Mobile Networks and Applications (2018) 23:419–431
https://doi.org/10.1007/s11036-018-1017-z

# Clustering-Driven Intelligent Trust Management Methodology for the Internet of Things (CITM-IoT)

**Mohammad Dahman Alshehri**[1,2] · **Farookh Khadeer Hussain**[1] · **Omar Khadeer Hussain**[3]

Published online: 22 February 2018


**Abstract**
The growth and adoption of the Internet of Things (IoT) is increasing day by day. The large number of IoT devices increases the risk of security threats such as (but not limited to) viruses or cyber-attacks. One possible approach to achieve IoT security is to enable a trustworthy IoT environment in IoT wherein the interactions are based on the trust value of the communicating nodes. Trust management and trust assessment has been extensively studied in distributed networks in general and the IoT in particular, but there are still outstanding pressing issues such as bad-mouthing of trust values which prevent them from being used in practical IoT applications. Furthermore, there is no research in ensuring that the developed IoT trust solutions are scalable across billions of IoT nodes. To address the above-mentioned issues, we propose a methodology for scalable trust management solution in the IoT. The methodology addresses practical and pressing issues related to IoT trust management such as trust-based IoT clustering, intelligent methods for countering bad-mouthing attacks on trust systems, issues of memory-efficient trust computation and trust-based migration of IoT nodes from one cluster to another. Experimental results demonstrate the effectiveness of the proposed approaches.

**Keywords** Internet of things (IoT) · Trust management · IoT scalability · Clustering · Bad-mouthing attacks

## 1 Introduction

It has been predicted that in time, multiple devices will be linked to the Internet, paving the way for the Internet of Things (IoT) [4, 12]. The coinage of the term 'IoT' is credited to Kevin Ashton, who used it in 1999 to refer to 'things' as being exemplified by one device linked to another [7]. According to Jabeur et al [7], the interest in the IoT has experienced an unprecedented increase due to projections that the number of devices could reach 50 billion by the year 2020.

The International Telecommunication Union (ITU) released a report describing a mode of connectivity that allows people to be connected anytime and anyplace [11]. The connectivity of the IoT refers to a network of wireless sensors that consist of distributed and varied components that use sensors to monitor neighboring sensor nodes [3, 17]. Each element of the IoT can be differentiated but is capable of engaging interoperably within the existing system [18, 19].

An Internet of Things system can be described as a set of devices, considered to be smart, that interact collaboratively to fulfill a particular goal [4]. The IoT ushers in a vast array of smart services, including applications used by individuals and organizations to handle real life challenges as they interact and connect with devices anytime, anywhere [2, 19]. With the increasing possibility of infusing smartness everywhere, these devices are being used to connect the physical world, in which field operations take place, and the cyber world, in which data processing and decision-making happen [1]. The devices (part of the physical world) use Internet protocols to communicate with other entities

✉ Mohammad Dahman Alshehri
Mohammad.Alshehri@uts.edu.au

Farookh Khadeer Hussain
Farookh.Hussain@uts.edu.au

Omar Khadeer Hussain
O.Hussain@adfa.edu.au

[1] Centre for Artificial Intelligence, School of Software, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW, Australia

[2] Computer Science Department, Computers and Information Technology College, Taif University, Taif, Saudi Arabia

[3] School of Business, Australian Defence Force Academy University of New South Wales Canberra, Canberra, ACT, Australia

such as Cloud platforms or other IoT nodes [6, 8]. In short, the IoT serves as a universal networking infrastructure that deploys data acquisition devices and communication resources to connect physical and virtual objects [1, 17].

The billions of deployed IoT node devices form complex networks or clusters, which are patterned on human social structures and entities [2, 18]. These structures and entities come in the form of houses, airports, and highways with each of them forming a cluster of IoT nodes. The vision of the IoT's capacity to link myriad things as they interact with the environment and receive information on the status of those interactions was not previously possible by simply observing such sets of things [20]. The clusters involved in the IoT are characterized by heterogeneity with a need for the development of trust from the interaction of one device to another device as well as from a device interacting with a user [9, 20]. Effective trust management in the IoT also provides an essential security characteristic to deliver secure and trustworthy computation between all devices that communicate with one another in the IoT network [1, 21].

Issues of trustworthiness of information, data security and authentication arise when there is data transfer from one cluster to another cluster in the IoT [5, 22]. Additionally, depending on the trust value of the IoT service provider node, the IoT service requester can make a reliable decision of interacting with the service provider. In other words the trustworthiness of IoT nodes serves as a basis for its service provisioning and other service requestors accepting that service. The use of various intelligent approaches for trust management in the IoT has been explored in the literature. However the shortcomings of the current proposed IoT trust solutions are that they are not scalable across billions of IoT nodes and they suffer from attack on their trust systems such as bad-mouthing.

To address the above-mentioned issues, we propose a clustering based approach, wherein IoT nodes are grouped into clusters based on their trust value. The IoT nodes within a cluster gain or lose trust values progressively as they interact with other nodes. As a part of our proposed trust-based IoT clustering approach, we propose intelligent solutions to critical issues related to IoT trust management issues in this paper: The first issue is the possible memory shortage induced by extreme memory usage of node services during the storage and computation of trust computations. Secondly, this paper explores and develops methods to counter bad-mouthing attacks. This is done based on identifying statistical trust outliers from all the values so that these values are not taken into account. Lastly, we draw a link between the trust outlier mechanisms found during trust value computation and memory usage on master nodes to allow IoT cluster nodes to join in/out clusters.

This paper is organized as follows. Section 2 investigates the current studies on IoT trust management and identifies key shortcomings. Section 3 introduces and presents an overview of our proposed framework for enabling trust protocol of IoT scalability (IoT-TM). In Section 4 we present the algorithmic solutions to some of the identified trust issues in IoT-TM. Section 5 presents the simulation set-up parameters and the simulation results. Finally, Section 6 concludes the paper.

## 2 Background work

The increase in the amount of data scattered across computing nodes has led to the concept of 'Big Data', which refers to large and growing amounts of data [13]. Thousands of smart devices that allow seamless communication as part of daily life will be the norm in the future [14]. IoT devices can be considered as service-oriented devices in which each node delivers a pre-defined service to the other nodes in the IoT ecosystem. As significant aspects of the smart revolution, these devices are capable of collecting data, sharing information, and initiating and executing services with minimal human intervention [14]. The Internet of Things is an emerging and improving paradigm that allows most of these physical devices to establish a connection with one another [7].

