A two-way trust management system for fog computing

Esubalew Alemneh, Sidi-Mohammed Senouci, Philippe Brunet, Tesfa Tegegne

▶ **To cite this version:**

Esubalew Alemneh, Sidi-Mohammed Senouci, Philippe Brunet, Tesfa Tegegne. A two-way trust management system for fog computing. Future Generation Computer Systems, 2020, 106, pp.206-220. 10.1016/j.future.2019.12.045 . hal-02885815

HAL Id: hal-02885815
https://hal.science/hal-02885815

Submitted on 19 Oct 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

A two-way trust management system for fog computing

Esubalew Alemneh [a,b,*], Sidi-Mohammed Senouci [a], Philippe Brunet [a], Tesfa Tegegne [b]

[a] DRIVE EA1859, Univ. Bourgogne Franche-Comt, F58000, Nevers, France
[b] Bahir Dar University, Bahir Dar Institute of Technology, Faculty of Computing, P.O. Box 26, Bahir Dar, Ethiopia

Fog computing is the next frontier of cloud computing since it can compute and store a massive amount of data generated by IoT devices near their sources. Indeed, transmitting all these data to the cloud will take up a huge amount of bandwidth. However, its features and flexibility of deployment make fog computing vulnerable to security and privacy attacks. The high-mobility support, dynamic environment, geographical distribution, location awareness, proximity to end users, and lack of redundancy are among these features that make the existing schemes not adequate to fog computing. Therefore, since trust management ensures security and privacy, we propose a two-way subjective logic-based trust management system that enables a service requester to verify whether a service provider can give reliable and secure services and lets the service provider check the trustworthiness of the service requester. Extensive evaluation of the system shows that the trust value of a node is accurate and converges in a very few trust computation cycles. The solution is also resilient to a large population of misbehaving nodes and it is able to thwart trust-based attacks successfully. Moreover, comparative analysis is made with different modification of the system and two baseline schemes, PeerTrust and EigenTrust. The comparison depicts that the two-way trust management proposals have lower overhead, allow a balanced load distribution, are effective in selecting the right service providers and are more accurate than the conventional one-way trust management systems.

1. Introduction

An immense number of physical devices all over the world are connected to the Internet creating the Internet of Things (IoT) paradigm. According to McKinsey [1], one trillion IoT devices are expected to be deployed by 2025. McKinsey also estimated the potential economic impact of IoT to dash to 11 trillion USD per year accounting 11% of the world economy by the same year. The IoT devices generate an unheard-of volume and variety of data resulting in big data. For instance, an electronic health track record system of a patient contains various wearable devices and sensors that produce an ocean of distinct types of data. Cloud computing's high-performance and storage capacity leverage processing and reposition of these data. However, transporting the data to far-situated cloud servers takes up large bandwidth. Hence, cloud computing is not suitable for real-time analytics and decision making. The aforementioned problems are alleviated by bringing data processing and storage down to the proximity of data production sites. The viability of the solution is founded on the fact that present-day edge devices processing, storage and networking capacity improves from time to time. One of such solutions is fog computing aka *fogging*.

Fog computing is a distributed computing paradigm that stretches computing, storage, communication, and networking services down to the fringes of the network to reduce latency, decrease bandwidth, and increase reliability [2]. The main components of a fog computing architecture are fog nodes which may be fog servers (service providers) or fog clients (service requesters) [3,4]. Fog servers can be set-top-boxes, access points, road-side units, cellular base stations, gateways, routers etc., while end devices like smartphones, sensors, smart watches, vehicles, cameras, etc., represent the fog clients [3]. Fog computing, which complements cloud computing rather than replacing it, supports mobility, geographical distribution, location awareness, heterogeneity, interoperability, and federation. By exploiting the characteristics of fog computing, many novel IoT applications and services have been suggested. Traffic safety [5], e-Health [6], web content delivery [7], augmented reality [8], and big data analysis [9] are some of the applications that suits fog computing. Moreover, establishment of consortiums like OpenFog [10] by hi-tech giant companies and renowned academic institutions to define an architecture for fog computing and build operational models and testbeds indicates wide acceptance of the computing paradigm.

* Corresponding author at: DRIVE EA1859, Univ. Bourgogne Franche-Comt, F58000, Nevers, France.
E-mail address: esubalew@gmail.com (E. Alemneh).

However, due to its features and flexibility of deployment, fog computing is highly susceptible to information security and user privacy violations [3,11–14]. High mobility support, dynamic environment, geographical distribution, location awareness, proximity to end users, and lack of redundancy are among characteristics of fog computing which have negative impacts on its security and privacy despite their proven merits. Fog computing confronts new privacy and security challenges in addition to those assumed from cloud computing [3]. Existing privacy and security remedies for cloud computing cannot be applied directly to fog computing as the architectures of the two computing paradigms are quite dissimilar [11,14]. Cloud computing is centralized where imposing security is relatively simpler because of the centralized component in comparison with distributed architectures like fog computing. Apt measures need to be taken on security and privacy issues of fog computing to maintain its pace of development and acceptance in academia and industry.

Another challenge on the dynamic fog computing environment, which is much related to security and privacy, is trust management. Trust is defined as the level of assurance that an object will behave satisfactorily [15]. The behavior pertains to the Quality of Service (QoS) or the security policies that the object has to possess. Thus, a trustworthy object conforms to QoS requirements without violating any security policy. The level of assurance depends on the deployment environment, the type of network application and the required level of security. For distributed environments like fog computing and for safety and security critical network application like traffic safety or health applications, a high degree of trust is needed. On the contrary, a low level of trust is demanded for centralized architectures for which other security mechanisms can be easily imposed and for applications whose reliability is not a priority [16]. Devices may encounter other strange devices in the network and the interaction with them should be carried out in caution since there is uncertainty on their behavior. The purpose of trust management systems in fog computing is to detect and deter bad fog servers and bad fog clients. A bad fog node is any malicious or rogue fog node which acts like a legitimate node nevertheless it is a compromised or replaced with fake one by intruders or malicious users [3]. A bad fog server may surreptitiously gather user data, provide wrong services to the clients and/or launch attacks. In case of fog clients, once such nodes get connected to the server, they may collude with other nodes to have high trust value (enabling it be accepted by any server) or may generate different types of attacks [17]. To predict the future trustiness of an entity and avoid any uncertainty or risk about an entity, trust management systems gather information about the entity from direct observations and recommendations of other entities. That is why Jin-Hee Cho et al. formulated trust management as one way of risk mitigation techniques which involves trust establishment, trust update and trust revocation tasks [18].

There are two entities in trust management: trustor and trustee. The trustor is an entity that puts faith on the other entity i.e., trustee. Trust can be delineated as trustors belief in trustees capabilities, honesty, reliability etc., [19]. Trust is directional in that trustfulness of trustee does not depict whether the trustor itself is trusty or not. Moreover, trust is subjective meaning that what is trustworthy for one entity may not be trustworthy for the other [20]. Trust management enables objects in a network to determine the level of trustworthiness of another object. In other words, it provides a mechanism to decide whether to put faith in an entity to which a communication is going to be established. It allows detection of damaged or misbehaving nodes and enables autonomous communication among entities in a network [16,21]. Trust is crucial for creating interaction in an uncertain environment [22]. Trust ensures information security and user privacy and it is also related to reliability, integrity, dependability, and the ability to provide the right services [21]. In this research, we have proposed a subjective logic based two-way trust management (TTM) system where trustor and trustee evaluate each other by exchanging their trust computation role to create a trusted data communication. The direct trust acquired from self-observation and indirect trust obtained from recommendations of neighboring nodes are used to determine the final trust value.

Trust computation enables to calculate trust value of a target entity dynamically. An effective trust computation method has five design dimensions: trust composition, trust propagation, trust aggregation, trust update, and trust formation [23]. Trust composition determines the information required for trust computation. This information can be QoS and/or social relationship information [24]. QoS trust refers to the belief of the trustor that the trustee can provide a service with the desired quality [25]. Since, social relationships among human beings are also reflected by devices they own [26,27], it can be used for trust computation. Trust propagation determines how trust values are stored and calculated. It can be either centralized or distributed. Trust update decides how often will the trust values of entities be updated. Trusts can be updated in event-driven and/or time-driven fashion. Trust formation describes how to combine the trust properties determined by trust composition. Trust can be formed from either a single trust property or multiple trust properties. Trust aggregation decides how to integrate trust evidences from different recommenders and from own experience. Weighted sum, Bayesian inference, fuzzy logic, subjective logic, and regression analysis are some examples of trust aggregation techniques. The proposed subjective logic-based bidirectional trust management system is distributed and event-driven trust management which uses both QoS and social trust information to calculate the trust values of fog nodes.

Even though trust management is a crucial and hot topic, due to flexibility of deployment fog nodes it is a challenging problem to enforce in fog environment. Fog nodes can be from different providers, they may be owned, operated and maintained by different individuals or providers independently and new nodes can join or leave the network anytime [3]. Geo-distribution and proximity of fogging to end users imply that the nodes are easily accessible making them susceptible to corruption and rogue node built up by adversaries. Lack of redundancy, dynamicity, high mobility support, and low processing power of nodes [12] are among other properties of fog computing that adds complication to trust management in fog computing. Due to these reasons, fog servers are potential threats to not only fog clients but also to other fog servers. The same is true from fog clients perspective.

In this paper, we present a novel two-way scalable and efficient trust management solution which aggregates trust using a specific version of belief theory called subjective logic [28]. Subjective logic is a kind of probabilistic logic that explicitly takes uncertainty and source trust into account. In general, subjective logic is suitable for modeling and analyzing situations or propositions involving uncertainty and relatively unreliable sources. The proposition is expressed as a probability in the range of 0 to 1. Besides its power to express uncertainty, subjective logic is helpful to aggregate the truth values of propositions that form a general and objective belief [29]. Section 4 deals with details of subjective logic concept. In fog networks trust should work in two-ways so that fog clients can verify fog servers that they can provide the right, reliable and secure services. On the reverse, fog servers must be able to check the legitimacy and roguery of fog clients. The contributions of this work are:

- We proposed a peer-to-peer two-way subjective logic-based trust management system that allows service requesters to check the trustworthiness of service providers and service providers to verify service requesters are the genuine ones. The distributed and event-driven trust management system considers both QoS and social trust metrics to determine the trust of a fog node. Final trust value is calculated by dynamically combining information obtained from self-observation and recommendations of neighboring nodes.
- We demonstrate the accuracy, convergence, and resilience of the solution by conducting an extensive evaluation using a simulation tool developed for this purpose. The evaluations made include the effect of the weight of direct and indirect trust values on the final trust values and the effect of the number of malicious or bad nodes in the network.
- We produced different derivatives of the trust management system and made a comparative analysis on the effectiveness of selecting the right service providers among a set of good and bad servers. We have also demonstrated through experiment that two-directional trust management surpasses one-way trust management systems in opting more trusted service providers. Moreover, we have compared our trust management systems with two baseline schemes, PeerTrust and EigenTrust. The result shows that the two-way trust management systems have superiority in terms of trust convergence and accuracy.

