Computer Networks 178 (2020) 107318

Contents lists available at ScienceDirect

Computer Networks

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

TruSD: Trust framework for service discovery among IoT devices

Kübra Kalkan[a,*], Kasper Rasmussen[b]

[a] Ozyegin University, Istanbul, Turkey
[b] University of Oxford, Oxford, UK

ARTICLE INFO

Keywords:
Trust
P2P
IoT
DHT
Service discovery

ABSTRACT

IoT provides an environment which enables access to a plethora of different services. In order to reach these services, devices need to decide if the providers are trustable or not. The decision to trust a node with whom one has not communicated earlier becomes more critical when the system has unrecoverable damages with inaccurate services. In this paper, we propose a framework which enables trusted communication among devices during service discovery. It focuses not only on the communication between the known devices but also the stranger communications which have not contacted earlier. Our framework works in a decentralized manner on top of a structured P2P network based on a Distributed Hash Table (DHT). In our system, for each device there are several nodes which are responsible for holding a trust value for this device. These responsible nodes are called Reference Holders for this device. By utilizing DHT, we propose a novel way of choosing Reference Holders that prevents the malicious nodes to control these nodes. Our protocols provide trust aggregation, service provision and feedback aggregation. In our threat model, attacker provides on-off, bad mouthing, ballot stuffing and selective attacks. We present closed form of probabilistic analysis and provide simulations that manage to give network-wide probabilistic security guarantees. Our results suggest that until 60% of the devices are captured, the results are perfect. Also, just three reference holders are enough to get accurate services through the network. Additionally, we analyze the framework in terms of memory, computational cost and communication overhead since we propose the framework for IoT devices. Due to these analysis, our framework is affordable for IoT devices.

1. Introduction

Internet of Things (IoT) is the network of huge number of devices that can communicate at any time at any place. This technology makes it possible to integrate physical and digital world by utilizing any type of thing which can connect to other devices and exchange data. This "Thing" concept in IoT can refer to any type of devices such as smoke detectors, cameras, heart monitoring implants, chips on farm animals, smart pills, smart phones and supercomputers. Things can be considered as inextricable mixture of hardware, software, data and services [1].

One of the main aims of IoT is to provide services available through the Internet to the users. In order to have efficient and reliable service provisions, trust mechanisms which enable analysis and computation of reputations among devices are provided [2–7].In particular, a feasible trust mechanism for IoT system must deal with scalability and heterogeneity since a typical IoT system contains both resource constraint and resource rich devices in the same ecosystem. Unlike there exist plenty of trust works for P2P MANETs trust computation for IoT remains an important issue as it needs to consider scalability, node mobility, heterogeneity and service experiences together. In [8], a survey is provided that classified the existing trust works for IoT environment. They provide classification according to five design dimensions trust composition (what components are considered?: QoS/social), trust propagation (how to propagate trust evidence to peers?: distributed/centralized), trust aggregation(trust evidence collection strategy: weighted sum/ regression analysis/ fuzzy logic/ bayesian system/ belief theory), trust update (after each service or periodically: event-driven/time-driven) and trust formation (how many trust parameter is considered in trust protocol: single trust/multi trust). According to this classification our work is in Class 6, which takes only QoS for trust composition, is distributed in terms of trust propagation, utilizes static weighted sum for trust aggregation, is event-driven in terms of trust update. There is only one more work [9] in Class 6. In this paper, devices evaluate trust considering only direct information generated from direct communication with the nodes. They did not gather any information from other devices experiences which means that they cannot differentiate malicious nodes before they communicate at least one. With the help of the trust mechanisms, while devices are receiving services, they can evaluate and use their experiences for further communications (first-hand information). Additionally, they can share their experiences with the other devices in order to protect

* Corresponding author.
E-mail addresses: kubra.kalkan@ozyegin.edu.tr (K. Kalkan), kasper.rasmussen@cs.ox.ac.uk (K. Rasmussen).

https://doi.org/10.1016/j.comnet.2020.107318
Received 11 September 2019; Received in revised form 1 May 2020; Accepted 18 May 2020
Available online 22 May 2020

their network. This information is also known as second-hand information [10]. This collaboration can provide accurate decisions even for the communications with strangers which is the most difficult case in trust decisions.

In this work, we propose a trust framework which is not only dealing with known devices but also mainly focusing on trust decisions for communications with a device that is not contacted earlier. This information is gathered from second-hand information after the warming period of the system. If a device does not have any experience with a stranger device, it will act according to only second-hand information which our results suggest that the system gives highly accurate decisions with only second-hand information. Our main motivation is to provide a framework which gives accurate decisions (trust to benign, not trust to malicious) about a service provider which is not contacted earlier. This is necessary when the system is not tolerable to wrong services or the system has unrecoverable damages with inaccurate services. For instance, if the service provides sensitive data storage, in case of inaccurate service the system will give a corrupted data which is unrecoverable. Also, our proposal can be preferable for computational services which need long duration such as a day. If the service does not work properly, it will be noticed after a day, which is annoying. Moreover, fetching an update service can be a good example for unrecoverable services. If a malicious code is provided in an update, the damage will be devastating. Our model can only provide this property after the warming period of the system. It cannot avoid unrecoverable damages from the beginning, as our proposal depends on second-hand information, some devices should provide their feedback according to their experiences. On the other hand, after the warming period, TruSD provides accurate trust decisions with high probability on the long run.

There exist centralized trust mechanisms in which a manager provides trust computation and analysis [11–14]. Regardless of their well-established application, the dependency of these centralized mechanisms on the availability of the connection link between the central administrative point and each device makes them vulnerable to single-point failures as when the connection link is down, trust values cannot be achieved. Also, the fact that the same entity handles all the requests can result in network bottlenecks making this solution not inherently scalable, an essential property of IoT infrastructures that have to accommodate a significant and continuously increasing number of connected devices. Also, every single device in the network needs to trust this manager which is not trivial to determine such type of Trusted Entity for large networks such as smart cities. Thus, these issues motivate the use of decentralized communication models to be used to allow for all of these endowed components to interact with each other. In this paper, we study the use of structured peer-to-peer (P2P) overlays. In structured P2P overlays, nodes communicate with each other in a decentralized manner without being obliged to communicate with a hub or a cloud backend, making the system more reliable and robust. The load is uniformly distributed among the network devices themselves; thus all the data are handled locally mitigating personal information leaks. The devices do not have specific roles and can be service requester and provider at the same time. In this system, trust value for each device is held by several peers in the system. When a node wants to decide about another device who is providing a service, it will ask opinions of the other devices who are holding trust values for the Provider device. These trust holding devices are called Reference Holders. We propose to utilize any of the existing structured P2P system (Distributed Hash Tables) for finding addresses of these devices who are holding the trust values (Reference Holders). Our aim is to decrease the impact of malicious Reference Holders of devices. In DHT, routing scheme that is followed by the peers provides scalability as queries can be resolved with logarithmic complexity in the size of the network, without creating bandwidth bottlenecks or requiring the nodes to be within distance to directly communicate with each other.

Contributions: In this paper, we propose a novel way of choosing reference holders that prevents the malicious nodes to control the reference holders. We propose protocols that provide trust aggregation, service provision and feedback aggregation. Then, we provide closed form of probabilistic analysis and make simulations that manages to give network-wide probabilistic security guarantees. Our trust framework focuses on not only the communication between the known devices but also the communications with new devices that have not contacted earlier. This type of communication becomes more critical when the system has unrecoverable damages with inaccurate services. Devices decide about the unknown nodes according to the second-hand information. Since the system depends on second-hand information, warming period is needed. To the best of our knowledge, this is the first work which provides a trust framework mainly focusing on the communications with stranger devices by utilizing a structured P2P network in IoT environment. Our results suggest that our model is robust against on-off attacks, bad mouthing, bad stuffing and selective attacks. Also, these results emphasize the significance of second-hand information for stranger communications.

In Section 3, we review the literature and we provide background on distributed networks and trust in Section 3 and Section 4 respectively. We explain our system and threat models in Section 5 and we describe our framework with its protocols in Section 6. We analyse the security of our framework in Section 7. Then, we give details about simulations and security results in Section 8 and analyse the framework in terms of computational, storage and communication overhead in Section 9 and we discuss our work design in Section 10. Finally, we conclude our work in Section 11.

2. Related work

IoT environment involves plethora of devices that can contain resource-limited ones which can be captured easily by an adversary due to its limited cryptographic security [10]. The adversary can reprogram the captured node and can cause devastating results in the system. It can become an insider attacker that cryptography cannot cope with. Thus, it is important to provide a trust mechanism that protects the system from catastrophic results.

Some of the well known attacks that is provided by insider devices in a trust management mechanism are on-off attack, selective attack, bad mouthing and ballot stuffing attacks [15]. On-off attack exploits the property of giving more weight to recent recommendations of trust systems. Attacker can behave well for a period of time and regain the trust. A bad mouthing attack occurs when malicious nodes provide dishonest recommendations to drop trustworthiness of honest parties. Inversely, ballot stuffing attack is the manipulation of reputation of compromised devices to increase trustworthiness. Bad mouthing and ballot stuffing attacks are reputation collusion attacks that the malicious nodes are acting in collusion with other malicious nodes. Also, in selective attack, the malicious nodes behave alternatively badly and well between requests when they are providing services.

