DCONST

Detection of Multiple-Mix-Attack Malicious Nodes Using Consensus-Based Trust in IoT Networks

Ma, Zuchao; Liu, Liang; Meng, Weizhi

Published in:
Proceedings of 25th Australasian Conference on Information Security and Privacy

Link to article, DOI:
10.1007/978-3-030-55304-3_13

Publication date:
2020

Document Version
Peer reviewed version

DCONST: Detection of Multiple-Mix-Attack Malicious Nodes Using Consensus-based Trust in IoT Networks

Zuchao Ma[1], Liang Liu[1] and Weizhi Meng[2⋆]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China
[2] Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark

Abstract. The Internet of Things (IoT) is growing rapidly, which allows many smart devices to connect and cooperate with each other. While for the sake of distributed architecture, an IoT environment is known to be vulnerable to insider attacks. In this work, we focus on this challenge and consider an advanced insider threat, called multiple-mix attack, which typically combines three sub-attacks: tamper attack, drop attack and replay attack. For protection, we develop a Distributed Consensus based Trust Model (DCONST), which can build the nodes' reputation by sharing particular information, called *cognition*. In particular, DCONST can detect malicious nodes by using the K-Means clustering, without disturbing the normal operations of a network. In the evaluation, as compared with some similar models, DCONST can overall provide a better detection rate by increasing around 10% to 40%.

Keywords: IoT network · Malicious Node · Trust Management · Consensus · K-means Method.

1 Introduction

The Internet of Things (IoT) is becoming an increasingly popular infrastructure to support many modern applications or services, like smart homes, smart healthcare, public security, industrial monitoring and environmental protection [14]. These IoT devices could be used to collect information from surroundings or control units to help gather information and make suitable strategies. Also, these devices can use various IoT protocols [20], with the aim of transferring their data including ZigBee, WiFi, Bluetooth, etc.

The IoT topology is flexible (e.g., multihop network), but it also suffers from many insider threats, where an attacker can launch an intrusion inside a network. For example, attackers can compromise some devices in an IoT network and then use these devices to infer sensitive information and tamper data. Therefore, it

⋆ Corresponding author

is very essential to design an effective security mechanism to identify malicious nodes in IoT networks.

Motivation. Most existing studies often focus on a single attack scenario in an IoT, but an advanced attacker may choose to perform several attacks simultaneously. Hence a stronger and more advanced attacker should be considered - who can control some internal nodes in IoT networks and perform a multiple-mix-attack. For example, Liu et al. [8] discussed a scenario of multiple-mix-attack by combining data tampering, packet dropping and duplication. They proposed a perceptron-based trust to help detect malicious nodes, but their method has to inject many packets to analyze the nodes' reputation, resulting in a disturbance of network operations. In addition, their detection accuracy depends heavily on network diversity and attack probability, i.e., the low network diversity and the high attack probability may cause low detection accuracy.

Contributions. Motivated by the literature, our work develops a Distributed Consensus based Trust Model (DCONST), which can achieve the self-detection via consensus among IoT nodes. It can work without disturbing the normal network operations and build the nodes' trust by sharing a kind of particular information called *cognition* among nodes. To mitigate the impact caused by the low network diversity and the high attack probability, DCONST makes a strategy that malicious nodes should receive more punishment while benign nodes should obtain more award. The contributions can be summarized as below.

- In this work, we formalize system models and propose DCONST by using consensus of nodes to improve the detection performance. In particular, our DCONST can generate punishment evidence to reduce the trust values of potential malicious nodes, and provide award evidence to improve the reputation of benign nodes. A base station can collect all cognitions of nodes to complete a final trust evaluation.
- We use the K-Means method to cluster nodes into benign group and malicious group. For the performance analysis, we compare our approach with two similar approaches: Perceptron Detection with Enhancement (PDE) [8] and Hard Detection (HD) [9]. The experimental results demonstrate that DCONST could achieve better detection performance.

Organization. The remaining parts are organized as follows. Section 2 introduces related work on trust-based detection in IoT networks. Section 3 formalizes the network model and message model. Section 4 describes DCONST in detail. Section 5 discusses our experimental environment and analyzes evaluation results. Finally, Section 6 concludes our work with future directions.

2 Related Work

The Internet of Things (IoT) is beneficial for its wide adoption and sustainable development, which can be divided into four layers including perception layer, network layer, middle-ware layer and application layer. Due to the distributed architecture, insider attacks are one big challenge in IoT networks.

Trust-based detection. Building a proper trust mechanism is a promising solution to discover insider attacks [13]. For instance, Cho et al. [2] proposed a provenance-based trust model, called PROVEST, for delay tolerant networks to handle the trust evaluation of nodes by using both direct and indirect trust information. Dinesh et al. [1] developed a detection scheme, named BAN-Trust for identifying malicious nodes in body area networks according to the nature acquired through the nodes by their own as well as partner nodes. BAN-Trust could conceive the common behavior among nodes and gather the information to measure the trust. Liu et al. [8] proposed a perceptron-based detection using machine learning in the multiple-mix-attack environment. The trust is evaluated according to the reputation of paths, but the detection accuracy depends heavily on the network diversity. Some other related studies can refer but not limited to [3,7,15,21].

Consensus-based trust. In this work, our goal is to detect malicious nodes based on the knowledge of all network nodes, which is a typical group decision making (GDM) problem. Consensus-based trust models are widely used in solving this problem with two typical steps: 1) *trust estimation* that evaluates the trust from a single group member, and *trust aggregation* that predicts the trust based on the aggregated knowledge from trust estimation. For example, Rathore et al. [17] presented a consensus-aware sociopsychological model for detecting fraudulent nodes in WSNs. They used three factors for trust computation, such as ability, benevolence, and integrity. Sharma et al. [18] proposed a consensus framework for mitigating zero-day attacks in IoT networks. Their approach uses the context behavior of IoT devices as a detection mechanism, working with an alert message protocol and a critical data sharing protocol. Mazdin et al. [12] analyzed the application of a binary trust-consensus protocol in multi-agent systems with switching communication topology. The trust in their work represents a belief of one agent that another one is capable of executing a specific task. Some other related studies can refer to [4,5,22].

Advanced attack. In this work, similar to former work [8], we consider a multiple-mix attack in which an insider can perform three typical attacks: tamper attack, drop attack and replay attack. More specifically, *tamper attack* is one of the most harmful internal threats, where malicious nodes along a multihop path can modify the received packets (randomly or with specific goals) before they reach the destination [6]. Under a *drop attack*, malicious nodes can drop received packets (randomly or with specific goals) to prevent these packets from reaching the destination [19]. The third attack is *replay attack*, where malicious nodes can send the received packets repeatedly to cause overhanded data flow, aiming to consume the link bandwidth and mislead network functions [11].

