Malicious Node Detection using Fog Computing in Smart City

Abstract: In any Smart City Applications like Smart Parking, Smart Home Automation, Smart Medicine etc. if any legal node becomes malicious, then the transferred data can be tempered easily or it can provide the false data. As the malicious node is also the part of the system that is internal node, its behavior will be very difficult to detect if we only apply some cryptographic techniques and so, many serious accidents can be caused by this behavior. We need to research how we can detect the internal malicious node from the available unencrypted data through some lightweight methods. To detect the latency and malicious node on smart parking system, we have used here two approaches: i) Cloud Computing  ii) Fog Computing. It is observed that it is better to implement it through Fog Computing as it solves latency issues as well as it provides substantial enhancements such as network usage, on-demand scaling, security, and the resource mobility too. In our research, we utilized simulations in iFogSim to assess the efficacy of our proposed approach, focusing on latency and network usage. We recognize fog computing to integrate heterogeneous computing resources dispersed throughout edge networks, ultimately providing enhanced computing services to users. Hence, our study introduces a Fog Computing-based Malicious Node Detection Scheme tailored for Smart City Applications. Central to our approach is a reputation calculation mechanism employed by the fog server to evaluate suspicious nodes, leveraging the correlation between outlier detection of acquired data and node influence. Our experiments demonstrate that our proposed scheme effectively and efficiently detects malicious nodes, enabling the fog server to attain more reliable data.

Keywords: Smart City Applications, Fog Computing, Smart City, Smart Parking, Cloud Computing, Malicious Node Detection

I. INTRODUCTION:

The term "fog computing," coined by Cisco, refers to extending cloud computing to the network edge within an organization, often referred to as fogging or edge computing. This facilitates networking, storage, and computing services for end devices and computing data centers. Fog nodes comprise the fog infrastructure, situated between physical hosts and the cloud, providing data, applications, computation power, and storage primarily for the host's benefit. Processing occurs closer to the data source, enhancing speed and ensuring improved security and system efficiency.

"Faulty nodes" are those that operate irregularly due to malfunction or accident, while "malicious nodes" are compromised nodes, posing threats such as distributed denial of service (DDoS), fake data injection (FDI), and other attacks. If a suspicious node's trust value does not increase over time, it is classified as malicious, and the source node avoids using its route for packet transmission to ensure security.

Malicious actions of nodes, such as falsely reporting traffic accidents, can lead to serious consequences, affecting drivers and traffic safety. Detecting internal hostile nodes, especially if a legitimate node becomes malevolent and tampers with data, poses challenges for cryptographic techniques.

Fog servers, located at the network edge in fog computing, resemble lightweight cloud servers, providing computing, data storage, and communication services. They enable direct communication between fog devices and servers.

1 Akash K. Mehta
2 Dr. Minal P Patel

1 Ph.D. Scholar (CE-IT), Gujarat Technological University, Ahmedabad, India
ORCID ID: 0000-0000-9124-3435
Email ID: akash.mehta.it@gmail.com

2 Assistant Professor, Department of Computer Engineering, Devang Patel Institute of Advance Technology and Research (DEPSTAR), Charotar University of Science and Technology (CHARUSAT), Changa - 388421, Gujarat, India
ORCID ID : 0000-0002-2002-2907

Copyright © JES 2024 on-line : journal.esrgroups.org
II. BACKGROUND:

Cloud computing has several drawbacks, including increased reaction times and communication expenses [1]. The cloud and fog were compared by Aazam et al. [2]. Through an examination of various computing paradigms, it has been established that fog computing significantly mitigates processing latency. By offloading specific processing tasks and computation from the cloud server to fog node, we have concluded that fog computing emerges as an effective solution for addressing the demands of high-throughput and low-latency applications. [3]. The potential for fog computing to reduce cloud computing's latency and network utilization problems is enormous [4].

