Detection of Packet Dropping Attack Based on Evidence Fusion in IoT Networks

Weichen Ding,1 Wenbin Zhai,1 Liang Liu,1 Ying Gu,2 and Hang Gao1

1College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
2School of Engineering and Applied Sciences, Columbia University, New York, NY, USA

Correspondence should be addressed to Liang Liu; liangliu@nuaa.edu.cn

Received 7 April 2022; Revised 6 June 2022; Accepted 16 June 2022; Published 9 July 2022

Copyright © 2022 Weichen Ding et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Internet of Things (IoT) is widely used in environmental monitoring, smart healthcare, and other fields. Due to its distributed nature, IoT is vulnerable to various internal attacks. One of these attacks is the packet-dropping attack, which is very harmful. The existing packet-dropping attack detection algorithms are unsuitable for emerging resource-constrained IoT networks. For example, ML-based algorithms always inject numerous packets to obtain the training dataset. However, it is heavyweight for energy-limited nodes to forward these extra packets. In this paper, we propose a lightweight evidence fusion-based detection algorithm (EFDA), which leverages the packet forwarding evidence to identify malicious nodes. Firstly, EFDA finds the sequence numbers of dropped packets and their corresponding source nodes. Then, it traces the routing path of each dropped packet and collects evidence for detection. The evidence stored by nodes around the path record the node’s forwarding behaviors. Finally, the collected evidence is fused to evaluate the trust of nodes. Based on nodes’ trust, the K-means clustering is used to distinguish between malicious nodes and benign nodes. We conduct simulation experiments to compare EFDA with ML-based algorithms. The experimental results demonstrate that EFDA can detect the packet-dropping attack without injecting packets and achieve a higher detection accuracy.

1. Introduction

In the last decade, the Internet of things (IoT) has become a popular infrastructure to support many applications, such as intelligent transportation [1] and smart home [2]. IoT is a system consisting of interrelated computing devices, which collect and process the data acquired from the environments. These devices (such as sensors) cooperate with each other through the IoT protocol, including ZigBee [3], Wi-Fi [4], and Bluetooth.

With the rapid development and application of IoT, it is prone to varied attacks, among which the packet-dropping attack is very hard to detect and prevent. In the packet-dropping attack, malicious attackers can invade and control legitimate devices to discard some essential packets halfway, causing the base station loses the important information. For example, malicious nodes drop the vital packets in the healthcare wireless sensor network (WSN) that contains the alarm information for the patient’s health parameters such as blood pressure and heart rate [5]. If the alarm information is not transmitted to the doctors but dropped halfway, the patients will be at risk. It is vital to detect malicious nodes.

1.1. Motivation. In recent years, many traditional packet-dropping attack detection algorithms have been proposed, but they are not suitable for the emerging resource-constrained IoT networks. For instance, traditional machine learning (ML)-based detection algorithms [6–9] identify malicious nodes by training detection models. The performance of the detection models depends on the size of the training dataset. To get a large size of the training dataset, numerous labeled packets need to be injected into the IoT networks. However, it is heavyweight for energy-limited nodes to forward numerous injected packets. It is crucial to propose a lightweight algorithm for resource-constrained...
IoT networks. To overcome this problem, we propose a lightweight evidence fusion-based detection algorithm (EFDA), which uses the packet forwarding evidence (PFEs) to identify malicious nodes. The PFE is generated during the packet forwarding process. When a node in the network forwards a packet, it locally stores a packet forwarding record. Due to the broadcast characteristic of wireless communication in IoT networks, each neighbor of the node can sniff the packet and generate a PFE.

As shown in Figure 1, EFDA contains three phases.

1. Getting the dropped packet set: the base station needs to find the dropped packets and their corresponding source nodes. For this purpose, the base station divides received packets into groups according to their source nodes. Then, the base station sorts the received packets in each group based on their sequence numbers. The dropped packets can be found because their sequence numbers are not in the groups. Figure 1 shows an example that the base station divides the received packets into two groups according to two source nodes: $N_1$ and $N_2$. After sorting the packets in each group, it finds that the dropped packet is $N_1.Packet_1$, which is the identifier of the packet whose corresponding source node is $N_1$.

2. Collecting PFEs: for each dropped packet, the base station traces its routing path and finds the suspicious nodes. In Figure 1, the base station sends a request to $N_1$ to ask it for the next forwarding node of $N_1.Packet_1$; $N_1$ searches its forwarding records and finds that the next forwarding node of $N_1.Packet_1$ is $N_2$, on behalf of the base station, $N_1$ asks $N_2$ for the next forwarding node of the packet; $N_2$ searches its forwarding records and finds that the next forwarding node is $N_3$; $N_2$ continues to ask $N_4$ for the next forwarding node of the packet; $N_4$ reports the next forwarding node of the packet is $N_6$. But after $N_4$ asks $N_6$, $N_6$ reports that it has never received the packet. At this moment, the base station finds a logic conflict between $N_4$ and $N_6$, and then, it identifies $N_4$ and $N_6$ as suspicious nodes. To resolve the logic conflict and find the liar, the base station collects PFEs stored by the neighbors of $N_4$, namely $N_2$, $N_3$, and $N_5$.

3. Fusing PFEs: the base station fuses the collected PFEs. Because $N_2$, $N_3$, and $N_5$ provide PFEs to prove that $N_4$ has forwarded $N_1.Packet$, to $N_6$, the base station discovers that $N_6$ lies to hide its dropping packet behavior.

As mentioned above, EFDA does not need to inject extra packets to obtain the training dataset to train the detection model, and it utilizes the existing PFEs in the network to perform logical reasoning and identify malicious nodes.