The existing initial works in the literature are based on beta distribution for updating reputation RRS [16] and RFSN [17]. RRS [16] utilizes both positive and negative reputations and the second-hand information is put under a deviation test. The receiving node decides whether to integrate the deviating information with its current records depending on the level of trustworthiness of the reporting node. These behaviors are also similar to our work but they give more weight to current experience than past experiences. However, this makes the system more prone to on-off attacks. The attacker gains the reputation rapidly and gives harm the network easily. The other work RFSN [17], also utilizes both first-hand and second-hand information. They only allows the positive feedbacks. However, this makes the system vulnerable to ballot stuffing attacks. With only positive information, nodes cannot share their bad experiences which can be catastrophic as each node needs to learn badness from their own experiences [10]. Then, stranger communication accuracy will be very low. Also, malicious nodes survival time can

be extended with the help of excessive number of positive feedbacks of other malicious nodes.

Bao and Chen [18–20] proposed to utilize several trust properties including QoS trust such as honesty and cooperativeness and social trust such as community-interest. The authors evaluate the proposed strategy in the presence of bad mouthing and ballot stuffing attacks[21]. utilizes a simple game approach which achieves Bayes equilibrium[22]. and [23] used Bayesian inference to aggregate self observations into direct trust (first-hand trust). Social similarity metric (friendship, social-contact, and community of interest) is utilized to rate a recommender. Social similarity weighted sum is used to aggregate recommendations into indirect trust (second-hand trust). Bad mouthing and ballot stuffing attacks are tolerated whereas on-off attack is not considered.

Another work in the literature is [9] which is in the same class with TruSD according to [8]. This work is also distributed and event-driven where the trust value is updated after each query. In this work, only first-hand information is utilized during trust calculation. Their work is resilient against on-off attacks but as their does not consider second-hand information, each device depends on its own experience which makes it vulnerable to stranger communications. This can be detrimental for unrecoverable services. To cope with this problem, our TruSD mechanism considers not only first-hand but also second-hand information. TruSD is also resistant against on-off attack, bad mouthing attack, ballot stuffing attack and selective attack. Additionally, all the works above only shows the results for overall communications which are expected to decrease when they only include the case where the trust decision is needed for a stranger communication that the device has no personal experience with. Our analysis not only shows the results for overall communications but also focuses on this case which includes stranger communications between devices who have not contacted earlier.

3. Distributed hash tables (DHT)

P2P networks which have key based routing mechanisms are stated as “Structured P2P” networks. They utilize Distributed Hash Tables (DHT) to provide key based routing. DHT is a distributed version of a traditional hash table which allows each peer to be responsible for a specific part of the content in the network. There are different protocols that construct structured P2P networks (e.g., CAN [24], Chord [25], Pastry [26], Kademlia [27] etc.) which show differences mainly in terms of the organization and the policies applied on the routing and forwarding of the messages/requests across the network.

The nodes and the data are assigned unique identifiers from a large ID space (Node_{ID} and $\text{Object}_{\text{ID}}$). The assignment of these identifiers is provided by applying a hash function on a property of the resource generating an m-bit sequence [28]. $Node_{ID}$ is calculated by applying a hash function on the MAC address of a node. Also, in our case the objects are holding the addresses of the Reference Holders who are holding the trust values. $Object_{ID}$ is provided by applying a hash function on the $Node_{ID}$ of the provider node, $h(Node_P||i)$ for each i where $i \in \{0, 1, 2, \ldots, c\}$. c is a system parameter which shows the number of reference holders per device.

Each object is held by a unique live node, called the object’s responsible node. The responsible node is chosen as the one whose Node_{ID} is the closest to the data’s identifier, in the ID space.

An example of a structured P2P network following the Chord protocol is illustrated in Fig. 1. Chord uses a circular identifier space of 160-bit integers modulo 2^{160} and maps the objects to the live node with the closest Node_{ID} clockwise from the data’s identifier. In this figure, there are four devices which obtain their $Node_{ID}$s that corresponds to logical points at the Chord circle. Objects are hold by the closest nodes in the circle. $Object_A$, $Object_B$ and $Object_C$ are stored in $Node_L$. Similarly, $Object_D$ and $Object_E$ are stored in $Node_N$ and $Object_F$ is kept in $Node_M$. The routing of messages is based on the unique identifiers of nodes ($Node_{ID}$). All nodes maintain a routing table (indicated as finger table in Chord) consisting of the $Node_{ID}$ and the communication address (e.g., IP) of several other nodes. Nodes route messages after advising their routing table by following a forward mechanism that leads progressively closer to the identifier that is each time specified. Let’s suppose $K = 6$, $L = 10$, $N = 33$ and $M = 47$ in Fig. 11 and the circle space is $2^6 - 1$ and then we can specify the finger table for Node K as shown in Table 1. In this topology, if N_K is trying to reach an object with $Obj_{ID=57}$, then it will forward the request to N_M as it is the closest node in its finger table to the final destination

In our framework, we do not specify any type of DHT since we utilize a lookup function which is provided by any type of DHT. Lookup operation is provided by sending the request to the closest node that is known by the querior. Then, it sends this query to the closest node it knows and this continues until it reaches the requested node. This is illustrated in Fig. 2. Let’s suppose that *NodeA* is looking for the destination node *NodeC*. *NodeA* can only communicate with the nodes in blue color. It sends the request to *NodeB*, since it is the closest node to *NodeC* that it knows. As *NodeB* only knows the white coloured nodes then it will send the request to the closest node to the requested node *NodeC*. Every node sends the request to the closest node it knows, until the distance between the destination becomes “0” which means that destination is reached. If the destination node does not exist in the network, then the closest node to the destination will inform the lookup query initiator node that the requested node is not in the network anymore.

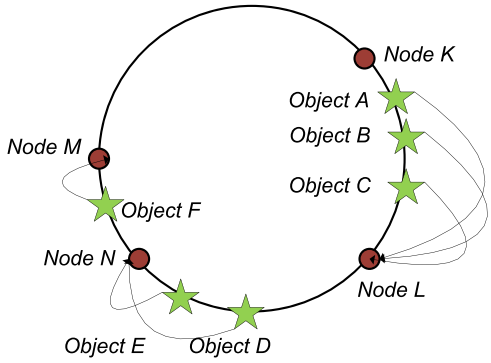

Fig. 1. An example for chord.

Table 1
Finger table of $N_{K=6}$.

Interval	Successor	IP/port
$N_6 + 2^0 = 7$	$N_{L=10}$	$IP_{N10}/Port_{N10}$
$N_6 + 2^1 = 8$	$N_{L=10}$	$IP_{N10}/Port_{N10}$
$N_6 + 2^2 = 10$	$N_{L=10}$	$IP_{N10}/Port_{N10}$
$N_6 + 2^3 = 14$	$N_{N=33}$	$IP_{N33}/Port_{N33}$
$N_6 + 2^4 = 22$	$N_{N=33}$	$IP_{N33}/Port_{N33}$
$N_6 + 2^5 = 38$	$N_{M=47}$	$IP_{N47}/Port_{N47}$

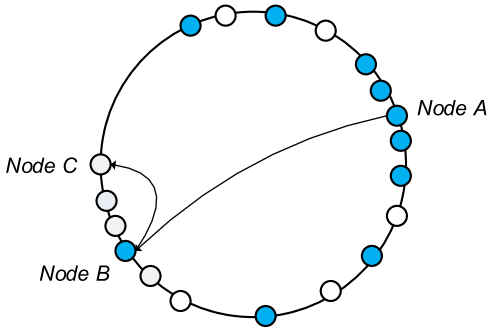

Fig. 2. Lookup operation.

Table 2
System parameters.

Term	Explanation
$Node_{ID}$	ID of a device in DHT layer
TL	Trust List of a device
AT_P	Aggregated Trust value for a provider P
tv_P	Trust Value for a Provider device P, calculated after its own experiences
tv_{RH}	Trust Value for a Reference Holder device RH, calculated after its own experiences
$tv_{RH \to P}$	Trust Value for a Provider device P sent by Reference Holder device RH
$tv_{R \to P}$	Trust Value for a Provider device P sent by Requester device R
θ	Trust Threshold

In our framework, we need the address of Reference Holders and it is obtained by calling the following function $Address_X \leftarrow lookup(X)$ where X is the object calculated by applying the hash as follows $h(Node_P||i)$ as explained above. With the help of this DHT layer, addresses of the devices who are holding the trust values are distributed over the network.

For device P, reference holders are determined with the following hash operations $h(Node_P||1)$, $h(Node_P||2)$, ..., $h(Node_P||c)$ where our system suggests having c number of reference holders per device. The nearest device just after $h(Node_P||1)$ point (clockwise) in the DHT circle, will be the first reference holder for device P, the nearest device after $h(NodeP||2)$ point in the DHT circle will be the second reference holder for device P, ... It is calculated like this for determining c reference holders of device P. In this way, reference holders are determined and the trust values for $Node_P$ will be hold by these devices. Also after this determination the addresses of Reference Holders are stored in DHT layer, and when DHT method lookup ($h(NodeP||1)$) is called then it will return the address of first reference holder of device P.

4. Trust

Trust is a way to measure the fulfillment of the requester. In a service query, a requester device wants to know if the provider device is acting in a way that it is supposed to be. This 'acting' verb is strongly related to the context that trust framework is utilized. For instance, if it is used for file sharing between devices, it will measure if the device provides requested file as it is wanted in a decent time. Thus, the devices evaluate trust values according to their experiences.