The remainder of the paper is organized as follows. Section 2 discusses related works in trust management in fog computing and other related computing environments. The system model considered and trust metrics comprised in the trust management system are discussed in Section 3. The proposed subjective logic-based trust management system is explained in Section 4. Performance evaluations conducted are presented in Section 5. Finally, Section 6 concludes the paper.

2. Related work

Because of limited works on trust management in fog computing and similarities of the computing paradigm to IoT systems and cloud computing, selected related researches in these domains are reviewed. Many trust management mechanisms are introduced for IoT systems [21,23,30–36], and cloud computing [37–44]. These mechanisms allow to select trusted nodes for reliable and secure communications and take single or multiple QoS and social trust metrics [30,31]; the QoS trust metrics being more studied than social trust metrics [23].

In [34] a fuzzy reputation-based trust management system that considers only QoS trust metrics is presented. The final trust is computed from trust information obtained from direct observation and from recommended indirect trusts. The main drawback of this work is ignorance of social relationships among devices in the Internet. A series of works by Bao et al. [30–33] emphasized on social relationships among IoT devices to define trust management systems for IoT applications. Trusts are calculated from information obtained from both direct observation and opinions of other nodes based on trust metrics like honesty, cooperativeness, Community of Interest (COI), friendship etc. Their solutions are evaluated mainly for trust assessment accuracy and convergence. [33] focuses on addressing the problem of misbehaving nodes whose characters may change over time. A scalable, adaptive and survivable trust management system is presented in [32]. Scalability of the trust management system is achieved by keeping trust information of the subset of nodes encountered using a storage strategy defined by them. The main contribution of [30] is the introduction of a novel adaptive filtering technique to determine the best way to combine direct and indirect trusts so that convergence time and trust estimation bias are minimized. More recent works on trust management on Social Internet of Things (SIoT) include [35,36]. A recommendation and reputation-based trust computation model for distributed SIoT networks which is able to converge in few iterations is discussed in [35]. However, the method depends solely on social trust metrics and final trust scores do not include knowledge from direct observation in the trust computation. A context-aware trust management system for SIoT is proposed in [35] to prevent attacks of malicious nodes which acts dishonestly based on a context. Three trust contexts are identified using context-aware QoS and social similarity-based trust metrics are used to identify honest and dishonest devices effectively.

Trust in cloud computing enables service consumers to select a cloud service provider with desired reliability, quality, and performance. If the service consumer has no prior experience with a cloud service provider, it is challenging to put faith in the service offered [37]. Trust is vital for fast adaption and growth of cloud computing [36,38]. However, non-transparent nature of cloud services has made trust management in a cloud environment challenging [36]. Yet there are many published works on recommendation, prediction, reputation and policy-based trust management in cloud computing [40]. Most of these works rely on verification of Service Level Agreement (SLA) [41] and QoS attribute information [42]. There are also some works that consider identity and interaction history in trust computation. A behavior graph and service grouping based trust evaluation method that encompasses relationship parameters such as identity, and interaction evolution and service quality attributes such as availability and reliability are proposed in [43]. Other trust metrics that depict social relationships like honesty and sincerity can also be used in trust computation for a cloud computing environment. Ing-Ray Chen et al. proposed a scalable trust protocol which depends on social trust metrics for mobile cloud IoT systems [44]. The protocol named IoT-HiTrust allows IoT devices report their experiences and query the service trustworthiness of other IoT device through cloudlets.

Trust management in fog environment is different from trust management in cloud environment in various ways from their architecture to ways of deployment. The distributed nature of fog architecture complicates trust computation because of lack of a global centralized entity that enables to impose traditional security mechanisms like authentication and access control to allow secure and trusted communication [11,45]. Secondly, mobility support, location awareness, a huge number of nodes, the low processing power of nodes are among the features of fog computing that pushes to strive for dynamic, scalable, and computationally efficient trust management system. Thirdly, the flexibility of deployment of fog computing makes fog environments more vulnerable to trust-based attacks [3]. Trust in cloud environments is more-or-less unidirectional. In contrary, the two-way requirement of trust is another issue that makes trust issue a formidable challenge in fog computing [11]. Fog nodes that provide services must be able to evaluate the trust level of nodes that request services and service requestors must also able to check whether to depend on service providers.

Hence, trust is one of the issues that have to be addressed to boost the acceptance of fog computing in industries [46]. However, little work is done on trust management in fog computing [47]. Most of the works that deal with trust in fog computing merely affirm imperativeness of trust management in the environment. We found only a handful of works that suggest particular methods for trust computation in fog computing till end of 2018. Rahman et al. in [45] identified a fuzzy logic configuration that affects trust values of a fog node. Distance, latency,

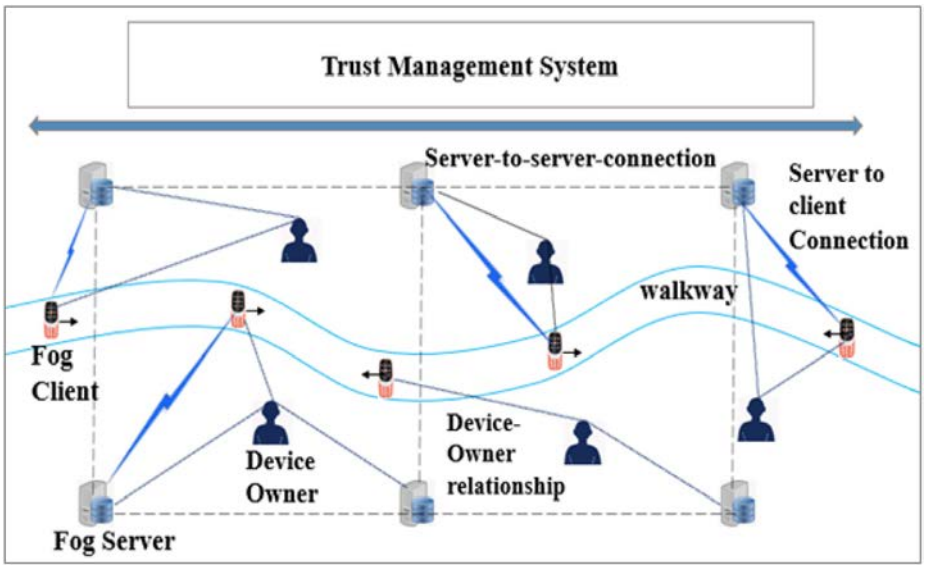


Fig. 1. System model of the two-way trust management system.

and reliability are trust metrics considered in the configuration. The work provides some insights to the definitions of trust and advantage of fuzzy logic for trust evaluation for fog computing. The same authors as in [45] extended their work to propose a broker-based trust evaluation framework for fog service allocation [48]. However, the proposed work conceives only QoS trust metrics and it is unidirectional. In addendum usage of broker implies malfunctioning of the broker results in a complete cessation of the trust evaluation framework. A summary of some related trust management works in IoT, cloud computing and fog computing research domains from the five trust dimensions of trust computation perspective are presented in Table 1.

Our work is different from the aforementioned ones. Firstly, it is a two-way trust computation where fog server checks trustworthiness of fog clients and fog clients checks back if the server is fit to provide the services. Secondly, we have considered QoS trust information besides considering social relationship information among nodes. The trust management system relies on both self-observations and recommendations from neighboring nodes which are combined adaptively. Thirdly, our bidirectional trust management solution does not depend on any third-party component. Moreover, multiple recommendation trusts are assembled using a trust aggregation technique called subjective logic. It is a kind of a belief theory and suggested to be most appropriate for fog computing [16,23].

3. System model

The two-way trust management system is based on a simplified fog computing environment whose system model is shown in Fig. 1.

Though multi-layered fog environment can be considered [49], without losing generality we have conceived single-layered one. We assume a fog node is a single static device though it may comprise of multiple devices integrated as one fog node [4]. Fog server can communicate with neighboring 1-hop fog nodes i.e., fog servers and fog clients. A fog client in fog computing can be user carried devices like smartphones, laptop, other computers or non-user accompanied devices like smart lighting, smart washing machine, CCTV etc. [50]. This research is about the first group of devices. Fog clients are mobile on a predefined trajectory. Fog clients can communicate with other neighboring fog servers. Each fog node has an owner and a person may own more than one node. The high-level description on how the proposed system works, deployment example and trust-related information considered to compute trust levels of fog nodes are presented in what follows.

3.1. How it works?

A fog client intending to get a service, requests a fog server for a connection. The fog server then wants to ensure that it is connected to a trusted (non-rogue) fog client. Therefore, to capture a malicious or bad fog client, the server calculates the trust value of the client by consulting neighboring servers and from its direct observation. Detected fraudulent clients will then be refused for the service and its trust value will be stored to monitor the node in the future. The value will also be sent to servers that request for trust level of the client as recommendations. A fog client which is allowed to connect to the server, in its turn wants to make sure that the fog server is trustable and can give the right service. A malicious fog server may give wrong or incorrect service. The fog client consults neighboring fog servers and adaptively combines with its direct observation to determine the final trust status of the server. Just like fog servers, fog clients also propagate their experience about the server to other nearby servers. Trust management systems do not require disseminating trust information over the entire network [51]. Therefore, nodes only keep and exchange trust information about neighboring nodes within the radio range for computational efficiency.

The sequence of communication between fog client and fog server in order to create a trusted connection is depicted in Fig. 2. The conversation is summarized as follows. A fog client sends a connection request to a fog server. The server evaluates the trust value of the client and allows the connection if the client is trustworthy; otherwise, it refuses the connection. If the client is granted connection, it evaluates the trust value of the server and establishes connection if the server is trustworthy. If the server refuses the connection or the server is found to be untrustworthy the client sends a connection request to another server. This conversation continues until a trusted service provider is found among neighboring fog servers.

Table 1
Summary of related trust management systems from the five dimensions of trust computation.