In summary, the contributions of this paper are as follows.

We propose a lightweight evidence fusion based packet dropping attack detection algorithm (EFDA) for the resource-constrained IoT networks. EFDA uses the packet forwarding evidences to detect malicious nodes, which achieves a high detection accuracy with a low cost. We conduct simulation experiments to systematically evaluate our detection algorithm. The experimental results show that EFDA provides better detection accuracy than ML-based algorithms.

1.2. Organization. The remainder of this paper is organized as follows. Section 2 introduces the related work of the packet dropping attack detection in IoT. Section 3 formalizes the packet dropping attack. Section 4 details our detection algorithm, EFDA. Section 5 shows the results of the simulation experiments. Section 6 concludes this paper.

2. Related Work

To resist the packet dropping attack, a wide variety of algorithms are proposed in recent researches, which can be divided into five categories: monitor-based algorithms, acknowledgment-based algorithms, camouflage-based algorithms, ML-based algorithms, and other algorithms.

2.1. Monitor-Based Algorithm. The core of monitor-based algorithms is to place monitoring nodes among communication nodes and classify them into “normal” and “abnormal” by the collected traffic data [10]. Watchdog [11] is a basic technology for the packet dropping attack detection, where a monitoring node sniffs the traffic of the next hop to detect the attacks of malicious nodes. Li et al. [12] applied the watchdog to monitor the behavior of nodes rather than traffic data, and they detected malicious nodes by comparing the interval of sending and receiving packets with the threshold. In the monitor-based detection (CMD) [13], each node monitors the packet loss rates of its preferred parent node and its one-hop neighbor nodes. By comparing the
packet loss rate of its preferred parent and one-hop neighbor nodes, the monitoring node can find the abnormal behaviors of its preferred parent node.

2.2. Acknowledgment-Based Algorithms. The acknowledgment-based algorithms depend on the acknowledgment (ACK) packet to detect malicious nodes [14]. Each node is responsible for monitoring the forwarding behaviors of its next node and reporting it to the base station by sending ACK packets. In adaptive acknowledgment-based approach (AAA) [15], each node monitors its one-hop and two-hop downstream nodes. After forwarding a data packet, the node overhears the forwarding behavior of its one-hop downstream node and waits to receive an ACK packet from its two-hop downstream node. Once receiving no ACK packet, the node identifies its one-hop downstream node as a malicious node. In single checkpoint-based detection (SCAD) [16], the source node randomly selects an intermediate node on the routing path as the checkpoint node for each packet. After receiving the packet, both the sink node and checkpoint node need to reply an ACK packet to the source node. If receiving no ACK packet, other intermediate nodes will send an alarm packet to the source node to suspect their downstream nodes, which are identified as malicious nodes.

2.3. Camouflage-Based Algorithms. In the energy harvesting motivated networks (EHNets), some nodes called energy harvesting node need to periodically harvest energy from an immediate environment. In camouflage-based active detection (CAM) [17], each node actively disguises it as an energy harvesting node and pretends not to overhear its adjacent nodes. But actually, each node monitors any forwarding behaviors of its adjacent nodes. Once finding abnormal behaviors, they identify that adjacent node as a malicious node. In the EYES [18], each node not only actively disguises itself as an energy harvesting node to overhear the forwarding behaviors of its adjacent nodes but also validates any previous uncertain forwarding behavior to detect malicious nodes.

2.4. ML-Based Algorithms. Machine learning (ML) is a common and efficient technology, which has been widely used in malicious node detection. Akbani et al. [19] combined the ML with the reputation systems (RS), which automates the process of designing the RS model. Liu et al. [20] proposed a trust system, which calculated the trust of each node by the trust of each routing path. Based on the trusts of nodes, they were divided into malicious or benign group. Liu et al. [21] improved this scheme, and they used the method of linear regression to calculate the trust of nodes, which was more accurate than [20]. Also, they took into account the possibility that nodes launched the multiple-mix-attack. Yang et al. [22] considered a more fine-grained attack named selective-edge packet attack, and they argued that malicious nodes may be more intelligent to launch an attack selectively. Also, they selected the best scheme after sifting through various types of regression algorithms and clustering algorithms.

2.5. Other Algorithms. In [23], due to most of the detection algorithms are for the centralized networks, blockchain-based multimobile code-driven trust mechanism (BMCTM) is proposed to detect malicious nodes in decentralized networks. It combines the blockchain technology and trust system, which detects nodes as malicious nodes according to their low trust. A secure routing framework is proposed in [24], which leverages a new type of packet called dummy packet to detect malicious nodes. The dummy packet scheme is used to find the critical routes and detect malicious nodes in the critical routes. In [25], considering malicious nodes may lie to attract and drop packets during route establishment phase, and a robust hybrid method is proposed to strengthen the route security.

Most of the above algorithms are heavyweight for the emerging resource-constrained IoT networks. For monitor-based algorithms, acknowledgment-based algorithms, and camouflage-based algorithms, the energy-limited nodes need to monitor the forwarding behaviors of their neighbor nodes and to converge collected data all the times. For ML-based algorithms, the energy-limited nodes need to assist them to obtain the training dataset by forwarding numerous injected packets. They are heavyweight for energy-limited nodes. Therefore, in this paper, we propose a lightweight evidence fusion-based detection algorithm (EFDA) to achieve a high detection accuracy.

3. Network and Attack Model

In this section, the network model is introduced, and the packet-dropping attack is formalized. Table 1 exhibits a list of notations for later reference.

