Computers and Electrical Engineering 92 (2021) 107123

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Context-oriented trust computation model for industrial Internet of Things☆

Ayesha Altaf [a,b], Haider Abbas [a,*], Faiza Iqbal [c], Farrukh Aslam Khan [d], Saddaf Rubab [a], Abdelouahid Derhab [d]

[a] National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan
[b] University of Gujrat (UoG), Gujrat, Pakistan
[c] University of Lahore (UoL), Lahore, Pakistan
[d] Center of Excellence in Information Assurance, King Saud University, Riyadh 11653, Saudi Arabia

ARTICLE INFO

Keywords:
IoT
Trust
Direct observation
Recommendation
Malicious
Ballot Stuffing
Bad Mouthing
Edge intelligence

ABSTRACT

The Industrial Internet of Things (IIoT) has revolutionized the industrial sector by providing advanced and intelligent applications. The objects and nodes communicate with one another to collect, exchange, and analyze a large amount of sensing data using techno-social systems, thereby challenging the security and trustworthiness of the data. To achieve effective communication in IIoT, trustworthy relationships must be established among these objects. This makes trust an important security parameter in an IoT-based environment to achieve secure and reliable service communication at the edge nodes. In this paper, we propose an adaptive Context-Based Trust Evaluation System (CTES), which calculates distributed trust at the node level to achieve edge intelligence. Each edge node takes recommendations from its context-similar nodes to calculate the trust of serving nodes. This collaborative trust calculation mechanism helps in filtering out malicious nodes in the network. The weighing factor "μ" is dynamically assigned based on the previously calculated trust score experienced by the edge node. This research also focuses on formal verification of the proposed CTES model. We analyze the efficiency of CTES in terms of accuracy, dynamic assignment of μ, and resiliency against Ballot Stuffing and Bad Mouthing attacks to avoid malicious nodes. The results ensure the significance of the proposed CTES model for dynamic assignment of μ and provide satisfactory results against EigenTrust, ServiceTrust, and ServiceTrust++ in terms of detecting malicious nodes and isolating them from providing recommendations.

1. Introduction

Industry 4.0 is the most significant industrial revolution, which is focused on creating smart factories by using smart machines [1]. Generally, Industry 4.0 is used for the automation and exchange of data between smart machines for manufacturing purposes, which comprise the Internet of Things (IoT), Industrial Internet of Things (IIoT), and Cyber–Physical Systems (CPS). The IoT connects physical objects through the Internet using sensors, RFID tags, and various smart devices. Sensing devices are used to get the stimulus from the environment and respond to the system. IoT has become a fundamental part of the smart environments and provides multiple services in safety, transport, healthcare, surveillance systems, education, and more importantly, in the industrial domain.

☆ This paper is for special section VSI-eiia. Reviews processed and recommended for publication by Guest Editor Dr. Jiafu Wan.

* Corresponding author.
E-mail address: dr.h.abbas@ieee.org (H. Abbas).

https://doi.org/10.1016/j.compeleceng.2021.107123
Received 3 June 2020; Received in revised form 4 March 2021; Accepted 12 March 2021
Available online 5 April 2021

The communication in IIoT is based on IoT-enabled devices that can run numerous applications for collaborative communication in smart manufacturing to generate large amounts of data. This data needs to be trustworthy and secure by isolating false data generated by the malicious nodes. This can be achieved through edge intelligence, which refers to the process of data collection, analysis, and related calculations at the node that captures or generates the data. IIoT is providing solutions to smart manufacturing in combination with security mechanisms to ensure the reliability of data and to improve communication between smart machines. Although this makes IoT helpful in everyday life, it also opens the doors of threats and vulnerabilities [2]. For the service exchange plan, devices need to be in contact with each other to share the data to provide services in industrial applications. There should be mutual trust between the service requester and the service provider. Therefore, it is imperative to measure the reliability of service providers.

Trust formation is important and can have different definitions according to the requirement. Trust is defined as the only requirement that is used to access the resources and information that need to be shared. The best definition of trust found in the IoT literature is, "Trust is the edge that links the intelligent object with the technological ecosystem" [3]. The presence of trust is helpful in making decisions in the network that comprises multiple systems, which makes it a significant part of the system for devices to perform the requested services. In an IoT network, trust is established among different nodes to complete the requested task. The important factor is when they need to associate with the unknown devices. At that time, devices need to have some trust establishment procedure to communicate with unfamiliar ones. IoT is a blend of various types of devices in the same network; thus, trustworthy communication between all associated entities is important [3].

This research proposes an intelligent trust model, which helps the edge nodes to request services from a reliable service provider instead of a malicious one. Since data is the most significant part of an IIoT architecture, we must consider the trust among entities to share the data for any given task. It is important to consider the context of a service provider in a network for trust calculation. The nodes that are used in the trust calculation process belong to various contextual environments; therefore, the Trust Management System (TMS) must include the context-similar nodes only while formulating the trust. A context in our research is to consider multiple factors, as discussed in Section 3, while taking the recommendations from other nodes. Context represents the environment of users and servers under which they interact to establish a service interaction. This represents the current user environment, server location, and Quality of Service (QoS) parameters.

The main contributions of this research are as follows:

1. We propose a context-based adaptive IoT trust model for edge intelligence. The novelty is in using collaborative filtering to collect trust feedback from nodes that get services under the same context.
2. We develop a direct trust calculation mechanism using actual user satisfaction experience, selected between the range (0–1), based on the percentage of positive observations instead of binary representation, which was previously used in the literature.
3. We develop an adaptive calculation mechanism to allocate weights to direct trust and indirect observations based upon the current user experience while considering the context.
4. The proposed Context-based Trust Evaluation System (CTES) model is designed and verified using High-Level Petri Nets (HLPN).
5. We evaluate CTES to analyze its effectiveness in avoiding malicious nodes against ballot stuffing and bad-mouthing attacks.

The rest of the paper is organized as follows: The introduction section is followed by Section 2, which presents the related work in the domain of IoT. Section 3 describes the proposed CTES model, which gives a clear explanation and working of the mentioned techniques for trust calculation. Section 4 presents the formal modeling and verification of the proposed CTES model. Section 5 provides the performance evaluation of CTES on IoT edge nodes in terms of their accuracy and dynamic assignment of weights to calculate the trust score. It also demonstrates the effectiveness of malicious node avoidance using CTES in the presence of ballot stuffing and bad-mouthing attacks. Finally, Section 6 concludes the paper with our findings and future directions.

2. Related work

IIoT is currently playing a significant role in smart manufacturing for Industry 4.0. CPS, IoT, and IIoT are collectively working for data generation and to analyze the collected data. Cognitive capabilities for the collected data for smart manufacturing are proposed in [4]. Multiple reference architectures that help for smart manufacturing by using CPS and IIoT are discussed in [5] along with their pros and cons. An architectural classification for Industry 4.0 is proposed to divide it into layers. The policy is formed after having a detailed analysis of the manufacturing process [6]. Recently, many studies have analyzed the implementation of Industry 4.0 for smart manufacturing [7]. Similarly, trust and privacy framework has been proposed by Mannhardt et al. [8]. Jeong et al. have described trust issues in IIoT-based smart manufacturing process [9]. To start the communication across the network for sharing resources and services, trust in IIoT is imperative. There exist several studies that have analyzed security, privacy, and trust issues in IIoT [10]. TMS is the mechanism to evaluate, update, and revoke the assurance based on trust score. To make the system upright, it is imperative to gain an insight into the methods by which the trust is being measured by the devices. The requesting device needs to predict the trust score by calculating direct observations from the service provider.

Malicious node involvement by having multiple service-oriented attacks is an important aspect to consider while designing the trust management mechanism in an IoT network. A dynamic trust model is presented in [11] to mitigate the malicious and misbehaving nodes by assigning weighted recommendations in the trust calculation process. Multiple reputation-based systems are also proposed to mitigate the impact of malicious nodes while calculating the trust score. Malicious nodes that are used for On–Off Attacks (OOA) on IoT trust models are discussed in [12,13] along with their mitigation strategies. Many studies have identified the

gaps in trust management models proposed for IoT; thus, the research work done in [14] and [2] are more focused on context-based trust management systems in IoT. Malicious node detection or avoidance while calculating trust or taking recommendations from malicious users need to be considered to avoid uncertainties in trustworthiness. A limited number of research articles considered this important aspect in IoT trust calculation [15].

A study in [16] used context in terms of the social relationship of nodes for trust level calculation. Each node in the proposed system is served by its owner and the owner is responsible for trust calculation in the system. The calculation of direct and indirect trust scores is used to compute the trust level while giving weight to the importance of context, power usage, and computation power. Another centralized trust calculation procedure is introduced in [17] to work with the context of a node while considering various metrics of a node. There exist multiple research articles that proposed trust computation procedures for IoT; however, only few of them [16–18] have given importance to context while calculating trust scores. Furthermore, the literature reveals that binary values are being used to represent a direct trust score. If the node is trusted, then the value is 1, whereas 0 is assigned for complete distrust. This research compares the results while taking the values for direct trust in the range between 0 and 1.

The literature highlights the limitation that most of the research has been done on binary values for trust calculation. Weights assigned to direct and indirect trust must be dynamic based upon the context of the data. Literature also reveals that many available trust management systems do not consider the context of data while calculating the trust level. CTES proves that if the context is being considered, then the trust will converge more quickly towards the ground truth. These limitations are also highlighted in our research presented in [2] for better understanding. These limitations motivate us to develop a trust-oriented computational model based on direct user experience and recommendation of context-similar users. In this research, we have focused on context-oriented trust formation. The total trust calculation process incorporates actual user satisfaction experience for the direct trust of client nodes towards the service providers. An adaptive weight parameter is used for taking the recommendations from context-oriented nodes. This process ensures the filtering of malicious nodes so that their recommendations do not make any major contribution to the total trust value.