Different approaches have been proposed to address the problem of trust management in IoT. However, it is acknowledged and argued that trust management is beset by the great challenge of scalability [7, 11]. According to [11], scalability is a key consideration in the creation of trust or designing reliable protocols for management. In order to address this issue, IoT networks need to evolve. A key consideration is to develop intelligent next-generation trust management methods for IoT networks that accommodate the joining and leaving of nodes and can handle networks at large scale. [7]. This suggests that the protocols designed for trust management should be able to seamlessly address IoT nodes joining and leaving the network. Additionally, the developed IoT trust management solutions should be equipped with high resilience to handle trust-related attacks such as bad mouthing, carry out memory efficient trust computation locally, intelligently cluster nodes based on their trustworthiness score, to maintain security even in the most unfavorable environments. In the next remainder of this section, works on IoT trust management will be discussed. The scalability of these proposed approaches and their shortcomings will also be discussed.

Jabeur et al. [7] proposed a solution for the management of trust in the Internet of Things. In their method, the researchers designed a protocol in which the trust relationship data are stored in sets of nodes within the

framework [7]. Nodes that come and go in the process rapidly build up trust with others because of the provision of convergence in the framework [7]. Despite the constrained storage space, storage is effectively utilized to adapt to a large-scale application. However, they fail to address critical issues related to developing scalable trust management IoT solutions such as bad-mouthing in IoT trust systems, the issue of memory shortage of in a node taking into account trust computations etc.

Khan et al. [9] made use of a system that provides security and scalability for accumulating data in the IoT using a tool for multi-coefficient utilization. In this work, files were transferred into shares and given out (allocated) to peers within the IoT storage system. Same shares are generated when the detected inconsistency is fixed. However, they do not address the scalability issues of IoT-based trust solutions.

Kogias et al. [10] proposed a layered and distributed architecture called Distributed Internet-like Architecture for Things (DIAT) and demonstrated its distributiveness. The architecture is also capable of tackling various technical challenges such as heterogeneity, scalability and interoperability. The protocol accommodates objects that are heterogeneous and implements security and privacy aspects using data usage control policies. Automation, intelligence, dynamicity, and zero conguration form part of DIAT. However, DIAT does not implement or propose intelligent scalable trust management solutions for IoT.

Li et al. [11] propose an approach for achieving IoT trust and privacy. Their proposed approach uses SecKit enforcement components, and progress in this approach is monitored by Policy Enforcement Point (PEP). The framework "operates on the implementation of enforcement rules, integrated specifications of the IoT system, middleware equipped with the privacy preserving behavior-driven services for adaptation to the context, specification of Context, Identity, and Role models, and secure and privacy preserving data gathering and transmission at a device level according to security and privacy policies". However, similar to the DIAT framework [10] and Khan et al. [9], this framework fails to address the scalability issues of the proposed IoT trust solution.

Sicari et al. [15] proposed a trust model (TRM-SIoT) in which the previous experiences of each node are mathematically modeled and represented as its trust value. This model uses distributed and centralized trust and reputation models for the creation of a hybrid trust model, resembling a human entity or authority. This model combines solutions proposed for Peer-to-Peer and mobile ad-hoc networks and applies them to the IoT environment. Nodes are used to model human behaviors and relationships during the interaction. However, the primary shortcoming of the proposed method is that it can be exploited to obtain a higher trust level. Similar to the other approaches discussed so far, TRM-SIoT does not address the issue of developing scalable IoT trust solutions and the various issues encountered during developing scalable IoT trust solutions such as bad mouthing of trust value(s), memory efficient trust computation etc.

Varghese et al. [16] presented an innovative architecture for trust modeling and computation. Message Queuing Telemetry Transport (MQTT) and Constrained Application (CoAP) protocols were used to make scalable gateways to efficiently integrate the IoT-to the cloud. The results indicate a guaranteed scalable support for constrained devices, and a high message delivery rate was achieved in busy network conditions without anomalies. However, the model lacks usage optimization of the Central Processing Unit (CPU) and memory, and security support through Datagram Transport Layer Security (DTLS) protocol [16]. Similar shortcomings to that of Sicari et al. [15] apply to the methods proposed by Varghese et al. [16].

## 3 Architecture of the approach for scalable trust management in IoT (IoT-TM)

In this section, we will present the architectural overview for scalable trust management solution (IoT-TM). Subsequently, in Section 4 we present the algorithmic working for addressing the key issues related to scalable trust solutions identified in Section 2.

The IoT-TM provides trustworthy platform for communication between all the devices that communicate with other nodes in the IoT environment. In IoT-TM the Master Node stores the trust values of all the nodes within its cluster. The Super Node in turn stores the trust value of all the MNs. The IoT-TM architecture allows heterogeneous IoT devices and applications to contact each other in trusted heterogenic-device communication.

The IoT-TM architecture (Fig. 1) is a distributed architecture that consists of Cluster Nodes (CN), Master Nodes (MN), Clusters and a Super Node (SN). The IoT-TM architecture consists of a Cluster Node (CN), which resides on a node, and a Master Node (MN). The CN acts as a communication node with responsibility for transporting the data generated or collected by the CNs to their MN. The MN manages many CNs in the cluster, and additionally stores the received data sent by CNs in the MN memory.

In the IoT-TM distributed architecture illustrated in Fig. 1, the Super Node (SN) is the main node in the IoT-TM. It is responsible for all IoT trusted environments and contains an API, referred to as the trust management API. The API allows the SN to communicate with many MNs in the clusters. The SN also has a repository that stores the trust value and address of each MN and CN. The SN repository