In our framework we do not specify the service since we do not want to restrict our framework. It can be utilized for any type of service that will only change the parameters for the trust function T utilized in service provision protocol. Provider device is providing a service and the Requester asks a service from it. According to the service, each device evaluates the nodes and calculates trust values for the nodes that they have communicated.

There are lots of trust calculation functions [29–31] in the literature which can be chosen according to the context. Thus, we do not want to restrict our scheme and we suppose that there is a trust function called T which gathers some parameters according to the context and returns a trust value tv_P for the provider $Node_P$: $tv_P \leftarrow T(parameter_1, parameter_2, \ldots)$. In order to find out the Reference holders of the Provider in DHT layer, the following hash operation is calculated: $h(Node_P||i)$. Hash values are computed for each i where $i \in \{0, 1, 2, \ldots, c\}$. ($c$ is the number of reference holders per device). Each calculated hash value shows one of the Reference Holder device for Provider.

Each device has its own internal trust list TL that contains trust values for any node that it has communicated earlier. After Requester receives the service from $Node_P$, it will add tv_P to its internal trust list TL. Then, for the further communications Requester node will use this list.

Since the aim of this work is to provide a decision before the service is requested from Provider node, Requester node needs to ask opinions of Reference Holder nodes. According to their opinions an aggregated trust value AT_P is calculated. Then a decision algorithm is utilized and AT_P is compared with a threshold θ. Then it is decided as trustable or not. These are explained in details in Section 6.

5. System and threat model

Our framework is above a P2P network that is based on a Distributed Hash Table (DHT). We suppose that DHT will hold the trust values for each device in the framework. We do not specify the DHT technology, it can be any type such as CAN [24], Chord[25], Kademlia[27] etc. operations. It has a dynamic structure that nodes can join and leave the network anytime.

Since we are providing a trust framework for a service provision in a P2P network, a service will be requested by a device whereas another device will provide it. As far as it is a P2P network, all devices can have different roles at the same time. In our system model, devices can have the following roles: Provider, Requester and Reference Holders (RH). Provider is the device whose trust will be queried whereas Requester is the one which wants to communicate with the Provider device if it is trustable. Reference Holders are the nodes who are holding trust values for the Provider node. In our framework, we provide a trust mechanism which aims to provide accurate information for especially a new communication in which a node who wants to receive service from another node that has not been communicated earlier. The main goal is to give accurate decisions about the devices that are not contacted before. In some cases it can be devastating to trust an unknown node. An example is illustrated in Fig. 3. Lets suppose that in the smart home, there is a temperature sensor which needs an update from Internet. Then, the sensor will have the Requester role and named as R. It firstly learns the providers list PL from an indexed table (which is not in our scope) and learns that smart watch and camera can provide this service (point 0). However, the sensor did not communicate with any of them earlier. Let's say it picks smart watch as the Provider P from the provider list. Then, R firstly needs to communicate with reference holders of the smart watch and receive their opinions about it. In order to reach the reference holders, Requester needs to learn the communication address of reference holders via lookup function of DHT $Address_X = lookup(X)$ (point 1), in which X is the object calculated by applying the hash as follows $h(Node_P||i)$. Then, R communicates with all reference holders via Reference Query Protocol (point 2). Then, it calculates an aggregated trust value, compares this value with a threshold and decides to trust or not. If R decides to trust, it needs to learn the communication address of Provider P via lookup function of DHT $Address_P = lookup(P)$ (point 3). Then, it requests the service via Service Provision Protocol (point 4). After R reaches the service, it evaluates the service and provides a feedback report. Then it sends the reports to all reference holders via Feedback Report Protocol (point 5). If the smart watch is malicious, it can send a malware which will be devastating and not possible to restore. Thus, we propose a trust framework for such type of stranger communications which decreases the possibility of such devastation.

We suppose that the main goal of an attacker is to obtain a control over the network to provide as many malicious services as possible. He aims to manipulate the benign nodes to trust and ask for services from malicious nodes. We allow the adversary to capture k out of N devices

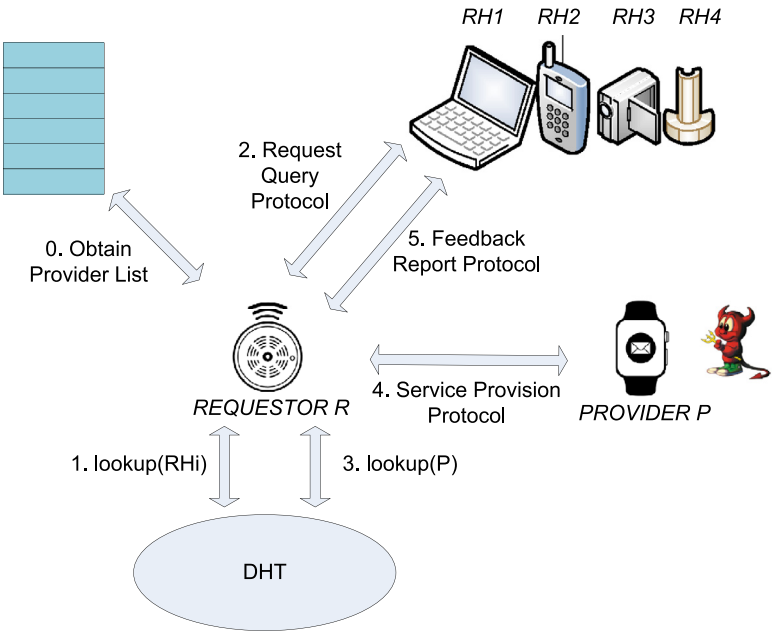

Fig. 3. System and threat model.

and manage to direct most of the benign nodes to malicious nodes. If an attacker manages to become a reference holder, he can provide positive references for his collaborators whereas negative ones for benign nodes in order to manipulate. Attacker can also compromise the legitimate devices and prevent their ability to provide services. We suppose that they can collaborate and they have perfect instant communication between themselves. They can control their own behaviours, they can decide to lie but they cannot reach to other devices messages.

The compromised nodes can provide on-off attack, selective attack, bad mouthing and ballot stuffing attacks [15]. On-off attack exploits the property of giving more weight to recent recommendations of trust systems. Attacker can behave well for a period of time and regain the trust. Our TruSD model handles this issue by giving more weight to the previous trust values. A bad mouthing attack occurs when malicious nodes provide dishonest recommendations to drop trustworthiness of honest parties. Malicious devices when they have requestor or reference holder roles, are sending negative feedbacks for benign devices. Similarly, ballot stuffing attack which is the manipulation of reputation of compromised devices to increase trustworthiness. In our system, malicious devices are sending positive feedbacks for captured nodes. TruSD is resistant against these attacks as the requestor compares its own experience with the reference holder's recommendation. Then, the system can easily notice the trial of the manipulation of malicious nodes. Also, selective attack is provided with the help of camouflage probability that the malicious nodes behave alternatively badly and well between requests when they have provider roles. This attack's affect is diminished in TruSD, since trust decisions are provided by considering not only one but also several reference holders' recommendations. We do not consider jamming and DDoS attacks.

6. Protocols

In order to provide a trust framework for a P2P network, we propose the following protocols that consist of the messages between the nodes who have the roles as Requester, Provider or Reference Holder. Initially, Requester device communicates with the Reference Holders and asks for their opinion about the Provider device and aggregates trust values. Then if the Provider node is trustable, Requester receives the service from the Provider Node and sends the feedback report to the Reference Holders. The details of the protocols are provided in the following subsections. The symbols utilized in the following sections are shown in the following table.

6.1. Reference query protocol

When a device needs a service, it will request this service from one of the devices who can provide it. Due to their roles, the device who wants the service will be Requester and the other device who is capable to provide the service is the Provider. Initially, Requester needs to decide if the Provider device is trustable or not. In order to come up with a decision, he needs to ask opinions of Reference Holders who are holding trust values for the Provider device. The communication messages for reference query is shown in Fig. 4.

In order to learn which devices are the Reference Holders for the Provider device, Requester calculates hash values $h(Node_P||i)$ for each i where $i \in \{0, 1, 2, \dots, c\}$. ($c$ is for the number of reference holders per device). Requester node learns the addresses of the reference holders with the lookup query in DHT layer, $Address_{h(Node_P||i)} = lookup(h(Node_P||i))$. After Requester learned the addresses, it communicates with all the reference holders. In Fig. 4, we show the communication between the Requester and one of the reference holder *RH*s, but the Requester device makes the same communication with all *RH*s of $Node_P$.