Research domain	IOT	Cloud computing	Fog computing
Trust composition	Social trust [30–32,35], QoS trust [34] and both [33,36]	QoS trust [42,43], QoS from SLA [41]	QoS including mobility and distance [48]
Trust propagation	Distributed [30–36]	Centralized [41–43]	Distributed [48]
Trust update	Time-driven [30,33,34] & event-driven [30,32]	Event-driven [41–43]	Event-driven [48]
Trust formation	Multi-trust [30–33] & single-trust [34,35]	Multi-trust [41–43]	Multi-trust [48]
Trust aggregation	Bayesian systems [30,32], weighted sum [30,33,35,36], fuzzy-logic [34]	Weighted sum [41–43]	Fuzzy-logic [48]

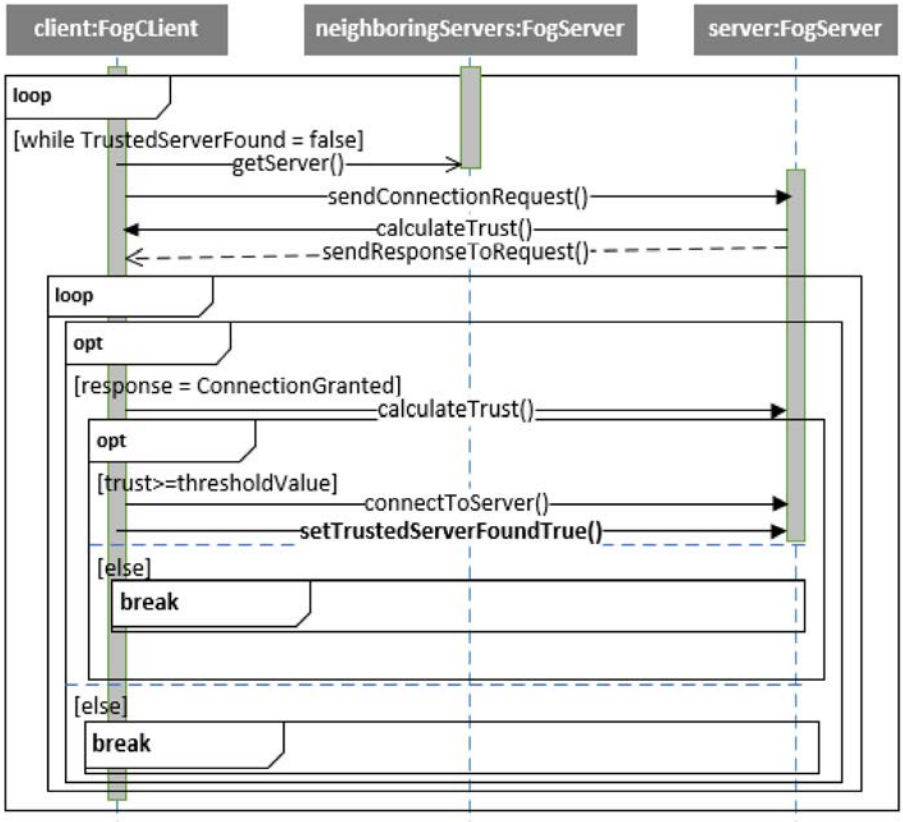


Fig. 2. Dialog between fog clients and fog servers.

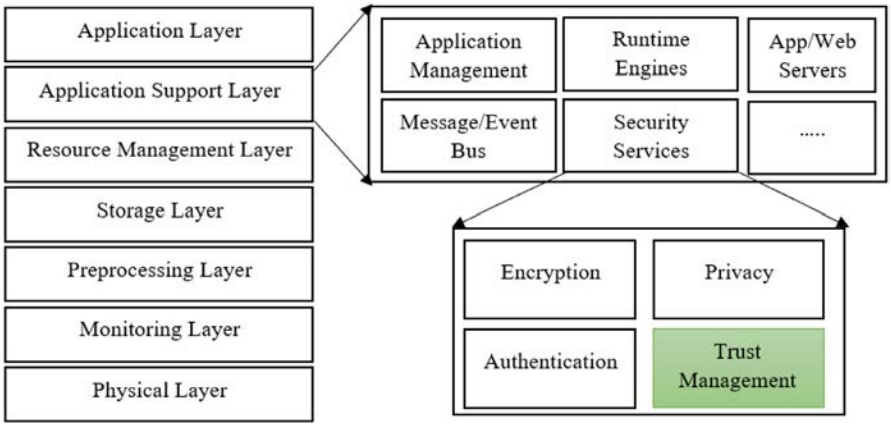


Fig. 3. Trust management in fog computing architecture.

3.2. Deployment example

In the reference architecture proposed by OpenFog [52] and other fog computing architectures [53], the proposed trust management system is deployed on top of resource management layer in application support layer which encompasses security services, see Fig. 3. Predominance of wireless connection in fog environment means security problems are very crucial in fogging. That is why security is considered as one of the main components of fog computing architecture [53]. Apart from trust management issues all security-related matters including encryption, privacy, authentication, intrusion detection and prevention, etc. are handled by this component. Trust management systems enables to establish trusted connections among nodes in fog environment.

The proposed trust management algorithm runs under applications that require trusted communications. For instance, let us take a traffic safety application for Vulnerable Road Users (VRU) proposed in [5]. In this application drivers and VRUs, which could be pedestrians, cyclists and powered two-wheelers send their geolocation and other associated data periodically to fog servers for collision risk prediction. The server predicts any imminent collisions and sends warning messages to both drivers and VRUs. In this type of systems, a bad fog server may provide wrong traffic accident prediction and a bad fog client may send wrong locations of VRUs to the servers [54]. In both cases, the road users will be in the jeopardy of traffic accidents. The trust management algorithms can be used to identify trusted fog servers and clients. Once the fog nodes are found to be trusted, VRUs and drivers can send cooperative awareness messages data for collision risk prediction to the servers. The servers also entertain clients which are identified as trustworthy by the trust management algorithm. Hence, the trust management system helps nodes check their trustworthiness each other before the actual service provisions.

3.3. Trust metrics

A trust metric or property is information needed to calculate the trust level of a node. In the trust management system, more than one trust properties are used to evaluate the trust of fog servers and clients. Choice of trust property depends on the issue the trustor is interested in [22,25,37]. QoS trust evaluates the capability of a fog server to successfully complete a requested mission. Since fog clients interest is to select a server that can provide the service properly more QoS are used and selection of clients by servers largely relies on social relationships. In the two-way trust management system proposed, the metrics used by fog servers to evaluate trustiness of fog clients are friendship, honesty, and ownership, while fog clients use latency, Packet Delivery Ratio (PDR) and ownership to evaluate trust value of a server. The definition of the metrics and how they are calculated are explained underneath:

- *Latency*: is the time required by a fog server to provide a service to a fog client. High latency and irregularities in response time predict possible intrusions in the system [55]. The value is returned from the latency response-time model produced using log-normal distribution [56] with the mean and standard deviation of 60 ms and 20 ms, respectively. The mean and standard deviation is taken from our previous work on fog computing-based traffic safety architecture for VRUs [5].
- *Packet Delivery Ratio*: is the ratio of packets successfully received to the total sent. It is the ratio between the number of packets received by the application layer of destination nodes to the number of packets sent by the application layer of the source node. In our solution, PDR is modeled using a well-known packet loss model called Gilbert–Elliott model [57]. The parameters of the model are obtained from packet loss observation of links of a good and bad node [58]. Hence, the probability of transferring from good state to the bad state, and the probability of transferring from the bad state to the good state are generated based on the PDR result of fog computing environment as experimented in [5].
- *Ownership*: each fog node has an owner. This metric is included with the assumption that devices owned by the same person are trustable each other [26]. Therefore, if a node encounters another node owned by the same person, the trust value is set to one; otherwise, it is zero.
- *Friendship*: refers degree of closeness of a node in comparison with other nodes. Instead of defining it initially in friendship matrix like [30,59], calculation of friendship in our case relies on interaction history. It follows the maturity model proposed in [51] in that the more positive interaction experiences between two nodes implies more trust and confidence between them. Friendship is calculated as the ratio of the number of successful connection requests of a client to the maximum connection of all requests. A connection request is said to be successful if the request is accepted by the server. A server accepts clients connection request if the clients trust is above the required application dependent threshold value.
- *Honesty*: evaluates the belief that a node is dependable based on another nodes direct observation over a given period of time [30,33,59]. It is calculated by keeping a count of suspicious dishonest experiences of a trustee node as observed by trustor node during a time period using a set of anomaly detection rules such as a high discrepancy in recommendation, as well as interval, retransmission, repetition, and delay rules. Hence, we figured out honesty as the ratio of valid trust propagations and realized connection requests. Realized connection requests are those resulted in a trusted connection between trustor and trustee. Exaggerated fog clients recommendations are conceived as invalid propagations and connection requests from nodes of low trust values are rejected.

4. Subjective logic-based trust management system

Trust computation enables to calculate trust value of a target entity dynamically. If the trust value is acceptable, then trusted data communication follows; otherwise, entities abstain from untrusted interaction to other entities. To establish trust values, an effective trust computation method has five design dimensions [23]. As stated earlier, TTM system uses QoS and social trust metrics and it is distributed, event-driven, and multi-trust system. In this section, we discuss subjective logic and its application for trust aggregation, how final trust values of fog nodes are calculated from direct and indirect trusts, the detail of TTM algorithm and the justification for the resilience of the proposed system in thwarting trust-based attacks.

4.1. Subjective logic and trust computation

The TTM system staged in this paper uses subjective logic to aggregate recommendations from neighboring fog servers. Though subjective logic has the ability to defend trust-based attacks because of its discounting step, it is less explored [23]. In this section, we briefly introduce subjective logic and show how it is used in the trust management system proposed.

Standard logic is designed for an idealized world where propositions can be evidently either true or false [29]. However, in the

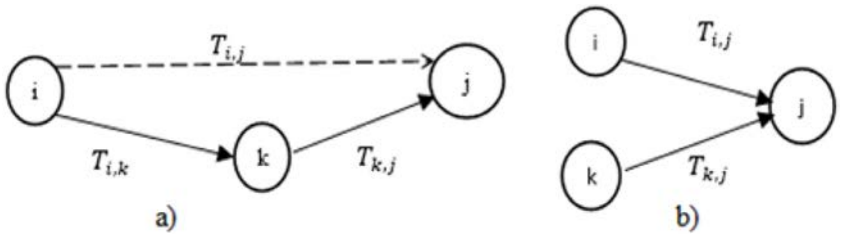


Fig. 4. (a) Discounting and (b) Consensus operators.

real world, nobody can be absolutely certain whether a proposition is true or false and the assessment of the proposition is individual i.e., not general and objective [60]. Therefore, many calculi and logic-based methods which consider uncertainty and ignorance have been proposed. These methods allow drawing conclusions about a proposition with insufficient evidence. Trust is one of such propositions since it is a statement or assertion that expresses a judgment or opinion about an object. Subjective logic which is a special form of belief theory builds on the belief that trust is subjective and it is differently experienced by everyone [61]. It is not practical for recommenders to consider all pertinent trust metrics to evaluate the trust value of a node. This implies that trust is computed with insufficient evidence and each node in a fog computing environment computes its own trust value subjectively for each node it encounters.

In subjective logic uncertain probabilities are represented with belief model as opinion. Opinion or degree of trust of a node x, ω_x, is defined with 4-tuples [28] as:

$$\omega_x = (b_x, d_x, u_x, a_x) \tag{1}$$

where b_x is the belief that the node is trustable, disbelief d_x denotes the doubt that the node is trustable, u_x is uncertainty to conclude that a node is trustable or not, and atomicity a_x is the prior probability of x without any evidence. If the value of atomicity is 0.5, an opinion has an equal probability of giving true or false output. Note that the sum of belief, disbelief, and uncertainty must be equal to 1. The degree of trust represented as a 4-tuple opinion can be converted into a single-valued trust value using the equation:

$$P(x) = b_x + a_x * u_x \tag{2}$$

To illustrate this using an example, suppose the probability of rain falling in Paris on 28 February 2018 is 0.4, the chance of no rain is 0.3, and the uncertainty of raining is 0.3. Assuming equal chance of giving a true or false output, this can be represented as opinion in the form $\omega_x = (0.4, 0.3, 0.3, 0.5)$ and uncertain probability is 0.4 + 0.5*0.3 = 0.55.

