A outperform the random selection in speedup. A two-tailed test also shows a significant difference in the overall average speed obtained per session between random selection and CTRUST, as shown in Table II. Hence, we conclude that incorporating the trust model into the CD protocol does not negatively impact on the performance of the protocol. The difference in speedup over the course of a session between the two modes of C-Trust is statistically insignificant. However, CTRUST-A computes fresh trust scores after each interaction. Therefore, this mode is more sensitive and adaptive to changes in node behavior within a session. This adaptability, known as trust resilience, is

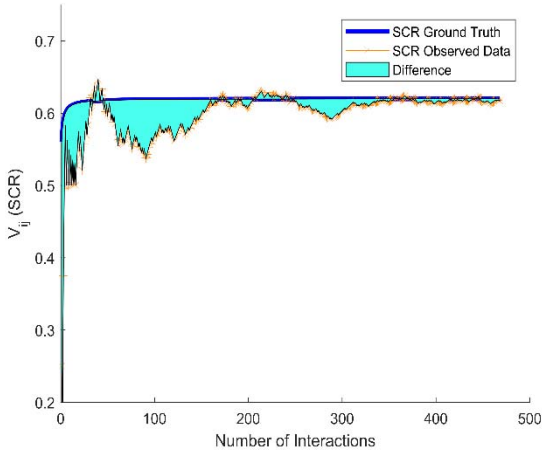

Fig. 5. Convergence of SCR to ground truth.

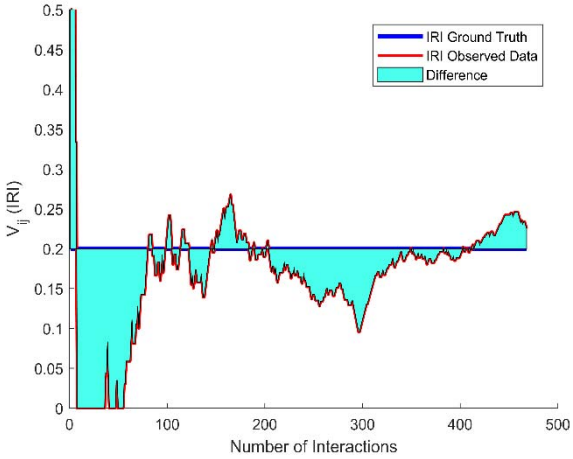

Fig. 6. Convergence of IRI to ground truth.

a desired property and is evaluated below. For this reason, CTRUST-A is the default mode used in our experiments.

D. Evaluating Trust Model Accuracy and Convergence

We now compare the trust scores obtained by the model to the ground truth status, and how long it takes to converge to ground truth status. The ground truth status is obtained by computing what the trust score should be based on the randomly assigned nodal characteristics. It is the truth value that would be assigned to the node if the trustor had perfect knowledge of its behavior. This comparison is important because it shows the effectiveness of the model in accurately estimating trustworthiness of nodes in a reasonable time. The results obtained are presented in Figs. 5–7.

The results show that the trust value of the node being assessed converges to the ground truth status after about 250–300 interactions. This number, though seemingly high, is to be expected. This collaboration context requires a short timeframe for each interaction. Therefore, a relatively high number of interactions would be required to accurately compute trust values. Also, the fluctuations that can be observed are to be expected because the behavior of a node may be *perceived*

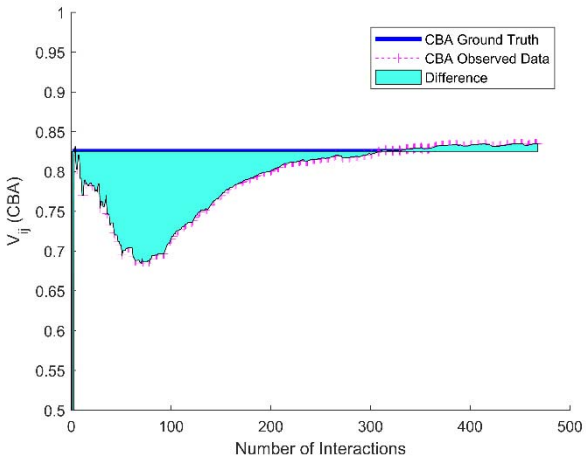

Fig. 7. Convergence of CBA to ground truth.

differently due to environmental or external factors. However, it should be noted that if the node characteristic remains the same, the trust value will converge back to the ground truth.