3. Context-based trust evaluation system model

The proposed Context-Based Trust Evaluation System (CTES) is a distributed trust management system. Each user calculates its trust score, which depends upon the context of the requested service from the service provider. In order to achieve scalability, each user calculates the trust score towards a limited set of requested services to which it interacted. Each user stores this information in the form of lists. The information list of user U_X includes the following:

1. A list of server IDs denoted by a set $S_X = \{s_1, s_2, s_3, \ldots., s_n\}$, which represents server IDs with which user U_X interacted and obtained services.
2. A list of server locations represented by a set $L_{Xj} = \{l_1, l_2, \ldots., l_n\}$, which represents the location of server S_j that provided services to user U_X.
3. A list of services represented by set $P_i = \{p_1, p_2, \ldots\ldots, p_n\}$, which represents the type of service user U_X requested from server S_j.
4. A list of user experiences represented by $E_{X_{ij}} = \{(a_{Xij}, b_{Xij}), \ldots\ldots\ldots., (a_{Xnm}, b_{Xnm})\}$, where a_{Xij} and b_{Xij} represent the positive experiences of the user and negative experiences of user U_X towards server S_j requesting service P_i, where $i = \{1, 2, \ldots\ldots, n\}$ and $j = \{1, 2, \ldots\ldots, m\}$.
5. A list of trust scores user U_X has towards server S_j providing services P_i. It is represented by a set $T_X = \{t_{Xij}, \ldots\ldots\ldots, t_{Xnm}\}$, where $i = \{1, 2, 3, 4, \ldots\ldots, n\}$ and $j = \{1, 2, 3, 4, \ldots\ldots, m\}$.

The network model that depicts the user profiles having these lists is displayed in Fig. 1, where U_X = User, which requests the service, and S_j = Server, which is accessed by user U_X for providing service. The profile list of User U_X represents the information, which is stored by user U_X related to server S_j.

3.1. Direct user interaction experiences

Direct trust is calculated by the Naive Bayesian method using direct user interaction experience values. The significance of Naive Bayesian in trust measurement and reputation calculation motivates us to use this well-established method. We calculate the user experience, which depends upon multiple available contexts of the service provider. These multiple contexts are further used to calculate trust scores using the Bayesian framework [19]. The important contexts that we consider in this research include the server's capability in terms of its service provided, the location from which the server is providing service, and the type of server used. Quality of service is taken as a context in terms of response time for any requested service in this research. In service computing, a user can provide a feedback rating of a service provider after direct interactions depending upon non-functional characteristics. These non-functional characteristics include response time, throughput, server availability, cost of service, etc.

Existing trust management solutions do not include these context based nonfunctional characteristics while computing trust scores of service providers. In our proposed CTES trust model, the current user interaction experience of user U_X towards service provider S_j providing service P_i at location L_X is evaluated based upon existing context and is represented by E_{Xij}. We have extended the simple case proposed by Chen et al. in [15], in which they have used a binary value to represent user satisfaction experience, where value 1 indicates the satisfied user and value 0 indicates that the user is not satisfied. On the scale of the trust measures in [20], by giving more flexibility to trustor to variate between trust and distrust, our proposed approach takes the user

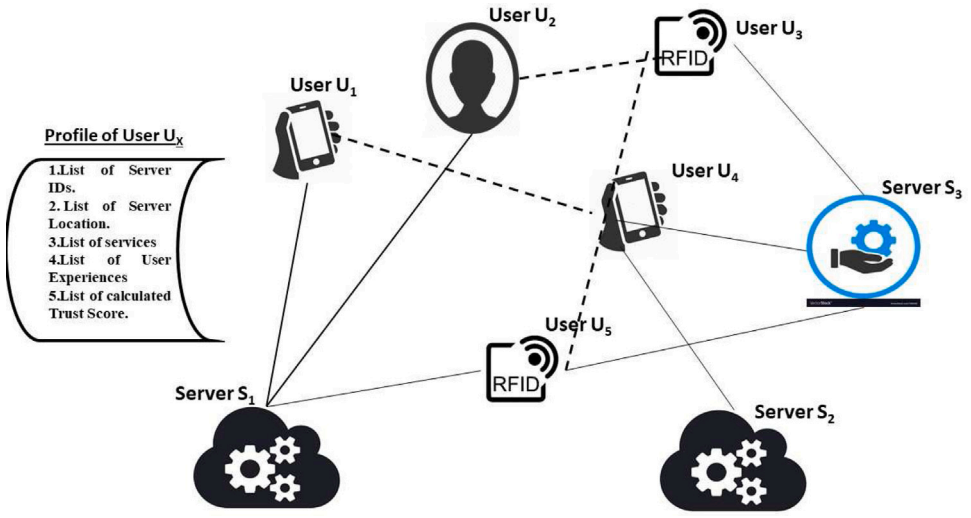

Fig. 1. User Profile with Attached Profile Lists.

Table 1
Class label values with various scales of trust measurement.

Parameters	Values	Satisfaction Level
E_0	0	Ignorance
E_1	0.1	Complete Distrust
E_2	0.2	Very High Distrust
E_3	0.3	High Distrust
E_4	0.4	High Medium Distrust
E_5	0.5	Low Medium Distrust
E_6	0.6	Low Medium Trust
E_7	0.7	High Medium Trust
E_8	0.8	High Trust
E_9	0.9	Very High Trust
E_{10}	1	Complete Trust

satisfaction experience $E_{X_{ij}}$'s value between 0 and 1 such that 0 represents not satisfied and 0.5 represents partially satisfied, as discussed in Table 1.

Table 1 lists all values of user experience in which a user node U_X can assign a value of $E_{X_{ij}}$. When it experiences trustworthy communication with its peer nodes, then depending on the level of trust, a value from $0.6-1$ is assigned, where 1 being a fully trusted interaction. Similarly, while detecting the malicious activity of neighboring nodes, user node U_X can assign a value between $0-0.5$, where 0 being a fully malicious interaction and 0.5 represents a lower-middle level of distrust. These values, assigned to $E_{X_{ij}}$, are used to calculate direct and indirect user trust. Therefore, depending upon the respective interactions of each user, malicious and non-malicious nodes are categorized based on total trust computation.

Eq. (1) shows that the parameters a_{Xij} and b_{Xij} are updated based upon trust decay considering current user satisfaction experience $E_{X_{ij}}$.

$$\begin{aligned} a_{X_{ij}} &= e^{-\phi\Delta t} \times a_{X_{ij}}(old) + E_{X_{ij}} \\ b_{X_{ij}} &= e^{-\phi\Delta t} \times b_{X_{ij}}(old) + (1 - E_{X_{ij}}) \end{aligned} \tag{1}$$

Eq. (1) describes that $E_{X_{ij}}$ contributes to positive experiences and $(1 - E_{X_{ij}})$ provides negative user experiences. To update $a_{X_{ij}}$ and $b_{X_{ij}}$, we consider an exponential decay $e^{-\phi\Delta t}$ on old values of $a_{X_{ij}}$ (old), where ϕ represents a decaying factor, which is a small number to formulate small trust decay over time and Δt represents the trust update cycle.

The direct trust ($t^d_{X_{ij}}$) of user U_X towards server S_j providing service P_i is calculated, as represented in Eq. (2).

$$t^d_{X_{ij}} = \frac{a_{X_{ij}}}{a_{X_{ij}} + b_{X_{ij}}} \tag{2}$$

The value of $a_{X_{ij}}$ and $b_{X_{ij}}$ are initially set to 1 as no prior information is available about these values.

3.1.1. Algorithm for direct user trust calculation

Algorithm 1 presents the step-by-step procedure for direct trust calculation for user U_X towards server S_j.

Algorithm 1 Calculation of Direct Trust

Input : E_{Xij}
Output : t^d_{Xij}
Set $a_{X_{ij}} \leftarrow 1$ and
$b_{X_{ij}} \leftarrow 1$ and
$t \leftarrow 1$
for $i \leftarrow 1$ to n **do**
    Update Interval
    $\Delta t \leftarrow (t_i - t_{(i-1)})$
end for
for all E_{Xij} **do**
    $a_{X_{ij}} \leftarrow e^{-\phi \Delta t}$
    $b_{X_{ij}} \leftarrow e^{-\phi \Delta t}$
    $t^d_{X_{ij}} = \frac{a_{X_{ij}}}{a_{X_{ij}} + b_{X_{ij}}}$
end for
Return : $t^d_{X_{ij}}$

3.2. Indirect user trust based on recommendations

Devices can exchange their interacted servers and a list of obtained services to provide trust recommendations. This is referred to as indirect user trust. To select trust feedback from devices working under the same context, a requesting node will first measure its similarity with the recommender to connected servers, the respective location of the servers, and a list of requested services. The reason to consider these metrics is that entities in any network can form a similarity network based upon some agreed similarity measures [21]. We have considered cosine similarity to calculate the similarity of two context lists of recommending nodes and the requesting node. The similarity measure is a value between 0 and 1, where $0.5 - 1$ is similar and $0 - 0.499$ represents dissimilar. The reason to choose cosine similarity lists is the low complexity and computational efficiency for IoT devices.