Now the challenge is how to get the values of the tuples of subjective trust from the interaction among fog nodes. To calculate the degree of trust of nodes in fog networking, the values of belief (b_x), disbelief (b_x) and uncertainty (u_x) of a node x can be obtained from its positive and negative experience [29]. If positive and negative experiences are denoted by p and n respectively, then the three variables can be calculated using the following set of equations:

$$\begin{aligned} b_x &= \frac{p}{n+p+1} \\ d_x &= \frac{n}{n+p+1} \\ u_x &= \frac{1}{n+p+1} \end{aligned} \tag{3}$$

Fog nodes count good and bad experiences of nodes they come across and forward these values as a subjective trust when they are asked for recommendations. Recommended trusts have to be weighted and combined to get the final trust values. There are two ways of combining trust recommendations in subjective logic; *discounting* and *consensus*. They are described hereafter.

Discounting. A node computing trust value of another node scales the recommendations it received using discounting (denoted by the operator $\oplus$) with the trust values the node has about the recommenders. Hence, trust values from trusted recommenders will have more weight than the less trusted ones. This is essential to defend trust-based attacks. Testimonial requester has trust values of the recommenders from previous communications. Suppose a trustor node i has a subjective trust value of a recommender node k as $T_{i,k} = (b_{i,k}, d_{i,k}, u_{i,k}, a_{i,k})$ and the recommender has trust value of trustee node j as subjective trust $T_{k,j} = (b_{k,j}, d_{k,j}, u_{k,j}, a_{k,j})$. See the visual representation of the statement in Fig. 4(a). Then the indirect trust of node j as evaluated by i based on ks recommendation is calculated as:

$$T_{i,j} = (b_{i,k}b_{k,j}, b_{i,k}d_{k,j}, d_{i,k} + u_{i,k} + b_{i,k}u_{k,j}, a_{k,j}) \tag{4}$$

Consensus. Recommendations from several recommenders are combined using consensus (denoted by operator $\otimes$). Suppose node i and k have recommendations about node j as $T_{i,j} = (b_{i,j}, d_{i,j}, u_{i,j}, a_{i,j})$ and $T_{k,j} = (b_{k,j}, d_{k,j}, u_{k,j}, a_{k,j})$, respectively, see Fig. 4(b). The combined recommendation for j is given by the following equation:

$$T_{ik,j} = (\frac{b_{i,j} * u_{k,j} + b_{k,j} * u_{i,j}}{k}, \frac{d_{i,j} * u_{k,j} + d_{k,j} * u_{i,j}}{k}, \frac{u_{i,j} * u_{k,i}}{k}, a_{ik,j}) \tag{5}$$

where, $k = u_{i,j} + u_{k,j} - u_{i,j} * u_{k,j}$ and $a_{ik,j} = \frac{a_{(i,j)} * u_{k,j} + a_{k,j} * u_{i,j} - (a_{i,j} + a_{k,j}) * u_{i,j} * u_{k,j}}{u_{i,j} + u_{k,j} - 2u_{i,j} * u_{k,j}}$.

Overall indirect trust is calculated by applying *discounting* and *consensus* together on the recommendations obtained from all neighboring 1-hop nodes as subjective trust. So as to get more reliable recommendations and to be more resistant to trust-based attacks, recommendations can be taken only from trusted recommenders by applying threshold-based filtering [25]. However, in the case of the proposed solution, as the number of neighboring nodes is limited all recommendations are taken and trust-based attacks are taken care of by the trust aggregation method. Suppose a node i has trust value of recommenders $r_1, r_2, \ldots, r_k$ at time t as $T_{i,r1}(t), T_{i,r2}(t), \ldots, T_{i,rk}(t)$ respectively, and the recommenders have trust towards node j at time t as $T_{r1,j}(t), T_{r2,j}(t), \ldots, T_{rk,k}(t)$, see Fig. 5, then by applying discounting and consensus operations the final cumulative indirect trust of node j as evaluated by i is calculated using Eq. (6).

$$\begin{aligned} T_{i,j}^{Indirect}(t) = &(T_{i,r1}(t) \otimes T_{r1,j}(t)) \\ &\oplus (T_{i,r2}(t) \otimes T_{r2,j}(t)) \oplus \cdots \oplus (T_{i,rk}(t) \otimes T_{rk,j}(t)) \end{aligned} \tag{6}$$

The trust management system relies on the trust value of a node calculated from direct observation in addition to recommendations. Trust metrics defined in Section 3.3 are used to calculate

Table 2
List of important notations for the algorithm.

Notation	Description
Clients $= C_1, C_2, \ldots, C_n$	List of fog clients
Servers $= S_1, S_2, \ldots, S_m$	List of fog servers
$T_{i,j}$	Trust of node j as evaluated by node i
$T^x_{i,j}$	Trust of node j as evaluated by node i with respect to trust metrics or trust type X
$I[2][Ci]$	Storage of interactions between a server and client C_i
$R[2][Ci]$	Storage of interactions realized by client C_i
$P[2][Ci]$	Storage of propagations made by client C_i
ST_i	Subjective trust of a node defined as belief, disbelief, uncertainty and atomicity
$\alpha 1, \beta 1, \mu 1, \gamma 1$	Adaptively calculated weighting factors for trust computation of clients
$\alpha 2, \beta 2, \mu 2, \gamma 2$	Adaptively calculated weighting factors for trust computation of Servers

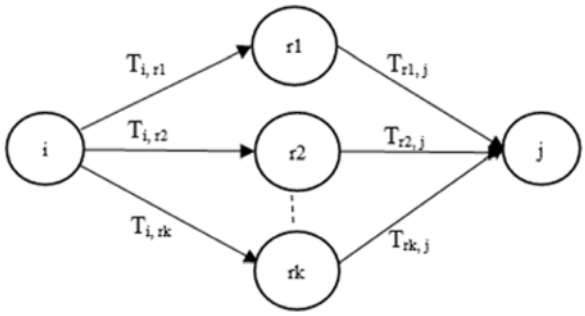


Fig. 5. Computation of overall indirect trust.

direct trust. Direct trust value of a node j as it is evaluated by node i at time t is calculated using the following formula:

$$T^{Direct}_{i,j}(t) = \alpha * T^x_{i,j}(t) + \beta * T^y_{i,j}(t) + \mu * T^z_{i,j}(t) \tag{7}$$

where x, y, and z are trust metrics and α, β, and μ are weighting factors of individual trust metrics. For fog servers x, y, and z are latency, PDR and ownership, respectively, while for fog clients they are friendship, honesty, and ownership, respectively. The weighting factors of the QoS and social trust metrics are adjusted adaptively based on belongingness of trustor and trustee to an owner and reputation of the trustee. Reputation in this case refers overall trust value of a node in previous encounters. If the trustor and trustee are owned by the same person or reputation of trustee is above threshold trust value that splits complete trust and distrust, average values of the existing trust metrics are taken. Otherwise, half the weight provided for the other trust metrics is assigned to ownership metrics. This simple weighting procedure encourages nodes with good reputations and penalizes bad ones. Furthermore, the procedure is suitable to resource constrained fog clients.

Calculating direct and indirect trust values paves a way to compute the overall final trust level of a node. It is calculated using the succeeding formula:

$$T_{i,j}(t) = \gamma * T^{Direct}_{i,j}(t) + (1-\gamma) * T^{Indirect}_{i,j}(t) \tag{8}$$

γ is the factor that specifies the contribution of direct and indirect trusts on the overall trust value. The contribution of indirect trust value ($\gamma_{Indirect} = 1-\gamma$) to overall trust is adaptively decided based on the trust value of trustee in preceding trust computation ($T_{i,j}(t - \triangle t)$), number of recommenders ($n(rec)$), and maximum possible recommenders the configuration allows ($max(rec)$), see formula (9).

$$\gamma_{Indirect} = \frac{(n(rec) * T_{i,j}(t - \triangle t))}{(max(rec) + n(rec) * T_{i,i}(t - \triangle t))} \tag{9}$$

The indirect trust weighting factor is normalized to past experience and current number of recommenders relative to maximum possible number of recommenders a fog node can have in the setup. The formula ensures that the contributions of indirect trust do not exceed more than half of the overall trust and an increase of the contribution proportionally with number of recommenders. Providing more weight to direct trust and using more recommendations leads to accurate and fast converging trust values [33].

4.2. Two-way trust computation algorithm

The detailed steps of the two-way trust computation algorithm are described in Algorithm 1. The final trust value is in the range of [0, 1], where 0 implies complete distrust and 1 is complete trust [14]. The ignorance or threshold point at which trust and distrust are dissected depends on the application on hand. For most applications, it is 0.5 but for safety-critical and health applications the value is usually higher. Table 2 displays important notations used in the algorithm.

Fog clients initiate trust computation by sending a connection request to a nearby fog server. The fog server checks the trustworthiness of the client by computing direct trust from friendship, honesty and ownership trust metrics, steps 4–11, and by consulting and aggregating recommendations from neighboring servers using subjective logic, steps 12–17 of Algorithm 1. At step 18 direct and indirect trusts are combined based on their respective weights to determine the final trust value of the client. If the final trust is greater than or equal to the minimum threshold trust value the server expects, it allows the connection. In this case, it is fog clients turn to assure if the fog server is a trusted service provider. Hence, it executes steps 19–33 to find out the trust value of the server. Two QoS trust information, latency and PDR, and a social relationship trust information, ownership, are used to calculate the direct trust value of the server. Recommendations are collected from neighboring servers and aggregated to form the indirect trust value. Next, the last trust value of the server is decided from the weighted sum of direct and indirect trusts. If the server has acceptable trust value, a trusted connection is established between the fog server and fog client. If either fog server or fog client is untrustworthy, then the client sends a connection request to another server and the steps described above are repeated. The name two-way trust management is given because of trust computation from fog server to fog client and from fog client to fog server.