The large dip in Fig. 7 after initial convergence is due to the sensitivity of the parameter (IRI) to slight changes in assessment. This is a property of this collaboration context. Unlike in [19], where the trust value is tracked over 100 h, the results here show the trust value over the interactions in one session. This is more logical in our opinion, as trust scores are a function of interactions over time. For comparison, however, the usual duration of one session is about 2 h.

E. Evaluating Trust Model Resilience

Resilience is the ability of the model to adapt to changes in the collaboration community and maintain a high efficiency under such circumstances. The three major factors affecting this are bad recommendations from other nodes, change in node behavior, and increase in the proportion of malicious nodes in the collaboration community. By design, bad recommendations have very little effect on CTRUST, as explained in Sections III–V. Recommendations are only used at the beginning of a new session of interactions with a node. A bad recommendation will show only as a minor initial fluctuation on the graph. The adaptive mode of CTRUST ensures this. Therefore, this is not discussed further.

CTRUST adapts to change in a node's behavior both within a session and between sessions. The former has been illustrated in the previous section. The results show that any nodes can reliably assess one another without the necessity of recommendations after about 250 direct interactions between them. Thus, trust maturity is reached after 250 interactions, i.e., $\Gamma = 250$. Figs. 8–10 illustrate the resilience of CTRUST to changes in node behavior across two sessions, using the same node as in the previous simulations. The second session begins with the 470th interaction, at which point $\phi_i = 0.8$. For comparison, there are two plots for each parameter: one with the trust decay function described in Section III-D and the other without it.

The results show that without the trust decay function, it takes much longer to start to converge and it may never reach

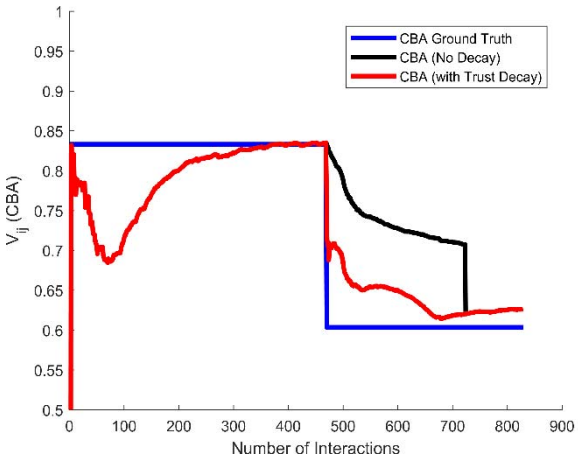

Fig. 8. Resilience of CTRUST to change in CBA.

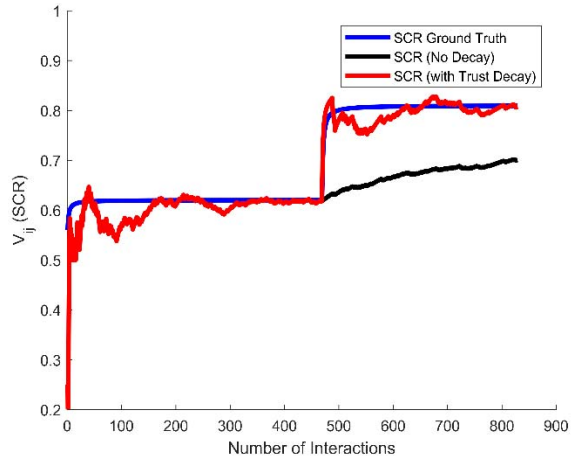

Fig. 9. Resilience of CTRUST to change in SCR.