Most existing research studies focus mainly on a single / separate attack, but in fact an advanced attacker can handle multiple types of attacks at the same time. In such case, it is more difficult to identify malicious nodes precisely. Thus, in this work, we consider an advanced attacker who can launch a multiple-mix-attack by combining these three attacks with a probability.

3 System Model

In this work, we adopt the same attack model in [8], called multiple-mix-attack, which consists of tamper attack, drop attack and replay attack. This section formalizes our network model and message model.

3.1 Network Model

Node Model. In this work, we consider that malicious nodes can make a tamper attack, a drop attack and a replay attack at the same time or choose combining some of them intentionally. We assume that a node can be represented by using the following equation:

$$Node =< id, T, pu, pr, cogl, p_{TA}, p_{DA}, p_{RA} > \tag{1}$$

where, id represents the unique identifier of the node; T represents the trust of the node; pu and pr represent a public-private key pair of the node to be used for encryption and decryption when network nodes transfer data and cognitions; $cogl$ is the list of cognitions that includes all cognitions of the nodes about others. p_{TA} is the probability of node N making a tamper attack, p_{DA} is the probability of node N making a drop attack, and p_{RA} is the probability of node N making a reply attack. For a benign node, the probability of node's p_{TA}, p_{DA} and p_{RA} should be all zero, whereas a malicious node's p_{TA}, p_{DA} and p_{RA} should be a positive number.

Path Model. The path of a packet can be represented as:

$$Path =< node_1, node_2, node_3 ... node_n > \tag{2}$$

where, if packet A arrives at destination with a $Path =< node_1, node_2, node_3 ... node_n >$, then it means packet A is delivered through $node_1$, $node_2$, $node_3$, ... $node_n$ in a sequence.

3.2 Message Model

We assume that the IoT network contains a trusted authority (TA) that can distribute keys to provide IoT nodes with a shield to defeat both external attacks and insider attacks. The key management method can refer to [2]. Distributed keys can be divided into two types:
1. A symmetric key K for encryption to defend external attackers;
2. Asymmetric key pairs $< pu, pr >$ used for encryption and signing to defeat insider attackers, referred in Equ. (1).

Message can be formalized as

$$M = [(D)_{pu_R}, Si_D, Cog_{x_1,y_1}, Si_{x_1,y_1}, ..., Cog_{x_m,y_m}, Si_{x_m,y_m}]_K \tag{3}$$

where, D represents the original data that need to be transferred, pu_R represents using the public key of the receiver to encrypt the data, and Si_D is the

signature. Cog_{x_i,y_i} represents the cognition of node x_i about node y_i. Si_{x_i,y_i} represents the signature of Cog_{x_i,y_i} that is generated by node x_i. Cognition will be introduced in Section 4.3, which is the assessment about the node trust. K means the symmetric key and K is used to encrypt the whole message.

To defend against insider attacks, the sender of the communication uses the public key of the receiver to encrypt the data-part of transferred packets. The cognition-part of packets do not need to be encrypted, because relay nodes in the transferred path may need to access the cognition. Besides, it is important to protect data and cognitions from being tampered and there should be a notification if tampered behavior is detected. Hence the data owner and the cognition should provide their signatures.

4 DCONST

This section introduces our proposed DCONST, including core workflow, trust model, cognition, cognition aggregation, cognition sharing, punishment and award, trust evaluation, and the K-means based detection.

4.1 Core Workflow

Fig. 1 shows the core workflow of DCONST, mainly consisting of trust evaluation and detection of malicious nodes. For trust evaluation, nodes have to measure the reputation of others within the network by sharing particular information, called *cognition* (refer to Section 4.3). To identify malicious nodes, the base station in the IoT network has to collect all cognitions from nodes and create the Cognition-Matrix, where Cog_{xy} represents the cognition of node x about node y. The base station should perform a central trust evaluation and obtain the trustworthiness of each node. For detection of malicious nodes, nodes' trust should be forwarded to the K-means clustering module. The output includes *Tamper-malicious set* - malicious nodes that launch tamper attacks; *Drop-malicious set* - malicious nodes that launch drop attacks; and *Replay-malicious set* - malicious nodes that launch replay attacks; and Benign set - benign nodes.

4.2 Trust Model

To evaluate the reputation of nodes by considering three attack types, we design the trust model with three dimensions accordingly - honesty, volume and straight. **Honesty** is the dimension that demonstrates whether a node tampers the received data or not. An evidence about tamper behavior can reduce the honesty of the node. **Volume** illustrates whether a node drops the received data or not. An evidence about drop behavior can reduce the volume of the node. **Straight** validates whether a node replays the received data or not. An evidence about replay behavior can reduce the straight of the node. Hence the trust of a node can be formalized as follow:

$$Node.T =< H, V, S > \tag{4}$$

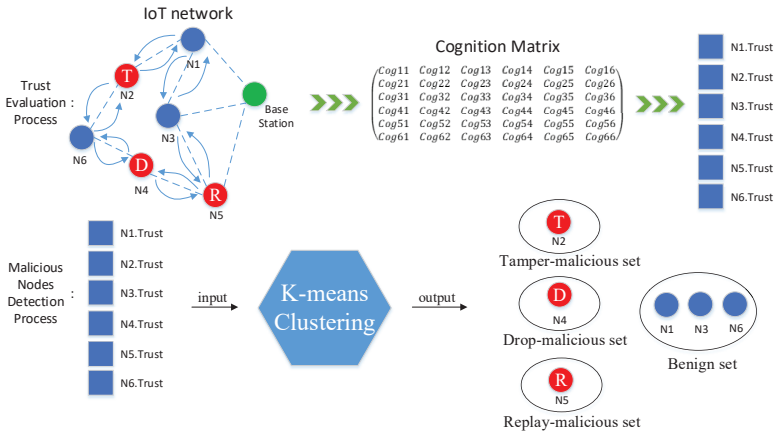

Fig. 1. Core Workflow

where, H represents the honesty; V represents the volume; S represents the straight. All of them range from 0 to 100; the negative side is 0 and the positive side is 100. The higher value reached by the H, V, S of a node, the better trust it possesses. Note that H, V, S are all integers rather than decimals, which can be space-saving in the data communication.