Algorithm 1: Two-way Trust Computation Algorithm

Input : $\{C_1, C_2, ..., C_n\}, \{S_1, S_2, ..., S_n\}, \alpha1, \beta1, \mu1, \gamma1, \alpha2, \beta2, \mu2, \gamma2$
Output: Trusted connection established

foreach *i between 1 and n* **do**
  ns = neighboring Servers of Client C_i
  foreach *j between 1 and ns.size* **do**
    $T_{i,j}^{Friendship} = \frac{C_i.I[1][j]}{Max(C_i.I[1][k]|k=1,2,...,n)}$
    $T_{i,j}^{Honesty} = \frac{C_i.P[1][j]+C_i.R[1][j]}{C_i.P[0][j]+C_i.P[1][j]+C_i.R[0][j]+C_i.R[1][j]}$
    if C_i *and* S_j *are owned by same person and Trust of* $C_i \geq thereshold_1$ **then**
      $T_{i,j}^{Ownership} = 0.33$;
    else
      $T_{i,j}^{Ownership} = 0.2$;
    end
    /* Calculate direct trust of client, C_i */
    $T_{i,j}^{Direct} = \alpha1 * T_{i,j}^{Friendship} + \beta1 * T_{i,j}^{Honesty} + \mu1 * T_{i,j}^{Ownership}$
    ST_i = Initial Subjective Trust Value of C_i
    foreach *a between 1 and ns.size* **do**
      $discounting = T_{i,a} \oplus T_{a,i}$
      $ST_i = ST_i \otimes discounting$
    end
    /* Calculate indirect trust of client, C_i */
    $T_{i,j}^{Indirect} = ST_i.Belief + ST_i.Uncertainty * ST_i.Atomicity$
    $T_{i,j} = \gamma1 * Ti, j^{Direct} + (1 - \gamma1) * T_{i,j}^{Indirect}$
    if $T_{i,j} \geq thereshold_1$ **then**
      $Tj, i^{Latency} = logNormalResponse(mean, SD)$
      $Tj, i^{PDR} = gilbertElliottModel(p, r)$
      if C_i *and* S_j *are owned by same person and Trust of* $S_j \geq thereshold_2$ **then**
        $T_{j,i}^{Ownership} = 0.33$;
      else
        $T_{j,i}^{Ownership} = 0.2$;
      end
      /* calculate direct trust of server, S_j */
      $T_{j,i}^{Direct} = \alpha2 * T_{j,i}^{Latency} + \beta2 * T_{j,i}^{PDR} + \mu2 * T_{j,i}^{Ownership}$
      ST_j = Initial Subjective Trust Value of S_j
      foreach *a between 1 and ns.size* **do**
        $discounting = T_{j,a} \oplus T_{a,j}$
        $ST_j = ST_j \otimes discounting$
      end
      /* Calculate indirect trust of server, S_i */
      $T_{j,i}^{Indirect} = ST_j.Belief + ST_j.Uncertainty * ST_j.Atomicity$
      $T_{j,i} = \gamma2 * T_{j,i}^{Direct} + (1 - \gamma2) * T_{j,i}^{Indirect}$
      if $T_{j,i} \geq thereshold_2$ **then**
        /* make trusted connection */
      end
    end
  end
end

4.3. Resistance to trust-based attacks

Service requesters must have the desired level of trust value since service providers entertain only trusted requesters. Service providers want to be profitable by serving as many service requesters as possible. Hence, nodes related in trust computation and sharing may be involved in trust-based attacks. A bad or a malicious node may transmit wrong information about another entity or itself, collude with others to control service or may also act incorrectly to mischief those in trust relation [14]. An inexhaustive list of trust-based attack models [23,24] with definitions and how effective is the TTM algorithm to thwart those attacks is presented in this section.

- *Self-Promotion Attack (SPA)*: if a malicious object is requested about its trust value it provides good recommendation and once it is selected, it may provide poor service or abuse the network. The trust management system presented is fully resilient to SPA since a node cant recommend for itself.
- *Ballot-Stuffing Attack (BSA)*: a type of collusion attack where malicious recommender gives exaggerated trust value about a bad object to trust information requester with an intention of increasing reputation of the object. In the proposed trust management system, a bad fog server may send incorrect recommendations about a fog node and a bad fog client may also propagate false trust information to neighboring servers. The problem is addressed by the selected trust aggregation method. In the method, recommendations are weighted based on the trust level of recommenders. Consequently, a recommendation of a bad node will have a very small contribution to the overall indirect trust. The second solution applied to fight back BSA is ignoring exaggerated recommendations from recommendations list.
- *Bad-Mouthing Attack (BMA)*: despite the fact that an object is trustable, colluding recommenders provide a false trust value to vituperate false information about the object. When malicious nodes are selected because of false information, they get access to resources posing security threats to the entire system. This kind of attack is viable to happen but solutions applied for BSA also address the attack.
- *Opportunistic-Service Attack (OSA)*: trust management systems decrease the reputation of malicious nodes from time to time. When the node notices that its reputation is dropping, it performs good service and when its reputation is high it starts giving bad service. OSA is dealt with monitoring the behavior of a node and removing it from the network if its behavior is fluctuating over a certain period of time.
- *On-Off Attack (OOA)*: with this attack, a malicious object performs good and bad services randomly to level itself as a normal or good object. To foil this attack, the same solution proposed for OSA is used.

Table 3
List of simulation parameters.

Parameters	Values
Number of fog servers	40
Number of fog clients	200
Number owners	20
% of bad nodes (PD)	20%
Simulation period (Cycle)	25

5. Trust management algorithm performance

The trust management system discussed is evaluated in a simulated environment. Evaluation setup and the results of the evaluations are presented in this section.

There are some simulation tools for fog computing; iFogSim [62] for measuring the impact of resource management techniques, Discrete Event System Specification (DEVS) based tool [63] for evaluating impact of deploying fogging, EmuFog [64] is extensible emulation framework for fog computing environments without mobility feature, and FogTorch [65] for QoS-aware deployment of multi-component IoT applications to fog infrastructures. However, there is no full-fledged simulation tool for the new computing paradigm. Therefore, we developed our own Java-based simulation tool for the scenario explained in the system model section. The tool contains classes like fog node, topology manager, mobility manager and trust computer as the most important components and other many miscellaneous components. Reporting detail of the tool is beyond the scope of this report. The list of parameters used and their default values are indicated in Table 3. We considered a fog environment where there are 200 fog clients and 40 fog servers randomly distributed over 20 owners. Maximum number of 1-hop neighboring nodes that could send recommendation is six. At the beginning of the simulation, fog clients which are placed on a walkway are connected to fog server. While fog servers are assumed to be static, the clients are mobile. The clients travel in the direction they are originally set with a pedestrian speed randomly assigned based on the values obtained from [66]. The clients walk from left to right and right to

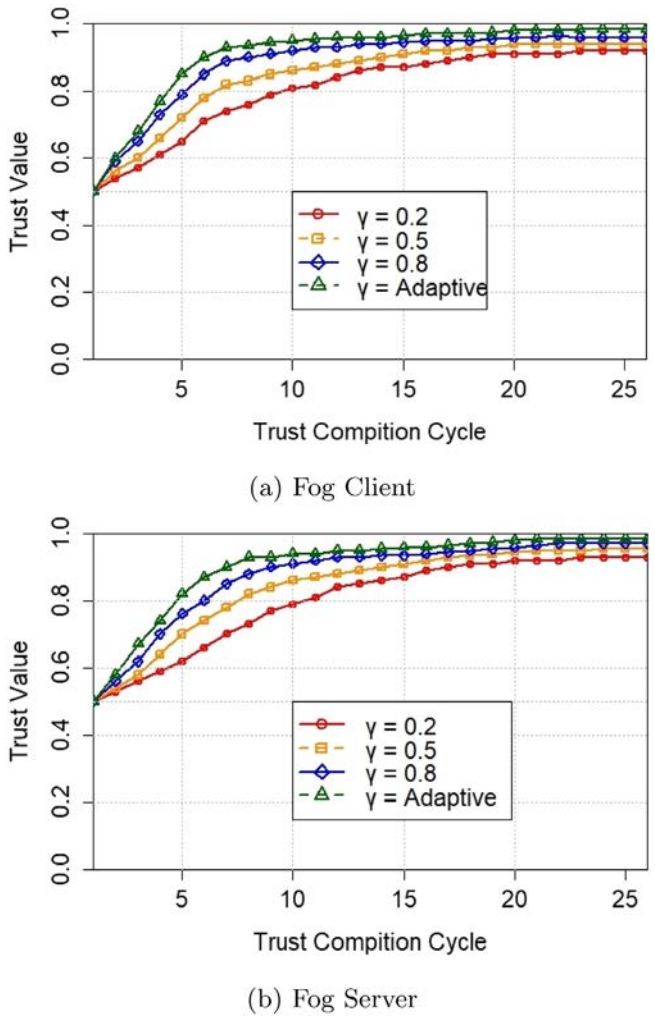


Fig. 6. Trust values of (a) Good fog client and (b) Good fog server over trust computation cycle.

left continuously on defined trajectory throughout the simulation period. The proposed system is an event-driven trust computation where, when a moving client arrives to the trust computation zone, it starts exchanging information with fog servers in order to create a connection to trusted fog server. Trust computation zones are regions in the simulation environment where a fog client is leaving the network coverage area of currently connected service provider. A trust computation cycle (simulation cycle) is equal to travel from one end of the walkway to the other. If there are multiple trust zones between the two ends of the road, the average value of trusts computed in all zones is taken. We first presented evolutions of trust accuracy, trust convergence and how resistant is the system to increased number of malicious nodes. Next, comparative analysis of different derivatives of the proposed algorithm is given. Finally, our algorithms are also compared against baseline trust computation methods.

5.1. Evaluation of trust accuracy, convergence and resilience

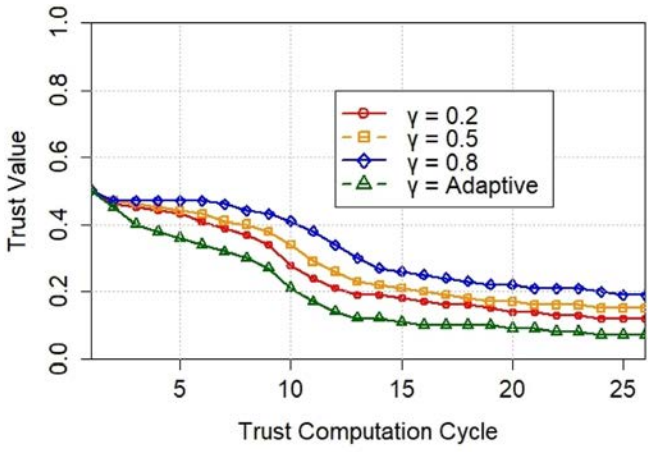
The evaluation results of accuracy, convergence and resilience of the proposed algorithm are presented in the next paragraphs. Unless specified, default parameters shown in Table 3 are used.

Fig. 6 shows trust accuracy and convergence of randomly picked normal or good fog client (Fig. 6(a)) and good fog server (Fig. 6(b)) with adaptive and static weighting factors (γ) among direct and indirect trusts. The trust management that relies on adaptive weighting factor is the most accurate and the fast converging one. This is because indirect trust gets higher weights only when it is obtained from larger number of recommenders. For static weighting, as the values of γ increase, the trust converges faster and it has better accuracy. This confirms that the trust value obtained by self-observation better describes the final trust value of an entity. This is because recommendations are affected by the presence of malicious nodes [67]. In our system, by default 20% nodes are dishonest or bad nodes. Trust value of 0.9 is achieved at 17th cycle when more weight is given to indirect trust for fog clients while only at 9th cycle the same value is achieved if more weight is provided for direct trust. As shown in the diagram, though it takes a longer time to converge, relying on indirect trust doesn't prevent the algorithm from convergence. Therefore, we can say that indirect trust has a very important role, especially in cases direct observation is imprecise or not possible. There is no difference between trust values of fog clients and fog servers except in case of fog servers the graph is a bit smoother due to a constant set of neighboring servers that feed recommendations.