Once a direct interaction of a user U_X about server S_j is done, the user will require recommendations of its neighboring user U_Y who has interacted with the respective server in the same context. Here, we consider the similarity of a server, the location of the server, and the provided services working under the same context to be used as a measure of recommendation. Now, we describe the context similarity calculation dynamically as follows:

3.2.1. Server similarity (SI^S_{XY})

The server similarity measure is necessary to recognize that the two users U_X and U_Y are accessing the services of the same server. The recommendations given by U_Y to U_X about server S_j can only be considered if the two users are connected to the same server under the same circumstances. First, the two users U_X and U_Y will exchange their servers list, S_X and S_Y. Next, the binary vector of the two lists $V\vec{S}_X$ and $V\vec{S}_Y$ are produced having size $|S_X U S_Y|$. Given the cosine angle between $V\vec{S}_X$ and $V\vec{S}_Y$, the cosine similarity measure representing the SI^S_{XY} is as follows:

$$SI^S_{XY} = \frac{V\vec{S}_X \cdot V\vec{S}_Y}{\|V\vec{S}_X\| \cdot \|V\vec{S}_Y\|} \tag{3}$$

If the computed similarity index is between the range $0.5 - 1$, then it will be added in the list $L^s_{Xj} = [U_{Y1}, U_{Y2}, \ldots.]$. Here, the calculation process is represented with an example. Let $S_X(S_1, S_2, S_4)$ represents the list of servers accessed by user U_X and $S_Y(S_3, S_1, S_5)$ represents the list of servers accessed by user U_Y. The vector list $V\vec{S}_X$ with size $|S_X U S_Y|$ will contain values $[S_1, S_2, S_3, S_4, S_5]$. According to our example, the binary representation of the set $V\vec{S}_X = [1, 1, 0, 1, 0]$ shows the interactions with servers from S_X with the value 1. Similarly, the vector set $V\vec{S}_Y$ will have same elements as those of $V\vec{S}_X$ and its binary set contains the values $[1, 0, 1, 0, 1]$. Using cosine similarity, the computation of similarity of these two vectors mentioned in the above example will result in 0.33. Thus, $\vec{V}_Y$ will not be added in the list L^S_{Xj} because its similarity is less than 0.5 and U_X will not rely on the recommendation from U_Y. On the other hand, considering the same example, if U_Y interacted with more than two same servers as those of V_X, the similarity computation will be 0.67, which shows that U_X and U_Y are similar in the context of connected servers and thus U_X can rely on the recommendations of U_Y. The list L^S_{Xj} will contain users having the similarity index of $0.5 - 1$ with U_X. After the calculation of server similarity, this list will be further evaluated for similarity in terms of server location.

3.2.2. Server location similarity (SI^L_{XY})

The server location similarity index is computed to identify the users/members of the list L^S_{Xj} computed in Eq. (3) to further evaluate those users who have interacted with the server on a similar location as of U_X. The user U_X will exchange their server

location list L_Xj with all users U_{Yi} (a member of the list L^S_{Xj}) one by one. The similarity index of the server's location SI^L_{XY} is calculated using cosine similarity as follows:

$$SI^L_{XY} = \frac{V\vec{L}_X \cdot V\vec{L}_Y}{\|V\vec{L}_X\| \cdot \|V\vec{L}_Y\|} \tag{4}$$

If the computed similarity index is between the range $0.5 - 1$, then it will be added in the list $L^L_{Xj} = [U_{Y1}, U_{Y2}, \ldots.]$. Next, this list will further be sent for evaluations in the context of the type of services obtained by users from the same server having similar locations.

3.2.3. Service similarity (SI^P_{XY})

The type of service similarity is calculated to further evaluate the connected users in terms of similarity in the context of the same service. The user U_X will exchange the service list with the user U_{Yi}(from list L^L_{Xj}). The cosine similarity computation is performed to calculate the similarity index SI^P_{XY} as follows:

$$SI^P_{XY} = \frac{V\vec{P}_X \cdot V\vec{P}_Y}{\|V\vec{P}_X\| \cdot \|V\vec{P}_Y\|} \tag{5}$$

If the computed similarity index is between the range $0.5 - 1$, then it will be added in the list $L^P_{XiY} = [U_{Y1}, U_{Y2}, \ldots.]$. Finally, this list will represent all those users who interacted with S_j at location L_j accessing service P_i, similar to U_X.

Now, U_X can use the recommendations of the users listed in L^P_{XiY}. Each user can exchange their trust recommendations request to its filtered list L^P_{XiY} periodically at Δt interval. Once a user U_X receives recommendations, it will select top n recommendations from n users based upon the highest similarity value with U_X and is represented in L_{Xij}. Based on these values, the indirect trust will be calculated as:

$$t^r_{Xij} = \sum_{L_{Xij}} \frac{SI_{XY}}{\sum_{L_{Xij}} SI_{XY}} \cdot t^d_{Yij} \tag{6}$$

Here, L_{Xij} represents the set of up to k users whose similarity values are the highest, and t^d_{Xij} represents direct trust of user U_Y towards server S_j at location L_j providing service P_i, and SI_{XY} represents a value calculated in Eq. (4). The weight is assigned to every recommendation that is considered after taking the filtered list against the proportion of the calculated similarity score of the recommender to the total similarity of all the recommenders.

3.3. Total trust

The total trust value of user U_X towards server S_j providing service P_i is represented as t_{Xij} and is calculated by combining direct and indirect trust scores obtained from Eq. (2) and Eq. (6) respectively.

$$t_{Xij} = \mu\, t^d_{Xij} + (1 - \mu) t^r_{Xij} \tag{7}$$

3.4. Function for assigning weight to weighing parameter μ

Algorithm 2 represents the procedure for dynamically calculating the value of μ based on direct trust and indirect recommendations. The design concept of the proposed trust management protocol CTES is to provide the highest trust to those devices that give positive user satisfaction experiences E_{Xij}. This leads to the calculation of μ depending upon the difference of current user satisfaction experience E_{Xij} with the direct trust t^d_{Xij}, indirect recommendations t^r_{Xij} and the current total trust evaluation t_{Xij}. The μ when assigned γ has the value ranging between $(0.81 - 1.0)$. Similarly, μ has a range between $(0 - 0.20)$ when it is assigned Γ. κ shows the value ranges between $(0.41 - 0.60)$ and χ and ϕ have value ranges in $(0.61 - 0.80)$ and $(0.21 - 0.40)$ respectively. The value of μ is divided into five categories, as shown in Algorithm 2.

Line 2 in Algorithm 2 estimates the difference of current user satisfaction experiences E_{Xij} with the most recent direct user experience t^d_{Xij} observed by user U_X within the previous trust interval Δt. If this difference is nearer to 0 and the value of direct trust t^d_{Xij} observed by user U_X is not equal to the recommended trust value t^r_{Xij} received by user U_X from its recommenders, then μ will be assigned a value between a range of $0.81 - 1.0$. It shows that the trust will mostly depend upon the direct user experiences because the current user experience can be a good indicator compared with the average direct user satisfaction experiences obtained over a while and so is true for the rest of the conditions.

The value of μ is dynamically adjusted considering the recommendations. If the recommender is context similar to the trustor node, then the value of μ is high, while in the latter case, it is low. The complexity analysis for CTES algorithm is presented in our research in [22].

Algorithm 2 Calculation of μ (Weighing Parameter)

Input : E_{Xij}, t^d_{Xij}, t^r_{Xij}
Output : μ

procedure WEIGH PARAMETER(E_{Xij}, t^d_{Xij}, t^r_{Xij})
  for all S_i **do**
    if $(t^d_{Xij} - E_{Xij}) \approx 0 \wedge (t^d_{Xij} \neq (t^r_{Xij})$ **then**
      $\mu = \gamma$
    else if $(t^r_{Xij} - E_{Xij}) \approx 0 \wedge (t^d_{Xij} \neq (t^r_{Xij})$ **then**
      $\mu = \Gamma$
    else if $(t^d_{Xij} - E_{Xij}) \approx 0 \wedge (t^r_{Xij} - E_{Xij}) \approx 0 \wedge (t^d_{Xij} \approx (t^r_{Xij})$ **then**
      $\mu = \kappa$
    else if $(t_{Xij} - E_{Xij}) \approx 0 \wedge (t^d_{Xij} - E_{Xij})$ has minimum difference $\wedge (t^r_{Xij} - E_{Xij})$ has maximum difference **then**
      $\mu = \chi$
    else if $(t_{Xij} - E_{Xij}) \approx 0 \wedge (t^r_{Xij} - E_{Xij})$ has minimum difference $\wedge (t^d_{Xij} - E_{Xij})$ has maximum difference **then**
      $\mu = \Phi$
    else
      Re-Calculate μ
    end if
  end for
end procedure
Return : μ

Algorithm 3 Calculation of Total Trust

Input : E_{Xij}, t^d_{Xij}, t^r_{Xij}
Output : t_{Xij}

After every Δt
for all do E_{Xij}
  μ = Weigh Parameter($E_{Xij}, t^d_{Xij}, t^r_{Xij}$)
  Calculate Total Trust
  $t_{Xij} = \mu t^d_{Xij} + (1 - \mu) t^r_{Xij}$
end for
Return : t_{Xij}

4. Formal modeling and verification of context-based trust evaluation system (CTES)

This section presents the verification and formal modeling of the proposed CTES model for an IoT environment. The main purpose of using formal modeling and verification is to verify the working process of the CTES based algorithms through mathematical rules. The High-level Petri Nets (HLPN) are used to display the formal modeling and verification of CTES calculations. HLPN is defined as a 7-tuple in the form of $N = (P, T, F, \varphi, R_n, L, M_0)$, where P is the set of places, T is the set of transitions, and R_n denotes the defined rules for the transitions. L defines a label on F, and M_0 describes the initial marking. In HLPN, transitions are displayed as rectangular boxes. Also, circles and arrowheads describe the places and data flow respectively. The complete system is represented by defining the set of $P(Places)$ and data types that are linked with the set. The sets are rules that need to be defined and are involved in HLPN.

The proposed algorithms are described in Section 3. In HLPN model, we present the CTES algorithms in the form of mathematical properties (i.e., Rules). The variables and places used in this model are defined in Table 2 and Table 3 respectively. Fig. 2 depicts the HLPN of the proposed CTES.

The HLPN model of the CTES system takes as input the user satisfaction experiences $E_X ij$ to update the trust update cycle Δt value in the update cycle in transition **U-Interval**.

$$\mathbf{R(U-Interval)} = \forall\, i2 \in x2 \wedge \forall\, i3 \in x3|$$
$$(i3[1]_{n_{\forall\, i3[1]_n \in x3}}) := Upd - interval(i2[3] - i2[4]) \wedge$$
$$x3' := x3 \cup \{i3[1]\} \quad \mathbf{Rule1}$$

$$\mathbf{R(D-T-Calculation)} = \forall\, i4 \in x4 \wedge \forall\, i5 \in x5 \wedge i6 \in x6|$$
$$i4[1] := i5[1] \wedge i4[2] := i5[1] \wedge$$

Table 2
Types used in HLPN for CTES.