4.3 Cognition

Cognition is the assessment about the reputation of nodes in the view of the cognition owner (a type of node). In DCONST, nodes obtain the trust of others by exchanging their cognitions. We formalize the cognition as

$$Cog =< Sub, Obj, H, V, S, isNei > \tag{5}$$

where, Sub represents the subject of the cognition and Obj represents the object of the cognition. Cog means the cognition of Sub about Obj. The parameters of H, V, S are the same in Equ. (4). $isNei$ represents whether Sub is the neighbor of Obj. For example, if node X and node Y are neighbors, and we set the honesty, the volume and the straight of node Y as 100, 100 and 50 respectively, then there is a cognition $< X, Y, 100, 100, 50, 1 >$ of node X.

4.4 Cognition Aggregation

When a node receives a new cognition from other nodes, it updates its own cognition via cognition aggregation. Essentially, cognition aggregation is the process about mixing own cognition (old cognition) with the new cognition in a specific ratio. This ratio depends on the credibility of the new cognition, which can be evaluated from two aspects: Reliability and Fluctuation.

Reliability. If the new-cognition provider is reliable in the old-cognition of the receiver, it makes sense to believe the new-cognition more. This is because

the precision of cognition of a node could be affected by its malicious behavior. For example, assume there is a packet passing a path that contains a malicious node N. If this packet is attacked by node N, then there will be a punishment evidence created, which we will discuss in Section 4.6. As this punishment evidence is negative, the cognition of node N would lack persuasion. To evaluate the reliability of a node n, we can have the following:

$$r(n) = \frac{c.H + c.V + c.S}{H_{max} + V_{max} + S_{max}} \tag{6}$$

where, c is the cognition about the node n; parameters of H, V and S are the same in Equ. (5); H_{max}, V_{max} and S_{max} are the maximum of H, V and S. In our system, H_{max}, V_{max} and S_{max} are set as 100.

Fluctuation. If the new-cognition differs from the old-cognition greatly, the new-cognition provider may be suspicious as the persuasion of the new-cognition is poor. The apparent difference may be caused by a wrong cognition or some negative evidence, and therefore we can reduce its weight in the aggregation.

To evaluate the fluctuation of a new-cognition compared to the old-cognition, we can have the following:

$$f(ncog) = \frac{|ncog.H + ncog.V + ncog.S - ocog.H - ocog.V - ocog.S|}{H_{max} + V_{max} + S_{max}} \tag{7}$$

where, $ncog$ is the new-cognition; $ocog$ is the old-cognition of the receiver; H_{max}, V_{max} and S_{max} are the maximum of H, V and S.

Thus, the credibility of the new-cognition can be evaluated as

$$cred(ncog) = \begin{cases} \rho_d * (1 - f(ncog)) * r(ncog.Sub) & \text{if } ncog.Obj \text{ is a} \\ & \text{neighbor of } ncog.Sub \\ \rho_i * (1 - f(ncog)) * r(ncog.Sub) & otherwise \end{cases} \tag{8}$$

where, $ncog$ is the new-cognition; $ncog.Sub$ is the provider of the cognition that can be referred in Equ. (5); whether $ncog.Obj$ is a neighbor of $ncog.Sub$ can be concluded according to $ncog.isNei$ in Equ. (5). ρ_d and ρ_i are the maximum of the credibility with $\rho_d > \rho_i$. The setting of ρ_d and ρ_i is mainly based on the common sense that a cognition from a node's neighbor about the node can be more objective and accurate.

Based on the credibility of the new-cognition, we update the old-cognition of the receiver by aggregating the new-cognition as follows:

$$\begin{aligned} &[ucog.H, ucog.V, ucog.S] = \\ &[\log_2(1 + \frac{ncog.H * cred(ncog) + ocog.H * (1 - cred(ncog))}{H_{max}}) * H_{max}, \\ &\log_2(1 + \frac{ncog.V * cred(ncog) + ocog.V * (1 - cred(ncog))}{V_{max}}) * V_{max}, \\ &\log_2(1 + \frac{ncog.S * cred(ncog) + ocog.S * (1 - cred(ncog))}{S_{max}}) * S_{max}] \end{aligned} \tag{9}$$

where, *ucog* is the cognition of the receiver updated after the cognition aggregation. The motivation of using the log function is due to that it needs more positive cognitions if any H, V or S improves. On the contrary, H, V or S of a node could decrease fast when there are negative cognitions.

4.5 Cognition Sharing

The cognition sharing of DCONST can be defined with two operations - Cognition Extraction and Cognition Spread. *Cognition Extraction* is the process to determine which cognition should be sent with the packet when the packet passes a node. Considering the bandwidth of a network, it is not an efficient way to send all cognitions, so that a wise strategy should balance both performance and cost. *Cognition Spread* aims to check whether a cognition can be aggregated by other nodes, as the aggregation process will bring some cost when the node calculates the credibility of a new cognition.

- **Cognition Extraction.** When a packet is transferred by a node, the node can create a new packet by adding three cognitions to the end of the packet. As the new packet is sent by the node, the cognition of the node is spread. For implementation, there are three prior queues, negative queue (NQ), reduction queue (RQ) and improvement queue (IQ) in the memory of each node. NQ sorts cognitions by their sum of honesty, volume and straight (the head of queue is the cognition having the smallest sum of honesty, volume and straight); RQ sorts cognitions by their reduction (the head of queue is the cognition that reduces the most and is updated recently); and IQ sorts cognitions by their improvement (the head of queue is the cognition that improves the most and is updated recently). A cognition exists in three queues, and if it is removed from a queue then it will be also removed by other two queues. DCONST selects the heads of these queues to spread including the most negative cognition, the cognition with the latest and the largest reduction, and the cognition with the latest and the largest improvement that has not been sent. That is, DCONST attempts to spread negative cognitions and cognitions whose fluctuation is evident promptly.
- **Cognition Spread.** Cognition transferred with packets can be aggregated by both relay nodes and destination node. The relay nodes can update their cognitions when they receive packets. They also need to add their extracted cognitions to the transferred packet and forward them to the next node. During idle time, nodes can send their extracted cognitions to their neighbors aiming to accelerate the process of updating cognitions.

The whole process of Cognition Sharing is shown in Fig. 2. Green blocks with Cog_{i1}, Cog_{i2} and Cog_{i3} represent the cognition selected from node n_i. Green full-line-arrow means the cognition can be aggregated by the pointed node. Green dotted-line-arrow shows that nodes can send the extracted cognitions to their neighbors during idle time.

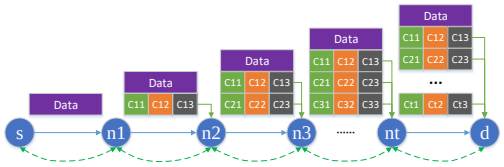

Fig. 2. DCONST Cognition Sharing

4.6 Punishment and Award

To identify malicious nodes in IoT networks, it is important to create the trust gap between benign nodes and malicious nodes, through punishing malicious nodes and awarding benign nodes. Essentially, award is the reverse process of punishment, which aims to recover the correct cognition about those nodes that are misunderstood by some wrong punishment evidence. The trigger of punishment is different from the trigger of award - one malicious transmission can trigger one punishment while accumulating enough successful transmission may only bring one award. This is because few successful transmission cannot prove that there are no malicious nodes existing in the path. To analyze different attack types, we design corresponding punishment and award as below.