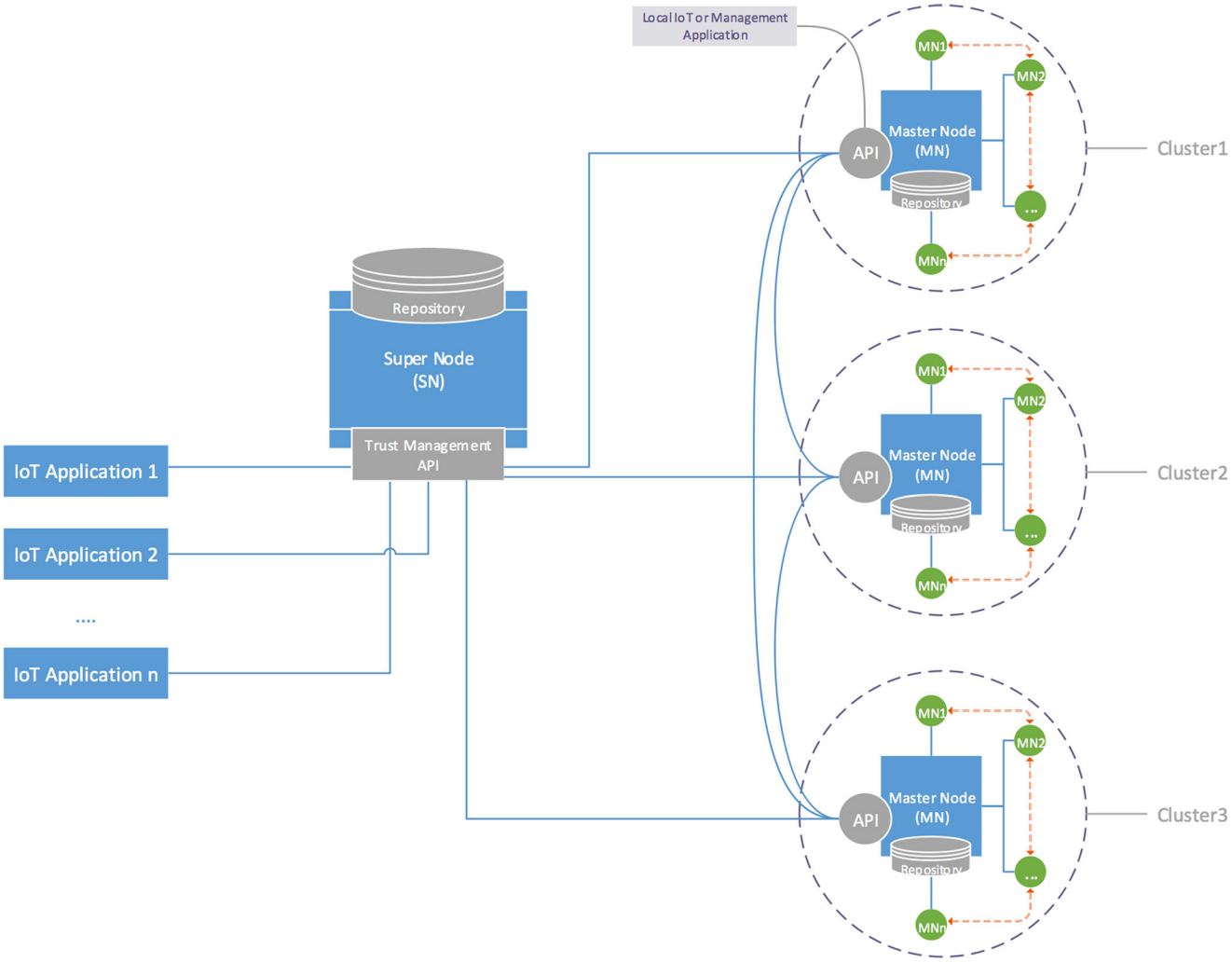



**Fig. 1** Architecture of the IoT-TM

is hierarchical (tree-structured), and each entry relevant to a CN is addressed through the MN's unique ID, thus the SN repository does not store any data collected from the CNs. It only stores their trust value and address information, i.e. through which the MNs and CNs can be accessed. An IoT application running with the SN can provide services based on combined data collected from various CNs. Therefore, IoT applications and services are built on top of the IoT by supporting communications between nodes via the SN.

The IoT-TM architecture provides a centralized model of several clusters and a Master Node (MN), which allows for the central trust management of things over a local area network. The IoT-TM distributed architecture of several MNs and clusters creates a trust distributed system where CNs communicate with each other; MNs communicate with the CNs in their cluster and the SN in a cooperative manner. This architectural flexibility is specifically designed for the communications requirements of the IoT, given that most IoT devices may play different roles in both centralized and distributed operations setups, especially for trust management in the IoT. Additionally, it is this architectural flexibility that allows IoT-TM to scale to a large number of nodes.

## 4 Algorithms for IoT-TM scalability

In order to ensure that our proposed architectural approach for trust management is scalable, we propose the following four algorithms, as follows:

a) Algorithm 1 proposes a new mechanism of clustering (in IoT-TM) by calculating trust value boundaries for each clusters according to memory boundaries. (Section 4.1)
b) Algorithm 2 defines the conditions in which a cluster node is able to change to a specified new master node in IoT-TM. (Section 4.2)

c) Algorithm 3 is used to address the issue of bad-mouthing of IoT nodes. It does so by eliminating the bad-mouthed values (outliers) of a set of floats using Tukey's fences. This algorithm is a proposed solution for extreme bad-mouthing attacks in IoT-TM. (Section 4.3)
d) Algorithm 4 proposes methods by which master nodes will monitor cluster node trust values and try to move some cluster nodes away. A check will then be made on the memory of the current master node to decide whether further cluster nodes need to be removed. (Section 4.4)

In the following sub-sections, we will present the workings of each of these four algorithms.

### 4.1 Algorithm for trust-based cluster boundary calculation

Algorithm 1 is run by the Super Node (SN) to allocate trust value thresholds and memory thresholds for all Master Nodes (MNs). The trust values of all MNs are collected from other MNs. An average of these trust values is calculated and input into a set trustMn. The total memory Mt is then retrieved from all MNs with memory permanently used by service, or Ms. Ms is calculated on the MNs by multiplying Mt with a global predefined service rate Rs to simulate a heavy use of memory.

$$Ms = Mt \times Rs \tag{1}$$

The total memory available Ma will be calculated by subtracting Ms from Mt. Ma will be a part of set Sma.

$$Ma = Mt - Ms \tag{2}$$

Based on Ma, the maximum number of nodes connected to this master node MNm will be calculated by Eq. 3 below. The result will be placed into the set Smnm

$$MNm = \frac{Ma}{Me} \tag{3}$$

$Me$ is a constant defining one set of connection information. It consists of the IP address and trust value of a cluster node. Next, Sma and Smnm are sorted according to trustMn from smallest to largest. In the later part of this algorithm, this sorting ensures consistency of trust values between MNs and CNs connected. The next step is to compute memory rates from Nmc the sum of Smnm elements. Nmc is the total number of cluster nodes possible in the system. The memory minimum bound rate Rmm and memory upper bound rate Rum will be calculated by the equations below.

$$Rmm = \frac{numCn}{Nmc} \tag{4}$$

$$Rum = \frac{Rmm + (1 - Rmm)}{256} \tag{5}$$

$numCn$ is the total number of cluster nodes in the system. The intention of Rum is to give every MN a small amount of memory to allow some memory for cluster node movement between MNs. The number 256 is selected to provide a suitable memory allowance. The higher this number is, the lower the allowance. A high allowance will lead to an unbalanced distribution of cluster nodes to master nodes. A low allowance will result in a high level of restriction on cluster node movement. The set of minimum memory boundary and the set of upper memory boundary are formed from Eqs. 6 and 7.