3.1. Network Model. In this paper, the IoT network is a multihop wireless network consisting of sensor nodes, which communicate with each other through the routing protocol for low-power and lossy networks (RPL). The sensor nodes collect data and encapsulate them into packets. The packets are forwarded by relay nodes to the base station. A typical IoT network is shown in Figure 2.

A node is represented as $N_i$ $(i \in [1, M])$, and the base station is represented as $S$. Each node has at least one routing path to the base station $S$. A routing path is represented as $Path_j$ $(j \in [1, K])$, which is expressed as

$$Path_j = [N_1 \rightarrow N_2 \rightarrow \cdots \rightarrow N_j \rightarrow S],$$

where it represents a packet which is sent from $N_1$, forwarded through $N_2, \ldots, N_j$, in a sequence, and finally received by the base station $S$.

Then, the network is expressed as

$$\text{Network} = (N, S, P),$$

$$N = \{N_1, N_2, \ldots, N_i, \ldots, N_M\},$$

$$P = \{Path_1, Path_2, \ldots, Path_j, \ldots, Path_K\},$$

where $N$ is the set of nodes in the network, and $P$ is the set of routing paths in the network. Network is the network...
consisting of the sensor nodes, the base station, and the routing paths.

### 3.2. Packet Dropping Attack Model
If there are no malicious nodes, a packet will arrive at the base station. However, the packet may be discarded halfway if there are malicious nodes.

As shown in Figure 3, the malicious node $N_6$ drops the packet $N_1 \cdot \text{Packet}_2$. In this paper, the malicious nodes may launch the packet-dropping attack with a certain probability $P_d$ ($P_d \in (0, 1)$). We use $N_i, P_d$ to represent the probability that $N_i$ launches a packet-dropping attack. Considering the harmfulness of the packet-dropping attack and the constrained resources of the IoT network, the malicious nodes should be detected with a high accuracy and a low overhead.

### 3.3. PFE Model
The packet forwarding evidences (PFEs) are generated during the packet transmission. Due to the broadcast characteristic of wireless communication, when a node $N_f$ forwards a packet $N_i \cdot \text{Packet}_j$, all neighbors of $N_f$ can sniff the packet. The receiving node $N_r$ will receive the packet, and other neighbors of $N_r$ generate PFEs to record the forwarding behavior of $N_f$. The generated PFE can be represented as

$$\text{PFE} = (N_f, N_i \cdot \text{Packet}_j, N_r), \quad (3)$$

where it represents the neighbors of $N_f$ witness that $N_f$ has forwarded the packet $N_i \cdot \text{Packet}_j$ to $N_r$.

As shown in Figure 4, $N_1$ wants to forward the packet $N_1 \cdot \text{Packet}_2$ to $N_6$. Due to the broadcast characteristic of wireless communication, all neighbors of $N_4$ can sniff the packet. $N_6$ receives the packet, and the other neighbors ($N_2, N_3, N_5, N_{10}$) of $N_4$ generate a PFE, namely ($N_4, N_1 \cdot \text{Packet}_2, N_6$).

### 4. Algorithm
In this section, we introduce our evidence fusion-based detection algorithm (EFDA), which is divided into three phases. (1) Getting the dropped packet set: the base station finds the dropped packet set and the source node of each dropped packet. (2) Collecting PFEs: for each dropped packet, the base station traces its routing path and finds the suspicious nodes. PFEs stored by neighbors of suspicious nodes are collected. (3) Fusing PFEs: the base station fuses the collected PFEs to detect malicious nodes.

#### 4.1. Getting the Dropped Packet Set
The source nodes collect data from the environment, encapsulate them into packets, and then upload the packets to the base station. After receiving the packets, the base station divides the received packets into different groups $G_i$ ($i \in [1, M]$) according to their source nodes $N_i$. For each group, the base station sorts the packets according to their sequence numbers. After grouping and sorting the received packets, the base station can find the dropped packets and their corresponding source nodes.

As Figure 5 shows, the base station divides the received packets into $M$ groups and sorts the packets for each group. For the first group of the source node $N_1$, the base station receives the packets with sequence number $N_1 \cdot \text{Packet}_1, N_1 \cdot \text{Packet}_2$, and $N_1 \cdot \text{Packet}_3$, except $N_1 \cdot \text{Packet}_4$. So it finds that $N_1 \cdot \text{Packet}_4$ is dropped. After checking all groups, the base station can obtain the dropped packet set.

#### 4.2. Collecting PFEs
After finding all the dropped packets, the base station traces the routing path of each dropped packet. In the process of tracing, the base station investigates the nodes on the routing path hop by hop. In the final hop, it...
can find two suspicious nodes that may drop the packet. To judge the suspicious nodes, PFEs about them are collected at the base station. We propose an evidence collection protocol (ECP) to trace the routing path and collecting PFEs.

In order to assist ECP to trace the routing path of each dropped packet, each node in the network needs to generate records about its packet forwarding behaviors. Each node maintains a packet forwarding record table (PFRT) to store the records, which is shown in Table 3. It contains three fields: Last Node, Packet-ID, and Next Node. Last Node means the last node that forwards the packet, Packet-ID means the identifier of the forwarded packet, and Next Node means the next node where the packet is forwarded. After a node receives a packet and forwards the packet to another node, it will update its PFRT to record the forwarding behavior.

As shown in Figure 6, during the transmission of the packet $N_j$.Packet$_2$, $N_4$ receives the packet from $N_2$ and forwards it to $N_6$. To record this forwarding behavior, $N_4$ inserts a record $(N_2, N_1$.Packet$_2, N_6)$ into its PFRT. Besides, malicious nodes may not update their PFRTs because they drop packets instead of forwarding them.