Punishment of Tamper Attack. The evidence of punishment about tamper attack is called Indirect Tamper Punishment Evidence(ITPE). Note that relay nodes only check whether the cognition in the packet is tampered because they do not have the key to decrypt the data part in the packet. Only the last node (with the key) can check both the cognitions and the data. ITPE could be generated in the following two cases:

Case A (Destination Punishment). Assume there is a packet transferred via a path $p =< s, n_1, n_2, n_3, ..., n_t, d >$ and node d checks whether the original data or cognition is tampered with their signature. If a tamper attack is detected, an ITPE will be created by node d. Then node d can use ITPE to update its cognitions about n_1, n_2, n_3, ..., n_t. If the data part of a packet is tampered, n_1, n_2, n_3, ..., n_t will be pointed by ITPE. While if Cog_{n_x,n_y} (refer to Equ. (3)) is tampered, $n_i(i > x)$ will be pointed by ITPE. This is because only latter nodes can tamper the cognitions in the packet from previous nodes.

Case B (Middle Punishment). Assume there is a packet transferred via a path $p =< s, n_1, n_2, n_3, ..., n_t, d >$ and node n_k checks whether the cognition is tampered with their signature. If a tamper attack is detected, an ITPE can be created by node n_k. While if Cog_{n_x,n_y} (refer to Equ. (3)) is tampered, $n_i(i > x)$ will be pointed by ITPE. In this case, node n_k can use ITPE to update the cognition about n_{x+1}, n_{x+2}, ..., n_{k-1}. To evaluate the punishment of tamper attack, when an ITPE is created, the cognition about all nodes pointed by ITPE will be updated by the producer of ITPE (the punishment provider) as below:

$$ucog.H = \log_2(1 + \frac{ocog.H - \theta_{hi}}{H_{max}}) * H_{max} \tag{10}$$

where, *ucog* is the updated cognition of the punishment provider (i.e., relay nodes and the destination node), *ocog* is the old cognition, and θ_{hi} is a parameter that determines the reduction of the honesty with ITPE.

Award about No Tamper. Award about No Tamper is the reverse process of Punishment of Tamper Attack. Cnt_{hi} is a counter in each node to record the positive behavior of relay nodes. When the value of a node in this counter meets the threshold σ_{hi}, an indirect honesty award evidence (IHAE) can be created to increase the node's honesty. We use $Cnt_{hi}.n.val$ to represent the value of node n in Cnt_{hi}. If a packet is successfully transmitted, all nodes related to the packet will be awarded. Taking the above Case A as an example, if the tamper attack does not exist, the value of each relay node in Cnt_{hi} can be increased by 1, i.e., $Cnt_{hi}.n_1.val = Cnt_{hi}.n_1.val + 1$, $Cnt_{hi}.n_2.val = Cnt_{hi}.n_2.val + 1$, ..., $Cnt_{hi}.n_t.val = Cnt_{hi}.n_t.val + 1$. Finally, if we have $Cnt_{hi}.n.val = \sigma_{hi}$, then an IHAE about node n will be created. If an IHAE or an ITPE about node n is created, $Cnt_{hi}.n.val$ will be set as zero.

To evaluate the award of no tamper attack, when an IHAE is created, the cognition about all nodes pointed by IHAE will be updated by the producer of IHAE (the award provider) as below:

$$ucog.H = \log_2(1 + \frac{ocog.H + \theta_{hi}}{H_{max}}) * H_{max} \tag{11}$$

where, *ucog* is the updated cognition of the award provider and *ocog* is the old cognition. θ_{hi} is a parameter that determines the improvement of the honesty with IHAE and its value is equal to that in Equ. (10).

Punishment of Drop Attack. The evidence of punishment about drop attack is called Indirect Drop Punishment Evidence (IDPE). Suppose there is a packet transferred via a path $p =< n_0, n_1, n_2, n_3, ..., n_t, d >$ and if node d receives the packet, it has to send an acknowledgement (ack) to node n_0 with the path $rp =< d, n_t, ..., n_3, n_2, n_1, n_0 >$. If we assume the ack cannot be faked and the node does not receive the ack after transferred the packet, then a drop attack can be detected and an IDPE can be created. Node n_i can use IDPE to update the cognition of n_{i+1}. For example, if node n_3 is a malicious node that drops the packet and node n_0, n_1, n_2 cannot get the ack from node d, then node n_0 can create an IDPE about node n_1; node n_1 can create an IDPE about node n_2; and node n_2 can create an IDPE about node n_3. Thus, IDPE can be regarded as a chain to connect all potential drop-malicious nodes.

To evaluate the punishment of drop attack, when an IDPE is created, the cognition about the node pointed by IDPE will be updated by the producer of IDPE (the punishment provider) as below:

$$ucog.V = \log_2(1 + \frac{ocog.V - \theta_{vi}}{V_{max}}) * V_{max} \tag{12}$$

where, *ucog* is the updated cognition of the punishment provider, *ocog* is the old cognition, and θ_{vi} is a parameter that determines the reduction of the volume with IDPE.

Award about No Drop. Award about No Drop is the reverse process of Punishment of Drop Attack. Cnt_{vi} is the counter to record the positive behavior with the acknowledgement from the destination node. When the value of a node in this counter meets the threshold σ_{vi}, an indirect volume award evidence (IVAE) can be created to increase the volume of the node. Similar to the *award about no tamper*, if a packet successfully arrives at its destination, the value in Cnt_{vi} of all nodes related to the packet can be increased by 1.

When an IVAE is created, the cognition about the node pointed by the IVAE will be updated by the producer of IVAE (the award provider) as below:

$$ucog.V = \log_2(1 + \frac{ocog.V + \theta_{vi}}{V_{max}}) * V_{max} \tag{13}$$

where, *ucog* is the updated cognition of the award provider and *ocog* is the old cognition. θ_{vi} is a parameter that determines the improvement of the volume with IVAE and its value is equal to that in Equ. (12).