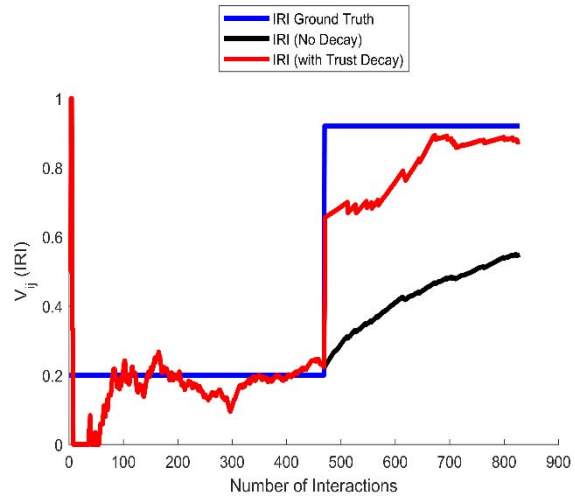

Fig. 10. Resilience of CTRUST to change in IRI.

the new ground truth status. With the trust decay function however, the new ground truth is reached after about 250 interactions, thus, keeping the trust maturity index, Γ, constant in the collaboration context. This proves the effectiveness of the trust decay function.

In Fig. 8, an early dip in the curve is noticeable. This is due to the nature of the parameter being assessed and the behavior of the node being assessed. CBA is a cumulative measure of reliability. Therefore, if a node achieves little in initially completing workloads, then it requires a steeper curve to reach ground truth status afterwards. The ground truth value for CBA, in this case, was 0.82. The assessed value dropped to 0.68 due to some failed blocks in the simulation. The dynamic update method in CTRUST-A implies that trust scores are updated after every interaction. This accounts for the variations that can be seen in Figs. 8–10. In the nonadaptive mode of CTRUST, the variations would balance out over the course of a session of interactions. The dynamic mode is preferred, however, because changes in the behavior of collaborating nodes are continuously tracked and this makes it easier to spot malicious behavior or a sudden drop in service level.

The initiator can define a minimum expected level of speedup to address the problem of suspected increase in the proportion of malicious nodes in the community, especially when the derived utility (speedup) suddenly drops significantly. The initiator can also alter the relative weights of the parameters to achieve its desired service level. For instance, if there are a lot of failed blocks, the weight of SCR may be increased by 20%. The same rule applies to every other parameter. If after two iterations the service level is not met, then the initiator may never achieve its minimum utility. The initiator should terminate the collaboration at this point because its objective cannot be achieved.

V. Related Work

A survey of trust models in the existing literature has already been reported in Section II. In this section, we present a comparative analysis of this paper to the previous work done in trust management for IoT applications. We do this to show the importance and distinction of this paper, and the contributions it makes to this field of study.

A detailed trust model for social IoT systems is presented in [15]. However, the model lacks a distinct trust decay function. Instead, two parameters are introduced; one weights past experiences versus direct assessments and the other weights recommendations versus past experiences. This introduces several problems in the trust computation problem. First, every direct assessment that is followed by a recommendation reduces the importance of past trust because it is weighted twice in both interactions. This does not allow for graceful degradation of trust. Moreover, if the trustor receives several consecutive recommendations on the same node, the impact of the past trust score and the trustor's direct assessment on that node rapidly declines with each recommendation. This is the case even if the trustor's direct assessment were made about the same time as the recommendation. Thus, malicious nodes can come together to influence the trust rating of one node with another node. Also, the model does not consider the time value of trust in that the recommendation made, even if genuine, could be based on an interaction farther time than the trustor's last direct assessment of the node on which a recommendation is being received. In our model, we have

separate recommendation and trust decay functions, each of which takes the temporal nature of trust into consideration. The recommendation function aggregates multiple recommendations received around the same time on the same node into a group recommendation score. Finally, we combine weighted values of direct assessments, past trust scores, and group recommendations to compute present trust scores. This ensures that trust scores are weighted once, and it mitigates the risks posed by a ring of bad recommenders.