Requester starts the protocol with sending a query message that contains his ID ($Node_R$), ID of the provider $Node_P$, i and the request for reference *ReqRef*. *RH* firstly calculates $h(Node_P||i)$ to see if he is really a *RH* for $Node_P$. Then it checks its own trust list TL_{RH} if a trust value exist for $Node_P$. At initial communications, it is possible that *RH* does not receive any feedback and it will not have any trust values for $Node_P$. In this case, reply message contains *NACK* otherwise it contains $tv_{RH \to P}$ which is the trust value for the Provider P given by the *RH*. After Requester receives the reply message, it waits for all messages from all *RH*s of $Node_P$. Then, it calculates aggregated trust value AT_P for $Node_P$ as follows:

$$AT_P = \frac{\sum_0^c tv_{RH \to P} \times tv_{RH}}{c} \tag{1}$$

tv_{RH} is utilized from the Requester's own list TL_R. If there is not any entry for *RH* in its list, a default initialization trust value for an unknown reference holder is utilized in calculations. Also, c is the number of reference holders for $Node_P$. Requester compares aggregated trust value AT_P with a trust threshold θ, if it is under this value it decides not to trust the Provider Node, otherwise it decides to trust. This equation reduces the

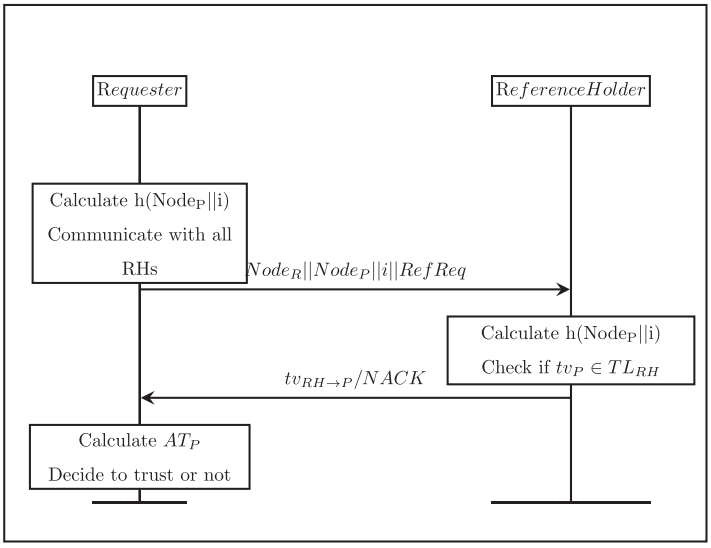

Fig. 4. Reference querypProtocol.

effects of selective attacks where the malicious nodes acting sometimes bad sometimes good. As this equation suggests deciding according to several reference holder's decision, it decreases the effect of malicious node.

At the very beginning of the system since the nodes does not have enough communications, the Requester mostly receives *NACK* messages from all the reference holders. In this case, it will trust with probability *pb*.

All the reference query protocol is mostly critical for the nodes which have not contacted earlier, but of course the same strategy can be applied for the known nodes (mostly for the untrusted ones). It is possible that Provider is not contacted for awhile and Requester wants to learn about the current trust value about it. Then it will calculate the trust value by considering aggregated trust value AT_P and its own past experience tv_P as follows:

$[(1-w)\times AT_P]+[w\times tv_P]$. In this formula, w determines the weight for its own past experiences. The higher w means that past experiences are considered as more important than aggregated opinions of others. If it is fixed that $w=1$, then Requester does not ask for references if it has contacted with a node earlier.

Additionally, the known nodes who are evaluated as untrusted are not plunged into darkness forever, if the Requester node does not want to operate reference query protocol. If it is tv_P value is under θ service is still requested with a probability *ps*, which gives them a chance. All these operations are summarised in the algorithm in the following algorithm.

if $Dev_P \notin TL$ **then**
  if Any $RH_i \in TL$ **then**
    Calculate AT_P
  else
    Trust device P with probability p
  end if
else
  if $tv_P > \theta$ **then**
    Trust device P
  else
    Trust device P with probability ps
  end if
end if

6.2. Service provision protocol

If Requester decides that Provider node is trustable, it will request the service as shown in Fig. 5. Requester node sends service request *ReqSer* and its node ID $Node_R$ to the Provider Node. Then Provider checks if a

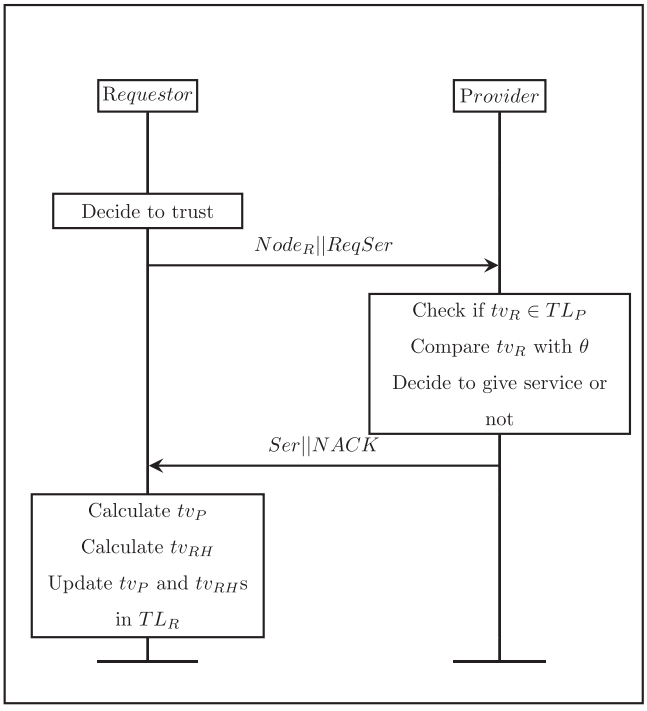

Fig. 5. Service provision protocol.

trust value tv_R exists in its trust list TL_P. If the Provider device communicated with Requester device before-time, then it has evaluated Requester and has calculated a trust value tv_R. Provider compares this value with a threshold θ to decide if Requester is trustable or not. If Requester is not trustable, it provides the service with p probability, if it decides not to give the service then it sends *NACK* message. If it is not in the list or its *tv* shows that it is trustable, it will give the requested service *Ser*. Since we do not specialize the service we suppose that service is provided in one message, but it is also possible that more messages are exchanged between the devices. Our attitude has the tendency of not giving services to untrusted devices since there can be fragile resources that can be affected by any malicious communication.

After the service is provided by the Provider node, Requester evaluates the service according to the trust function $T(parameter1, parameter2, \ldots)$ which takes several service specific

Table 3
Attacker behaviours

Security Threat	Service Provision / Camouflage probability (selective attack)	Reputation for Benign Nodes (bad mouthing)	Reputation for Malicious Cooperatives (ballot stuffing)	TruSD property
Malicious collectives	0%	Normal	High	Eq. 2
Malicious collectives with camouflage	*f*%	Normal	High	Eq. 1
Driving down the reputation of a reliable peer	0%	Low	High	Eq. 2

parameters. This function returns a trust value tv_P for the Provider node. Additionally, Requester node evaluates the RHs according to their suggestions. The trust values for Reference Holders, tv_{RH}s are calculated according to the following formula:

$$tv_{RH} = 1 - (\mid tv_P - tv_{RH \to P} \mid) \tag{2}$$

$tv_{RH \to P}$ shows the trust value for Provider, that is sent by corresponding reference holder *RH* in Reference Query Protocol. Also, tv_P is the trust value for Provider that is calculated by Requester. If RH suggested the Provider node as trustable and Requester is satisfied with the service of Provider node, then RH is evaluated as trustable. Similarly if the opinion of Requester about the Provider node is conflicted with the suggestion of RH then it is evaluated as untrusted. This equation provides resistency for bad mouthing and ballot stuffing attacks as these attacks are trying to manipulate the decisions by giving bad feedbacks to good ones and good feedbacks to bad ones. As this equation compares its own experience with the suggestion of the Reference Holder, different suggestions will be punished by decreasing its trust value in *TL*. For instance, if the malicious *RH* sends feedback for the Provider node as 0.1 (bad mouthing attack) whereas the Requestor experienced the service and evaluated trust value as 0,9 then it will evaluate the trust value for *RH* as $1 - (\mid 0.9 - 0.1 \mid) = 0.2$. Similarly, if *RH* is providing ballot stuffing attack and sends a feedback for a malicious Provider as 0.9 and the Requestor experienced the service and evaluated trust value as 0,1 then it will evaluate the trust value for *RH* as $1 - (\mid 0.1 - 0.9 \mid) = 0.2$.

According to an aggregation function *F*, the trust values for both provider and reference holders are updated in the list TL_R. This function calculates the current trust value as follows:

$$tv_{(RH||P)k} = \frac{(tv_{(RH||P)k-1} x(k-1)) + tv_{(RH||P)}}{k} \tag{3}$$

The aim of this equation is to update trust values by considering the previous experiences. This function updates the current trust value for RH,tv_{RHk}, and provider,tv_{Pk}, by considering the calculated tv_{RH} in Eq. 2 and tv_P calculated from function *T* and the previous trust values tv_{RHk-1} and tv_{Pk-1}. In this equation, *k* refers to the number of communications between these entities. tv_{Pk-1} or tv_{RHk-1} is multiplied with $k-1$ to preserve the effects of the past experiences. This preservation is important which makes our model resistant to on-off attacks since this attack exploits the forgetting property of trust systems. When the attacker starts to behave well, devices are deceived immediately and start to trust the compromised devices. In order to be resistant to this attack, we give more weight to the previous experiences. This function is called for both provider and reference holders and accordingly their trust values are updated in the internal trust list of Requester TL_R. At the end of this protocol, a service is provided to the Requester node and the Provider is evaluated according to this service.