Types	Description
$E_X ij$	User satisfaction experience
Δt	Trust update cycle
Exp Decay	Exponential decay
D-Trust	Direct trust
S_v Similarity	Server similarity
SI^P_{XY}	Server service similarity
SI^S_{XY}	Service similarity
L^S_x	List of users based on server similarity
SI^L_{XY}	Server location similarity
S_I Similarity	Location similarity
S_S Similarity	Service similarity
L^l_x	List of users based on location similarity
F-List	Final list of users or members
R-Trust	Recommended trust
C- μ	Calculated value of weighting parameter
Total-Trust	Total trust

Table 3
Mapping of data types on places.

Places	Description
$\varphi(E_X ij)$	$P(a_{x,j} \times b_{x,j} \times t_i \times t_{i-1} \times E'_{xij})$
$\varphi(\Delta t)$	P(T-U-C)
φ(Exp Decay)	$P(e^{-\phi \Delta t})$
φ(D-Trust)	$P(t^d_{Xij})$
$\varphi(S_v)$ Similarity	$P(V\vec{S}_x \times V\vec{S}_Y \times U)$
$\varphi(SI^S_{XY})$	$P(SI^S)$
$\varphi(SI^P_{XY})$	$P(SI^P)$
$\varphi(L^S_x)$	P(L-u-srvs)
$\varphi(SI^L_{XY})$	$P(SI_L)$
$\varphi(S_I$ Similarity)	$P(V\vec{L}_x \times V\vec{L}_Y \times SI^P \times U)$
$\varphi(S_P$ Similarity)	$P(V\vec{P}_x \times V\vec{P}_Y \times SI^P \times U)$
$\varphi(L^l_x)$	P(L-u-loc)
φ(F-List)	$P(L_{xij})$
φ(R-Trust)	$P(t^r_{Xij})$
φ(C-μ)	$P(\mu)$
Total-Trust	$P(t_{Xij})$

$$(i6[1]_{m_{\forall\ i6[1]_m \in x6}}) := [(i4[1])\ /\ ((i4[1]) + (i4[2]))]_{m_{\forall\ (i4[1],i4[2])_j \in x4}} \wedge$$
$$x6' := x6 \cup \{i6[1]\} \quad \textbf{Rule2}$$

Eq. (6) performs a calculation of recommended trust t^r_{Xij} as follows: First, the server similarity is calculated by invoking the similarity procedure, which is calculated using Eq. (3). If the similarity value is ≥ 0.5, then the user U_Y will be added to the list of trustworthy users and this list will be further checked for server location similarity. The server similarity is calculated by invoking the similarity procedure again and if the similarity value is ≥ 0.5, then user U_Y will be added to the list of trustworthy users, and this list will be further checked for service similarity. The final filtered list is generated after checking all the similarities. Finally, the recommended trust t^r_{Xij} based on similarity values is calculated. Rule 3 to Rule 7 represent the above-mentioned process.

$$(\mathbf{C-Sv-Similarity}) = \forall\ i7 \in x7 \wedge \forall$$
$$i8 \in x8 \wedge \forall\ i9 \in x9 \wedge i10 \in x10|$$
$$i8[1] := C - SIM(i7[1], i7[2]) \wedge (i8[1] \geq 0.5)$$

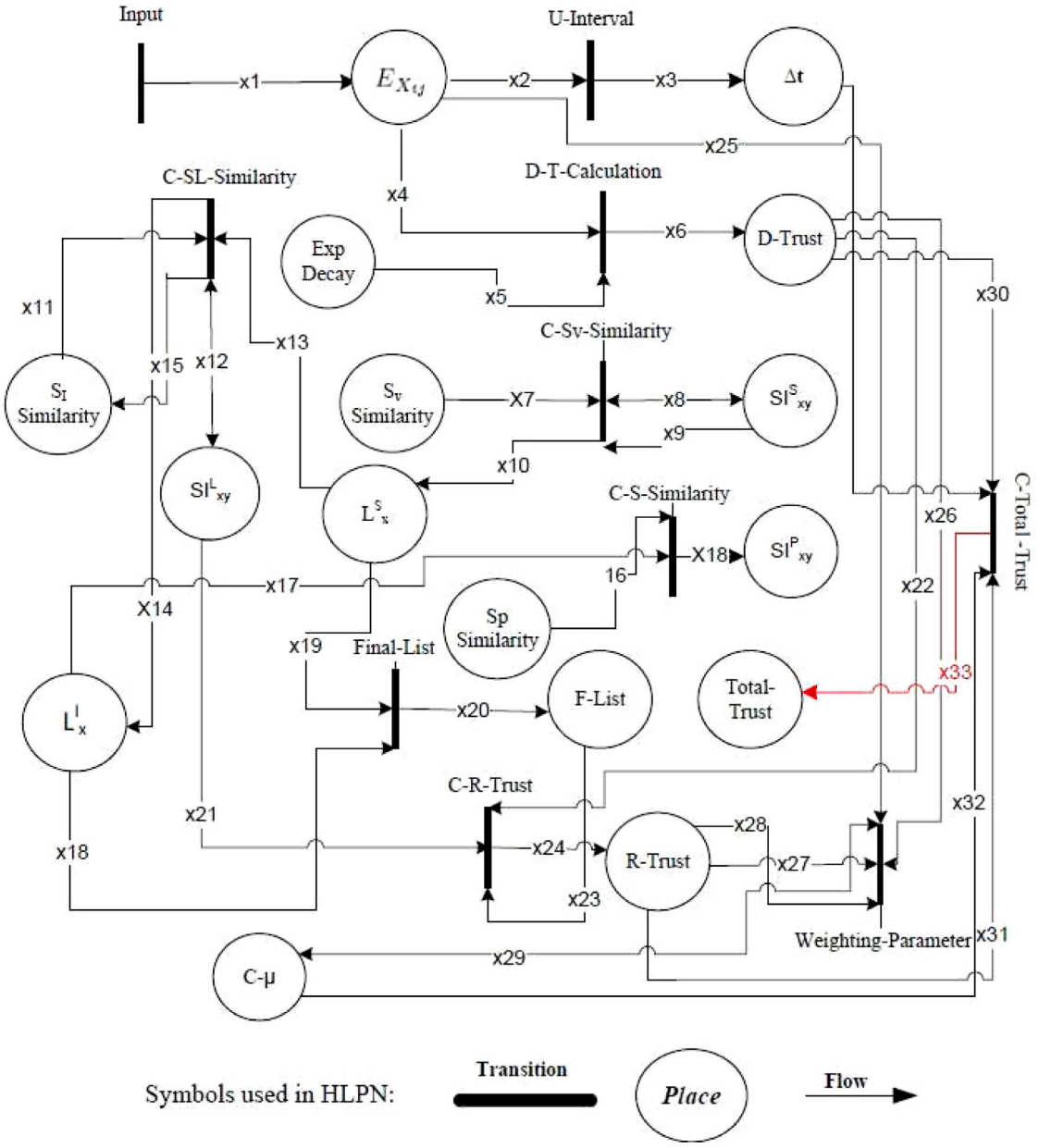

Fig. 2. HLPN of Context-Based Trust Evaluation System (CTES).

$\rightarrow (i10[1] := i10[1] + i7[3])$

$x10' := x10 \cup \{i10[1]\}$ **Rule3**

$(\mathbf{C-SL-Similarity}) = \forall\, i11 \in x11 \wedge \forall$

$i12 \in x12 \wedge i13 \in x13 \wedge$

$i14 \in x14 \wedge i15 \in x15|$

$i12[1] := C - SIM(i11[1], i11[2]) \wedge ((i12[1] \geq 0.5)$

$\rightarrow (i14[1] := i14[1] + i12[3]) \vee$

$(i12[1] \ngeq 0.5) \rightarrow Remv(i13[1]) \wedge x12' := x13 \cup \{i13[1]\})$

$\wedge\, x12' := x12 \cup \{i12[1]\}$ **Rule4**

$$\mathbf{R(C-S-Similarity)} = \forall\, i16 \in x16 \wedge \forall$$
$$i17 \in x17 \wedge \forall\, i18 \in x18 \wedge$$
$$i14 \in x14 \wedge i15 \in x15|$$
$$i12[1] := C - SIM(i11[1], i11[2]) \wedge$$
$$((i12[1] \geq 0.5) \rightarrow (i14[1] := i14[1] + i12[3]) \vee$$
$$(i12[1] \ngeq 0.5) \rightarrow Remv(i13[1]) \wedge x12' := x13 \cup \{i13[1]\})$$
$$\wedge\, x12' := x12 \cup \{i12[1]\} \quad \textbf{Rule5}$$

$$\mathbf{Final-List}) = \forall\, i18 \in x18 \wedge \forall\, i19 \in x19 \wedge \forall\, i20 \in x20|$$
$$i20[1] := F - list(i18[1] \cup i19[1]) \wedge$$
$$x20' := x20 \cup \{i20[1]\} \quad \textbf{Rule6}$$

$$\mathbf{Final-List}) = \forall\, i21 \in x21 \wedge \forall\, i22 \in x22 \wedge$$
$$\forall\, i23 \in x23 \wedge \forall\, i24 \in x24|$$
$$i24[1] := F - list(i21[1], i22[1], i23[1]) \wedge$$
$$x24' := x24 \cup \{i24[1]\} \quad \textbf{Rule7}$$