In Section II, we discussed the risks inherent in using trust models that are solely based on reputation for IoT, such as [19], [23], and [24]. Our model utilizes both objective and subjective trust parameters in computing trust values for nodes. The trustor also decides the relative importance of each parameter. Hence, rather than having to use two different trust models in an IoT application context where both QoS and social trust must be considered, our trust model ensures we can compute one aggregate trust value in such contexts. Our model also dynamically adapts to changes in the trust community similar to [14] and [15]. Both models correctly identify that bad recommendations make it difficult to reach a new ground truth status quickly, as was also discussed in this paper. In those models however, two different model parameters must be tuned to achieve a high degree of trust resilience. The method with which we manage recommendations in our model ensures that we achieve a similar level of trust resilience without having to continuously tune system parameters during interactions.

In most of the trust models we have cited, the reputation of a node providing a recommendation is the only factor that is taken into consideration in deciding the weight of that recommendation. In CTRUST, the recommendation function also considers how recently direct interactions were made between the trustor and the node on which a recommendation is being provided, and the difference between the past trust value and the recommendation value for that node. This makes recommendations more robust and effectively deters opportunistic service attacks. We also determined the point at which the trust between two nodes reaches maturity. When trust maturity is reached within a session, a node can reliably compute trust values based on direct assessments alone. This reduces both the processing and storage overhead involved in trust computation, which is vital in the IoT platform where low computing power is a major characteristic.

It should be noted that while the importance of trust-based security in IoT has been recognized, trust management for IoT is still evolving. We believe that the work presented in this paper is a major contribution to this field, fills important research gaps, and will provide a better well-rounded approach to trust modeling in IoT.

VI. Conclusion

In this paper, we enumerated the suitable properties for a TMS for collaborative IoT applications. We then designed CTRUST in accordance with these properties. We have modeled trust as a result-oriented metric evaluated on parameters that are mapped to the functional requirements in the applied context. The model was evaluated in a collaborative download context. The analysis shows that the model is effective, and its trust estimation and performance show a high degree of accuracy, reliability, and resilience. The model is adaptable to several collaborative contexts. CTRUST effectively addresses self-promotion, good-mouthing, ballot-stuffing, opportunistic service, and on–off attacks. It requires little computing and energy resources for trust computation.

CTRUST is flexible since several design parameters can be set up based on the applied context. We implemented a robust trust decay function and mathematically modeled the acceptance of recommendations based on insights from social interactions. We were also able to determine the number of direct interactions required to achieve trust maturity between any two nodes. From the literature available to us, we conclude these are novel and important contributions to the study of trust management in IoT.

In normalizing values on each parameter, we have used a linear value function. However, sometimes the utility function of the initiator is marginally not linear. Collaborative applications generally address service provision and service composition. To model trust more accurately in these applications, we shall consider defining a utility function and threshold scales for each parameter in future work. We will consider how to set parameter weights dynamically and automatically, in response to changes in the collaboration community. We will also evaluate our trust model within other collaborative contexts in IoT.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[3] H. Chaouchi, "Introduction to the Internet of Things," in *The Internet of Things*. Hoboken, NJ, USA: Wiley, 2013, pp. 1–33.

[4] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1497–1516, Sep. 2012.

[5] S. Sicari, A. Rizzardi, L. A. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.

[6] J. A. Stankovic, "Research directions for the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, Feb. 2014.

[7] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.

[8] C. D. Jensen, "The role of trust in computer security," in *Proc. 10th Annu. Int. Conf. Privacy Security Trust*, Paris, France, 2012, p. 236.

[9] W. Leister and T. Schulz, "Ideas for a trust indicator in the Internet of Things," in *Proc. 1st Int. Conf. Smart Syst. Devices Technol. (SMART)*, 2012, pp. 31–34.

[10] U. E. Tahta, S. Sen, and A. B. Can, "GenTrust: A genetic trust management model for peer-to-peer systems," *Appl. Soft Comput.*, vol. 34, pp. 693–704, Sep. 2015.

[11] D. McKnight and N. Chervany, "The meanings of trust," Manag. Inf. Syst. Res. Center, Univ. Minnesota, Rep. MISRC 96-04, 1996.