Based on the packet forwarding records stored by nodes, ECP can trace the routing path of each dropped packet. For a dropped packet $N_j$.Packet$_j$, the process of tracing the packet can be described as follows.

The base station finds the dropped packet $N_j$.Packet$_j$ and its corresponding source node $N_i$. It constructs a TM message (shown in Table 4) $tm$ ["Packet-ID": "$N_j$.Packet$_j$"], and sends it to $N_i$. The message $tm$ is used to ask $N_i$ for the next forwarding node of $N_j$.Packet$_j$. After that, the base station initializes the tracing progress as $[N_i]$. Once receiving $tm$, node $N_i$ searches its PFRT for the packet forwarding record about the packet. It finds the next forwarding node as $N_j$. It constructs a RM message (shown in Table 5) $rm$ ["Successor": "$N_j$"], and sends it to the base station. The message $rm$ is used to report the tracing progress to the base station. Besides, $N_i$ forwards $tm$ to $N_j$ to ask it to continue to trace the routing path of the packet. When the base station receives $rm$, it updates the tracing progress as $[N_i \rightarrow N_j]$. After receiving $tm$, node $N_j$ searches its PFRT for the packet forwarding record about the packet. After several steps of tracing, the tracing progress is updated to $[N_i \rightarrow N_j \rightarrow \cdots \rightarrow N_k \rightarrow N_m]$, and a malicious node $N_m$ receives $tm$.

As shown in Figure 7, the base station finds the dropped packet $N_j$.Packet$_j$ and its corresponding source node $N_i$. Then, it sends a TM message $tm$ to $N_i$ and initializes the tracing progress as $[N_i]$. Once receiving $tm$, node $N_i$ searches its PFRT and finds the next forwarding node as $N_j$. It sends a RM message $rm_1$ to the base station and forwards $tm$ to $N_j$. When the base station receives $rm_1$, it updates the tracing progress as $[N_i \rightarrow N_j]$. Once receiving $tm$, node $N_j$ continues to trace the routing path of the packet. After several steps of tracing, the tracing progress is updated to $[N_i \rightarrow N_j \rightarrow \cdots \rightarrow N_k \rightarrow N_m]$, and $N_j$ receives $tm$ from $N_i$. We assume that node $N_6$ is a malicious node.

When a malicious node $N_m$ receives $tm$, there are three possible cases as follows.

**Case 1.** The malicious node does not respond to the base station.

The continued tracing process is described as follows. After receiving $tm$, the malicious node $N_m$ does not respond to the base station. Once receiving no response from $N_m$, the base station identifies $N_m$ as a malicious node.
Security and Communication Networks

As shown in Case 1 of Figure 7, $N_4$ has received $tm$ from $N_6$, but it does not respond to the base station. The base station identifies $N_6$ as a malicious node.

**Case 2.** The malicious node responds that it has never received the packet forwarded by its predecessor.

The continued tracing process is described as follows:

After receiving $tm$, the malicious node $N_m$ denies that it has received $N_i$ Packet from $N_k$. So, it constructs an IM message (shown in Table 6) $im$ ("Packet-ID": $N_i$.Packet$_j$,"Impeaching-Node": $N_m$"Impeached-Node": $N_k$) and sends it to the base station. The message $im$ is used to report the base station that $N_m$ has never received $N_i$ Packet from $N_k$. When the base station receives $im$, it finds a logic conflict between $N_k$ and $N_m$. It identifies $N_k$ and $N_m$ as suspicious nodes.

**Table 3: Packet forwarding record table of $N_6$.**

<table>
<thead>
<tr>
<th>Last node</th>
<th>Packet-ID</th>
<th>Next node</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_2$</td>
<td>$N_1$.Packet$_2$</td>
<td>$N_6$</td>
</tr>
</tbody>
</table>

**Figure 5: The process of getting the dropped packets set.**

**Table 4: Tracing message (TM).**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet-ID</td>
<td>The identifier of a dropped packet</td>
</tr>
</tbody>
</table>

**Table 5: Reporting message (RM).**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successor</td>
<td>The next forwarding node of the dropped packet</td>
</tr>
</tbody>
</table>

As shown in Case 2 of Figure 7, $N_6$ denies that it has received $N_i$ Packet$_j$ from $N_k$. Then, it sends an IM message $im_1$ to the base station. After receiving $im_1$, the base station identifies $N_4$ and $N_6$ as suspicious nodes.

**Case 3.** The malicious node responds that it has forwarded the packet to a neighbor, but actually not.

The continued tracing process is described as follows:

After receiving $tm$, the malicious node $N_m$ lies that it has forwarded $N_i$ Packet$_j$ to $N_k$. So, it sends $rm$ ("Successor": $N_m$"Impeached-Node": $N_k$) to the base station. Besides, it forwards $tm$ to $N_m$. When the base station receives $rm$, it updates the tracing progress as [$N_i \rightarrow N_j \rightarrow \ldots \rightarrow N_k \rightarrow N_m \rightarrow N_n$]. Once receiving $tm$, node $N_n$ searches its PFRT but finds no packet forwarding record about $N_i$ Packet$_j$. So, it sends $im$ ("Packet-ID": $N_i$.Packet$_j$,"Impeaching-Node": $N_m$"Impeached-Node": $N_k$) to the base station to deny that it has received $N_i$ Packet$_j$ from $N_k$. When the base station receives $im$, it finds a logic conflict between $N_m$ and $N_n$. It identifies $N_m$ and $N_n$ as suspicious nodes.