$$Smm = Sma \times Rmm \tag{6}$$

$$Sum = Sma \times Rum \tag{7}$$

All CN trust values stored on MNs are subsequently gathered and sorted from lowest to highest into a set Stcn. A temporary 2-dimensional set tempTrust is created. Trust values in Stcn will be considered from the middle to the boundaries and added to tempTrust. Values will first be added to the middle of tempTrust and then toward the boundaries. During the process of adding a temporary memory usage is calculated as below:

$$Tmemory = (tempTrust[i].size + 1) \times Me \tag{8}$$

If $Tmemory$ exceeds $Smm[i]$, it will add the value to the next set of $tempTrust$. Lastly, the trust upper bound $Tupper$ and trust lower bound $Tlower$ is computed. When calculating $Tupper$ for $tempTrust[i]$, $tempTrust[i + j]$ is considered. It continues to look for the next $tempTrust$ until the highest number of $tempTrust[i]$ is not equal to the highest number of $tempTrust[i + j]$. $Tupper$ is then calculated by Eq. 9.

$$Tupper = \frac{max(tempTrust[i]) + min(tempTrust[i + j])}{2} \tag{9}$$

$Tlower$ is calculated in a similar way using $tempTrust[i - j]$. It continues to look for the next $tempTrust$ until the lowest number of $tempTrust[i]$ is not equal to the highest number of $tempTrust[i - j]$. $Tupper$ is then calculated by Eq. 10.

$$Tupper = \frac{min(tempTrust[i]) + max(tempTrust[i - j])}{2} \tag{10}$$

Lastly, all $Smm$, $Sum$, $Tupper$ and $Tlower$ are assigned to the consistent MNs.

**Algorithm 1** Allocating trust value thresholds and memory thresholds for master nodes

**Require:** Number of master nodes as numMn, trust values of Master nodes as a set trustMn, Number of cluster nodes as numCn, trust values of cluster nodes as a set Stcn
**for all** master nodes **do**
get trust values of master node from other master nodes into set Strust
Algorithm 3 (Strust)
**if** Strust is not empty **then**
Add average of trust into set trustMn
**else**
Add 0.0 to set trustMn
**end if**
**end for**
**for all** master nodes **do**
$Ma \leftarrow Mt - Ms$
Add Ma to set Sma
$Mnm \leftarrow Ma \div Me$
Add Mnm to set Smnm
**end for**
Sort Sma, Smnm, according to trustMn from lowest to largest
$Nmc \leftarrow sumofSmnmelements$
$Rmm \leftarrow numCn \div Nmc$
$Rum \leftarrow Rmm + (1 - Rmm) \div 256$
$setSmm \leftarrow Sma \times Rmm$
$setSum \leftarrow Sma \times Rum$
Sort Stcn from lowest to largest
**for all** master nodes from the middle to the boundary in the sequence of trustMn **do**
**repeat**
add current float into the set tempTrust[i]
**until** $(tempTrust[i].size + 1) \times Me > Smm[i]$
**end for**
**for all** float sets in set tempTrust **do**
$max \leftarrow$ largest number in tempTrust[i]
$min \leftarrow$ smallest number in tempTrust[i]
j = 0
**repeat**
$maxj \leftarrow$ largest number in tempTrust[i+j]
$minj \leftarrow$ smallest number in tempTrust[i+j]
j++
**until** max != maxj
**if** i + j + 1 is the size of tempTrust **then**
Add max to set STupper
**else**
Add average of max and minj to set STupper
**end if**
j = 0
**repeat**
$maxj \leftarrow$ largest number in tempTrust[i-j]
$minj \leftarrow$ smallest number in tempTrust[i-j]
j–
**until** min != minj
**if** i - j == 0 **then**
Add min to a set STlower
**else**
Add average of maxj and min to STlower
**end if**
**end for**
**for all** master nodes **do**
Assign trust upper bound from STupper
Assign trust lower bound from STlower
Assign memory minimal bound from Smm
Assign memory upper bound from Sum
**end for**

## 4.2 Algorithm for trust-driven node migration from one cluster to another

In our proposed framework (IoT-TM) scalability is achieved my clustering the IoT nodes into groups or clusters based on their trust values. An IoT node (Master Node) takes care of the trust management process of each Cluster Node (CN) within a given cluster. When the trust value of a given CN changes, the MN uses Algorithm 2 to transfer the IoT node to a different cluster. Algorithm 2 is used to accept a request to transfer a (CN) between clusters. It gathers the CN trust values from cluster peers of the current cluster and cluster peers of the target cluster into a list $trustValues$. Algorithm 3 (Section 4.3) is run for $trustValues$ to eliminate outliers, and $Tv$ is calculated as the average of $trustValues$. If $Tv$ is in the trust bound of the target (MN) and adding CN will not exceed the memory threshold of the target MN, the CN and its trust value record will be removed from the current MN and transferred to the target MN.

**Algorithm 2** Request to switch cluster

**Require:** cluster node CN, current master node MN, a new master node NMN, trust boundaries of NMN, memory used by NMN to store cluster node information as Mi, memory threshod as memThres
**for all** cluster peers of CN **do**
Send trust value of CN to the list trustValues on MN
**end for**
**for all** cluster nodes belongs to NMN **do**
Send trust value of CN to the list trustValues on MN forward by NMN
**end for**
Algorithm 3 (trustValues)
$Tv \leftarrow$ average of values in trustValues
**if** Tv not within the trust boundaries of NMN **then**
**return** false
**else**
**if** Mi + Me <memThres **then**
remove CN from its current cluster
remove CN's trust value from MN's trust list
add CN to the cluster of NMN
add Tv as trust value of CN to NMN's trust list
**return** true
**else**
**return** false
**end if**
**end if**

## 4.3 Algorithm to eliminate outliers for bad-mouthing attacks in the IoT

The first step in Algorithm 3 is to break the set of input float values from lowest to uppermost into two parts. Q1 will

be the medium of first part and Q3 will be the medium of the second part in the interquartile range. The interquartile range is expressed as below:

$$Qrange = Q_3 - Q_1 \tag{11}$$

Then the upper threshold of the set will be computed

$$Thupper = Q_3 + 1.5 \times Qramge \tag{12}$$

The lower threshold of the set will be computed in a similar way

$$Tlower = Q_1 - 1.5 \times Qramge \tag{13}$$

Finally, all floats outside of the ranges Tupper and Tlower will be eliminated.