[12] Y. D. Wang and H. H. Emurian, "An overview of online trust: Concepts, elements, and implications," *Comput. Human Behav.*, vol. 21, no. 1, pp. 105–125, Jan. 2005.

[13] 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," *Comput. Security*, vol. 39, pp. 351–365, Nov. 2013.

[14] I.-R. Chen, J. Guo, and F. Bao, "Trust management for SOA-based IoT and its application to service composition," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 482–495, May/Jun. 2016.
[15] I.-R. Chen, F. Bao, and J. Guo, "Trust-based service management for social Internet of Things systems," *IEEE Trans. Depend. Secure Comput.*, vol. 13, no. 6, pp. 684–696, Nov./Dec. 2016.
[16] Z. Lin and L. Dong, "Clarifying trust in social Internet of Things," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 234–248, Feb. 2018.
[17] D. Gessner, A. Olivereau, A. S. Segura, and A. Serbanati, "Trustworthy infrastructure services for a secure and privacy-respecting Internet of Things," in *Proc. IEEE 11th Int. Conf. Trust Security Privacy Comput. Commun.*, Liverpool, U.K., 2012, pp. 998–1003.
[18] J. Guo and I.-R. Chen, "A classification of trust computation models for service-oriented Internet of Things systems," in *Proc. IEEE Int. Conf. Services Comput.*, New York, NY, USA, 2015, pp. 324–331.
[19] F. Bao and I.-R. Chen, "Dynamic trust management for Internet of Things applications," in *Proc. Int. Workshop Self Aware Internet Things (Self-IoT)*, San Jose, CA, USA, 2012, pp. 1–6.
[20] D. Chen *et al.*, "TRM-IoT: A trust management model based on fuzzy reputation for Internet of Things," *Comput. Sci. Inf. Syst.*, vol. 8, no. 4, pp. 1207–1228, 2011.
[21] H. Al-Hamadi and I. R. Chen, "Trust-based decision making for health IoT systems," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1408–1419, Oct. 2017.
[22] F. Bao and I.-R. Chen, "Trust management for the Internet of Things and its application to service composition," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, San Francisco, CA, USA, 2012, pp. 1–6.
[23] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in *Proc. 12th Int. Conf. World Wide Web (WWW)*, Budapest, Hungary, 2003, pp. 640–651.
[24] L. Xiong and L. Liu, "PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 7, pp. 843–857, Jul. 2004.
[25] S. Asiri and A. Miri, "An IoT trust and reputation model based on recommender systems," in *Proc. 14th Annu. Conf. Privacy Security Trust (PST)*, Auckland, New Zealand, 2016, pp. 561–568.
[26] S. K. M. Yi, M. Steyvers, M. D. Lee, and M. J. Dry, "The wisdom of the crowd in combinatorial problems," *Cogn. Sci.*, vol. 36, no. 3, pp. 452–470, Apr. 2012.
[27] N. S. Nizamkari, "A graph-based trust-enhanced recommender system for service selection in IOT," in *Proc. Int. Conf. Inventive Syst. Control (ICISC)*, 2017, pp. 1–5.
[28] J. Guo, I.-R. Chen, and J. J. P. Tsai, "A survey of trust computation models for service management in Internet of Things systems," *Comput. Commun.*, vol. 97, pp. 1–14, Jan. 2017.
[29] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, Mar. 2007.
[30] E. ElSalamouny, K. T. Krukow, and V. Sassone, "An analysis of the exponential decay principle in probabilistic trust models," *Theor. Comput. Sci.*, vol. 410, no. 41, pp. 4067–4084, Sep. 2009.
[31] G. Ananthanarayanan, V. N. Padmanabhan, L. Ravindranath, and C. A. Thekkath, "COMBINE: Leveraging the power of wireless peers through collaborative downloading," in *Proc. 5th Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, 2007, pp. 286–298.
[32] A. Sharma, V. Navda, R. Ramjee, V. N. Padmanabhan, and E. M. Belding, "Cool-Tether: Energy efficient on-the-fly WiFi hot-spots using mobile phones," in *Proc. ACM Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, Rome, Italy, 2009, pp. 109–120.
[33] P. Jassal *et al.*, "Unity: Collaborative downloading content using co-located socially connected peers," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM Workshops)*, San Diego, CA, USA, 2013, pp. 66–71.