As shown in Case 3 of Figure 7, $N_6$ lies that the next forwarding node is $N_8$. It sends $rm_4$ to the base station and forwards $tm$ to $N_8$. When the base station receives $rm_4$, it updates the tracing progress as [$N_1 \rightarrow N_2 \rightarrow N_4 \rightarrow N_6 \rightarrow N_8$]. Once receiving $tm$, node $N_8$ sends $im_2$ to the base station to deny that it has received $N_i$ Packet$_j$ from $N_k$. When the base station receives $im_2$, it identifies $N_6$ and $N_8$ as suspicious nodes.

After the tracing process of a dropped packet like $N_i$ Packet$_j$, the base station can get two suspicious nodes like $N_k$ and $N_m$ ($N_m$ and $N_n$). To find the liar in them, the base station needs to collect PFES about them as follows.

Without loss of generality, we suppose the suspicious nodes are $N_k$ and $N_m$.

As shown in Case 2 of Figure 7, $N_6$ denies that it has received $N_i$ Packet$_j$ from $N_k$. Then, it sends an IM message $im_1$ to the base station. After receiving $im_1$, the base station identifies $N_4$ and $N_6$ as suspicious nodes.

As shown in Case 3 of Figure 7, $N_6$ lies that the next forwarding node is $N_8$. It sends $rm_4$ to the base station and forwards $tm$ to $N_8$. When the base station receives $rm_4$, it updates the tracing progress as [$N_1 \rightarrow N_2 \rightarrow N_4 \rightarrow N_6 \rightarrow N_8$]. Once receiving $tm$, node $N_8$ sends $im_2$ to the base station to deny that it has received $N_i$ Packet$_j$ from $N_k$. When the base station receives $im_2$, it identifies $N_6$ and $N_8$ as suspicious nodes.

**Figure 6: An example of updating the PFRT.**

As shown in Case 2 of Figure 7, $N_6$ denies that it has received $N_i$ Packet$_j$ from $N_k$. Then, it sends an IM message $im_1$ to the base station. After receiving $im_1$, the base station identifies $N_4$ and $N_6$ as suspicious nodes.

As shown in Case 3 of Figure 7, $N_6$ lies that the next forwarding node is $N_8$. It sends $rm_4$ to the base station and forwards $tm$ to $N_8$. When the base station receives $rm_4$, it updates the tracing progress as [$N_1 \rightarrow N_2 \rightarrow N_4 \rightarrow N_6 \rightarrow N_8$]. Once receiving $tm$, node $N_8$ sends $im_2$ to the base station to deny that it has received $N_i$ Packet$_j$ from $N_k$. When the base station receives $im_2$, it identifies $N_6$ and $N_8$ as suspicious nodes.
The base station constructs a CEM message (shown in Table 7) cem [“Expected-PFE”: “(Nm, Nj, Packetj, Nm)”) and sends it to the precursor Nk of the suspicious nodes. The message cem is used to collect PFEs. Once receiving cem, Nk constructs a REM message (shown in Table 8) rem [“Expected-PFE”: “(Nm, Nj, Packetj, Nm)”) and sends it to its neighbors. The message rem is used to request the neighbors to send the relevant PFEs to the base station. When each neighbor of Nk receives rem, it searches its PFET for the expected PFE. Once getting the matched PFE, each neighbor constructs an EM message (shown in Table 9) em [“PFE”: “(Nm, Nj, Packetj, Nm)”) and sends it to the base station. But, the accomplice does not submit correct PFE by sending em [“PFE”: “(No Evidence)”) or submits a faked PFE to the base station. After receiving all EM messages, the base station extracts the PFEs in them.

As shown in Case 2 of Figure 7, the base station finds the suspicious nodes N4 and N6. It sends a CEM message cem1 to the precursor N4. Once N4 receives cem1, it sends a REM message rem1 to its neighbors. When the neighbors (N2, N3, N5, and N10) receive rem1, benign neighbor N2 (N3, N5) searches its PFETs and sends the matched PFE by an EM message em1 (em2, em3) [“PFE”: “N4, N1, Packet1, N6”) to the base station, but the accomplice N10 sends em1 [“PFE”: “(No Evidence)”) to the base station. After receiving all EM messages, the base station extracts all PFEs.

Table 6: Impeaching message (IM).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet-ID</td>
<td>The identifier of a dropped packet</td>
</tr>
<tr>
<td>Impeaching-node</td>
<td>The node impeaches that impeached node has not forwarded the packet to it</td>
</tr>
<tr>
<td>Impeached-node</td>
<td>The node is impeached</td>
</tr>
</tbody>
</table>

Table 7: Collecting evidences message (CEM).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected-PFE</td>
<td>The PFE is expected by the base station</td>
</tr>
</tbody>
</table>

Table 8: Requesting evidence message (REM).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requested-PFE</td>
<td>The PFE is requested by the sending node</td>
</tr>
</tbody>
</table>

After tracing the routing path of a dropped packet by ECP, the base station can get two suspicious nodes and their relevant PFEs.

4.3. Fusing PFEs. After the base station finds the suspicious nodes and collects their relevant PFEs by ECP, we propose an evidence fusion algorithm (EFA) to fuse these PFEs and detect malicious nodes.