**Algorithm 3** Eliminate outliers

**Require:** A set of floats as Sinput
Sort Sinput from lowest to highest
**if** Sinput.size is event **then**
Slower ← first half set of Sinput
Supper ← second half set of Sinput
**else**
Slower ← first half set of Sinput eliminating the middle number
Supper ← second half set of Sinput eliminating the middle number
**end if**
**if** Slower.size is even **then**
Q1 ← average of the middle two numbers
**else**
Q1 ← the middle value
**end if**
**if** Supperr.size is even **then**
Q3 ← average of the middle two numbers
**else**
Q3 ← the middle value
**end if**
$Qrange \leftarrow Q3 - Q1$
$Thupper \leftarrow Q3 + 1.5 \times Qrange$
$Thlower \leftarrow Q1 - 1.5 \times Qrange$
**for all** float E in Sinput **do**
**if** E >Thupper or E <Thlower **then**
remove E from Sinput
**end if**
**end for**
**return** Sinput

### 4.4 Algorithm for updating and checking the trust values of cluster and master nodes

The intention of Algorithm 4 is to update and check the trust values of the Cluster Nodes (CNs) of Master Nodes (MNs). The memory usage of MNs will also be checked. If the trust value or memory exceeds the pre-defined trust boundaries or thresholds, the MN will use Algorithm 2 to determine whether it can move certain CNs. First, the trust values of each CN are collected from peer cluster nodes. The outliers are eliminated by Algorithm 3 and $Tv$ is calculated as an average. The trust value of each CN will be updated to $Tv$ and stored on the MN. If $Tv$ is not in range of $Tupper$ and $Tlower$ of the current MN, it is added to the list $trustOut$. For all CNs in $trustOut$, a list of MNs where $Tv$ of the CN is in range of the trust bounds of the MNs is computed. This list of MNs will be $trustMNodes$. Algorithm 2 is run for all MNs in $trustMNodes$ with the minimum memory bound of MN. If CN is removed, the algorithm is terminated. If all MNs of $trustMNode$s are considered for Algorithm 2 and CN is not removed from the current master node, Algorithm 2 is run again for all MNs in $trustMNodes$ with the upper memory bound for MNs.

The second part of Algorithm 4 is triggered when the memory of the current master node MN is over the upper memory bound. In this case, the number of nodes that are required to be removed will be computed. The current memory usage is required to calculate the value. This value does not consider the service memory and Ma will be the current memory available.

$$Mi = Mt - Ms - Ma \tag{14}$$

The number of cluster nodes (CNs) that need to be moved will be $Num$, and will be calculated with the upper memory boundary $mum$.

$$Num = (Mi - mum) \div Me \tag{15}$$

Algorithm 2 will then be run for all CNs targeting all other MNs using their minimal memory bound until the Num of CNs is removed. If the Num of CNs is not removed following this process, run Algorithm 2 again for all CNs, targeting all other MNs using their upper memory bound until the number of CNs that have been removed reaches $Num$.

**Algorithm 4** Check current cluster node status

**Require:** a master node MN, memory used by MN to store cluster node information as Mi, its cluster nodes CNs, trust boundaries of MN, a set of other master nodes MNs
**for all** CN in CNs **do**
**if** a trust value of CN exists on MN **then**
add trust value of CN stored on MN to set Stcn
**end if**
**for all** other CN in CNs **do**
add trust value of CN to set Stcn
**end for**
Algorithm 3 (Stcn)
$Tv \leftarrow$ average of values in Stcn
update trust value of CN on MN to Tv
**if** T **then**v is not within trust boundaries of MN
add CN to list trustOut
**end if**
**end for**
**for all** CN in trustOut **do**
Tcn is trust value of CN
**for all** Mn if other MNs **do**
$Tupper \leftarrow$ trust upper bound of MN
$Tlower \leftarrow$ trust lower bound of MN
**if** Tlower <Tcn <Tupper **then**
Add MN to list trustMNodes
**end if**
**end for**
**for all** MN in trustMNodes **do**
**if** Algorithm 2 (CN, current master node of CN, MN, trust boundaries of MN, memory used to store cluster node information in MN, minimum memory bound of MN) **then**
break
**end if**
**end for**
**if** CN not switched to other cluster **then**
**if** Algorithm 2 (CN, current master node of CN, MN, trust boundaries of MN, memory used to store cluster node information in MN, upper memory bound of MN) **then**
break
**end if**
**end if**
**end for**
**if** Mi >upper memory bound of current master node MN mum **then**
number of nodes needed to be removed $Num \leftarrow (Mi - mum) \div Me$
**end if**
**for all** CN in current MN's cluster **do**
**if** $Num \leq 0$ **then**
**return**
**end if**
**for all** MN in other MNs **do**
**if** Algorithm 2 (CN, current master node of CN, MN, trust boundaries of MN, memory used to store cluster node information in MN, minimum memory bound of MN) **then**
Num–
break
**end if**
**end for**
**end for**
**for all** CN in current MN's cluster **do**
**if** $Num \leq 0$ **then**
**return**
**end if**
**for all** MN in other MNs **do**
**if** Algorithm 2 (CN, current master node of CN, MN, trust boundaries of MN, memory used to store cluster node information in MN, minimum memory bound of MN) **then**
Num–
break
**end if**
**end for**
**end for**

## 5 Experimentation and results for the proposed IoT-TM algorithms

In this section, we experimentally evaluate the working of the proposed IoT-TM algorithms in Section 4. The parameters of the simulation set-up and testing are as below:

Table 1 gives the simulation parameters in the base case. We define the Memory Rate as the percentage of total memory used on other services. In our simulation, this chunk of memory is occupied permanently. We define the Trigger Outliers triggers Algorithm 3, eliminate outlier for bad-mouthing trust attacks in IoT in Algorithm 3. We define Trigger Bad Mouths as a trigger that initiates an extreme bad-mouthing, as a result of which some nodes will give an extremely low trust value number for other nodes. We define Trigger Memory Threshold triggers a comparison of the current memory being used with the memory bounds used. Lastly, we define Trigger Balanced Node Distributed as a measure that captures the even distribution of CNs across

**Table 1** Base case simulation parameters

| Parameter | Value |
|---|---|
| Number of nodes | 343 |
| Number of clusters | 9 |
| Simulation time | 200s |
| Master/Cluster node memory | 32 bytes |
| Memory rate | 0.5 |
| Trigger Outliers | True |
| Trigger Bad Mouths | False |
| Trigger Memory Thresholds | True |
| Trigger Balanced Node Distribution | False |

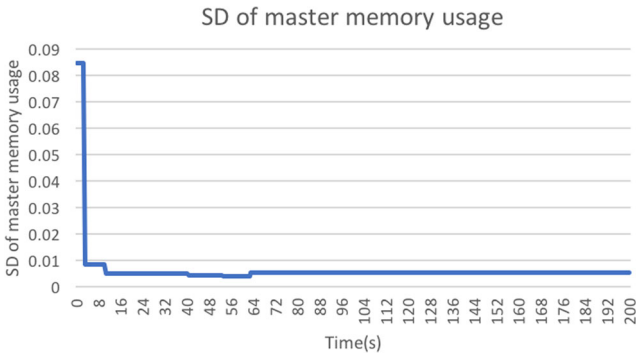



**Fig. 2** Base case SD of master memory usage

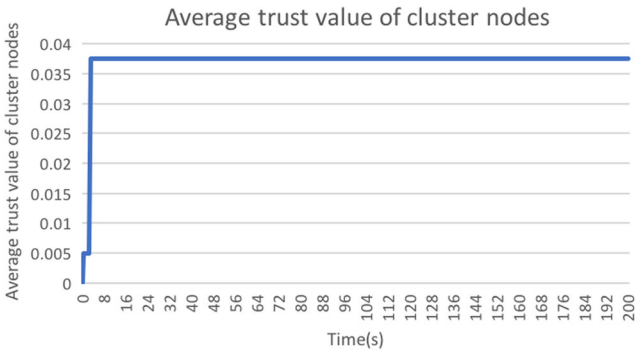



**Fig. 3** Base case average trust value of cluster nodes

MNs. If the Trigger Balanced Node Distribution is true, the cluster nodes will be distributed evenly between the master nodes. If it is false, the cluster nodes will be distributed to only two master nodes at the start of a simulation, otherwise it is evenly distributed across all the master nodes in the simulation.