Anuoluwapo A. Adewuyi (S'18) received the M.Sc. degree in advanced computer science with IT management from the University of Manchester, Manchester, U.K., in 2015. He is currently pursuing the Ph.D. degree at the PROTECT Research Centre, Liverpool John Moores University, Liverpool, U.K.

He is a Researcher with the PROTECT Research Centre, Liverpool John Moores University. His current research interests include cryptography, trust management, IoT security, and secure service composition.

Mr. Adewuyi is a Student Member of the IEEE Computer Society.

Hui Cheng (M'18) received the B.Sc. and M.Sc. degrees in computer science from Northeastern University, Shenyang, China, in 2001 and 2004, respectively, and the Ph.D. degree in computer science from Hong Kong Polytechnic University, Hong Kong, in 2007.

He was a Senior Lecturer with Liverpool John Moores University, Liverpool, U.K., from 2013 to 2018. He is currently a Senior Lecturer with the University of Hertfordshire, Hatfield, U.K. His current research interests include trust management, artificial intelligence, dynamic optimization, optical networks, and mobile networks.

Qi Shi received the Ph.D. degree in computing from the Dalian University of Technology, Dalian, China.

He was a Research Associate involved with an EU research project with the University of York, York, U.K. He joined Liverpool John Moores University, Liverpool, U.K., as a Lecturer and then a Reader, where he is currently a Professor of computer security and the Director of the PROTECT Research Centre. He has authored or co-authored over 200 papers in international conference proceedings and journals. He has many years of research experience in many security-related areas.

Dr. Shi has served on the Editorial Boards of several conferences and journals.

Jiannong Cao (M'93–SM'05–F'15) received the B.Sc. degree in computer science from Nanjing University, Nanjing, China, in 1982, and the M.Sc. and Ph.D. degrees in computer science from Washington State University, Pullman, WA, USA, in 1986 and 1990, respectively.

He is currently the Chair Professor of distributed and mobile computing with the Department of Computing and the Director of the University Research Facility in Big Data Analytics, Hong Kong Polytechnic University, Hong Kong. His current research interests include parallel and distributed computing, wireless networks and mobile computing, big data and cloud computing, pervasive computing, and fault tolerant computing.

Prof. Cao has served as the Chair and a member of Organizing and Technical Committees of many international conferences, including PERCOM, INFOCOM, SMARTCOMP, ICMU, ICPP, MASS, ICPADS, IWQoS, ICDCS, DSN, SRDS, ICNP, and RTSS. He has also served as an Associate Editor and an Editorial Board member of many international journals, including the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON COMPUTERS, *IEEE Network*, *ACM Transactions on Sensor Networks*, *Pervasive and Mobile Computing* (Elsevier), *Peer-to-Peer Networking and Applications* (Springer), and *Wireless Communications and Mobile Computing* (Wiley).

Áine MacDermott (GS'13–M'15) received the B.Sc. degree (Hons.) in computer forensics and the Ph.D. degree in network security from Liverpool John Moores University (LJMU), Liverpool, U.K., in 2011 and 2017, respectively.

She was as an Associate Tutor of computing with Edge Hill University, Ormskirk, U.K. She is currently a Senior Lecturer of cyber security and IoT with the Department of Computer Science, LJMU. She has authored or co-authored over 25 academic papers to date, presenting, and exhibiting her research both nationally and internationally. Her current research interests include Internet of Things, critical infrastructure protection, computer network security, collaborative intrusion detection in interconnected networks, and digital forensics.

Xingwei Wang received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 1989, 1992, and 1998, respectively.

He is currently a Professor with the School of Computer Science and Engineering, Northeastern University. His current research interests include future Internet, cloud computing, mobile computing, and mobile social networks.