Algorithm 2 shows the calculation of weighing parameter μ in transition **Weighing-Parameter**. It calculates the difference between current user satisfaction experiences $E_X ij$ and the most recent direct user experience t^d_{Xij} observed by user U_X within the previous trust interval Δt. If this difference is close to 0 and the value of direct trust t^d_{Xij} observed by user U_X is not equal to the recommended trust value t^r_{Xij} received by user U_X from its recommenders, then μ will be assigned a value in the range of $[0.81 - 1.0]$. It shows that the trust will mostly depend upon the direct user experiences t^d_{Xij} because the current user experience can be a good indicator compared to the previous average t^d_{Xij} satisfaction experiences. Similarly, other conditions are evaluated, as shown in Algorithm 2. The calculation of weighing parameter μ is given in Rule 8.

$$\mathbf{R(Weighing-Parameter)} = \forall\, i25 \in x25 \wedge \forall\, i26 \in x26 \wedge$$
$$\forall\, i27 \in x27 \wedge \forall\, i28 \in x28$$
$$\wedge \forall i29 \in x29|((i26[1] - i25[5]) \approx 0) \wedge$$
$$(i26[1] \neq i27[1]) \rightarrow i29[1]$$
$$:= (0.8 - 1.0) \vee ((i27[1] - i25[5]) \approx 0) \wedge$$
$$(i26[1] \neq i27[1]) \rightarrow i29[1]$$
$$:= (0 - 0.2) \vee ((i26[1] - i25[5]) \approx 0) \wedge$$
$$((i27[1] - i25[5]) \approx 0) \wedge (i26[1] \approx i27[1]) := (0.41 - 0.6)$$
$$\rightarrow i29[1] \vee ((i27[1] - i25[5]) \approx 0) \wedge min(i26[1], i25[5])$$
$$\wedge max(i27[1], i25[5]) \rightarrow i29[1](0.61 - 0.8)$$
$$\vee((i27[1] - i25[5]) \approx 0) \wedge$$
$$max(i26[1], i25[5]) \wedge min(i27[1], i25[5])$$
$$\rightarrow i29[1] := (0.21 - 0.4) \vee Re - cal(i29[1]) \quad \textbf{Rule8}$$

Algorithm 3 calculates the total trust t_{Xij} using the value of weighing parameter μ. The calculation is done by assigning the weights to direct trust t^r_{Xij} and indirect trust t^d_{Xij}. The transition **C-Total-Trust** represents the total trust calculation process, as given in Rule 9.

$$\mathbf{R(C-Total-Trust)} = \forall i30 \in x30 \wedge \forall i31 \in x31 \wedge$$
$$\forall i32 \in x32 \wedge \forall i33 \in x33|$$
$$i33[1] := [(i30[1] \times i32[1]) + \{((1 - i32[1]) \times i31[1])\}]$$
$$\wedge x33' := x33 \cup \{i33[1]\}\ \textbf{Rule9}$$

In this section, we have presented formal modeling, analysis, and verification of our proposed CTES, which is written in the form of HLPN and mathematical Rules in $Z3Solver$, and thus we verified that the proposed system satisfies the claimed properties in CTES.

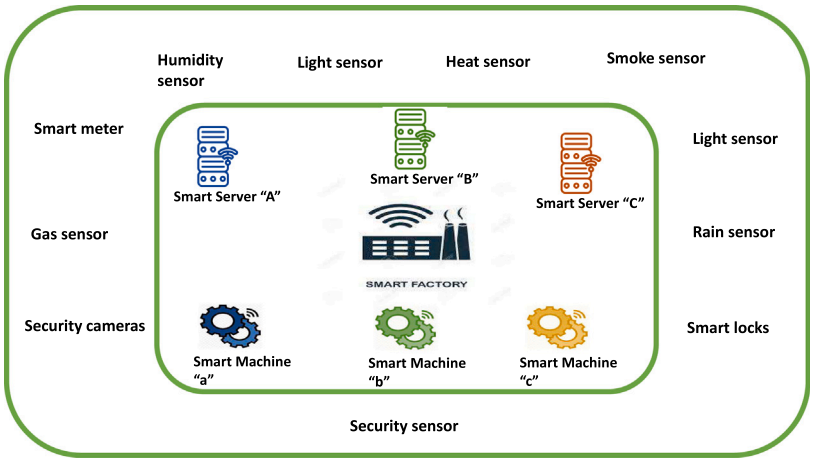

Fig. 3. An IIoT architecture in a Smart Factory for Data Generation.

5. Performance analysis of CTES

To evaluate and validate our CTES model, we use IoT Sentinel dataset, which represents the traffic of 31 IoT-based smart nodes of different types. It considers a network of 31 smart devices of 27 different types. The setup of IoT-based smart node is repeated 20 times for each device type to get better accuracy. The dataset directories contain several pcap files, and each pcap file represents the setup of the given scenario. We have used the parameters of existing servers, in addition to their location, services type, and corresponding response time values.

Fig. 3 shows the IIoT basic architecture, which is taken as the case study for data generation for a smart factory. The figure shows the availability of multiple servers, clients, and sensors that are cooperatively working in a smart factory to complete the task while communicating and collecting the data from various nodes. In this architecture, the servers provide information to machines for assigned tasks, and the machines are interacting with each other to share the services so that after the completion of a task by one machine, another machine starts execution. The calculation of trust using CTES is performed as follows:

1. When sensors send data to machines.
2. When a machine sends that data to servers.
3. When machines interact with each other for data exchange.
4. When servers share data with each other for task completion.

We analyze the performance of CTES in terms of accuracy and resiliency against malicious attacks using the data obtained from the real-time setup for IoT devices. In this section, we analyze the trust accuracy (Section 5.1), and demonstrate the effect of dynamic weight assignment on total trust calculation (Section 5.2). We also show the CTES malicious node avoidance in an IoT environment (Section 5.3). The accuracy of the protocol represents the correctness of the calculated trust score compared to the real scenario. We evaluate the accuracy of our protocol by considering the user satisfaction level generated by the dataset values as ground truth. Finally, we compare CTES with other existing work (Section 5.4).

5.1. CTES accuracy analysis

In this section, we observe the accuracy of trust convergence of our proposed CTES protocol to the ground-truth value. The direct user satisfaction experience of user U_X towards server S_i is calculated by Eq. (2). The input value of the user satisfaction experience is taken from the real data of the IoT environment, which may exhibit variations in terms of actual interactions. These deviations are not considered as noise but as actual observed values. Initially, user U_X sets the value of t^d_{Xij} (direct trust) to 0.5, which represents the initial trust of user U_X on server S_i. The trust is then dynamically updated using CTES protocol based on the interaction between users and servers, and hence the requested services are delivered. It depends upon the trust feedback acquired after every interaction. We first compare the variation of direct trust with recommended trust and its impact on total trust considering static control of μ. Fig. 4 represents the trust score of user U_X trusting a server S_i during service interactions. Direct trust is then mapped by comparing it with the recommended trust. It is observed that the direct and recommended trust calculations vary depending upon the response time-based user satisfaction experiences of respective users with server S_i. As the total trust (calculated using Eq. (7)) depends upon direct and recommended trust, the value varies depending upon two trust values. An important observation to note is that the static value of μ, which varies between 0.2, 0.5, and 0.9, affects the total trust. When the value of μ is lower, the total trust diverges towards recommendations, while the higher value of μ reflects more contributions of direct trust in total trust calculations.

Fig. 5 presents a case in which the accuracy of CTES has been measured. It shows the interactions that are based on the information of different contexts. The total trust is calculated based on different contexts and the results are observed. User U_X

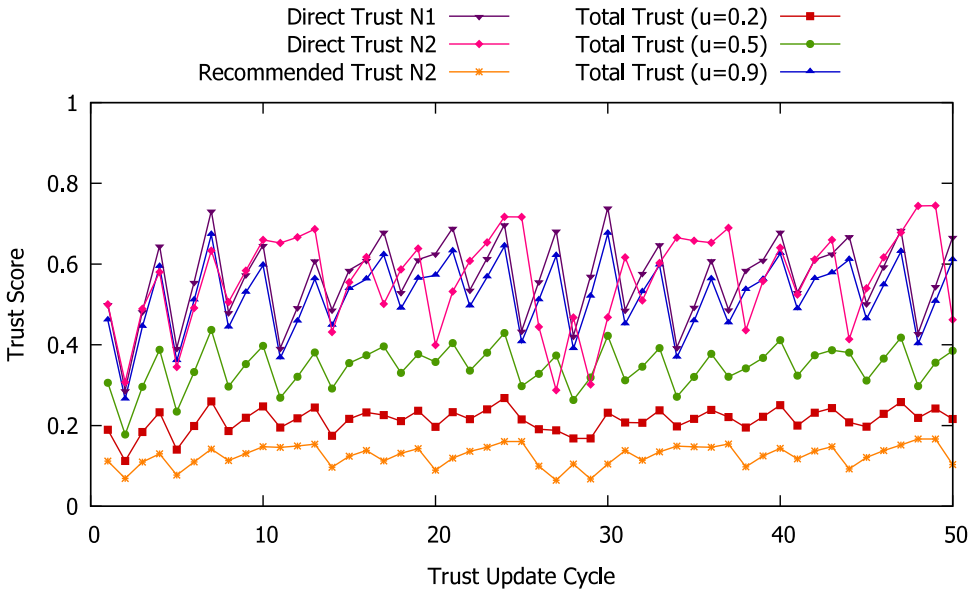

Fig. 4. Effects of using Static Values of μ for Services in Same Context.