Figure 2 uses Standard Deviation (SD) to demonstrate the difference in memory usage on the master nodes (MNs). Memory usage on every MN (for trust computations) based on memory without service occupied memory will be computed by Eq. 16.

$$Musa = 1 - \frac{Mavailable}{Mtotal - Mservice} \tag{16}$$

Standard deviation $\sigma$ will then be calculated using the mean of N number of Musa in a set SMusa by Eq. 17.

$$\sigma = \sqrt{\frac{1}{N}\sum(SMusa - \mu)^2} \tag{17}$$

A low $\sigma$ means there is an even distribution of memory is being used for all MNs for trust computation. This demonstrates the effectiveness of IoT-TM along two dimensions as follows: (a) the ability of MNs to carry out trust computations without dedicating large amounts of resources such as memory to it; and (b) the ability to intelligently and automatically group CNs with similar values into a cluster.

The average of values from Fig. 3 is computed by the trust values of all CNs stored on MNs using Eq. 18. $SclusterTrust$ is a list of these trust values.

$$\mu = \frac{1}{N}\sum SclusterTrust \tag{18}$$

This measure ($SclusterTrust$) is used to capture the effectiveness of the algorithm in eliminating the effects of bad-mouthing attacks.

Figure 4 shows the difference between the average cluster trust value from each cluster. During the initial part of the simulation the difference is high and as the simulation progresses the average trust value of all the clusters converge. This shows the effectiveness of IoT-TM in intelligent and automatically clustering IoT with similar trust values.

The average trust value of CNs from each cluster $\mu$ will be calculated by $SclusterTrust$ as a set of all trust values and put into a list $S_\mu$ by Eq. 19.

$$\mu = \frac{1}{N}\sum SclusterTrust \tag{19}$$

An average of $S_\mu$ will be computed by Eq. 20.

$$\mu s = \frac{1}{N}\sum S_\mu \tag{20}$$

Lastly, the standard deviation will be calculated using Eq. 21.

$$\sigma = \sqrt{\frac{1}{N}\sum(S_\mu - \mu s)^2} \tag{21}$$

### 5.1 Results of method proposed for countering bad-mouthing attacks in IoT-TM

Table 2 shows the case simulation values of parameters for bad mouthing attack.

The purpose of this simulation is to set up and simulate an extreme bad-mouthing attack environment and demonstrate

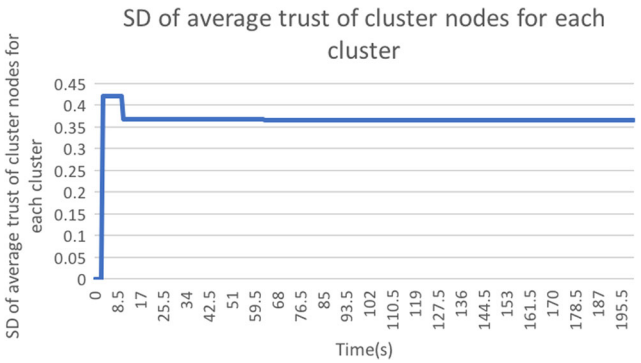



**Fig. 4** Base case SD of average trust of cluster nodes for each cluster

**Table 2** Bad-mouthing attack case simulation parameters

| Parameter | Value |
|---|---|
| Number of nodes | 343 |
| Number of clusters | 9 |
| Simulation time | 200s |
| Master/Cluster node memory | 32 bytes |
| Memory rate | 0.5 |
| Trigger Outliers | Switched |
| Trigger Bad Mouths | True |
| Trigger Memory Thresholds | True |
| Trigger Balanced Node Distribution | False |

the effectiveness of determining statistical outliers (bad-mouthed trust values) using Algorithm 3. To determine the effectiveness of our proposed method for countering bad-mouthing attacks we carry out the simulation with and without Algorithm 3. The blue line in Figs. 5, 6, and 7 shows the results with the proposed bad-mouthing algorithm, and the orange line demonstrates the results without our proposed bad-mouthing algorithm.

Figure 5 illustrates that without the outlier mechanisms, there is a higher SD of master memory usage. A higher SD means there is an unbalanced distribution of CNs to MNs based on master node memory. This comparison shows that in the case of extreme bad-mouthing attacks, the outlier mechanism is ensures the balanced distribution of memory on the MNs.

Figure 6 demonstrates that the outlier mechanism is able to filter out some of the bad-mouthing attack values, resulting in a reliable trust values being available for each CNs for trust computation. These simulation results also reveal a downside of the outlier mechanism, namely that a larger set of data is required for this mechanism to be effective. In the earlier stages of the simulation, fewer trust values accumulate, reducing the effect of the outlier mechanism. As the trust values accumulate the outlier

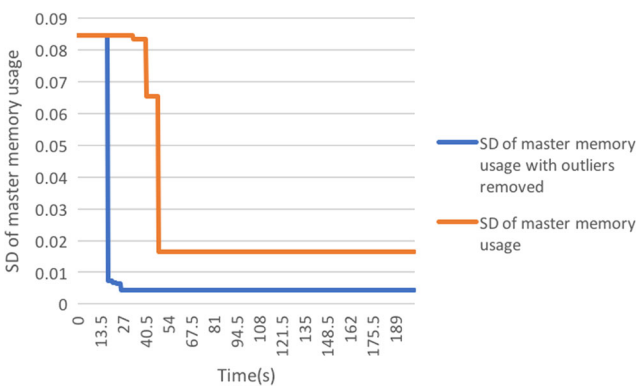



**Fig. 5** Comparison of SD of master memory usage with and without outliers (proposed bad-mouthing algorithm – Algorithm 3)

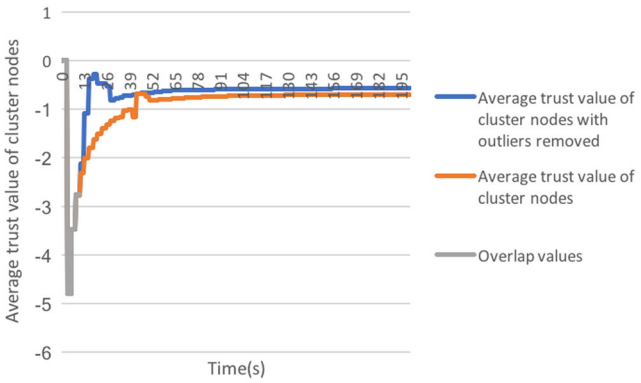



**Fig. 6** Comparison of average trust values of CNs with and without outliers

mechanism becomes more effectiveness and can effectively filter out bad-mouthed values.