Numerous studies have suggested architectures based on fog in several fields to improve the effectiveness, security, and cost-effectiveness of their systems. Vilalta and associates [5] reported a fog computing based architecture that offers combined cloud computing and fog computing properties for the deployment of network features at the furthest point of a telecom operator's network, including virtualization, mobile edge computing, and IoT (Internet of Things) services. A fog architecture based on software-defined networking was presented by Bi et al. [6] to facilitate mobility. In addition, the authors developed an efficient signaling operation and suggested a route optimization technique to give mobile end users clear and flawless mobility support.

The ability of fog to lessen the frequency of data transfers to the cloud is one of its main advantages. Consequently, the applications connecting with the cloud demand less network consumption, which finally improves response time. Numerous studies have demonstrated that fog has lower latency than cloud. For instance, the authors of [7], [8], and [9] showed that, as compared to the cloud based employment of systems, the latency in fog is minimal. Furthermore, fog computing facilitates scalability and lowers network traffic, making it an excellent fit for IoT frameworks [10]. For real-time applications, network use is always a crucial factor that can be substantially reduced with fog computing [11].

According to Passas et al. [12], a multi radio access technology environment offers a way to service consumers without clogging up the Paris metro pricing scheme by allowing the same waggons to be maintained for varying ticket prices.

Milioti et al. [13] recommended a weighted uniformly fair bandwidth distribution algorithm aimed at managing the data volume distributed via WiFi & implementing a pricing centered rate to mitigate network congestion issues.

Fog nodes, as highlighted in the research, are characterized by high computational power and geographic dispersion. This architecture involves a diverse array of heterogeneous devices connected at the network's endpoint, facilitating computation and storage services within the distributed fog computing infrastructure [14]. Fog computing minimizes network bandwidth usage, thereby establishing a more adaptable architecture and enhancing data processing methodologies [15].

When a new car enters the parking area, the smart car parking system uses fog computing and image processing to offer information about the availability of parking spaces. The idea behind the suggested approach's use of fog computing is that, as the data on available parking spots must be updated on a regular basis, it is preferable to minimize latency and network bandwidth usage. Fog computing, which prioritizes carrying out computations close to network edges, can efficiently address the problems and prevent frequent data transfers to the cloud [16].

To ascertain whether a parking space is available nearby, we use an image processing technique in the suggested framework. The picture of the car is taken, and then it is examined to determine if it is (i) parked, (ii) not parked, or (iii) driving out of the parking lot. The drivers can have the indication for Empty slots for the parking at the entrance through the LED (Light Emitting Diode) fitted at the entrance. The slot's state is updated in the fog once the driver parks their car there. In our study, we employed iFogSim to conduct simulations and construct scenarios for both cloud based and fog based executions of the system, aiming to evaluate the effectiveness of the proposed smart parking method. The experimental results indicate that, compared to the cloud-based implementation, deploying the smart parking environment in a fog based architecture leads to reduced latency and lower bandwidth utilization.

III. LITERATURE REVIEW

Malicious node detection has been a significant focus in various types of networks, including mobile ad hoc networks (MANETs) and mobile wireless sensor networks (MWSNs), prompting numerous research efforts.
In MANETs, Wange et al. [19] introduced a detection method based on the trust utilizing concepts like trust fluctuation and evidence chain. Their approach assesses nodes' trust values over time, utilizing trust fluctuation to indicate significant volatility and evidence chain to identify malicious behavior.

Ebinger et al. [20] proposed a cooperative intrusion detection technique for MANETs based on reputation exchange and trust assessment. They categorize credit data into exchanged reputation and trust, combining them for intrusion detection purposes.

For MWSNs, various trust management algorithms have been proposed primarily for data integrity, network security, and secure routing. Shaikha et al. [24] suggested a trust management plan based on grouping focusing on directly observed QoS metrics. Althunibat et al. [25] presented an efficient technique capable of detecting malicious nodes regardless of their type and quantity, recognizing that both dependent and independent harmful nodes impact target detection in WSNs.