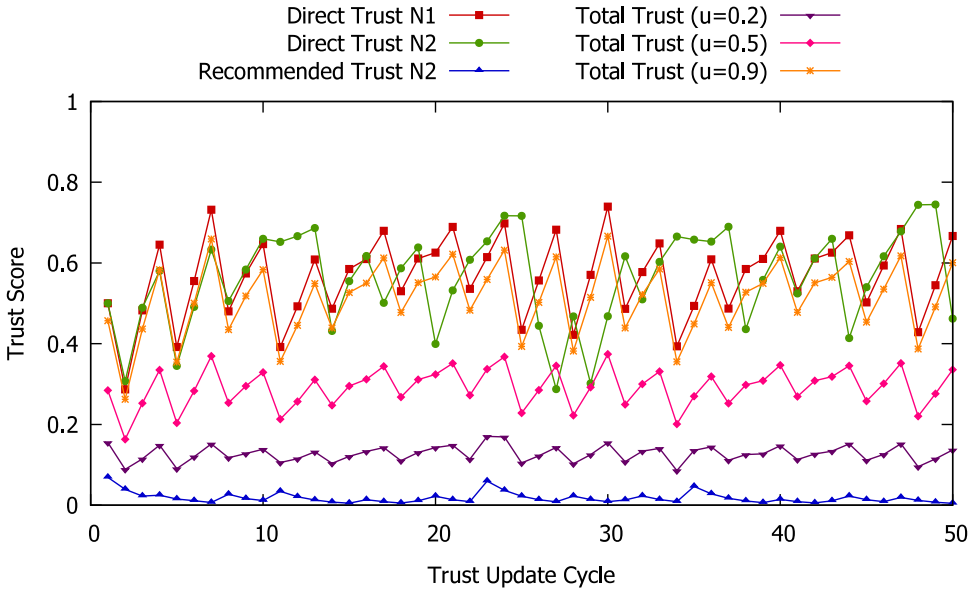

Fig. 5. Effects of using Static Values of μ for Services in Different Context.

accesses the service P_i from server S_i at location L_i. The direct trust score of user U_Y is represented as a green circle marked line and it accesses the same service P_i from another server S_j residing at location L_j. Thus, the two users are not operating in the same context. Considering the different contexts, the result shows the variations of the trust calculation process for user U_X and user U_Y. The blue triangle marked lines display the case when U_X uses the recommendations of user U_Y (which is working in a different context from U_X). The value of the total trust is decreased, and this can be verified through the difference in the average values of direct trust, recommended trust, and total trust. We also consider the variation in the value of the μ, which depends upon the recommended and direct trust. Therefore, the static value of μ cannot justify the calculation of trust based on contextual trust calculation. μ must be dynamically adjusted and it should consider similar contextual recommendations. CTES provides a mechanism to dynamically adjust μ, as explained in Algorithm 2 and shown in Fig. 6.

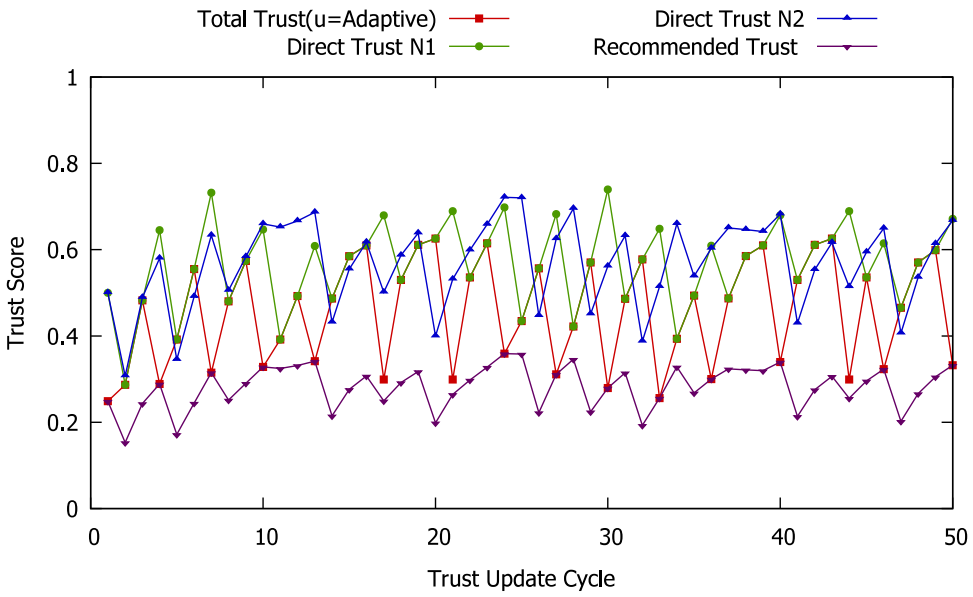

Fig. 6. Effects of using Adaptive Values of μ for Services in Same Context.

5.2. Adaptive control of μ

Fig. 6 presents the trust score calculation based on the dynamic adjustments of μ. It is calculated based on the following three factors:

1. The incoming user experiences E_{Xij} of U_X with server S_i during the next trust update cycle ΔT.
2. Previous direct trust score t^d_{Xij}.
3. Previous recommended trust t^r_{Xij}.

This dynamic adjustment of μ based on the above parameters provides a mechanism to find the total trust score, which depends upon the recommendations of contextually similar nodes of U_X. In Fig. 6, direct trust of U_X is represented by the green circle marked lines and U_Y is represented by the blue triangle marked lines. The recommended trust scores of contextually similar nodes are mapped using purple triangle marked lines. It can be seen that the total trust score varies between direct trust and recommended trust dynamically based upon the next user interaction experience in previous trust scores. In comparison, Fig. 7 presents the dynamically adjusted trust score of contextually different nodes. As the nodes are contextually different, the recommended trust score is reported lower than that of Fig. 6. Moreover, these contextual differences are dynamically adjusted, and the total trust scores highly fluctuate between direct and recommended trust scores.

5.3. Malicious node avoidance with CTES

We consider a scenario of malicious node avoidance by assuming that user U_X is interacting with server S_j and receives current user satisfaction experience E_{Xij} of 0.6, which represents low medium trust, as presented in Table 1. User U_X has stored a list of historical data related to direct trust t^d_{Xij}, recommended trust t^r_{Xij} and total trust t_{Xij}. In this scenario, we assume that the average value of t^d_{Xij} is 0.37 and the average value of t^r_{Xij} is 0.52 to calculate the average total trust t_{Xij}.

5.3.1. Ballot Stuffing attack scenario

This attack is executed when any malicious node is providing good recommendations of a service provider to boost its reputation, thus making more chances for that service provider to be selected for providing a service. In the case of IIoT architecture, any malicious smart machine provides a good recommendation for a malicious service provider so that it can be selected for providing the service. This communication will take place along with the trust calculation process, which includes the direct trust score and indirect trust calculation. Indirect trust calculation takes the recommendations from neighboring nodes. The recommendation can be taken from any malicious node, which is part of any ballot stuffing attack. In this case, the malicious recommender will provide a good recommendation of a server irrespective of the fact that how the service was provided by the service provider. We conduct an experiment of this scenario by using 70 nodes, and out of which 20 act maliciously for the execution of the attack. The malicious nodes provide good recommendations on all servers that offer service in all interactions.

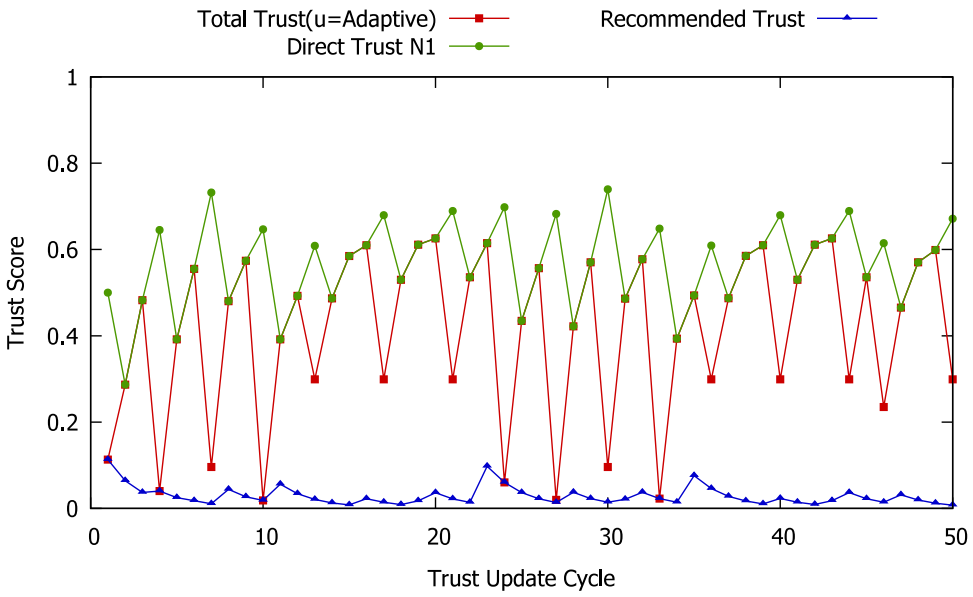

Fig. 7. Effects of using Adaptive Values of μ for Services in Different Context.

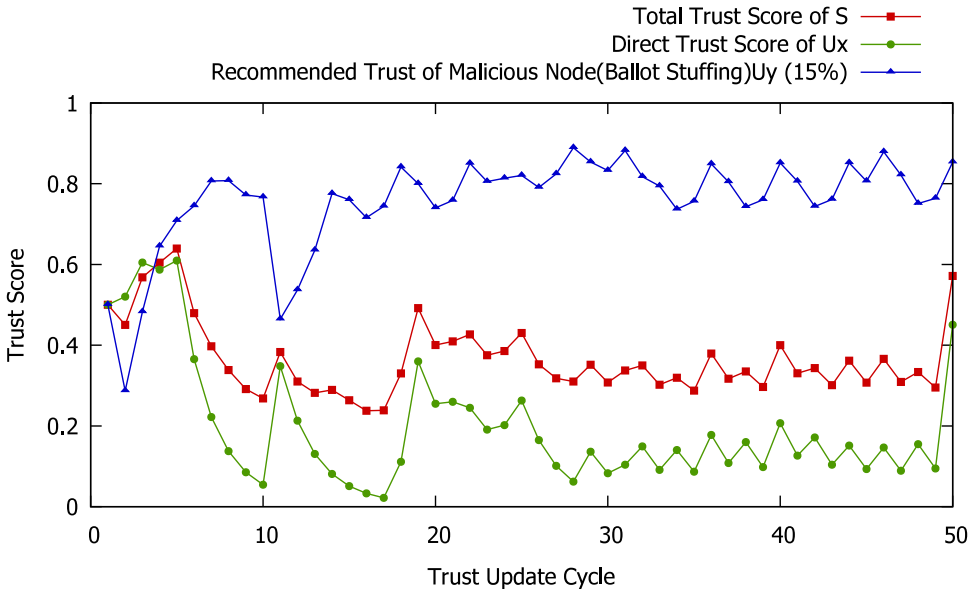

Fig. 8. Effects of Ballot stuffing with CTES.