Figure 7 shows that clustering is more effective with the outlier mechanism, based on the trust values. There is a difference around 46s of the simulation. This is the result of the uneven distribution of cluster nodes (CNs). At the start of this simulation, CNs are distributed only to two master nodes (MNs). After several trust values generated between cluster nodes, some clusters will have extremely low trust values due to the extreme bad mouth attackss and the other empty clusters will have 0, which is the default trust value. At the first few seconds, this difference between extreme bad mouth values and the default trust value is creating a huge difference between average trust values of each clusters. In the later stage of the simulation, the nodes are evenly distributed based on their trust value, creating a smaller difference in average trust values between clusters.

## 5.2 Results of method proposed for countering extreme memory

The parameters of the simulation approach used for countering extreme memory are shown below in Table 3.

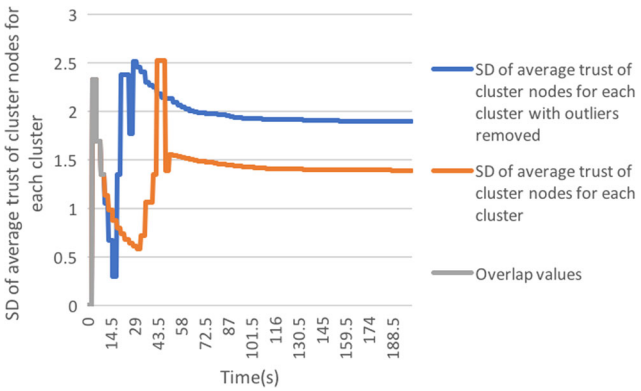



**Fig. 7** Comparison of average trust values of CNs with and without outliers

**Table 3** Extreme memory simulation parameters

| Parameter | Value |
|---|---|
| Number of nodes | 343 |
| Number of clusters | 9 |
| Simulation time | 200s |
| Master/Cluster node memory | 32 bytes |
| Memory rate | 0.9775 |
| Trigger Outliers | True |
| Trigger Bad Mouths | False |
| Trigger Memory Thresholds | Switched |
| Trigger Balanced Node Distribution | True |

Table 3 shows that, compared to the base case, the memory rate it switched from 0.5 to 0.9775, creating an extremely low memory condition. Instead of distributing the cluster nodes (CNs) to only two master nodes (MNs), the CNs are distributed equally to all MNs. The memory thresholds are switched 'on' and 'off' for comparison. When the memory threshold is switched 'on', the memory boundaries will be considered when clustering. When the threshold is switched 'off', the memory boundary mechanisms will be turned off.

Figures 8 and 9 respectively show the time it takes for the simulation to execute with and without the memory threshold. As can be seen from these two figures, the simulation successfully executes for much longer (195.5 seconds) with the memory threshold than without the memory threshold (2.5 seconds). As expected, the SD of master memory usage is 0 due to an even distribution of cluster nodes at the start. Figure 8 shows that, with memory thresholds, the simulation is able to run fully. Figure 9 illustrates that without memory thresholds, the simulation terminates due to the master node running out of memory. This difference between simulation times demonstrates the usefulness of memory thresholds under extreme memory conditions.

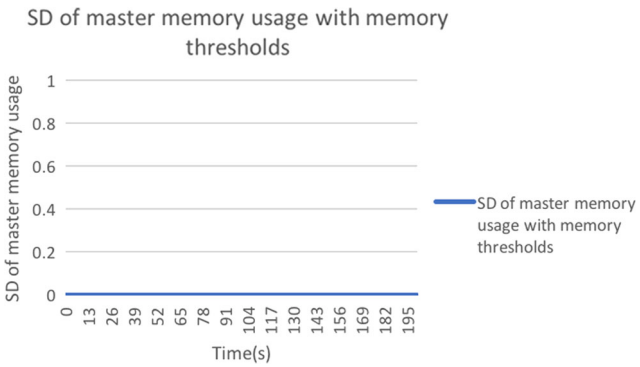



**Fig. 8** SD of master memory usage with memory thresholds

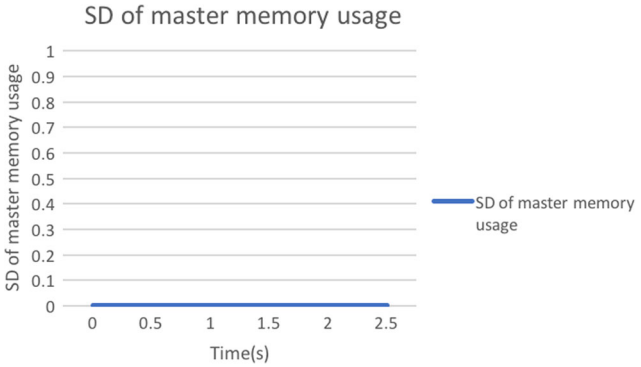



**Fig. 9** SD of master memory usage without memory thresholds

## 6 Conclusion

In this paper we proposed IoT-TM which is a scalable trust management solution for the Internet of Things. To ensure that the proposed IoT-TM is scalable, we developed four algorithms to provide practical solutions to IoT trust management. We proposed intelligent algorithms for: (a) eliminating outliers of trust values, to avoid bad-mouthing attacks and ensure that only correct trust values for an IoT service are taken into consideration; (b) intelligent trust-based formation of clusters; (c) intelligent trust-based migration of IoT nodes from one cluster to another; and (d) an algorithm to examine current IoT cluster node states based on trust values, to determine the appropriate cluster to join. The proposed solutions were evaluated using a prototype and our results demonstrate the applicability of the algorithms. These solutions are a key contribution to the development of trustworthy IoT solutions and their deployment. Our future work will involve deploying the solutions developed in this paper in a real-world environment and evaluating them. We will also explore offering the above solutions as-a-service, so that other nodes can use them to develop secure IoT solutions.