6.3. Feedback report protocol

After Requester node receives the service, it generates a feedback report for the Provider node, FB_P, and sends this report to all Reference holders of Provider as shown in Fig. 6. This report contains the trust value $tv_{R \to P}$ that is calculated in the previous section. Requester sends its own node ID $Node_R$, feedback report FB_P and the Provider's node ID $Node_P$ with *i*. Then, reference holder firstly computes $h(Node_P||i)$ to see

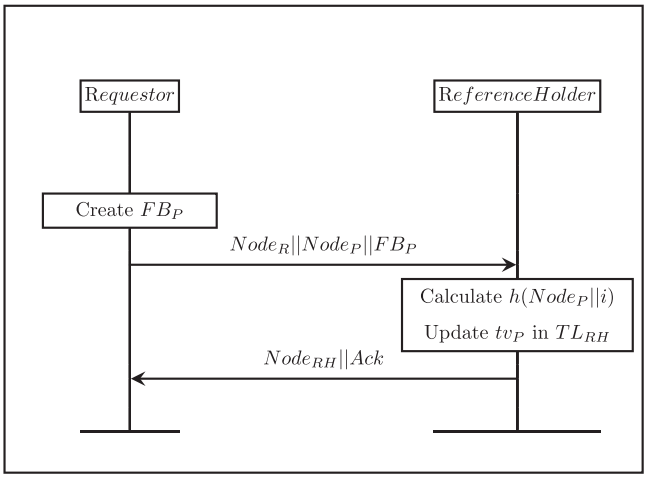

Fig. 6. Feedback report protocol.

if it is really the reference holder for *P*. Then it calculates the current trust value tv_{P_k} for $Node_P$ as follows:

$$tv_{P_q} = \frac{(tv_{P_{q-1}} x(q-1)) + (tv_{R \to P})}{q} \tag{4}$$

Previous trust value for Provider $tv_{P_{q-1}}$ is added up with the trust value $tv_{R \to P}$ in the feedback report. In this equation, *q* shows the number of updates for the trust value. $tv_{P_{q-1}}$ is multiplied with $q-1$ in order to preserve the effects of the past experiences. After the calculation, Reference Holder sends an *Ack* message back to Requester node to inform that its feedback is considered.

6.4. An overview

Security threats scenarios over distributed systems are categorised in [32]. Our threat model considers "collective malicious" attacks instead of individual ones since it requires more capable attackers and it is more difficult to cope with. We also consider "malicious collectives with camouflage" and "driving down the reputation of a reliable peer" attacks. The malicious attackers behaviours can be analyzed as service provision behaviour and reputation provision behaviour. Service provision shows if the attacker provides a good or bad service. Since some attacker behaviour suggest being camouflaged, this depends on the camouflage probability. Thus, service provision behaviour can be analyzed in terms of camouflage probability. Reputation providing behaviour can also be categorized as the attitude for benign nodes and malicious cooperatives. The following Table 3 shows these behaviours and the property of TruSD that meets the ball as a defence player.

According to this table, our model is resistant against malicious collectives with the help of Eq. 2 since this attack is provided by ballot stuffing. On the other hand, malicious collectives with camouflage attack is handled with Eq. 1 where a selective attack is provided. In driving down the reputation of a reliable peer attack, Eq. 2 provides the resistency of TruSD. This paper does not consider on-off attack, but it is handled with Eq. 3.

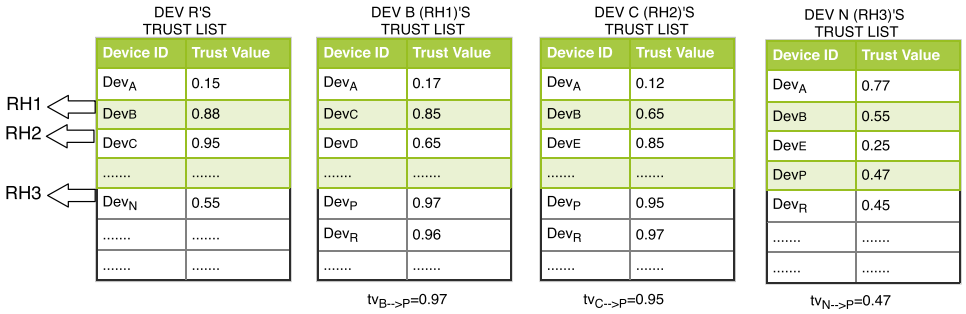

Fig. 7. Trust lists before service is requested from the provider.

Let's suppose that requester device R is trying to decide about the provider device P. Reference Holder's for device P are *DevB, DevC* and *DevN*. Trust Lists of these devices are shown in Fig. 7. In this scenario, $\theta = 0.6, kb = 10, kc = 15, kn = 20, qb = 3, qc = 4 and qn = 5$. All the steps can be summarised as follows:

- *R* operates the Reference Query Protocol with *DevB, DevC* and *DevN*.
- *R* obtains $tv_{B\rightarrow P}$, $tv_{C\rightarrow P}$ and $tv_{N\rightarrow P}$.
- *R* calculates
  $AT_P = (tv_{B\rightarrow P} x tv_B) + (tv_{C\rightarrow P} x tv_C) + (tv_{N\rightarrow P} x tv_N)/3$
  $AT_P = ((0.97x0.88) + (0.95x0.95) + (0.47x0.55))/3 = 0,66$
- *R* compares AT_P and θ.Since $\theta = 0.60$ and $AT_P > \theta$. Then *R* will trust *P*, request and obtain the service.
- After the service obtained, *R* evaluates the provider *P* according to a trust function:
  $tv_P = T(param1, param2, param3, param4) = 0.98$.
- *R* adds $tv_P = .98$ to its Trust List.
- *R* also evaluates the Reference Holders.Let's suppose that it is the 10th time that *R* communicates with Dev_B, whereas it is 15 for Dev_C and 20 for Dev_N ($kb = 10, kc = 15, kn = 20$). Then,
  $tv_{B10} = 1 - (tv_P - tv_{B\rightarrow P}) = 1 - (0.98 - 0.97) = 0.99$ [1–41].
  $tv_{C15} = 1 - (tv_P - tv_{C\rightarrow P}) = 1 - (0.98 - 0.95) = 0.97$
  $tv_{N20} = 1 - (tv_P - tv_{N\rightarrow P}) = 1 - (0.98 - 0.47) = 0.49$
- *R* updates its Trust list with the following trust values for the reference holders:
  $tv_B = \frac{(tv_{B9} x 9) + tv_{B10}}{10} = ((0.88x9) + 0.99)/10 = 0.89$
  $tv_C = \frac{(tv_{C14} x 14) + tv_{C15}}{15} = ((0.95x14) + 0.97)/15 = 0.951$
  $tv_N = \frac{(tv_{N19} x 19) + tv_{N20}}{20} = ((0.55x19) + 0.49)/20 = 0.54$
- Then *R* sends its feed back report to the reference holders.
- Each reference holder updates the tv_P in its own trust list considering the feedback report. Let's suppose that tv_P is updated 3th time on Dev_B, whereas it is 4 for Dev_C and 5 for Dev_N. ($qb = 3, qc = 4, qn = 5$). Then, all devices updates tv_P according to the following formula:
  $tv_P = \frac{(tv_{Pq-1} x (q-1)) + (tv_{R\rightarrow P})}{q}$
  - Dev_B calculates tv_P as follows:
    $tv_P = \frac{((0.97x4) + (0.98))}{5} = 0.972$
  - Dev_C calculates tv_P as follows:
    $tv_P = ((0.95x9) + (0.98))/10 = 0.953$
  - Dev_N calculates tv_P as follows:
    $tv_P = ((0.47x14) + (0.98))/15 = 0.50$
  - Trust list tables after the service provision is provided in Fig. 8.

7. Protocol analysis

In Reference Query Protocol, Reference Holder can be a liar or malicious device which wants to manipulate the Requester node to trust malicious nodes via sending inaccurate tv_P value to the Requester. Additionally, he can try to hide himself by sometimes giving right values. This is called camouflage probability that he sends accurate trust values with *f*% rate. Thus, in our protocol only a probabilistic guarantee can be provided to the Requester node. If there are *m* malicious nodes out of *N* devices in the system, then Requester node receives the right tv_P value from all the reference holders with the following probability:

$$((1 - \frac{m}{N}) + (f \times \frac{m}{N}))^c \quad (5)$$

where *c* is the number of reference holders per device. According to aggregated tv_P values Requester calculates aggregated trust value AT_P which not only depends on tv_P, but also tv_{RH}s (shown in Eq. 1). The tv_{RH} can be calculated as in Eq. 2 in the previous communications or the reference holder can be contacted as a Provider in previous communications and evaluated after a service according to function *T*. Since our framework is a P2P network and each node can have different roles in different protocols and query order is not deterministic, it is not possible to give an exact guarantee formula for having accurate AT_P calculation.

In Service Provision Protocol, service is provided in the last message. If Provider node is malicious (or captured), Requester node cannot reach the accurate service. Thus, our protocol suggests that Requester node can receive the right service with $(1 - \frac{m}{N}) + (f \times \frac{m}{N})$ probability.

Similarly, in Feedback Protocol, if Requester node is captured, inaccurate feedback will be utilized in trust value updates. Thus, the Reference Holder will receive the accurate feedback from a Provider with $(1 - \frac{m}{N}) + (f \times \frac{m}{N})$ probability.