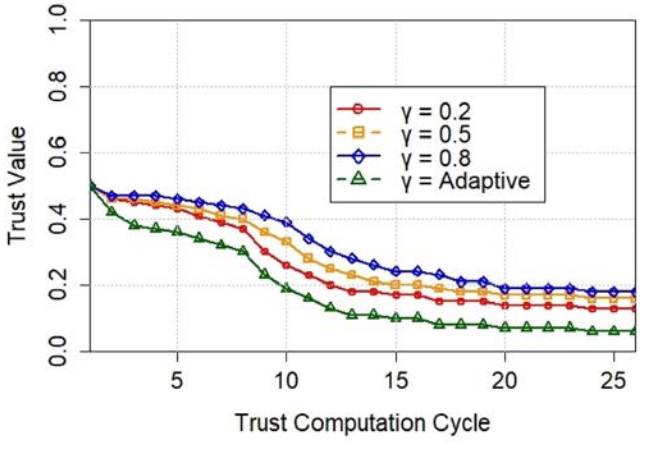
The evaluation of the trust value of a randomly selected bad fog client and bad fog server are shown in Fig. 7. If a node is bad either intentionally or unintentionally it acts undesirably resulting in its less trust value. Fog servers that have high latency and low PDR are bad servers. Hence, the servers latency and PDR which are out of range of good nodes are converted to trust values less than 0.5. Friendship and Honesty are calculated based on interaction history and propagated trust values. All the metrics are used to compute the direct trust value of the overall trust. The indirect trust which is obtained from recommenders may contribute to up to 50% (maximum) of the overall trust. The main objective of this evaluation is assuming that a node is bad, how is trust management system handling such situations without going through the nitty-gritty of malicious node detection. The default trust value of a node in this evaluation is 0.5 and the ground truth values of bad and good nodes are 0.0 and 1.0, respectively. Based on the type of a node the trust level increase or decrease in the course of trust computation cycle. The algorithm is able to turn bad nodes worst. This is especially true when the weighting factor of direct and indirect trust values is dynamically determined since the factor depends on preceding trust values. The reduction of the trust values helps other nodes abstain from creating connection to such nodes and to remove such nodes from the network whenever it necessitates.

Fig. 8 shows the effect of the algorithm to the increased percentage of bad fog clients with adaptive γ. The percentage of bad nodes are set to 20%, 40%, and 60%. It can be observed that as the percentage of malicious nodes increase it takes more time for a good fog clients trust values to converge. Moreover, the trust value at which convergence appears lowers as more population of malicious nodes are introduced. For instance, the trust value of a good fog client cant attain 0.80 till 22nd cycle of simulation if the percentage of bad nodes is 60%. However, the algorithm is resilient to hostility since it provides reasonably high trust value even in percentage of large population of bad nodes. The value meets the requirements of many applications although the population of bad nodes is large.

We modified the algorithm in such a way that bad nodes are expelled from the network. A fog client is expelled if it is unable to be accepted by any neighboring service provider. That is, it passes a trust computation zone without successfully connecting to another service provider due to its low trust value. A fog server is expelled if it is not selected for service provision by any service requester over a trust computation cycle. The percentage of expelled bad nodes in the simulation environment where the



(a) Fog Client



(b) Fog Server

Fig. 7. Trust values of (a) Bad fog client and (b) Bad fog server over trust computation cycle.

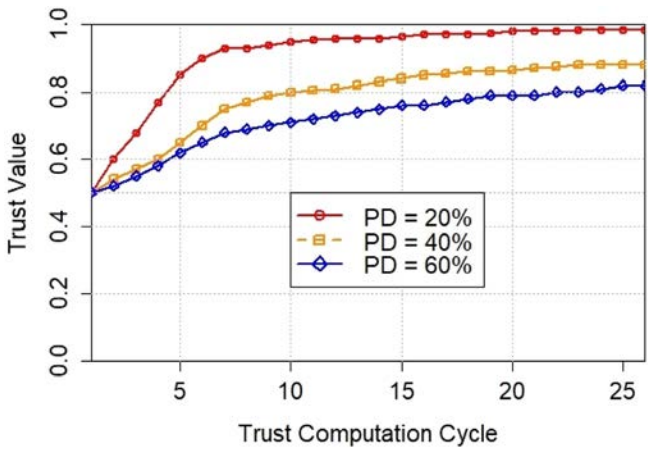


Fig. 8. Effect of percentage of bad nodes on trust values of a good fog client.

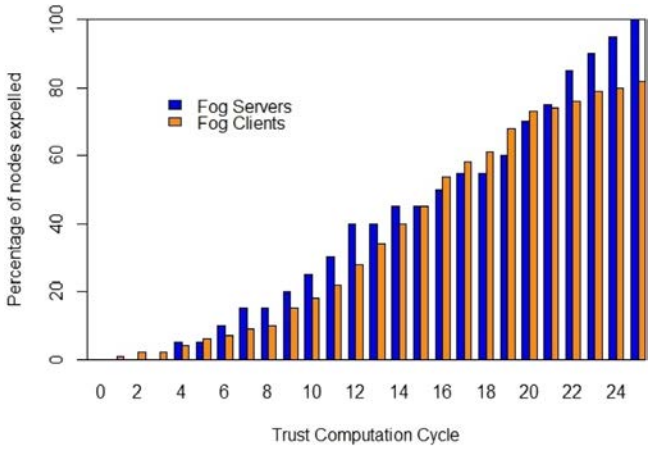
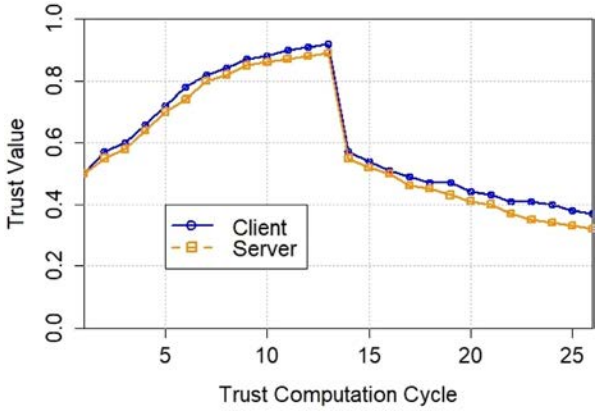
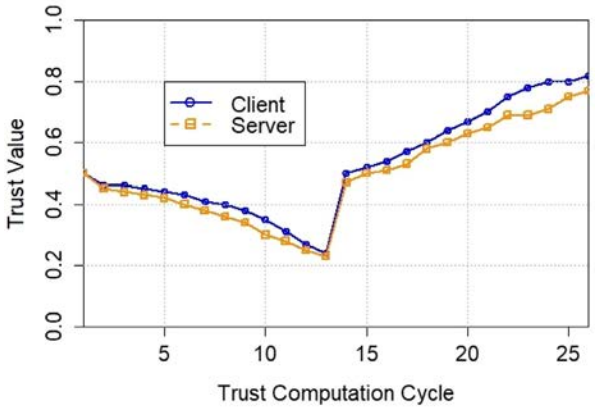


Fig. 9. Percentage of expelled bad nodes over trust computation cycle.



(a) Fog Client



(b) Fog Server

Fig. 10. Change of behavior of nodes (a) from Good to Bad and (b) from Bad to Good.

percentage of bad nodes is 50% are graphed in Fig. 9. This data is collected by counting nodes expelled up to a simulation cycle. At 25th trust computation cycle, all bad fog servers and 82% of bad fog clients are able to be expelled from the network.

The behavior of fog nodes may change over time [51,68]. A good node may change to a bad node because of malfunctioning or in order to pose attacks and a bad node may also change to good behaving node. Fig. 10 shows the trust value of a randomly selected good node whose behavior has turned to bad (Fig. 10(a)) and a bad node whose behavior has changed to good (Fig. 10(b)) at 12th simulation cycle. The algorithm is able to capture the behavior change of nodes correctly.

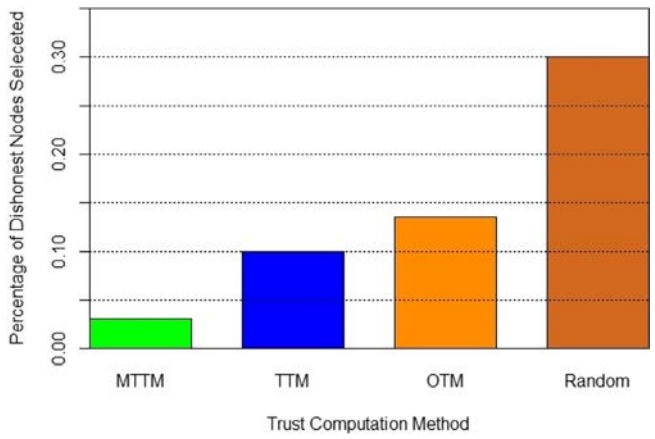


Fig. 11. Average percentage of bad fog servers selected for service provision.

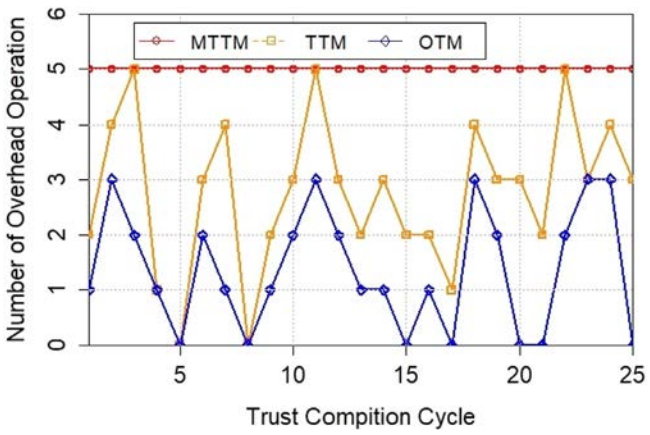


Fig. 12. Number of overhead operations over trust computation cycle.