The evidence fusion is actually a voting process. For the suspicious nodes Nk and Nm about Nc, Nk’s neighbors send either PFE (Nk, Nj, Packetj, Nm) or PFE (No Evidence) to the base station. PFE (Nk, Nj, Packetj, Nm) means a neighbor witnesses Nk has forwarded Nc, Packetj to Nm, and it votes for Nk. PFE (No Evidence) means a neighbor regards Nk as the liar, and it votes for Nm. The number of votes for a node is represented as V, and V (V) is the number of votes for Nk (Nm). Malicious neighbors may submit a faked PFE to vote for its accomplices. To mitigate the effects of the collusion among malicious nodes, we use nodes’ weights to multiply nodes’ votes. The weight of Ni is represented as ωi, and it is the ratio of Ni’s trust to the initial value, namely ωi = ti/T. The trust of Ni (i ∈ [1, M]) is represented as ti, and the initial value of ti is T. The base station maintains a trust and weight table (TWT) to record the trusts and weights of all nodes.
As the votes are weighted, the number of votes for a node is the sum of the weights of the neighbors that have voted for it. After fusing PFEs, the base station identifies the node with fewer votes as the liar and punishes it by decreasing its trust.

For the dropped packet \( N_1 \), packet 2 in Case 2 of Figure 7, the base station gets four PFEs \( \{ N_5, N_1, \text{Packet}_2, N_6 \} \), \( N_3 \), \( N_5 \), and \( N_6 \) witness that \( N_4 \) has forwarded \( N_1, \text{Packet}_2, N_6 \), and they regard \( N_4 \) as a benign node and vote for it. But \( N_5 \) denies that \( N_4 \) has forwarded the packet to \( N_6 \), and it regards \( N_6 \) as a benign node and votes for it. Because \( \omega_1, \omega_3, \omega_5, \) and \( \omega_{10} \) are initialized as 1, \( v_4 = \omega_2 + \omega_5 + \omega_6 = 3 \) and \( v_6 = \omega_{10} = 1 \). By comparing \( v_4 \) with \( v_6 \), the base station identifies \( N_6 \) as the liar and decreases its trust.

For a dropped packet, ECP traces its routing path to find two suspicious nodes and the relevant PFEs, and EFA fuses these PFEs to discover the liar and decreases its trust. After ECP traces all dropped packets and EFA punishes all liars over, the final trusts of nodes can be obtained. Based on the final nodes’ trusts, the K-means clustering is used to cluster nodes into two groups: malicious group (MG) and benign group (BG).

As shown in Algorithm 1, EFDA contains three steps:

1. Getting the dropped packet set: the base station divides the received packets into different groups \( G_i \) according to their source nodes (line 3–5) and sorts the packets according to their sequence numbers for each group (line 6–7). After grouping and sorting the received packets, the base station can find the dropped packet set, namely DPS (line 8–12).

2. Collecting PFEs: for each dropped packet, the base station sends a TM message to the source node \( N_i \) to start a tracing process (line 14–15). The current node \( N_{\text{cur}} \) finds the successor \( N_i \) and continues the tracing process until the base station finds two suspicious nodes \( N_k, N_m \) (line 17–23). The base station collects PFEs about \( N_k, N_m \) to judge them (line 24–28).

3. Fusing PFEs: for two suspicious nodes, the base station fuses their relevant PFEs to update their votes (line 29–35). The node with fewer votes is punished by decreasing its trust (line 36–41). Based on the final nodes’ trusts, the K-means clustering is used to cluster nodes to BG and MG (line 43).

### 4.4. Algorithm Analysis

#### 4.4.1. Algorithm Complexity Analysis

According to the pseudocode in Algorithm 1, the proposed approach contains three steps: (1) getting the dropped packet set; in order to get the dropped packet set, EFDA needs to traverse the received packet set (RPS). The complexity of the first step is \( \Omega_1 = \text{size}(\text{RPS}) \), where \( \text{size}(\text{RPS}) \) means to find the size of the set RPS. (2) Collecting PFEs: in order to collect PFEs, EFDA needs to traverse the dropped packet set (DPS). For each dropped packet, EFDA needs to trace the routing path of the dropped packet. The complexity of the second step is \( \Omega_2 = \text{size}(\text{DPS}) \times \text{size}(\text{Path}) \), where Path means the traced routing path. (3) Fusing PFEs: in order to fuse PFEs, EFDA needs to traverse the PFEs for each dropped packet. The complexity of the third step is \( \Omega_3 = \text{size}(\text{DPS}) \times \text{size}(\text{PFEs}) \). Therefore, the complexity of EFDA is represented as \( O = O_1 + O_2 + O_3 = \text{size}(\text{RPS}) + \text{size}(\text{DPS}) \times \text{size}(\text{Path}) + \text{size}(\text{PFEs}) \).

#### 4.4.2. Algorithm Overheads Analysis

1. **Energy Overheads of EFDA.** EFDA detects malicious nodes by tracing the routing paths of dropped packets. In addition, it can get the detection results in a limited number of dropped packets. We assume that the limited number of dropped packets is \( L \). For each tracing process, considering the worst case, all nodes on the routing path need to send a TM message and a RM message. Only some specified nodes need to send an IM message, a REM message, or an EM message. Therefore, each node sends no more than 3 extra messages for one tracing process. Because EFDA needs to trace \( L \) dropped packets, each node sends no more than \( 3 \times L \) extra messages for one time detection. Moreover, since the sizes of the above messages are small, the energy overheads of sending them are small.

2. **Storage Overheads of EFDA.** In EFDA, each node needs to store the packet forwarding records and PFEs, and we estimate the storage overheads of EFDA to prove its feasibility. The storage overheads of a node are affected by the sizes of its PFRT and PFET. A general IoT device forwards 1200 messages/minute according to the study in [27]. Assuming that EFDA is executed every hour to detect malicious nodes. During this period, there are 72000 messages forwarded and 72000 packet forwarding records stored by a node. For a packet forwarding record, it contains three fields, and its storage overheads are 3 Bytes. The storage overheads of PFRT are 72000 × 3 = 216000 Bytes ≈ 210 kB. For PFET, its capacity C (shown in Table 2) is approximate to the number of forwarded packets, namely 72000. For a PF, it contains three fields, and its storage overheads are 3 Bytes. The storage overheads of PFET are 72000 × 3 = 216000 Bytes ≈ 210 kB. So, the storage overheads of a node are 210 + 210 = 420 kB, which is far less than a general IoT device’s storage 2 GB [28].