Punishment of Replay Attack. The evidence of punishment about replay attack is called Indirect Replay Punishment Evidence(IRPE). Assume there is a packet transferred via a path $p =< n_0, n_1, n_2, n_3, ..., n_t, d >$ and if node d receives the packet, it has to send an acknowledgement (ack) to node s with the path $rp =< d, n_t, ..., n_3, n_2, n_1, n_0 >$. If any nodes in the path receive redundant acks, a replay attack can be detected. For example, when node n_0 sends the packet to node d, it should receive one ack about this packet from node d instead of two acks or more. Once a replayed attack happens, node d will receive two or more identical packets and return identical acks. In this case, node $s, n_1, n_2, ...n_t$ can realize there is a replay attack, and create an IRPE. Then node n_i can use IRPE to update the cognition of node n_{i+1} except for node d. For example, if node n_3 is a malicious node that replays the packet once, then node d can get two identical packets and return two identical acks to node s. Thus node s can create an IRPE about node n_1; node n_1 can create an IRPE about node n_2; and node n_2 can create an IRPE about node n_3. Basically, IRPE can be regarded as a chain to connect all potential replay-malicious nodes.

To evaluate the punishment of replay attack, when an IRPE is created, the cognition about the node pointed by the IRPE will be updated by the producer of IRPE (the punishment provider) as below:

$$ucog.S = \log_2(1 + \frac{ocog.S - \theta_{si}}{V_{max}}) * V_{max} \tag{14}$$

where, *ucog* is the updated cognition of the punishment provider and *ocog* is the old cognition. θ_{si} is a parameter that determines the reduction of the straight with IRPE.

Award about No Replay. Award about No Replay is the reverse process of Punishment of Replay Attack. Cnt_{si} is a counter to record the positive behavior with the acknowledgement from the destination node. When the value of a node in this counter meets the threshold σ_{si}, an indirect straight award evidence (ISAE) can be created to increase the straight of the node. If a packet successfully arrives at its destination and no redundant acks are found, the value in Cnt_{si} of all nodes related to the packet can be increased by 1.

When an ISAE is created, the cognition about the node pointed by the ISAE will be updated by the producer of ISAE (the award provider) as below:

$$ucog.S = \log_2(1 + \frac{ocog.S + \tau_{si}}{S_{max}}) * S_{max} \tag{15}$$

where, *ucog* is the updated cognition of the award provider and *ocog* is the old cognition. τ_{si} is a parameter that determines the improvement of the straight with ISAE and its value is smaller than θ_{si} in Equ. (14). Here is the explanation about $\tau_{si} < \theta_{si}$: when a node receives the supposed ack, it increases the Cnt_{si} without considering whether it will receive a duplicated ack in the future. So if the node receives the duplicated ack later (a replay attack is detected), it will create an IRPE to reduce the straight of its next node, and $\tau_{si} < \theta_{si}$ can guarantee the reduction of the straight could be larger than the increase. Otherwise, the reduction and the increase of the straight will counteract each other, i.e., making the punishment about the replay attack invalid.

4.7 Trust Evaluation

Trust evaluation has to be executed in the base station of an IoT network, when there is a need to analyze the security situation. The base station can collect all cognitions of every node and perform a centralized process. The final trust of a node should be evaluated based on the consensus (cognition) of all members.

In the base station, after all cognitions are collected, a Cognition Matrix can be created and defined as below:

$$CogMat = \begin{pmatrix} Cog_{1,1} & Cog_{1,2} & Cog_{1,3} & ... & Cog_{1,n} \\ ... & ... & ... & ... & ... \\ Cog_{n-1,1} & Cog_{n-1,2} & Cog_{n-1,3} & ... & Cog_{n-1,n} \\ Cog_{n,1} & Cog_{n,2} & Cog_{n,3} & ... & Cog_{n,n} \end{pmatrix} \tag{16}$$

where, $Cog_{x,y}$ represents the cognition of node x about node y. Note that we set $Cog_{x,x}$ as $< x, x, 0, 0, 0 >$, indicating that the recognition of a node about itself will be ignored.

Then we denote Re_{ix} as the reputation of node x according to node i. Re_{ix} can be regarded as the weight on the cognition of node i in all nodes' cognitions (except node x), when evaluating the trust of node x. The reputation of a node is based on the similarity with the cognition from their neighbors. It is believed that the cognition from a node's neighbors can help evaluate the reputation more persuasively. This is because most evidence of punishment and award comes from

neighbors directly. The higher the similarity is, the better the reputation can be. We define Re_{ix} as below:

$$Re_{ix} = \frac{w_{ix}}{\sum_{k=1}^{n} w_{kx}} \tag{17}$$

where, w_{kx} means the weight of the cognition when node k evaluates node x. w_{kx} can be defined as below:

$$w_{kx} = \begin{cases} 0 & \text{if } k = x \\ \frac{1}{n-1} + (1 - \frac{Distance(Cog_{k,x}, Aver(x))}{H_{max}+V_{max}+S_{max}}) * \frac{1}{n-1} & \text{otherwise} \end{cases} \tag{18}$$

$$\begin{aligned} Distance(Cog_{k,x}, Aver(x)) = &|Cog_{k,x}.H - Aver(x).H| + |Cog_{k,x}.V \\ &- Aver(x).V| + |Cog_{k,x}.S - Aver(x).S| \end{aligned} \tag{19}$$

where, $Distance(Cog_{k,x}, Aver(x))$ is the function to evaluate the similarity between $Cog_{k,x}$ and the cognition from neighbors of node x. $Aver(x)$ is the average of the cognition from neighbors of node x. $Cog_{j,x}.H$ represents the honesty of $Cog_{j,x}$, $Cog_{j,x}.V$ represents the volume of $Cog_{j,x}$, and $Cog_{j,x}.S$ represents the straight of $Cog_{j,x}$. In particular, $Aver(x)$ can be defined as:

$$\begin{aligned} &[Aver(x).H, Aver(x).V, Aver(x).S] = \\ &\frac{[\sum Cog_{nei(x),x}.H, \sum Cog_{nei(x),x}.V, \sum Cog_{nei(x),x}.S]}{\text{the count of the neighbors of node } x} \end{aligned} \tag{20}$$

where, $nei(x)$ represents the neighbor of node x. Finally, we can evaluate the trust of node i.

$$\begin{aligned} &[Node_i.T.H, Node_i.T.V, Node_i.T.S] = \\ &[\sum_{j=1}^{n} [Cog_{j,i}.H * Re_{ji}], \sum_{j=1}^{n} [Cog_{j,i}.V * Re_{ji}], \sum_{j=1}^{n} [Cog_{j,i}.S * Re_{ji}]] \end{aligned} \tag{21}$$

4.8 Detection based on K-means Method

When obtaining the trust of all nodes, an intuitive way is to identify malicious nodes by selecting a trust threshold: if the trust value of a node is higher than the threshold, then this node is benign; otherwise, the node is malicious. Here comes a question that how to choose a proper threshold in our scenario - when there is a mix-attack, it is very hard to analyze the threshold, since all types of attacks may influence the threshold selection. In this work, we advocate using the clustering method to classify groups and then identify malicious nodes. The adoption of K-means method in this work aims to facilitate the comparison with similar studies like [9,10].