In Multi-Wavelength Wireless Sensor Networks (MWSNs), Abdelhakim et al. [26] proposed a robust approach for detecting malicious nodes amidst Byzantine attacks, emphasizing adaptive data fusion in dynamic threat environments. Their method utilizes a trust model based on entropy for analysis. Vempaty et al. [28] introduced 2 different strategies to counter Byzantine attacks, one involving Byzantine node identification assuming uniform local quantizers, and the other combining this identification method with dynamic non-uniform threshold quantizers across sensors.

Malicious Node Detection Scheme

In this work, we address the critical issue of detecting malicious nodes in dynamic networks, particularly focusing on the potential risk they pose to traffic safety. Our approach leverages fog computing to score suspicious nodes based on reputation calculations, considering the correlation between collected data and network structure. Additionally, we introduce a novel method for reputation computation, utilizing outlier detection of acquired data and node influence. Our experimental results demonstrate the effectiveness of our approach in identifying malicious car nodes, thereby enabling the fog server to obtain more accurate data.

Further contributions of our work include:

1. Security Issues and Problem Formulation: We examine the security challenges associated with obtaining unencrypted data and propose a problem formulation to address these challenges. Our approach utilizes reputation calculation to score questionable or you can say suspect nodes, facilitating malicious nodes detection based on the relationship between collected data and network structure.
2. Fog Computing-based Data Gathering System: We propose a novel data gathering system for dynamic networks on the basis of fog computing architecture. Each node selects a neighboring node within its transmission scope as the next hop to transmit the data, eventually reaching the cluster head, which forwards the data to the fog server along with associated network topology.
3. Malicious Node Detection Method: Our method for detecting malicious nodes involves three key steps: determining node influence in dynamic network structures, detecting anomalies in data using outlier detection techniques, and establishing the relationship between outlier detection and node influence. We then compute the reputation score for each suspect node to identify rogue nodes effectively.
4. Experimental Evaluation: We conduct experiments to evaluate the effectiveness of our proposed approach. This includes assessing the computation time for various tasks, examining the relationship between node reputation, node influence, and anomalous data, and analyzing the effectiveness of identifying malicious and suspicious nodes under different scenarios.

Overall, our work contributes to enhancing the security and reliability of dynamic networks, particularly in scenarios where traffic safety is a concern. By leveraging fog computing and innovative reputation computation techniques, we provide a robust framework for detecting and mitigating malicious node behavior, thereby improving the overall performance and reliability of networked systems.

Proposed System Architecture for Cloud Vs. Fog
Figure 1: Fog based Smart Parking system set with the Sensors and Cameras – An Architecture

The suggested architecture in Figure 1, comprises three layers. The first layer contains cameras located above parking slots and they will capture the images of the outer space and will determine the exact occupancy for the parking space. A microcontroller device connects fog node with the cameras and so second layer of the architecture will be formed. The fog is connected to the cloud, which constitutes the third layer and manages and stores the picture data for extended periods.

As drivers approach the parking area, they will be able to see the parking slot status displayed on an LED screen, eliminating the need for smartphone verification. This approach directs drivers to designated spots at the entrance gate, reducing traffic congestion and fuel consumption.

Multiple cameras are installed to cover each parking lane, with the microcontroller acting as an intermediary between the cameras and fog. To share the Parking occupancy or where the Parking space available, Fog nodes communicate the details with the cloud and also takes help of the neighboring nodes. All the details like whether the parking spaces are available or not available, fog node will retrieve the information from the adjacent spacing and it will be displayed on to the LEDs.

While cloud computing can control the data for long duration of time, frequent connections for data transmission and access result in significant network bandwidth consumption and latency. To overcome this issue, Latency will be reduced through middle layer of fog node by minimizing the access to the cloud. Image storage and Image Processing functions will be performed at fog node with 2-way communication establishment between the cloud and fog node. For the long term storage, Pictorial data is sent periodically to the cloud.