We assume that U_{y2} is performing ballot stuffing attacks against U_X by providing good recommendations on server S_i. We also assume that U_X receives the value of 0.3 as current E_{Xij} from server S_i while U_{y2} provides recommendations t^r_{Xij} in the range $0.9 - 1$. This recommended value is used in Eq. (6) of CTES to calculate the similarity of the recommenders for U_X. This will filter out those recommenders that are providing dissimilar services (out of the context of U_X). Secondly, using the adaptive calculation process of calculating μ, the value of current E_{Xij} will be compared with the average value of direct trust t^d_{Xij} (which in this case is 0.37) and the average recommended value of trust t^r_{Xij} (which is 0.52 in this case). Thus, it will filter out the recommenders that are involved in ballot stuffing attacks (see Fig. 8).

5.3.2. Bad Mouthing attack scenario

The bad-mouthing attack provides bad recommendations on any service provider which is not malicious and offers good services. Let us consider the IIoT architecture presented in Fig. 3, where the smart machines request services from any available smart server. The direct trust calculation is based on the actual interaction between the client machine and the server. The recommendations are

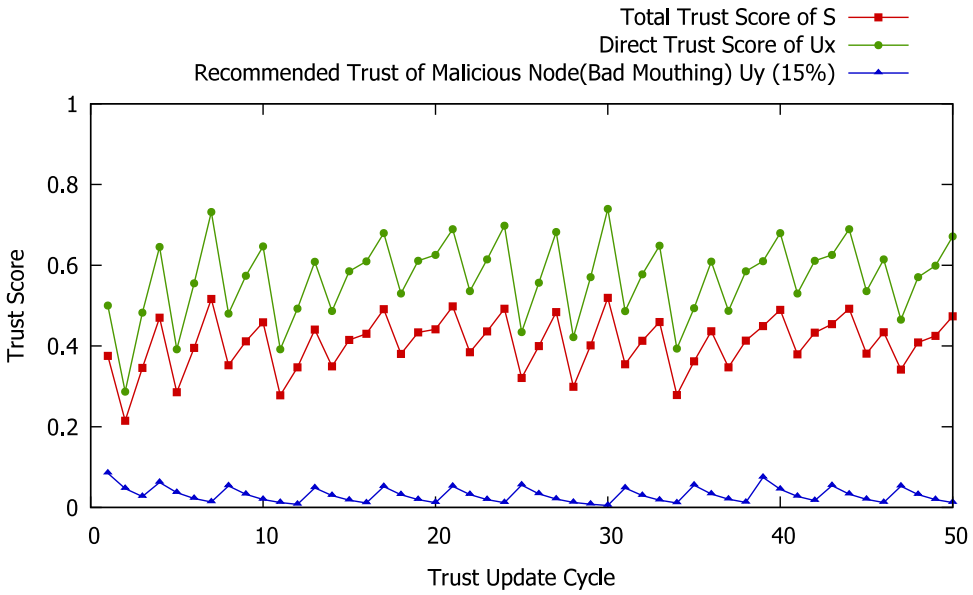

Fig. 9. Effects of Bad Mouthing with CTES.

needed for the indirect calculation process. The malicious nodes in bad-mouthing attacks provide bad recommendations to the client machines.

Now, consider that U_Y executes a bad-mouthing attack against server S_i and provides recommendations in a range of $0-0.2$ to U_X. The value of the current user experience E_{Xij} of U_X is 0.6. The adaptive μ calculation algorithm of CTES will compare the value of E_{Xij} with direct trust, recommended trust, and total trust, as the value of E_{Xij} is closer to the average recommended value. In this case, as the recommended trust value is taken as an average value from the historical data stored in U_X, U_{Y2} will not be able to execute the bad-mouthing attack when the CTES algorithm is used by U_X (see Fig. 9).

5.4. Performance comparison with existing work

We compare the performance of our proposed CTES with existing schemes: EigenTrust [23], ServiceTrust [24], and ServiceTrust++ [25] in terms of achieved trust score of the network in an IIoT-based scenario. EigenTrust formulates the global trust of the network based on the transitive local trust score of the client nodes. On the other hand, the calculation mechanism of ServiceTrust and its extended version ServiceTrust++ is based on uniform trust propagation of the network coupled with trust decay to deal with related malicious nodes.

Fig. 10 presents the performance comparison of CTES with EigenTrust, ServiceTrust, and ServiceTrust++ in terms of trust score. The network consists of 70 nodes, and there exist 20 random nodes that are acting as malicious nodes. In the network, 4 nodes are considered as pre-trusted nodes. The graph shows that initially all protocols try to converge at the same rate but with increased trust update cycles and incremented malicious nodes. CTES converges with more resilience. The comparison of CTES with EigenTrust shows that initially both schemes converge equally, but after around 10 trust update cycles, the trust bias gap of EigenTrust from the ground-truth value (which is the trust score of 0.81) widens. The improved resiliency and accuracy of CTES in comparison to EigenTrust is the result of dynamic adjustment of μ, which results in adjusting the trust value of malicious nodes possessing self-promoting and bad-mouthing attacks. Similarly, the comparison of CTES with ServiceTrust and ServiceTrust++ shows improved performance of CTES in breaking the chains of malicious nodes to achieve convergence, resilience, and accuracy against malicious nodes.

6. Conclusion and future work

In this paper, we have proposed and analyzed a distributed CTES model for trust calculation in smart objects, which allows the requesting object to trust the service provider in an IIoT environment. The direct observations of edge nodes (trustor) are represented between a range of $0-1$, which creates the flexibility between trustor and trustee to highlight actual observations. Since CTES is a context-based model, we have calculated the similarity index of existing context among neighboring nodes before taking indirect recommendations. This process filters out the dissimilar recommendations. The trustor only considers the recommendations from neighboring nodes if and only if they have been served by the same server under the same context. Furthermore, the dynamic weight is assigned to direct interactions and indirect recommendations to calculate the trust score. The verification of CTES is performed using formal methods, which shows that the system performs correctly. Finally, we have demonstrated the effectiveness of CTES in

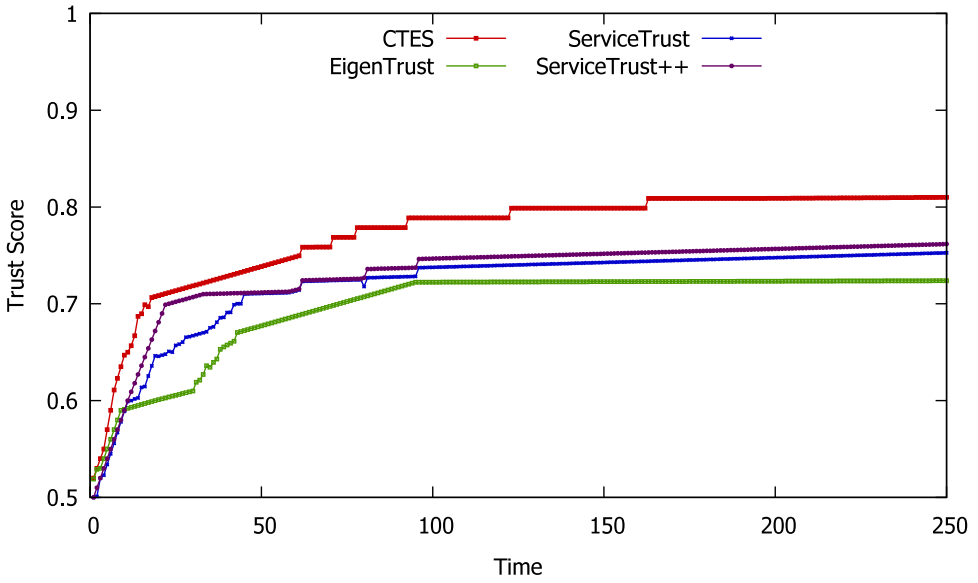

Fig. 10. Comparing CTES with EigenTrust, ServiceTrust, and ServiceTrust++ in terms of Accuracy and Resiliency.

terms of accuracy, dynamic assignment of weights, and prevention from malicious attacks, e.g., ballot stuffing and bad-mouthing. The results show that CTES performed well under a malicious environment and filtered out bad recommenders by providing maximum achievable performance. In the future, CTES will be further evaluated considering different applications for service compositions to filter out the service-oriented attacks.

Some of the limitations of the proposed study are as follows: Currently, we are formalizing the trust of serving nodes based on their type of service and location. We have not considered other contextual parameters of servers, which can be included in the future work. These include battery life, server's status as static or mobile, network's capability in terms of delay, response time and/or throughput, etc. Besides, the formulated trust of the server node is based on the interaction of client nodes and the recommendations of neighboring nodes. This study does not consider trust calculation of client nodes, which can be an interesting future direction.

CRediT authorship contribution statement

Ayesha Altaf: Conceive and designed the analysis, Methodology, Software (Matlab), Contributed data, Analysis tools, Data curation, Writing and documentation. **Haider Abbas:** Conceive and designed the analysis, Contributed data, Investigation of methodology and results, Data curation, Writing - review & editing. **Faiza Iqbal:** Methodology, Investigation of methodology and results, Writing and documentation, Writing - review & editing. **Farrukh Aslam Khan:** Contributed data, Investigation of methodology and results. **Saddaf Rubab:** Analysis tools, Writing and documentation. **Abdelouahid Derhab:** Writing and documentation, Writing - review & editing.