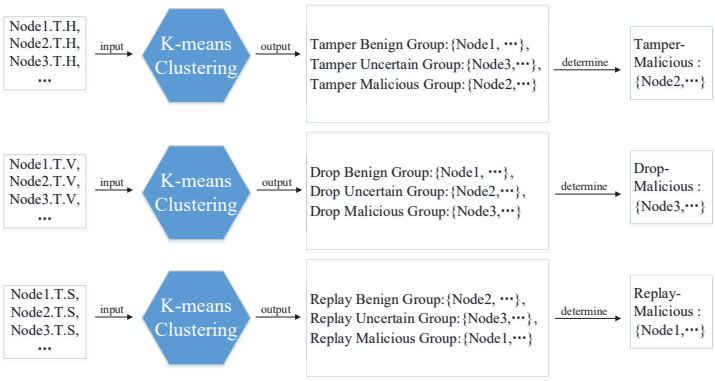

Fig. 3. K-means based Malicious Node detection

K-means method has been widely adopted in practice, which is a typical clustering method with unsupervised learning and the main argument is the count of clusters [16]. To identify malicious nodes, we use K-means method to cluster nodes in terms of honesty, volume and straight individually. **That is, K-means method will be executed three times to detect malicious nodes with three attack types and the input of K-means will be changed each time.** The three different inputs of K-means method are the tuple of honesty, the tuple of volume and the tuple of straight. In each cluster, we set the count of clusters to be three, then we can classify all nodes into three clusters - benign group, uncertain group and malicious group. **Only the nodes in the malicious group can be determined as malicious**, and this strategy aims to reduce the false positive rate. If the center of benign group and the center of malicious group are very close (i.e., a distance less than 10 in our evaluation), all nodes can be determined as benign. For example, when we input the tuple of honesty to detect tamper-malicious nodes, all nodes can be classified to benign group, uncertain group and malicious group. Those nodes in malicious group can be identified as tamper-malicious nodes. **It is vital to highlight that, a node can be classified into different malicious groups at the same time (e.g., tamper-malicious group and drop-malicious group) if it launches multiple attacks.**

The K-means based detection is described in Fig. 3. That is, malicious nodes include all nodes in the tamper-malicious group, the drop-malicious group, and the replay-malicious group. In this work, our main focus is to identify malicious nodes (without distinguishing the attack types), while the detection performance of different attack types will be addressed in our future work.

5 Evaluation

In the evaluation, we compare our DCONST with two similar approaches called Hard Detection (HD) [9] and Perceptron Detection with enhancement (PDE) [8].

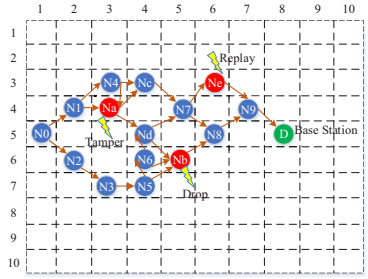

Fig. 4. A distribution of IoT nodes

Table 1. Environmental settings

Item	Description
CPU	Intel Core i7-4700MQ, 2.4GHz, 4Core(8 Threads)
Memory	Kingston DDR3L 8GB*2
OS	Ubuntu 18.04 LTS
Python	3.6.8
Scikit-learn	0.20

In more detail, **HD** is a mathematical method to detect malicious nodes that can perform a tamper attack. As the focus of HD is not fully the same in this work, we tune HD to make it workable in a multiple-mix attack environment. In particular, we added a module to help detect duplicated packets corresponding to replay attack, and enabled HD to search replay-attack malicious nodes. **PDE** is a detection scheme that uses both perceptron and K-means method to compute IoT nodes' trust values and detect malicious nodes accordingly. It also adopts an enhanced perceptron learning process to reduce the false alarm rate.

We use the accuracy as the main metric to evaluate the performance. When a malicious node is identified as malicious (even the attack type is labeled wrongly), it will be a True Positive (TP). When a benign node is identified as benign, it will be a True Negative (TN). Thus, if the total number of predictions is S, we can define $accuracy = (TP + TN)/S$.

5.1 Experimental Setup

In our environment, all IoT nodes are deployed in a 100 x $100m^2$ rectangle area discretely, and each node's communication range is 10m-15m. Our IoT network is generated randomly but it has a feature - for each node, there is at least one path from the node to the base station, enabling IoT devices to be connected. Fig. 4 shows an example of the distribution, where the green node is the base station, blue nodes are normal and red nodes are malicious - N_a is tamper-malicious, N_b is drop-malicious, N_e is replay-malicious.

To avoid result bias, we ran our simulation for each experiment in 10 rounds with 10 different networks. We then selected the average value to represent the final experimental result. In particular, we used Python to realize all algorithms, and used the scikit-learn, which is a famous machine learning tool library, to help cluster nodes according to their trust values via the K-means method. Our detection was deployed at the base station. Table 1 shows the detailed experimental settings. Besides, we set ρ_d in Equ. (8) as 0.4; ρ_i in Equ. (8) as 0.1; σ_{hi} in Section 4.6 as 3; σ_{vi} in Section 4.6 as 3; σ_{si} in Section 4.6 as 3; θ_{hi} in Equ. (10) as 40; θvi in Equ. (12) as 40 and θ_{si} in Equ. (14) as 40; and τ_{si} in Equ. (15) as 10, accordingly.

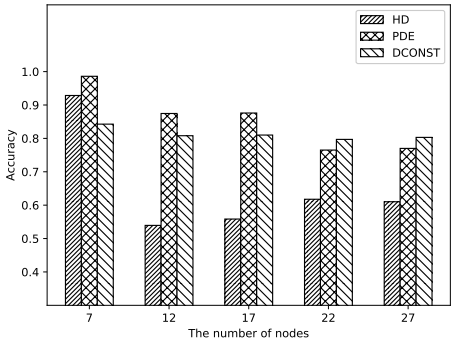

Fig. 5. The impact of the number of nodes on detection accuracy

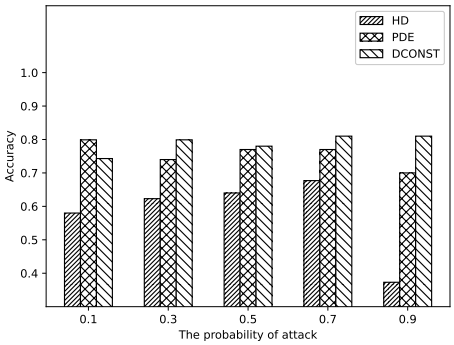

Fig. 6. The impact of the probability of attack on detection accuracy

5.2 Impact of the Number of Nodes

This variable means the scale of the topology, which can affect the detection performance of malicious nodes. To explore the performance, we consider a typical IoT network and a multiple-mix-attack, with the number of nodes as 7, 12, 17, 22 and 27, respectively. In this experiment, we set the number of passing packets as 20000; the probability of attack is 0.5; the percentage of malicious nodes as 0.3; and the diversity of network is all-type (use all paths).