Since the same node can have different roles in different protocols and these roles determine the order of filling the trust lists *TL* of nodes, we cannot provide exact functions for guarantees per protocol. Additionally, there are possible malicious activities that can trap the system. For instance, in the scenario in Fig. 3, let's suppose the requestor sensor R claims that it needs an update from Internet. It learns the provides list *PL* from an indexed table and learns that smart watch can provide this service. Let's suppose sensor is compromised and controlled by an attacker, then it can follow the feedback report protocol and can provide a negative feedback report to reference holders decreasing the trust levels of the smart watch.

In addition to these activities several attacks are also possible to the protocols since all messages are clear. Man-in-the-middle attack, modification, eavesdropping are some examples of possible actions. These activities cannot be completely blocked. That is why we give probabilistic guarantees under compromised node attack. This attack also covers the unsecure communication between the nodes. The device or the communication can be captured by the attacker. We can only give network wide probabilistic guarantees which show that until 60% devices are captured feedback mechanism works properly. In order to show the performance results we provide the simulation details in the following section.

8. Simulation details and results

Our protocols performance analysis are provided via simulations. In the presence of an attacker in our system, malicious devices can lie and manipulate the benign nodes. Since our protocols cannot provide se-

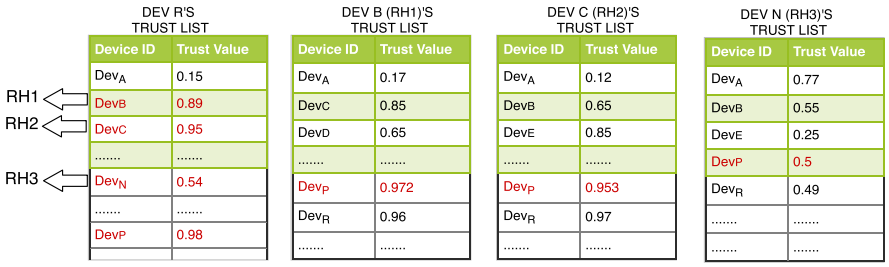

Fig. 8. Trust Lists after service is provided to the requester.

curity guarantees itself, we can only make probabilistic analysis. However, the probabilistic analysis does not provide deterministic guarantees since it depends on the order of receiving different roles in different queries. Thus, in order to see the network-wide performance of our protocols we provide simulations.

In our simulations, the malicious devices are in collaboration that they know each others. Their main goal is to manipulate the benign nodes to trust and ask for services from malicious nodes. According to the roles, malicious behaviours in the simulation can be summarized as follows:

- Provider: During service procurement, it can give good services with f probability. This is the camouflage probability that it sometimes provides good services and sometimes bad.
- Requester: While sending the feedback report to the reference holders it can give positive feedback for malicious nodes with $1-f$ probability and negative feedback for benign nodes with $1-f$ probability.
- Reference Holder: While providing references, it can give positive references for its collaborators with $1-f$ probability whereas negative ones for benign nodes with $1-f$ probability in order to manipulate.

Simulation of this framework is implemented by using Python for coding. We run the simulation on a PC which has 8 GB RAM and Intel Core i7-3610QM 2.3 GHz CPU. In our simulations there are 100 nodes which are part of a P2P network. Interconnected peers can either issue a query for a service, ask for a reference or give service/references. Our network consists of benign and malicious nodes. We suppose that in each run, there are 3000 queries where a random node is chosen as the Requester and the service can be supplied by Provider nodes. At each query a similar scenario in Fig. 3 is operated. There is a Provider List *PL* that consists of the providers for the query. We suppose that for each query, we choose a random node from all devices and it became a requestor. Then, provider list is generated with 10 random nodes. At each query, Requester chooses a node from *PL* and makes a decision to trust or not. If Requester decides to trust it requests the service, otherwise it chooses another node from *PL* and calculates trust value again. It continues to pick a node from *PL* until it receives the service. The initial 1000 query is not involved in results since the system is not stabilized and most of the devices have not contacted yet. Each simulation is run 100 times and each result is obtained by calculating the average of these runs. Also, if requestor is a benign node and receives a service from a provider (which is a benign device or giving the service on purpose for camouflage), it sends positive feedback ($fb = 1.0$) to the reference holders of the provider. If it cannot receive the service, then it sends a negative feedback ($fb = 0$). Malicious nodes will behave inversely and try to falsify the benign nodes.

We evaluated our framework in terms of three metrics: accurate trust decisions rate (ACC), choosing malicious nodes rate (FDR) and receiving bad services rate (GBS).

ACC shows the rate of the accurate decisions that benign nodes are deemed as trusted and malicious nodes are considered as untrusted. In our case, *TP* is the number of "trusted" decisions for benign nodes whereas *TN* is the number of "untrusted" decisions for malicious nodes. Similarly, *FP* is the number of "trusted" decisions for malicious nodes whereas *FN* is the number of "untrusted" decisions for benign nodes. In that vein, ACC is calculated as follows:

$$ACC = (TP + TN)/(TP + TN + FP + FN). \tag{6}$$

False Discovery Rate (FDR) is the probability of incorrectly choosing a malicious node. It is calculated as:

$$FDR = FP/(TP + FP). \tag{7}$$

Receiving Bad Service (GBS) is the probability of the Requester node requests the service from an untrusted node and receive bad service.

$$GBS = BadService/TotalQueries \tag{8}$$

According to these metrics, we provide performance analysis under different number of malicious nodes (m), different number of reference holders (c) and different camouflage probabilities (f). As it is explained in Section 7, c, f and m are the main factors in protocol analysis.

In Fig. 9, simulation results are illustrated for different rates of malicious nodes. In this simulation, each query is generated by picking a random device. We suppose that this device is the Requester and looking for a service. For each query, a Provider List *PL* is generated by picking random devices. These devices can be either a device who has already communicated with Requester earlier or a device which will have a stranger communication with Requester that they have not contacted earlier. Camouflage probability is considered as $f = 0$ during the simulation. According to Eq. 5 in Section 7, when m increases the performance of the framework gets worse. The results suggest that until 60% of the devices are captured, the results are perfect. *ACC* is nearly 1 and *GBS* and *FDR* are 0. When more than 60%s of devices are captured, *GBS* and *FDR* starts to increase and *ACC* starts to decrease as expected. If all the nodes become malicious, *ACC* decreases to 0 and system gives always wrong decisions. Similarly, *FDR* and *GBS* increases to 1 and the network always gives bad services.

Fig. 10 shows the results for different number of reference holders per device. These results are showing the behaviour of the framework for stranger communications. When a node asks for a service, the provider list *PL* is generated by picking random devices which will have a stranger communication with Requester that they have not contacted earlier. During the simulations, the camouflage probability is $f = 0$ and there are 20% malicious nodes in the network. When the percentage of reference holders per device increases, it is expected to have better results since the Requester node gives more accurate decisions by considering more references. This can also be interpreted from the Eq. 5 in Section 7. According to the simulations, after 3 reference holders, the framework gives perfect results. *ACC* is nearly 1 and *GBS* and *FDR* are 0. These results suggest that 3 reference holders are enough for obtaining accurate services. That means in order to obtain a accurate service from an unknown node, each device needs to communicate with 3 reference holders.

The results for different camouflage probability is shown in Fig. 11. These results show the success of our framework on bootstrapping communications. *PL* is generated by picking random devices that have not

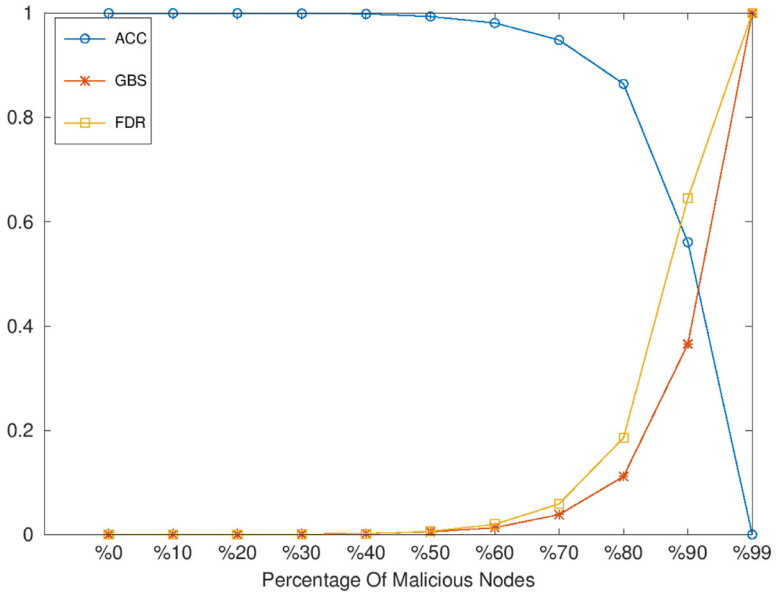

Fig. 9. Performance under different percentage of malicious nodes.