Acknowledgments

The authors extend their sincere appreciation to the Deanship of Scientific Research at King Saud University, Saudi Arabia, for funding this work through the Research Group No. RGP-214. This research is also supported by the Higher Education Commission (HEC), Pakistan through its initiative of National Center for Cyber Security for the affiliated lab National Cyber Security Auditing and Evaluation Lab (NCSAEL), Grant No: 2(1078)/HEC/M&E/2018/707.

References

[1] Pereira AC, Romero F. A review of the meanings and the implications of the Industry 4.0 concept. Procedia Manuf 2017. http://dx.doi.org/10.1016/j.promfg.2017.09.032, URL http://www.sciencedirect.com/science/article/pii/S2351978917306649, Manufacturing Engineering Society International Conference 2017, MESIC 2017, 28-30 June 2017, Vigo (Pontevedra), Spain.

[2] Altaf Ayesha, Abbas Haider, Iqbal Faiza, Derhab Abdelouahid. Trust models of internet of smart things: A survey, open issues, and future directions. J Netw Comput Appl 2019;137:93–111. http://dx.doi.org/10.1016/j.jnca.2019.02.024, URL http://www.sciencedirect.com/science/article/pii/S1084804519300839.

[3] Sfar Arbia Riahi, Natalizio Enrico, Challal Yacine, Chtourou Zied. A roadmap for security challenges in the Internet of Things. Digit Commun Netw 2018;4(2):118–37. http://dx.doi.org/10.1016/j.dcan.2017.04.003, URL http://www.sciencedirect.com/science/article/pii/S2352864817300214.

[4] Iarovyi Sergii, Martinez Lastra Jose Luis, Haber Guerra Rodolfo Elias, del Toro Raúl. From artificial cognitive systems and open architectures to cognitive manufacturing systems. 2015, http://dx.doi.org/10.1109/INDIN.2015.7281910.

[5] Moghaddam Mohsen, Cadavid Marissa N, Kenley C Robert, Deshmukh Abhijit V. Reference architectures for smart manufacturing: A critical review. J Manuf Syst 2018. http://dx.doi.org/10.1016/j.jmsy.2018.10.006, URL http://www.sciencedirect.com/science/article/pii/S0278612518301043.
[6] Lu Hsi-Peng, Weng Chien-I. Smart manufacturing technology, market maturity analysis and technology roadmap in the computer and electronic product manufacturing industry. Technol Forecast Soc Change 2018;133:85–94. http://dx.doi.org/10.1016/j.techfore.2018.03.005, URL http://www.sciencedirect.com/science/article/pii/S0040162517311514.
[7] Mittal Sameer, Khan Muztoba Ahmad, Romero David, Wuest Thorsten. A critical review of smart manufacturing & Industry 4.0 maturity models: Implications for small and medium-sized enterprises (SMEs). J Manuf Syst 2018. http://dx.doi.org/10.1016/j.jmsy.2018.10.005, URL http://www.sciencedirect.com/science/article/pii/S0278612518301341.
[8] Felix Mannhardt, Petersen, Oliveira Sobah Abbas, Fradinho Manuel. A trust and privacy framework for smart manufacturing environments. J Ambient Intell Smart Environ 2019. http://dx.doi.org/10.3233/AIS-190521.
[9] Jeong S, Na W, Kim J, Cho S. Internet of things for smart manufacturing system: Trust issues in resource allocation. IEEE Internet Things J 2018.
[10] Wang T, Luo H, Jia W, Liu A, Xie M. MTES: An intelligent trust evaluation scheme in sensor-cloud-enabled industrial internet of things. IEEE Trans Ind Inf 2020.
[11] Bao Fenye, Chen Ing-Ray. Dynamic trust management for internet of things applications. In: Proceedings of the 2012 international workshop on self-aware internet of things. New York, NY, USA: ACM; 2012, p. 1–6. http://dx.doi.org/10.1145/2378023.2378025, URL http://doi.acm.org/10.1145/2378023.2378025.
[12] Altaf Ayesha, Abbas Haider, Iqbal Faiza, Khan Malik Muhammad Zaki Murtaza, Rauf Abdul, Kanwal Tehsin. Mitigating service-oriented attacks using context-based trust for smart cities in IoT networks. J Syst Archit 2021 115:102028. http://dx.doi.org/10.1016/j.sysarc.2021.102028.
[13] Caminha Jean, Perkusich Angelo, Perkusich Mirko. A smart trust management method to detect on-off attacks in the internet of things. Secur Commun Netw 2018;2018:10, URL https://doi.org/10.1155/2018/6063456.
[14] Truong Nguyen B, Jayasinghe Upul, Um Tai-Won, Lee Gyu Myoung. A survey on trust computation in the internet of things. J Korean Inst Commun Sci 2016;33(2):10–27.
[15] Chen I, Guo J, Bao F. Trust management for SOA-based IoT and its application to service composition. IEEE Trans Serv Comput 2016;9(3):482–95. http://dx.doi.org/10.1109/TSC.2014.2365797.
[16] Rafey SEA, Abdel-Hamid A, El-Nasr MA. CBSTM-IoT: Context-based social trust model for the Internet of Things. In: 2016 International conference on selected topics in mobile wireless networking. 2016, p. 1–8. http://dx.doi.org/10.1109/MoWNet.2016.7496623, April.
[17] Saied Yosra Ben, Olivereau Alexis, Zeghlache Djamal, Laurent Maryline. Trust management system design for the Internet of Things: A context-aware and multi-service approach. Comput Secur 2013;39:351–65. http://dx.doi.org/10.1016/j.cose.2013.09.001, URL http://www.sciencedirect.com/science/article/pii/S0167404813001302.
[18] Lin Qin, Ren Dewang. Quantitative trust assessment method based on Bayesian network. In: 2016 IEEE advanced information management, communicates, electronic and automation control conference. 2016, p. 1861–4. http://dx.doi.org/10.1109/IMCEC.2016.7867540, Oct.
[19] Altaf A, Abbas H, Iqbal F. Context based trust formation using direct user-experience in the internet of things(IoT). In: 2019 IEEE international conference on cloud computing technology and science. 2019. p. 424–30.
[20] Marsh Stephen Paul. Formalising trust as a computational concept. Technical report, 1994.
[21] Fernandez-Gago Carmen, Agudo Isaac, Lopez Javier. Building trust from context similarity measures. Comput Stand Interfaces 2014;36(4):792–800. http://dx.doi.org/10.1016/j.csi.2013.12.012, URL http://www.sciencedirect.com/science/article/pii/S0920548913001864, Security in Information Systems: Advances and new Challenges.
[22] Altaf A, Abbas H, Iqbal F, Khan MMZM, Daneshmand M. Robust, secure and adaptive trust-oriented service selection in IoT-based smart buildings. IEEE Internet Things J 2020;1. http://dx.doi.org/10.1109/JIOT.2020.3040775.
[23] Kamvar Sepandar D, Schlosser Mario T, Garcia-Molina Hector. The eigentrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th international conference on world wide web. New York, NY, USA: Association for Computing Machinery; 2003, p. 640–51. http://dx.doi.org/10.1145/775152.775242.
[24] Su Z, Liu L, Li M, Fan X, Zhou Y. ServiceTrust: Trust management in service provision networks. In: 2013 IEEE international conference on services computing. 2013. p. 272–79.
[25] Su Zhiyuan, Liu Ling, Li Mingchu, Fan Xinxin, Zhou Yang. Reliable and resilient trust management in distributed service provision networks. ACM Trans Web 2015;9(3). http://dx.doi.org/10.1145/2754934.

Ayesha Altaf did her B.S. in Computer Science from COMSATS Lahore in 2006 and M.S. Degree in Information Security from National University of Sciences and Technology (NUST), Pakistan in 2009. She is continuing her PhD Degree in Information Security from NUST, Pakistan. Her research interests include Internet of Things security, Trust Modeling and Information Security.

Haider Abbas is currently heading the National Cyber Security Auditing and Evaluation Lab (NCSAEL) at MCS NUST. He is a Cyber Security professional, researcher and industry consultant who took professional trainings and certifications from (MIT), USA, Sweden, IBM, USA and EC-Council. He received his MS in Engineering and MIS (2006) and PhD in Information Security (2010) from KTH-Stockholm, Sweden. He is also an adjunct faculty and doctoral studies advisor at Florida Institute of Technology, USA and Manchester Metropolitan University, United Kingdom.

Faiza Iqbal received her M.S. and Ph.D. degrees in software engineering from NUST, Pakistan, in 2009 and 2015 respectively. She has been associated with Quaid-i-Azam University, Islamabad as Assistant Professor. Currently, she is working as Assistant Professor at The University of Lahore (UoL), Lahore. Her current research interests are knowledge based systems, network optimization modeling, and high performance protocol design.

Farrukh Aslam Khan is currently working as a Professor at the Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh, Saudi Arabia. He has over 20 years of teaching and research experience at various institutions. His research interests include Cyber Security, Wireless Sensor Networks and e-Health, Smart Grid, Bio-inspired and Evolutionary Computation, and the Internet of Things.

Saddaf Rubab is an experienced researcher with a demonstrated history of working in the higher education industry. Skilled in Data Science, Artificial Intelligence, Cognitive Science, Grid Computing, and Operations Research. Strong research professional with a Doctor of Philosophy (Ph.D.) focused in Information Technology from Universiti Teknologi PETRONAS, Malaysia.

Abdelouahid Derhab received the Engineer, Master, and Ph.D. degrees in computer science from University of Sciences and Technology Houari Boummediene (USTHB), Algiers, in 2001, 2003, and 2007 respectively. His research interests include: Mobile Adhoc Networks (MANETs), Wireless Sensor Networks (WSNs), Intrusion detection systems, Botnet detection, and Mobile security.