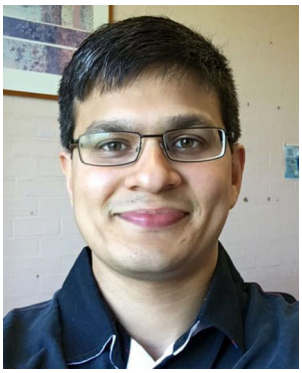
## References

1. Alshehri MD, Hussain FK (2015) A comparative analysis of scalable and context-aware trust management approaches for internet of things. In: International conference on neural information processing. Springer, Berlin, pp 596–605
2. Bao F, Chen R, Guo J (2013) Scalable, adaptive and survivable trust management for community of interest based internet of things systems. In: 2013 IEEE 11th international symposium on autonomous decentralized systems (ISADS). IEEE, pp 1–7
3. Bellavista P, Zanni A (2016) Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP. In: 2016 IEEE 2nd international forum on research and technologies for society and industry leveraging a better tomorrow (RTSI), pp 1–6. IEEE

4. Gharbieh M, ElSawy H, Bader A, Alouini M-S (2017) Spatiotemporal stochastic modeling of IoT enabled cellular networks: Scalability and stability analysis. IEEE Transactions on Communications
5. Chen R, Guo J, Bao F (2016) Trust management for SOA-based IoT and its application to service composition. IEEE Trans Serv Comput 9(3):482–495
6. Han S, NDN-Based WooH (2016) Pub/sub system for scalable IoT cloud. In: Cloud computing technology and science (CloudCom). IEEE
7. Jabeur N, Yasar AU-H, Shakshuki E, Haddad H (2017) Toward a bio-inspired adaptive spatial clustering approach for IoT applications. Future Generation Computer Systems
8. Jiang H, Shen F, Chen S, Li K-C, Jeong Y-S (2015) A secure and scalable storage system for aggregate data in IoT. Futur Gener Comput Syst 49:133–141
9. Khan ZA, Herrmann PA (2017) Trust based distributed intrusion detection mechanism for internet of things. In: 2017 IEEE 31st international conference on advanced information networking and applications (AINA). IEEE, pp 1169–1176
10. Kokoris-Kogias E, Voutyras O, Varvarigou T (2016) TRM-SIoT: A scalable hybrid trust & reputation model for the social internet of things. In: 2016 IEEE 21st international conference on emerging technologies and factory automation (ETFA). IEEE, pp 1–9
11. Li F, Vögler M, Claeßens M, Dustdar S (2013) Efficient and scalable IoT service delivery on cloud. In: 2013 IEEE 6th international conference on cloud computing (CLOUD). IEEE, pp 740–747
12. Yan Z, Zhang P, Vasilakos AV (2014) A survey on trust management for internet of things. J Netw Comput Appl 42:120–134
13. Ahmed A, Bakar KA, Channa MI, Khan AW (2016) A secure routing protocol with trust and energy awareness for wireless sensor network. Mobile Netw Appl 21(2):272–285
14. Ma T, Liu Y, Zhang Z-J (2015) An energy-efficient reliable trust-based data aggregation protocol for wireless sensor networks. Int J Control Autom 8(3):305–318
15. Sicari S, Rizzardi A, Grieco LA, Coen-Porisini A (2015) Security, privacy and trust in internet of things: the road ahead. Comput Netw 76:146–164
16. Varghese R, Chithralekha T, Kharkongor C (2016) Self-organized cluster based energy efficient meta trust model for internet of things. In: 2016 IEEE international conference on engineering and technology (ICETECH). IEEE, pp 382–389
17. Tayeb S, Latifi S, Kim Y (2017) A survey on IoT communication and computation frameworks: An industrial perspective. In: 2017 IEEE 7th annual computing and communication workshop and conference (CCWC). IEEE, pp 1–6
18. Williams R, McMahon E, Samtani S, Patton M, Chen H (2017) Identifying vulnerabilities of consumer Internet of Things (IoT) devices: A scalable approach. In: 2017 IEEE international conference on intelligence and security informatics (ISI). IEEE, pp 179–181
19. Vögler M, Schleicher JM, Inzinger C, Dustdar S (2016) A scalable framework for provisioning large-scale IoT deployments. ACM Trans Internet Technol 16(2):11
20. Wolf T, Nagurney A (2016) A layered protocol architecture for scalable innovation and identification of network economic synergies in the internet of things. In: 2016 IEEE 1st international conference on internet-of-things design and implementation (IoTDI). IEEE, pp 141–151
21. Yu Y, Jia Z, Tao W, Xue B, Lee C (2017) An efficient trust evaluation scheme for node behavior detection in the internet of things. Wirel Pers Commun 93(2):571–587
22. Ngu AH, Gutierrez M, Metsis V, Nepal S, Sheng QZ (2017) IoT middleware: A survey on issues and enabling technologies. IEEE Internet Things J 4(1):1–20


**Mohammad Dahman Alshehri** is an Academic Staff and PhD candidate at the School of Software and member of the Centre for Artificial Intelligence (CAI) at the University of Technology Sydney (UTS), Australia, also a Lecturer at the department of Computer Science at Taif University, Saudi Arabia. He received his Master degree in Information Technology from the University of Technology Sydney (UTS), and Bachelor degree in Computer Science from King Khalid University, Saudi Arabia. He did number of intensive advance courses in Internet of Things security. His main current research interest lies in the areas of Network Security, Trust and Trust Management for the Internet of Things (IoT).

**Dr. Farookh Khadeer Hussain** is an Associate Professor at the School of Software and the Centre for Artificial Intelligence (CAI) of the University of Technology Sydney (UTS). He is also the Head of Discipline (Software Engineering) in the School of Software. Within CAI, he formed and provides leadership to the Cloud Computing group. His key research interests are in Cloud-of-Things, Cloud Computing, Internet-of-Things, and Cloud-driven Data Analytics. He has procured external funding worth more than $ 2.5 million for various projects. His research has been funded by the prestigious Australian Research Council (ARC) and by other Australian government agencies such as the Department of Industry, Innovation and Science etc. He has successfully co/supervised more than 10 PhD students to completion. Currently, he is supervising around 6 PhD students. He has published widely in top journals such as IEEE Transactions on Industrial Informatics, IEEE Transactions on Industrial Electronics, The Computer Journal, Journal of Computer and System Sciences, Journal of Network and Computer Applications, Future Generations of Computer Systems, Knowledge Based Systems etc... His H-index is 26, i-10 index is 75 and his research has received more than 3050 citations.



**Dr. Omar Khadeer Hussain** is a senior lecturer at the University of New South Wales, Canberra. His research interests are in business intelligence, cloud computing and logistics informatics. In these areas, his research work focusses on utilizing decision making techniques for facilitating smart achievement of business outcomes. His research work has been published in various top international journals such as Information Systems, The Computer Journal, Knowledge Based Systems, Future Generation of Computer Systems etc. He has won awards and funding from competitive bodies such as the Australian Research Council for his research.