#### 4.4.3. Distinction between EFDA and ML-Based Algorithms

In this section, we analyze the distinction between EFDA and ML-based algorithms. For each injected packet, ML-based algorithms use it to calculate the trust of its routing path by mathematical reasoning. Then, they use the
Routing path’s trust to estimate the nodes’ trusts on the routing path. However, in order to estimate the nodes’ trusts more accurately, numerous packets need to be injected to get more routing path’s trusts, which are used as the input of the ML-based algorithms.

On the contrary, EFDA detects malicious nodes without injecting packets. It can trace the routing path of each dropped packet and find the suspicious nodes. The PFEs around suspicious nodes are collected to the base station, and EFDA fuses them to find the malicious nodes. A potential constraint for EFDA is how to resist collusive attacks. Suppose that a benign node is surrounded by many malicious nodes, they submit faked PFEs that cause the base station to misidentify the benign node as a malicious node. A possible extension is to use the causal inference algorithm to solve the problem.
5. Performance Evaluation

In this section, we evaluate the performance of our proposed EFDA and compare it with two typical ML-based algorithms, namely HD [20] and PDE [21].

Both HD and PDE need to inject numerous labeled packets into the network and collect them at the base station. Each labeled packet has a routing path, and each routing path has abundant labeled packets. For each routing path, not all labeled packets on the routing path can be collected by the base station due to the malicious nodes. HD and PDE define the trust of the routing path as a ratio, which is the number of collected labeled packets to the total number of labeled packets on the routing path. According to whether a node is on the routing path, the relationship between the trust of nodes and the trust of the routing path can be formalized as a mathematical equation. The mathematical equation can be solved by machine learning algorithms, and the trust of nodes can be obtained. Based on the trust of nodes, the clustering algorithm classifies them into benign group and malicious group.

We evaluate accuracy and error rate to compare detection performance. As shown in Table 10, the accuracy is defined as \( P_a = \frac{(TP + TN)}{(P + N)} \), and the error rate is defined as \( F_a = \frac{(FP)}{(FP + TN)} \).

### 5.1. Experimental Environment

#### 5.1.1. Environmental Settings

In our environment, all nodes are evenly distributed in a rectangle area of 100 × 100 m², and each node’s communication range is 10 m. Our IoT network is generated randomly, and there is at least one routing path from each source node to the base station.

To avoid bias, we run our simulation for each experiment in 10 rounds with 10 different networks generated randomly. The average value of 10 rounds’ result is calculated as the final experimental result of each experiment. In particular, we use the simulator in [21] and add our EFDA to it. Both EFDA and the ML-based algorithms are deployed at the base station.

#### 5.1.2. Environmental Variables

In the following experiments, we investigate the impact of the variables (shown in Table 11) on the detection performance. Unless otherwise specified, all experimental variables will remain the default, which is set as follows.

- The number of uploaded packets
- The number of nodes
- The percentage of malicious nodes
- The probability of attack
- The diversity of network

The number of uploaded packets is 500. The number of nodes is 15. The probability of an attack is 0.3. The percentage of malicious nodes is 0.3. The diversity of the network is 1.

### 5.2. HD vs PDE vs EFDA

In this section, we explore the performance comparison among HD, PDE, and EFDA through experiments.

#### 5.2.1. Impact of the Number of Uploaded Packets

The results in Figure 8 show that EFDA performs better than HD and PDE. When the number of uploaded packets is small, HD and PDE get a low \( P_a \). As the number of uploaded packets increases, HD and PDE can get a higher \( P_a \). EFDA gets a stale \( P_a \) in all cases. This is because as the number of uploaded packets increases, HD and PDE can calculate more routing path’s trust to estimate nodes’ trusts. Once more collected information is used to estimate nodes’ trusts, HD and PDE can get more exact nodes’ trusts and get more accurate detection results. EFDA can trace the routing path to find the suspicious nodes and detect malicious nodes in a smaller detection range. EFDA hardly needs abundant

### Table 10: Experimental evaluation.

<table>
<thead>
<tr>
<th>Actual result</th>
<th>Detection result</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative</td>
<td>True positive (TP)</td>
<td>False negative (FN)</td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>False positive (FP)</td>
<td>True negative (TN)</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>( P' ) (detect negative)</td>
<td>( N' ) (detect positive)</td>
<td>( P + N )</td>
</tr>
</tbody>
</table>

### Table 11: Variables and description.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of uploaded packets</td>
<td>The number of packets that are uploaded to the base station will influence the detection accuracy.</td>
</tr>
<tr>
<td>The number of nodes</td>
<td>It means the number of nodes deployed in the network, which can affect the scale of the network and the detection accuracy.</td>
</tr>
<tr>
<td>The percentage of malicious nodes</td>
<td>It means that how many nodes are malicious in the network, which can affect the detection result.</td>
</tr>
<tr>
<td>The probability of attack</td>
<td>Malicious nodes launch the packet-dropping attack with a probability, and less probability means that the node is more difficult to be detected. It can influence the detection accuracy.</td>
</tr>
<tr>
<td>The diversity of network</td>
<td>It essentially indicates the ratio of available routing paths that could be chosen by source nodes to upload packets. The diversity of network reflects the routing paths’ complexity, and it influences the detection accuracy.</td>
</tr>
</tbody>
</table>
routing path’s information to estimate nodes’ trusts, so it can get a stable detection results.