Fig. 5 shows that when the IoT network is small-scale, all schemes can reach a high accuracy rate in which PDE performed the best. With the increase of nodes, the accuracy of HD and PDE has an obvious decrease. By contrast, DCONST can maintain stable and outperform HD and PDE when the network scale becomes large. This is because when the network scale is too small, numerous passing packets may cause too many redundant cognitions and make the cognitions of all nodes similar to each other, resulting in a worse case for DCONST.

A small network scale indicates few paths available in the network, in which malicious nodes can be easily identified by HD and PDE. On the other hand, Fig. 5 shows that when the number of nodes reaches 17 or more, the network topology may become more complicated and it is more difficult for HD and PDE to identify all malicious nodes. In such scenario, our DCONST can outperform the other two schemes. As PDE can reduce the false alarm rate by applying perceptron, it can perform much better than HD.

5.3 Impact of Attack Probability

In practice, insider attackers (malicious nodes) can choose a strategy to launch attacks with a certain probability, which would influence the detection performance. To explore this variable, we set the probability of multiple-mix-attack to be 0.1,0.3,0.5,0.7 and 0.9, respectively. In this experiment, we set the number of nodes as 27; the number of passing packets as 2000; the percentage of malicious nodes as 0.3; and the diversity of network is all-type (use all path).

Fig. 6 shows that our DCONST could outperform HD in all cases, and the accuracy of HD had a significant decrease when the probability of attack reaches 0.9. When the attack probability is very low (like 0.1), PDE could outperform DCONST, while with the increase of attack probability, DCONST could work better than PDE. The main reasons are analyzed as below.

- It is more beneficial for DCONST to detect malicious nodes with the high attack probability than with the low attack probability. This is because high attack probability can trigger numerous punishment evidence and few award evidence to malicious nodes. However, when the attack probability is low, the most accurate cognitions are often owned by neighbors of the malicious node. Cognitions owned by other nodes of the network are not accurate and the cognition aggregation from those non-neighbor nodes may cause a negative impact on detection accuracy. With the increase of attack probability, there is a better chance to obtain accurate cognitions.
- On the other hand, it is difficult for HD and PDE to handle high attack probability like 0.9. This is because the detection accuracy of HD and PDE depends on the reputation of paths. If there is a node with a high attack probability along a path, then the path reputation might become very low, making it hard to analyze the trust of all nodes within this path.

5.4 Impact of the Percentage of Malicious Nodes

The percentage here means the number of malicious nodes in the IoT network, which may have an impact on the detection accuracy. In the experiment, we set the percentage of malicious nodes under multiple-mix-attack to be 0.1, 0.2, 0.3, 0.4 and 0.5, respectively. In this experiment, we set the number of nodes as 27; the number of injected packets as 2000; the probability of attack as 0.5; and the diversity of network is all-type (use all paths). Fig. 7 shows the detection performance, and below are the main observations.

- It is found that DCONST could outperform HD and PDE in all cases. When the percentage of malicious nodes is small, there is a very obvious gap between DCONST and HD, i.e., DCONST could perform the best while HD only achieved the lowest accuracy. This is because when the percentage is small, only limited nodes can be pointed by punishment evidence while most nodes should be pointed by award evidence, making it more accurate for DCONST to identify malicious nodes from the whole network. By contrast, for HD and PDE, a small percentage may result in a high false positive since there are fewer malicious nodes in a path. Again, PDE can achieve better performance than HD by reducing the false rates via perceptron.
- When the percentage of malicious nodes increases, the performance of both PDE and DCONST could decrease gradually. This is because with more malicious nodes, the fewer award evidence can be obtained pointing to benign nodes, which may cause more errors. While DCONST could still outperform the other two schemes under such scenario.

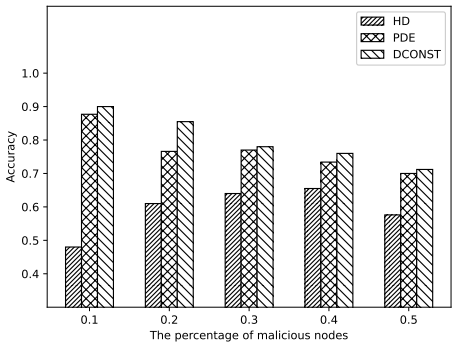

Fig. 7. The impact of the percentage of malicious nodes on detection accuracy

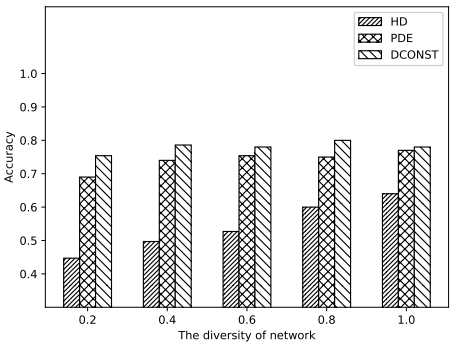

Fig. 8. The impact of the diversity of network on detection accuracy

5.5 Impact of Network Diversity

The diversity presents the type of paths through which packets can be delivered. To explore the detection performance under the multiple-mix-attack, we set the rate of valid paths to be 0.2, 0.4, 0.6, 0.8 and 1, respectively. In this experiment, we consider the number of nodes as 27; the number of injected packets as 2000; the probability of attack as 0.5, and the percentage of malicious nodes as 0.3.

Fig. 8 shows that when the network diversity is very low like 0.2, the detection accuracy of HD and PDE could not work well. This is because the detection of malicious nodes depends on analyzing the same nodes in different paths. As an example, assume that a path includes node A and node B. If there is an attack detected, either node A or node B could be malicious. We can only know node B is benign based on the other paths that include node B as well. If node B is determined, then it is easy to determine node A. This is why the detection accuracy could be increased with more valid paths.

As a comparison, the network diversity does not have a significant impact on the accuracy of DCONST. This is because DCONST relies on the strategy that enabling malicious nodes to receive more punishment and providing benign nodes with more award. In such case, even when the network diversity is low, DCONST can work well as long as there is an extra communication channel among benign nodes. For example, suppose there is a path $p =< n_1, n_2, n_3, n_4 >$ where n_2 is a malicious node. If n_3 and n_4 have another data communication channel, then they can award each other and improve their trust during the detection. This self-healing feature is a special merit of DCONST.