5.2. Comparative analysis

The TTM systems algorithm can be amended in different ways. Hence, we have modified the algorithm in two different ways. The first amendment is made in such a way that a fog client evaluates trust values of all neighboring servers and send a connection request to the server with the highest possible trust value. We named this change Modified Two-way Trust Management (MTTM) system. In the original two-way algorithm, any fog server encountered early and that fulfills trust value requirement of the client is selected as long as the client itself is found to be trustworthy. The second alteration of the algorithm is a One-way Trust Management (OTM) system where clients can evaluate trust of servers to which they are going to connect without their trust being evaluated by the servers. This method works in the same way as TTM except, in this case, clients trust is not evaluated. In addition to the two-modification random service provider selection which randomly opts service providers without regard to the trust values is another method on the plate for comparison. The four methods are evaluated in terms of average percentage of bad servers selected for service provision over 25 trust computation cycles by a randomly selected fog client. In a single trust computation cycle a fog client changes its parent (fog server) 20 times, which implies the total number of fog server selections are 500 in 25 trust computation cycle. In this experiment, a bad server has a chance to be selected for service version if a fog client cant find a good one. A bad server is the one with trust value less than 0.5. The objective of the evaluation is to check how effective is the proposed method in terms of selection of the best service provider with the assumption that the quality of services provided depends on the trust level of the service provider. For this evaluation the percentage of bad servers is set to 40% and the number of fog servers has the default value. The average percentage of bad service providers selected over the trust computation cycles, with confidence 95% level, is plotted in Fig. 11.

MTTM has chosen only one bad fog server on average in all trust computation cycles. It is the most effective method because of its ability to choose the most trusted node to create a connection. However, since the method involves high overhead (see Fig. 12) due to the computation of trust of all neighboring servers, it is not suitable for real-time systems. Moreover, MTTM results in an unbalanced load distribution among fog servers. That is, the algorithm heavily loads fog servers with highest trust values (see Fig. 13). TTM may select a server which does not possess the highest trust value but it is optimal in the selection of service providers with desired trust value and it has lesser computational overhead as well as balanced load distribution than MTTM. OTM has a bit higher percentage of bad fog server choices than TTM since clients will not be denied connection by any server. In TTM a bad server may reject connection request of a bad client due to a very low trust value.

Fig. 12 shows overhead of the three subjective logic-based trust computation algorithms over trust computation cycle. Overhead here referees the number of trust computations that doesn't result in a trusted connection. In MTTM trust value of all servers are computed regardless of trust values of the servers encountered. Then, connection request is sent to the one with the highest trust value. That is why it has the highest overhead. This makes MTTM not suitable for latency sensitive applications. OTM has high chance of getting servers quicker than TTM. Hence, it has the lowest overhead.

In addition to the high overhead, MTTM has a negative effect on load balancing. Servers with high trust values tend to attract a lot of fog clients whereas those with lower trust values are idle. Fig. 13 shows load distribution of randomly selected 10 fog servers. TTM and OTM have comparatively balanced load distribution while MTTM biases towards servers with high trust values. In MTTM, fog servers numbered 1, 3, and 9 have very high load shares whereas others have either very small or no loads.

The last set of experiments conducted are the evaluation of the trust levels of servers opted for service provision using MTTM, TTM, and OTM against two well-known peer-to-peer trust computation baselines, PeerTrust [69] and EigenTrust [70], for trust values of a randomly selected good node. The purposes of this evaluation are: (i) to check if two-way trust computation has indeed advantage over one-way trust computations, (ii) to compare our trust management algorithms against baseline schemes. Hence, trust convergence and accuracy of the trust computation methods on randomly selected good fog server are evaluated over trust computation cycles. Percentage of bad client nodes is set to have default value (i.e., 20%), and direct and indirect trust contributions to overall trust are decided dynamically. The result is shown in Fig. 14.

MTTM outperforms the other subjective logic-based trust management methods proposed both in accuracy and trust convergence though its difference with TTM is minor. This is bearing to its ability to choose the most trusted fog server. One-way trust management system is the least performing algorithm. The reason behind this is that, firstly, the trust value of the server they first encountered and selected may not have the highest trust value in comparison with neighboring servers. Secondly, the indirect trust computation of the servers misses recommendations from nearby servers. In the main algorithm, clients

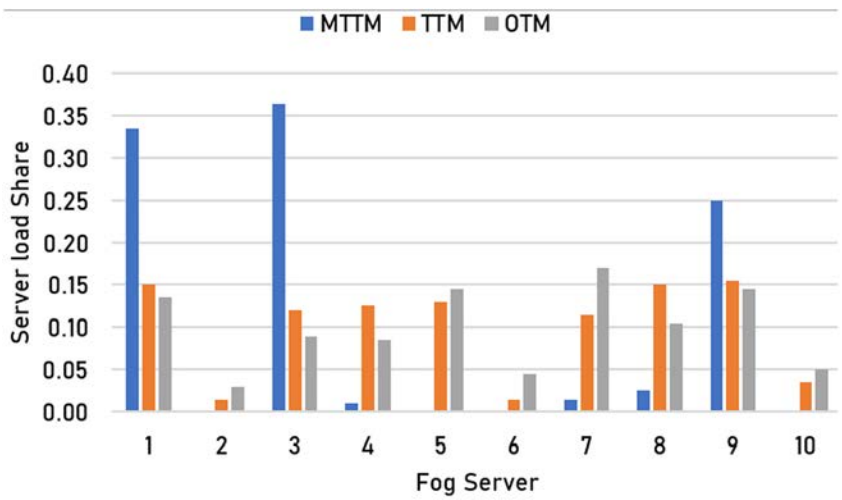


Fig. 13. Load distribution of randomly selected fog servers.

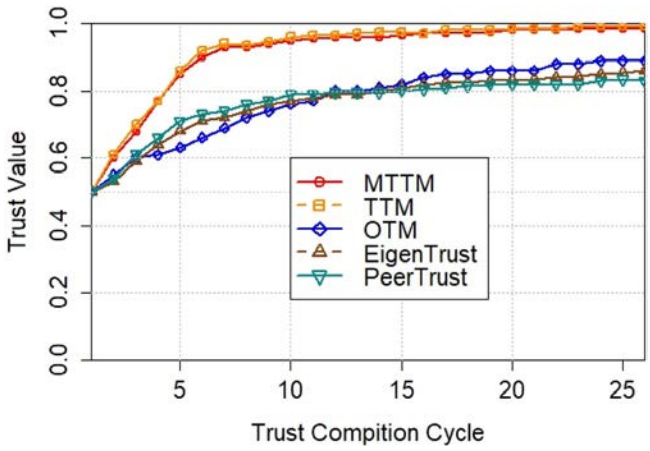


Fig. 14. Trust values of different derivatives of the algorithm and baseline schemes for randomly selected good fog server.

propagate trust values of the servers they have computed to establish trusted data communication to servers and the servers use this value to weight recommendations of the servers. Therefore, we can conclude that the two-way trust management system chooses most trusted service providers than the one-way trust management system. The baseline schemes have comparable trust values as OTM as all of the trust computation methods are one-way. However, PeerTrust and EigenTrust converge faster than OTW though they have lower accuracy as trust computation cycle increases. Due to two-wayness and adaptive nature, MTTM and TTM have highest accuracy and fast convergence time than the other one-way trust management systems.

6. Conclusion

Unprecedented amount of data is being generated by IoT devices posing challenges to cloud computing to find a pathway to the data to travel to its destination. Fog computing is a promising architecture that allows data processing near to where they are generated instead of sending them to the cloud. In doing so, it saves a great deal of bandwidth and responds in real time making it ideal solution for latency sensitive applications. To enjoy the bevy advantages of fog computing there are challenges that have to be addressed where security, privacy and trust management are among the most important ones. In this paper, we have presented a two-way trust management system that enables service requesters to evaluate trust level of service providers before awarding the service and that allows service providers to check the trustworthiness of the service demanders. The bidirectionality of our solution is the first of its kind for fog computing and it allows fog nodes abstain from connecting to untrustworthy nodes and ensures secure data communication with only trusted nodes. The system incorporates both QoS and social trust information to compute trust levels of fog nodes from adaptive combination of direct observation and from recommendations. Subjective logic is used to aggregate trust values obtained from neighboring recommenders. Extensive evaluation of the trust management system shows that it converges quickly, have high accuracy and is resilient to trust-based attacks. The solution is modified in two different ways and compared in terms of percentage of bad nodes selected for service provision and trust convergence. The first modification is that a service requester selects a service provider with the highest trust value instead of selecting a trusted service provider encountered first. The second amendment, whereas, is the conversion of TTM to the conventional one-way trust management system. We found that the proposed system gives an optimal solution since the modifications have either high overhead and unbalanced load distribution or do not converge in time or have less accuracy. The comparison of subjective logic-based algorithms with two baseline trust schemes shows the preeminence of the two-way versions of the proposed trust management system.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank anonymous reviewers of this paper for the insightful and detailed comments. This work is supported partly by the French government and Ethiopian Ministry of Education.

References

[1] M.P. Daniel Alsen, J. Shangkuan, The future of connectivity: Enabling the Internet of Things, 2017, URL https://www.mckinsey.com/featured-insights/internet-of-things/our-insights.

[2] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ACM, 2012, pp. 13–16.

[3] S. Yi, Z. Qin, Q. Li, Security and privacy issues of fog computing: A survey, in: International Conference on Wireless Algorithms, Systems, and Applications, Springer, 2015, pp. 685–695.

[4] E. Marín-Tordera, X. Masip-Bruin, J. García-Almiñana, A. Jukan, G.-J. Ren, J. Zhu, Do we all really know what a fog node is? Current trends towards an open definition, Comput. Commun. 109 (2017) 117–130.

[5] E. Alemneh, S.-M. Senouci, P. Brunet, PV-Alert: A fog-based architecture for safeguarding vulnerable road users, in: 2017 Global Information Infrastructure and Networking Symposium, GIIS, IEEE, 2017, pp. 9–15.

[6] A.M. Rahmani, T.N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, P. Liljeberg, Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach, Future Gener. Comput. Syst. 78 (2018) 641–658.

[7] C.-F. Lai, D.-Y. Song, R.-H. Hwang, Y.-X. Lai, A QoS-aware streaming service over fog computing infrastructures, in: 2016 Digital Media Industry & Academic Forum, DMIAF, IEEE, 2016, pp. 94–98.

[8] T. Fernández-Caramés, P. Fraga-Lamas, M. Suárez-Albela, M. Vilar-Montesinos, A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard, Sensors 18 (6) (2018) 1798.

[9] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, Q. Yang, Incorporating intelligence in fog computing for big data analysis in smart cities, IEEE Trans. Ind. Inform. 13 (5) (2017) 2140–2150.

[10] OpenFog, Building an open architecture for fog computing, 2018, URL https://www.openfogconsortium.org.

[11] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M.A. Ferrag, N. Choudhury, V. Kumar, Security and privacy in fog computing: Challenges, IEEE Access 5 (2017) 19293–19304.

[12] J. Ni, K. Zhang, X. Lin, X.S. Shen, Securing fog computing for internet of things applications: Challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2017) 601–628.

[13] Y. Guan, J. Shao, G. Wei, M. Xie, Data security and privacy in fog computing, IEEE Netw. 32 (5) (2018) 106–111.