5.2.2. Impact of the Number of Nodes. The results in Figure 9 show that when the number of nodes is small, all algorithms get a high $P_a$ and a low $F_a$, but when the number of nodes increases, the accuracy $P_a$ of all algorithms decreases, and the error rate $F_a$ of them increases. EFDA still performs better than HD and PDE in all cases. This is because when the number of nodes is 5, the network topology is simple, and malicious nodes are more easily to be detected; when the number of nodes increases and the network topology becomes more complex, the malicious nodes are more likely to hide their abnormal behaviors, and it is difficult to identify all malicious nodes. However, no matter how complex the network topology becomes, EFDA still reaches higher accuracy than HD and PDE.

5.2.3. Impact of the Percentage of Malicious Nodes. The results in Figure 10 show that EFDA gets the better results than the other two detection algorithms; but with the percentage of malicious nodes increases, the accuracy $P_a$ of EFDA is getting lower and the error rate $F_a$ of EFDA is getting higher, while the trends of HD and PD remain stable. This is because when the percentage of malicious nodes increases, the number of malicious nodes in the network will also increase that leads to more malicious nodes cooperate to resist EFDA. Assuming that most of the neighbors around a benign node are malicious, the malicious neighbors vote for its accomplice, which causes the benign node to be
misidentified as the liar, and its trust is decreased by EFDA. Because EFDA misidentified the benign node as a malicious node, it gets a lower $P_a$ and a higher $F_a$. Although we have optimizations for the collusion among malicious nodes, it is difficult to resist the collusion attacks from many malicious nodes.

5.2.4. Impact of the Probability of Attack. The results in Figure 11 show that EFDA performs better than HD and PDE. When the probability of attack is small, EFDA gets a small $P_a$ and a large $F_a$. However, when the probability of attack increases, the accuracy $P_a$ of EFDA begins to increase, and the error rate $F_a$ of EFDA begins to decrease. The trends of HD and PDE are similar, but their accuracy $P_a$ is lower than that of EFDA, and their error rate $F_a$ is higher than that of EFDA. This is because when the probability of attack is small, malicious nodes intend to hide their attack behaviors that make EFDA more difficult to detect them. However, when the probability of attack becomes larger, malicious nodes are more likely to launch a packet dropping attack that makes EFDA find more dropped packets. EFDA traces more routing paths of the dropped packets and finds more suspicious nodes, and it gets more accurate detection results.

5.2.5. Impact of the Diversity of Network. The results in Figure 12 show that when the diversity of network is low, both HD and PDE get a low $P_a$ and a high $F_a$. With the diversity of network increases, their accuracy $P_a$ becomes higher, and their error rate $F_a$ becomes lower. However, EFDA gets stable accuracy $P_a$ and error rate $F_a$ in all cases, and they are better than those of HD and PDE. This is because when the diversity of network is low, there are few...
routing paths for source nodes to upload packets to the base station. It means HD and PDE obtain few routing paths’ information to estimate the nodes’ trusts, and that causes HD and PDE to get the inaccurate nodes’ trusts. Therefore, they get negative detection results. As the diversity of network becomes larger, HD and PDE obtain more routing paths’ information to estimate the nodes’ trusts, and they get positive detection results. However, EFDA does not need more different routing paths’ information to estimate the nodes’ trusts. It can trace the path of dropped packet and accurately find the suspicious nodes on the path, and it only decreases the liar’s trust. So, EFDA detects malicious nodes more efficiently than HD and PDE.

5.3. Discussion and Limitations. In the experiments, we explore the performance comparison between HD, PDE, and EFDA on five variables, which are the number of uploaded packets, the number of nodes, the percentage of malicious nodes, the probability of attack, and the diversity of the network. Overall, it is observed that EFDA can achieve better detection performance compared with HD and PDE. EFDA can improve the detection rate by around 20% to 30%.

Although EFDA performs better than HD and PDE, there are some limitations that can be addressed in our future work. When the percentage of malicious nodes exceeds 50%, the detection performance of EFDA declines significantly, which indicates that EFDA is difficult to resist the collusion of numerous malicious nodes. In our future work, we plan to investigate how to resist the collusion of numerous malicious nodes.

6. Conclusion

Due to the distributed nature of the IoT networks, they are vulnerable to the packet-dropping attack. There are abundant detection algorithms to detect the packet dropping attack; however, most of them are heavyweight for the resource-constrained IoT network. In this paper, we propose a lightweight evidence fusion-based detection algorithm, namely EFDA. It uses packet forwarding evidence to detect malicious nodes. In EFDA, the received packets are grouped and sorted to find the dropped packets. For each dropped packet, the base station traces its routing path, finds the suspicious nodes, and collects evidence. The collected evidences are fused to find the liar, and EFDA punishes the liar by decreasing its trust. Based on nodes’ trusts, the K-means clustering is used to cluster nodes and detect malicious nodes.

Our experimental results demonstrate that EFDA has better detection performance than two typical ML-based algorithms: HD and PDE. EFDA detects malicious nodes without injecting packets, and it can improve the detection accuracy by around 20% to 30%.

Data Availability

Some or all data, models, or codes generated or used during the study are available from the corresponding author by request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under no. U20B2050 Public Service Platform for Basic Software and Hardware Supply Chain Guarantee under no. TC210804A, Secure and Controllable System for Malicious Code Analysis under no. TC200H02X.

References