Discussion. Based on the above results, our proposed DCONST could outperform the similar schemes of HD and PDE in most cases. In particular, the low network diversity and the high attack probability would cause a big impact on HD and PDE, while DCONST could still maintain the detection accuracy. Hence we consider that the performance of DCONST is overall better than HD and PDE, and in practice, PDE and DCONST can complement each other.

6 Conclusion and Future Work

Due to the distributed nature of IoT networks, there is a significant need to design proper security mechanisms to defeat insider attacks. Most existing studies mainly consider a single attack, but we notice that an advanced intruder may perform several attacks simultaneously to make a more harmful impact. In this work, we target on this issue and focus on a multiple-mix attack including three typical sub-attacks: tamper attack, drop attack and replay attack. We develop DCONST that uses both the consensus of nodes and the K-means method to help measure nodes' trust and detect malicious nodes. Our experimental results demonstrate that DCONST can provide a better detection rate by around 10% to 40% as compared with two similar methods of Hard Detection (HD) and Perceptron Detection with Enhancement (PDE).

As our work is an early study in applying consensus, there are some open challenges that can be considered in our future work. First, DCONST is a method based on consensus of distributed nodes and the detection could be affected by each node. Thus, it is hard to provide a sufficient strategy to control the whole detection process. Then, the parameters of DCONST could be further optimized to deal with different network settings (i.e., improving accuracy and stability). Also, our future work can further investigate the impact of attack types and the number of passing packets on the detection performance.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under Grant No.61402225 and the Science and Technology Funds from National State Grid Ltd. (The Research on Key Technologies of Distributed Parallel Database Storage and Processing based on Big Data).

References

1. Anguraj, D.K., Smys, S.: Trust-based intrusion detection and clustering approach for wireless body area networks. Wireless Personal Communications **104**(1), 1–20 (2019)
2. Cho, J.H., Chen, R.: Provest: provenance-based trust model for delay tolerant networks. IEEE Transactions on Dependable and Secure Computing **15**(1), 151–165 (2016)
3. Cho, J., Swami, A., Chen, I.: A survey on trust management for mobile ad hoc networks. IEEE Communications Surveys and Tutorials **13**(4), 562–583 (2011)
4. Hongning, L., Xianjun, L., Leilei, X.: Analysis of distributed consensus-based spectrum sensing algorithm in cognitive radio networks. In: 2014 Tenth International Conference on Computational Intelligence and Security. pp. 593–597. IEEE (2014)
5. Kaveri, A., Geetha, K., Kaveri, A., Geetha, K.: Enhanced secure data transmission in manet networks using consensus based and trust aware protocol. International Journal **4**, 14–25 (2018)
6. Komninos, N., Philippou, E., Pitsillides, A.: Survey in smart grid and smart home security: Issues, challenges and countermeasures. IEEE Communications Surveys & Tutorials **16**(4), 1933–1954 (2014)

7. Li, W., Meng, W., Kwok, L., Ip, H.H.: Enhancing collaborative intrusion detection networks against insider attacks using supervised intrusion sensitivity-based trust management model. J. Network and Computer Applications **77**, 135–145 (2017)
8. Liu, L., Ma, Z., Meng, W.: Detection of multiple-mix-attack malicious nodes using perceptron-based trust in iot networks. Future Generation Computer Systems **101**, 865–879 (2019)
9. Liu, X., Abdelhakim, M., Krishnamurthy, P., Tipper, D.: Identifying malicious nodes in multihop iot networks using diversity and unsupervised learning. In: 2018 IEEE International Conference on Communications (ICC). pp. 1–6. IEEE (2018)
10. Liu, X., Abdelhakim, M., Krishnamurthy, P., Tipper, D.: Identifying malicious nodes in multihop iot networks using dual link technologies and unsupervised learning. Open Journal of Internet Of Things (OJIOT) **4**(1), 109–125 (2018)
11. Mahmoud, R., Yousuf, T., Aloul, F., Zualkernan, I.: Internet of things (iot) security: Current status, challenges and prospective measures. In: 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST). pp. 336–341. IEEE (2015)
12. Mazdin, P., Arbanas, B., Haus, T., Bogdan, S., Petrovic, T., Miskovic, N.: Trust consensus protocol for heterogeneous underwater robotic systems. IFAC-PapersOnLine **49**(23), 341–346 (2016)
13. Meng, W.: Intrusion detection in the era of iot: Building trust via traffic filtering and sampling. Computer **51**(7), 36–43 (2018)
14. Meng, W., Choo, K.R., Furnell, S., Vasilakos, A.V., Probst, C.W.: Towards bayesian-based trust management for insider attacks in healthcare software-defined networks. IEEE Trans. Network and Service Management **15**(2), 761–773 (2018)
15. Meng, W., Li, W., Xiang, Y., Choo, K.K.R.: A bayesian inference-based detection mechanism to defend medical smartphone networks against insider attacks. Journal of Network and Computer Applications **78**, 162–169 (2017)
16. Nahiyan, K., Kaiser, S., Ferens, K., McLeod, R.: A multi-agent based cognitive approach to unsupervised feature extraction and classification for network intrusion detection. In: International Conference on Advances on Applied Cognitive Computing (ACC). pp. 25–30 (2017)
17. Rathore, H., Badarla, V., Shit, S.: Consensus-aware sociopsychological trust model for wireless sensor networks. ACM Transactions on sensor networks (TOSN) **12**(3), 21 (2016)
18. Sharma, V., Lee, K., Kwon, S., Kim, J., Park, H., Yim, K., Lee, S.Y.: A consensus framework for reliability and mitigation of zero-day attacks in iot. Security and Communication Networks **2017** (2017)
19. Wang, C., Feng, T., Kim, J., Wang, G., Zhang, W.: Catching packet droppers and modifiers in wireless sensor networks. In: 2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. pp. 1–9. IEEE (2009)
20. Withanage, C., Ashok, R., Yuen, C., Otto, K.: A comparison of the popular home automation technologies. In: Innovative Smart Grid Technologies-Asia (ISGT Asia), 2014 IEEE. pp. 600–605. IEEE (2014)
21. Yun, J., Seo, S., Chung, J.: Centralized trust-based secure routing in wireless networks. IEEE Wireless Commun. Letters **7**(6), 1066–1069 (2018)
22. Zou, J., Ye, B., Qu, L., Wang, Y., Orgun, M.A., Li, L.: A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services. IEEE Transactions on Services Computing (2018)