[14] N. Tariq, M. Asim, F. Al-Obeidat, M. Zubair Farooqi, T. Baker, M. Hammoudeh, I. Ghafir, The security of big data in fog-enabled IoT applications including blockchain: A survey, Sensors 19 (8) (2019) 1788.

[15] H. Li, M. Singhal, Trust management in distributed systems, Computer 40 (2) (2007) 45–53.

[16] T.S. Dybedokken, Trust Management in Fog Computing (Master's thesis), NTNU, 2017.

[17] I. Stojmenovic, S. Wen, The fog computing paradigm: Scenarios and security issues, in: 2014 Federated Conference on Computer Science and Information Systems, IEEE, 2014, pp. 1–8.

[18] Z.M. Aljazzaf, M.A. Capretz, M. Perry, Trust bootstrapping services and service providers, in: 2011 Ninth Annual International Conference on Privacy, Security and Trust, IEEE, 2011, pp. 7–15.

[19] Y. Wang, J. Vassileva, Trust and reputation model in peer-to-peer networks, in: Proceedings Third International Conference on Peer-to-Peer Computing, P2P2003, IEEE, 2003, pp. 150–157.

[20] B. Liu, A survey on trust modeling from a Bayesian perspective, 2018, arXiv preprint arXiv:1806.03916.

[21] Z. Yan, P. Zhang, A.V. Vasilakos, A survey on trust management for Internet of Things, J. Netw. Comput. Appl. 42 (2014) 120–134.

[22] J.-H. Cho, A. Swami, R. Chen, A survey on trust management for mobile ad hoc networks, IEEE Commun. Surv. Tutor. 13 (4) (2010) 562–583.

[23] J. Guo, R. Chen, J.J. Tsai, A survey of trust computation models for service management in internet of things systems, Comput. Commun. 97 (2017) 1–14.

[24] F. Azzedin, M. Ghaleb, Internet-of-things and information fusion: Trust perspective survey, Sensors 19 (8) (2019) 1929.

[25] R. Chen, J. Guo, F. Bao, J.-H. Cho, Integrated social and quality of service trust management of mobile groups in ad hoc networks, in: 2013 9th International Conference on Information, Communications & Signal Processing, IEEE, 2013, pp. 1–5.

[26] L. Atzori, A. Iera, G. Morabito, SIoT: Giving a social structure to the internet of things, IEEE Commun. Lett. 15 (11) (2011) 1193–1195.

[27] V.L. Tran, A. Islam, J. Kharel, S.Y. Shin, On the application of social internet of things with fog computing: a new paradigm for traffic information sharing system, in: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud, FiCloud, IEEE, 2018, pp. 349–354.

[28] A. Jøsang, Decision making under vagueness and uncertainty, in: Subjective Logic, Springer, 2016, pp. 51–82.

[29] A. Jsang, Subjective Logic: A Formalism for Reasoning Under Uncertainty, Springer Publishing Company, Incorporated, 2018.

[30] R. Chen, J. Guo, F. Bao, Trust management for service composition in SOA-based IoT systems, in: 2014 IEEE Wireless Communications and Networking Conference, WCNC, IEEE, 2014, pp. 3444–3449.

[31] F. Bao, R. Chen, Trust management for the internet of things and its application to service composition, in: 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM, IEEE, 2012, pp. 1–6.

[32] F. Bao, R. Chen, J. Guo, Scalable, adaptive and survivable trust management for community of interest based internet of things systems, in: 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems, ISADS, Citeseer, 2013, pp. 1–7.

[33] F. Bao, I.-R. Chen, Dynamic trust management for internet of things applications, in: Proceedings of the 2012 International Workshop on Self-Aware Internet of Things, ACM, 2012, pp. 1–6.

[34] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, X. Wang, TRM-IoT: A trust management model based on fuzzy reputation for internet of things, Comput. Sci. Inf. Syst. 8 (4) (2011) 1207–1228.

[35] U. Jayasinghe, N.B. Truong, G.M. Lee, T.-W. Um, Rpr: A trust computation model for social internet of things, in: 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress, UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld, IEEE, 2016, pp. 930–937.

[36] M. Khani, Y. Wang, M.A. Orgun, F. Zhu, Context-aware trustworthy service evaluation in social internet of things, in: International Conference on Service-Oriented Computing, Springer, 2018, pp. 129–145.

[37] M.B. Monir, M.H. AbdelAziz, A.A. AbdelHamid, E.-S.M. EI-Horbaty, Trust management in cloud computing: a survey, in: 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems, ICICIS, IEEE, 2015, pp. 231–242.

[38] M. Chiregi, N.J. Navimipour, Cloud computing and trust evaluation: A systematic literature review of the state-of-the-art mechanisms, J. Electr. Syst. Inf. Technol. (2017).

[39] T.H. Noor, Q.Z. Sheng, S. Zeadally, J. Yu, Trust management of services in cloud environments: Obstacles and solutions, ACM Comput. Surv. 46 (1) (2013) 12.

[40] J. Huang, D.M. Nicol, Trust mechanisms for cloud computing, J. Cloud Comput. Adv. Syst. Appl. 2 (1) (2013) 9.

[41] S. Chakraborty, K. Roy, An SLA-based framework for estimating trustworthiness of a cloud, in: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2012, pp. 937–942.

[42] P. Manuel, A trust model of cloud computing based on Quality of Service, Ann. Oper. Res. 233 (1) (2015) 281–292.

[43] R. Hajizadeh, N. Jafari Navimipour, A method for trust evaluation in the cloud environments using a behavior graph and services grouping, Kybernetes 46 (7) (2017) 1245–1261.

[44] R. Chen, J. Guo, D.-C. Wang, J.J. Tsai, H. Al-Hamadi, I. You, Trust-based service management for mobile cloud IoT systems, IEEE Trans. Netw. Serv. Manag. 16 (1) (2018) 246–263.

[45] F.H. Rahman, T.W. Au, S. Newaz, W.S. Suhaili, Trustworthiness in fog: A Fuzzy approach, in: Proceedings of the 2017 VI International Conference on Network, Communication and Computing, ACM, 2017, pp. 207–211.

[46] A. Alrawais, A. Alhothaily, C. Hu, X. Cheng, Fog computing for the internet of things: Security and privacy issues, IEEE Internet Comput. 21 (2) (2017) 34–42.

[47] R. Roman, J. Lopez, M. Mambo, Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges, Future Gener. Comput. Syst. 78 (2018) 680–698.

[48] F.H. Rahman, T.-W. Au, S.S. Newaz, W.S. Suhaili, G.M. Lee, Find my trustworthy fogs: A fuzzy-based trust evaluation framework, Future Gener. Comput. Syst. (2018).

[49] R. Mahmud, R. Kotagiri, R. Buyya, Fog computing: A taxonomy, survey and future directions, in: Internet of Everything, Springer, 2018, pp. 103–130.

[50] M. Wazid, A.K. Das, N. Kumar, A.V. Vasilakos, Design of secure key management and user authentication scheme for fog computing services, Future Gener. Comput. Syst. 91 (2019) 475–492.
[51] P.B. Velloso, R.P. Laufer, D.D.O. Cunha, O.C.M. Duarte, G. Pujolle, Trust management in mobile ad hoc networks using a scalable maturity-based model, IEEE Trans. Netw. Serv. Manage. 7 (3) (2010) 172–185.
[52] OpenFog, Open fog consortium, openfog reference architecture for fog computings, 2017, URL https://www.openfogconsortium.org/wpcontent/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL-1.pdf,2017.
[53] R.K. Naha, S. Garg, D. Georgakopoulos, P.P. Jayaraman, L. Gao, Y. Xiang, R. Ranjan, Fog computing: Survey of trends, architectures, requirements, and research directions, IEEE Access 6 (2018) 47980–48009.
[54] M. Arshad, Z. Ullah, N. Ahmad, M. Khalid, H. Criuckshank, Y. Cao, A survey of local/cooperative-based malicious information detection techniques in VANETs, EURASIP J. Wireless Commun. Networking 2018 (1) (2018) 62.
[55] S. Namal, H. Gamaarachchi, G. MyoungLee, T.-W. Um, Autonomic trust management in cloud-based and highly dynamic IoT applications, in: 2015 ITU Kaleidoscope: Trust in the Information Society, K-2015, IEEE, 2015, pp. 1–8.
[56] V. Paxson, Empirically derived analytic models of wide-area TCP connections, IEEE/ACM Trans. Netw. 2 (4) (1994) 316–336.
[57] G. Haßlinger, O. Hohlfeld, The Gilbert–Elliott model for packet loss in real time services on the Internet, in: 14th GI/ITG Conference-Measurement, Modelling and Evalutation of Computer and Communication Systems, VDE, 2008, pp. 1–15.
[58] A. Bildea, O. Alphand, F. Rousseau, A. Duda, Link quality estimation with the Gilbert–Elliot model for wireless sensor networks, in: 2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC, IEEE, 2015, pp. 2049–2054.
[59] R. Chen, J. Guo, Dynamic hierarchical trust management of mobile groups and its application to misbehaving node detection, in: 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, IEEE, 2014, pp. 49–56.
[60] A. Jøsang, A logic for uncertain probabilities, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 9 (03) (2001) 279–311.
[61] A. Jøsang, S. Marsh, S. Pope, Exploring different types of trust propagation, in: International Conference on Trust Management, Springer, 2006, pp. 179–192.
[62] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments, Softw. - Pract. Exp. 47 (9) (2017) 1275–1296.
[63] M. Etemad, M. Aazam, M. St-Hilaire, Using DEVS for modeling and simulating a Fog Computing environment, in: 2017 International Conference on Computing, Networking and Communications, ICNC, IEEE, 2017, pp. 849–854.
[64] R. Mayer, L. Graser, H. Gupta, E. Saurez, U. Ramachandran, Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures, in: 2017 IEEE Fog World Congress, FWC, IEEE, 2017, pp. 1–6.
[65] A. Brogi, S. Forti, QoS-aware deployment of IoT applications through the fog, IEEE Internet Things J. 4 (5) (2017) 1185–1192.
[66] H. Hamdane, T. Serre, C. Masson, R. Anderson, Issues and challenges for pedestrian active safety systems based on real world accidents, Accid. Anal. Prev. 82 (2015) 53–60.
[67] Y. Wang, Y.-C. Lu, I.-R. Chen, J.-H. Cho, A. Swami, C.-T. Lu, LogitTrust: A logit regression-based trust model for mobile ad hoc networks, in: 6th ASE International Conference on Privacy, Security, Risk and Trust, Boston, MA, 2014, pp. 1–10.
[68] M. Nitti, R. Girau, L. Atzori, Trustworthiness management in the social internet of things, IEEE Trans. Knowl. Data Eng. 26 (5) (2013) 1253–1266.
[69] L. Xiong, L. Liu, Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities, IEEE Trans. Knowl. Data Eng. 16 (7) (2004) 843–857.
[70] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, in: Proceedings of the 12th International Conference on World Wide Web, ACM, 2003, pp. 640–651.