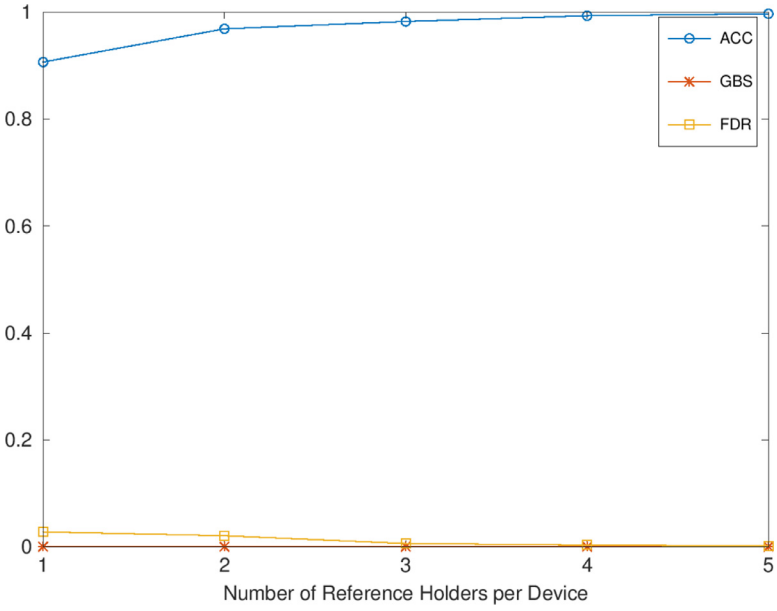

Fig. 10. Performance under different percentage of reference holders per device in stranger communications.

contacted with Requester earlier. There are 20% malicious nodes in the network. During the simulations, the camouflage probability *f* is varied. When a malicious node becomes a reference holder, he gives accurate trust value with probability *f*. Similarly, if a malicious node become a Provider in a query, he gives good service with probability *f*. It is expected to have wrong decisions about the malicious node when *f* increases. According to the simulations, the framework gives perfect results until f=30%s. After then *FDR* starts to increase and *ACC* starts to decrease. When *f* becomes 1.0, malicious nodes always gives good services and they cannot be differentiated. Thus, *GBS* is not affected since when *f* increases Requester still receives good service even it asks from malicious nodes. As there are 20% of malicious nodes, *FDR* increases to 20%s and *ACC* decreases to 80%s since any of the malicious nodes cannot be detected.

Another performance related issue is about number of steps until it decides to trust. If Requester device decides that the candidate Provider is not trustable, it will pick another device from the Provider List and aggregate trust values from reference holders for this device. It continues until it decides to trust. Thus, until "trust" decision is found, service is not queried from the Provider nodes, but references are aggregated from reference holders. We also make analysis to see in how many steps they can reach the service during the simulations and the analysis suggest that Requester can reach the accurate service at most in two steps. That is to say, it does not bring noteworthy burden for the nodes.

9. Performance analysis

In this section we provide computational, communication and memory analysis for the proposed protocols and operations in our framework TruSD. We suppose that our framework is working on top of DHT generic interface, we present the additional costs on DHT network.

9.1. Computational cost

According to our protocols at each service query, if a node has a requestor role, it makes one hash operation, c multiplication, c addition and one division operation for aggregated trust value AT_P calculation

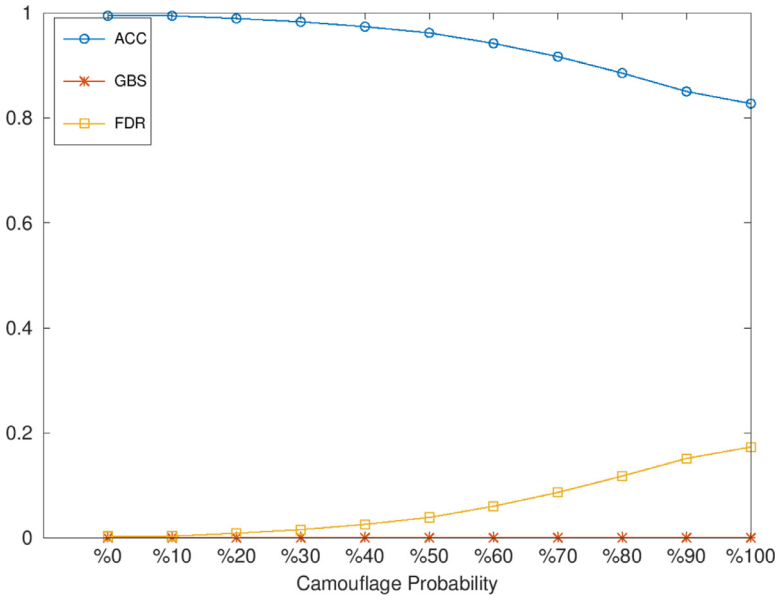

Fig. 11. Performance under different camouflage probability(f) in stranger communications.

Table 4
Computational overhead of TruSD.

Role	Computations
Requestor	$1H + 1D + c * (M + A + S)$
Reference Holder	$1H + 1M + 1D + 1A$
Service Provider	–

and c subtraction operations for trust list update. Similarly, if a node has a reference holder role, it operates one hashing, one multiplication, one addition and one division operation. If a node has a service provider role, then it needs to make comparisons to find the trust value for the requestor device in its trust list. Since comparison is a basic XOR operation, we do not even incorporate it into the Table 4.

According to the analysis in Fig. 10, when $c = 3$ framework gives favorable results. Thus, a few number of mathematical operations and one hashing is operated for each query which can easily be handled by primitive IoT devices. For hashing, there are lots of lightweight techniques [33] which can be applied on our trust framework.

9.2. Memory consumption

In our framework each node needs to hold trust list that contains device ID Dev_{ID} and trust value for the devices.

If there are N number of nodes in the network and c reference holders, then each node needs to be reference holder for c number of nodes which means that it holds c number of trust values. Additionally, each node holds trust values for the nodes that they have communicated earlier. If there are z number of communicated nodes earlier then there will be $c + z$ entries in this table. Each entry has a node identifier (eg. 32 bit) and a trust value (eg. 64 bit). Then total storage will be $96 \times (c + z)$ bits. According to the analysis in Fig. 10, when $c = 3$ framework gives favorable results. Then, the table size depends on z which shows the number of communicated nodes. According to our scenario in the simulations, even if it is supposed that all (100) nodes have communicated by a node and a trust value is entered to the table for each communication, then approximately 1, $1KB$ is needed to store this table which is an affordable amount of storage for an IoT device.

9.3. Communication cost

Since our framework is on top of DHT, join and lookup operations are utilized from this framework. In Chord, such a join procedure requires $O((logn)^2)$ messages where n shows the number of nodes in the network. Also, lookup operation demands $O(logn)$ number of messages [34].

In TruSD, all the protocols consist two messages. Then, for each service query there will be $2 \times ((c \times 2) + SM)$ messages in total. The reference query protocol will be performed with all the reference holders. Also, as we do not specialize the service, it can be provided in several messages. Thus, SM represents the number of messages that is needed to provide the service. At the end, feedback protocol is operated with the reference holders. According to the analysis in Fig. 10, when $c = 3$ framework gives favorable results. Then, for each service query, 14 messages will be exchanged in total if the service is provided in one message. Requestor device will transmit seven and receive seven messages, whereas reference holder will transmit six and receive six messages and provider will transmit and receive one message.

In terms of energy consumption, each device is receiving and transmitting a few messages which is eligible for sensors (as energy to transmit is about $59.2\mu J/byte$ and energy to receive is $28.6\mu J/byte$ [35]). Also, as far as our protocols have only simple mathematical operations and just one hashing operation (for each requestor role and reference holder role) for each query, it is favorable even for sensors (as hashing costs $5.6\mu J/byte$ and mathematical operations give negligible costs [35]).

10. Discussion

In our framework, the number of queried reference holder is adjustable. There is a trade off between the number of reference holders which are asked for opinions, and communication overhead. Requester node can decide how many Reference Holders it wants to ask for opinions. Our results suggest that just three Reference Holders are enough for receiving accurate service, which does not take a notable communication burden. Still, it is possible to define another threshold which is utilized to show highly trusted nodes. Then, in order to decrease the communication overhead, Requester can choose to ask just one Reference Holder (instead of 3) which is highly trusted. Additionally, it is important to state that Requester node does not have to communicate with the Reference Holders to obtain a trust value if it has communicated with the Provider earlier. It can utilize its own experiences expressed as

trust values in its trust list *TL*. This brings an advantage that, in such service provision systems where most of the nodes communicate with the other nodes at least ones after awhile, devices do not have to ask for opinions of Reference Holders.

Our protocols provide a trust framework which can be utilized in service discovery. We do not specialize the service that it can be any type. If the service requires confidentiality and it has existing setup for this requirement, it is possible to utilize TruSD on top of an existing secure communication network. Then, the system will provide services from trusted devices via confidential links.

Also, we argue that our framework is important for sensitive and unrecoverable data. It is possible to state that this type of data is not preferred to be transferred via an open channel. if it is required and devices have enough resources for this operations, it is possible to create a shared key among devices (i.e. Diffie Helman Key Exchange Protocol with certificates) and the service can be transferred through this link in Service Provision Protocol. The certificates are utilized to prevent man-in-the middle attack and they can be provided via an off-line CA (Certificate Authority) before the devices participate the system

Our model is not only considering constant but also camouflage attacks. The attackers are not always acting in the same pattern. There is a camouflage probability which makes the attacker unpredictable. This camouflage property not only provided during the service provision but also during giving the feedbacks. Making our system more resilient against more clever attacks such as collusion attacks [36] can be considered as a future work.

11. Conclusion

In this paper, we propose a framework that enables trusted communication among devices in service discovery. Our trust framework focuses on not only the communication between the known devices but also the stranger communications with new devices that have not contacted earlier. This type of communication becomes more critical when the system has unrecoverable damages with inaccurate services. All devices in the system can be provider and also the requester of a service at the same time that they constitute a P2P environment. Our framework works on top of a structured P2P network based on DHT. Any type of structured P2P protocol can be utilized, since we only use lookup function of a DHT. By utilizing DHT, we propose a novel way of choosing reference holders that prevents the malicious nodes to control the reference holders. We propose protocols that provide trust aggregation, service provision and feedback aggregation. Attacker provides on-off attack, selective attack, bad mouthing and ballot stuffing attacks in the threat model. Then, we provide closed form of probabilistic analysis and make simulations which give network-wide probabilistic security guarantees. According to the results, until 60% of the devices are captured, nearly all decisions are accurate. Also, just three reference holders are enough to get accurate services through the network. As the framework is provided for IoT devices, we also provide analysis in terms of memory, computational cost and communication overhead which show that our framework is affordable for IoT devices. As a future work, our model's security can be verified by using formal verification techniques as represented in the following papers [37–41]. Additionally, we plan to make experiments in a real testbed. This will make our results more realistic.

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.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2020.107318.

CRediT authorship contribution statement

Kübra Kalkan: Conceptualization, Methodology, Software, Writing - original draft, Visualization, Investigation, Writing - review & editing. **Kasper Rasmussen:** Conceptualization, Methodology, Visualization, Investigation, Writing - review & editing.

References

[1] I. Walden, G. Noto La Diego, Contracting for the internet of things: looking into the nest, Queen's University of Belfast (2016) 219.
[2] M. Raya, P. Papadimitratos, V.D. Gligor, J.-P. Hubaux, On data-centric trust establishment in ephemeral ad hoc networks, in: Proc. INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, IEEE, 2008, pp. 1238–1246.
[3] W. Feng, Z. Yan, Mcs-chain: decentralized and trustworthy mobile crowdsourcing based on blockchain, Future Generation Computer Systems 95 (2019) 649–666.
[4] Z. Yan, P. Zhang, T. Virtanen, Trust evaluation based security solution in ad hoc networks, in: Proceedings of the Seventh Nordic Workshop on Secure IT Systems, 14, Citeseer, 2003.
[5] W. Tang, Z. Yan, Cloudrec: A mobile cloud service recommender system based on adaptive qos management, in: 2015 IEEE Trustcom/BigDataSE/ISPA, 1, IEEE, 2015, pp. 9–16.
[6] 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.
[7] 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.
[8] 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.
[9] C.V. Mendoza, J.H. Kleinschmidt, Mitigating on-off attacks in the internet of things using a distributed trust management scheme, Int. J. Distrib. Sens. Netw. 11 (11) (2015) 859731.
[10] J. Sen, A survey on reputation and trust-based systems for wireless communication networks, arXiv preprint arXiv:1012.2529 (2010).
[11] G. Lize, W. Jingpei, S. Bin, Trust management mechanism for internet of things, China Commun. 11 (2) (2014) 148–156.
[12] S. Namal, H. Gamaarachchi, G. MyoungLee, T.-W. Um, Autonomic trust management in cloud-based and highly dynamic iot applications, in: ITU Kaleidoscope: Trust in the Information Society (K-2015), 2015, IEEE, 2015, pp. 1–8.
[13] Y.B. Saied, A. Olivereau, D. Zeghlache, M. Laurent, Trust management system design for the internet of things: a context-aware and multi-service approach, Computers & Security 39 (2013) 351–365.
[14] I.U. Din, M. Guizani, B.-S. Kim, S. Hassan, M.K. Khan, Trust management techniques for the internet of things: a survey, IEEE Access 7 (2018) 29763–29787.
[15] Y.B. Saied, A. Olivereau, D. Zeghlache, M. Laurent, Trust management system design for the internet of things: a context-aware and multi-service approach, Computers & Security 39 (2013) 351–365.
[16] B. Sonja, J. Boudec, A robust reputation system for peer-to-peer and mobile ad-hoc networks, in: Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems (P2PEcon 2004), Cambridge, MA, USA, 2004, pp. 4–5.
[17] S. Ganeriwal, L.K. Balzano, M.B. Srivastava, Reputation-based framework for high integrity sensor networks, ACM Transactions on Sensor Networks (TOSN) 4 (3) (2008) 1–37.
[18] 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, 2012, pp. 1–6.
[19] 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.
[20] R. Chen, F. Bao, J. Guo, Trust-based service management for social internet of things systems, IEEE Trans Dependable Secure Comput 13 (6) (2015) 684–696.
[21] W. Liu, L. Yin, B. Fang, X. Yu, An efficient trust evaluation approach in attacker dominated networks in internet of things, in: Proceedings of the Future Information Technology, Application, and Service, Springer, 2012, pp. 559–567.
[22] F. Bao, R. Chen, J. Guo, Scalable, adaptive and survivable trust management for community of interest based internet of things systems, in: Proceedings of the 2013 IEEE eleventh international symposium on autonomous decentralized systems (ISADS), IEEE, 2013, pp. 1–7.
[23] R. Chen, J. Guo, F. Bao, Trust management for soa-based iot and its application to service composition, IEEE Trans. Serv. Comput. 9 (3) (2014) 482–495.
[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, 31, ACM, 2001.
[25] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for internet applications, ACM SIGCOMM Computer Communication Review 31 (4) (2001) 149–160.
[26] A. Rowstron, Pastry: Scalable, distributed object location and routing for large-scale, persistent peer-to-peer storage utility, in: Proc. IFIP/ACM International Conference on Distributed Plarforms, 2001, 2001.
[27] P. Maymounkov, D. Mazieres, Kademlia: A peer-to-peer information system based on the xor metric, in: Proceedings of the International Workshop on Peer-to-Peer Systems, Springer, 2002, pp. 53–65.
[28] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, A survey and comparison of peer-to-peer overlay network schemes, IEEE Communications Surveys & Tutorials 7 (2) (2005) 72–93.

[29] 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, Computer Science and Information Systems 8 (4) (2011) 1207–1228.
[30] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decis. Support Syst. 43 (2) (2007) 618–644.
[31] M. Nitti, R. Girau, L. Atzori, Trustworthiness management in the social internet of things, IEEE Trans Knowl Data Eng 26 (5) (2014) 1253–1266.
[32] F.G. Mármol, G.M. Pérez, Security threats scenarios in trust and reputation models for distributed systems, computers & security 28 (7) (2009) 545–556.
[33] S. Singh, P.K. Sharma, S.Y. Moon, J.H. Park, Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions, J. Ambient. Intell. Humaniz. Comput. (2017) 1–18.
[34] F. Chowdhury, M.S. Ferdous, Performance analysis of r/kademlia, pastry and bamboo using recursive routing in mobile networks, International Journal of Computer Networks & Communications (IJCNC) 9 (5) (2017).
[35] A.S. Wander, N. Gura, H. Eberle, V. Gupta, S.C. Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: Proceedings of the Third IEEE international conference on pervasive computing and communications, IEEE, 2005, pp. 324–328.
[36] R. Abassi, Dealing with collusion attack in a trust-based manet, Cybern. Syst 49 (7–8) (2018) 475–496.
[37] M.Y. Becker, A. Russo, N. Sultana, Foundations of logic-based trust management, in: 2012 IEEE Symposium on Security and Privacy, 2012, pp. 161–175.
[38] N. Drawel, J. Bentahar, E. Shakshuki, Reasoning about trust and time in a system of agents, Procedia Comput. Sci. 109 (2017) 632–639.
[39] A. Aldini, Design and verification of trusted collective adaptive systems, ACM Transactions on Modeling and Computer Simulation (TOMACS) 28 (2) (2018) 1–27.
[40] R. Casadei, A. Aldini, M. Viroli, Towards attack-resistant aggregate computing using trust mechanisms, Sci. Comput. Program. 167 (2018) 114–137.
[41] F. Liu, E. Lorini, Reasoning about belief, evidence and trust in a multi-agent setting, in: Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems, Springer, 2017, pp. 71–89.

Kubra Kalkan is Assistant Professor in the Computer Science Department of Ozyegin University in Turkey. She received her BSc and MSc degrees in Computer Science and Engineering department from Sabanci University in 2009 and 2011, respectively. She received her PhD degree from Bogazici University in 2016. After Ph.D., she worked as a post doctoral researcher in University of Oxford. She also worked as a visiting researcher at various institutions such as École Polytechnique Fédérale de Lausanne (EPFL), Microsoft Redmond and Northeastern University during her M.Sc. and Ph.D. Her Ph.D. thesis is rewarded IEEE Turkey PhD Thesis Award and Bogazici University BAP Ph.D. Thesis Award. Her current research interests include network security, computer networks, wireless networks, IoT, P2P networks and software-defined networking (SDN).

Kasper Rasmussen is Associate Professor in the Computer Science Department. He joined University of Oxford in 2013 and in 2015 was awarded a University Research Fellowship from the Royal Society in London. He completed his masters degree in Computer Science from the Technical University of Denmark (DTU) in 2005. He did his Ph.D. with Prof. Srdjan Capkun at the Department of Computer Science at ETH Zurich. He worked as a post-doc at University of California, Irvine, with Prof. Gene Tsudik.