Analysis of Latency

For real-time performance requirements, minimizing latency is crucial. Fog computing minimizes latency by conducting computations at the network's edge and providing rapid responses to client devices. Fog nodes receive images for processing, with each node assigned to a specific location, ensuring sufficient processing capacity to analyze photos and update LED information promptly. Equation 1, derived from [17], is utilized to calculate latency, taking into account the architectural characteristics and operational parameters of the fog-enabled smart parking system.
Latency = \( \alpha + \mu + \phi \)

where \( \mu \) is the amount of time needed to upload images to the fog node for processing and storage, and \( \alpha \) is the Tuple CPU execution delay for taking photographs. Finally, \( \phi \) is the amount of time that the data takes to appear on Light Emitting Diode (LED) once the completions of processing at the fog node.

**Network Usage Analysis**

Whenever the traffic on cloud server increases, it will lead to the higher network usage as that time only cloud resources will be utilized. As network usage rises due to increased load, the data rates of the network may drop. However, by dispersing servers geographically, a single fog node can be assigned to handle requests from a specific region. This results in increased transmission rates for the remaining traffic while reducing overall network consumption.

The formula for calculating network utilization, denoted as Eq. 2 and sourced from [17], is:

\[
\text{Network usage} = \text{Latency} \times \partial
\]

Where \( \partial \) represents the tupleNWSize parameter.

The efficacy of our proposed smart car parking design based on fog is depicted in Figure 1, utilizing experimental findings to illustrate its performance benefits.

We simulated the circumstances where Smart Cameras of High-Definition capture the images of Parking slot and then they are transmitted to the fog node as per our investigational setup. To conclude the status of Parking Space/Slots, Image analysis process will be done by the fog node and the analysis will get displayed on SMART LED which is connected through Wi-Fi with the Fog node. A proxy server is used to connect the fog nodes with Central Cloud device. Each and every location will be having one allotted fog node for the work accuracy.

The simulation involved creating variables for parking lots and cameras, with four parking lots and four cameras initially assigned to each parking spot. These cameras, treated as sensors according to [17], are smart and Wi-Fi enabled, connected via microcontrollers.

We expanded the number of cameras so that we can evaluate the different scenarios and so we can find the impact with the increasing number of cameras on Network Usage and Latency in a fog node.

The iFogSim topology included four fog nodes, each with four cameras, to evaluate network utilization and latency. Picture capturing modules were installed in the cameras to capture images, while slot detection modules were installed in fog nodes to process images and identify available parking spots. The status of Parking Slots/Spaces will get updated through the fog nodes and it will get displayed on the associated SMART LED.

We observed that a latency and network consumption fog node increase with the number of cameras and LEDs managed by it. Processing activities on fog nodes significantly reduce the computational demand on the cloud. However, the Latency and Network usage increases if we directly connect the cameras via a router to the cloud server and so high network traffic consumption will also be there.

This highlights the effectiveness of fog computing in reducing latency and network usage by performing processing tasks at the network edge.

**Proposed Framework for Fog based Malicious node Detection Scheme:**

We introduce a specialized data acquisition framework designed for dynamic networks, utilizing principles of fog computing. In this framework, each node selects a neighboring node within its transmission scope as the next hop to transmit data until it moves and reaches the cluster head. The cluster head then forwards the data to the fog server through the associated base station, such as a Road Side Unit (RSU).
Figure 2. Proposed Framework for Fog based Malicious node Detection Scheme

In the networks based on fog computing, certain fog servers are strategically positioned in close proximity to specific networks, responsible for acquiring, storing, and analyzing data from these networks. Each fog server can communicate with multiple base stations. However, the varying management ranges of fog servers pose a challenge, as mobile nodes moving beyond a fog server's range necessitate relaying relevant information to the next fog server.

Nodes with significant influence in a network topology often yield larger deviations in data results, making vehicle nodes suspicious. To address this, a reputation mechanism is developed to assess the credibility of each suspicious node.

Node influence calculation

Node influence calculation primarily utilizes the degree centrality method, assessing it through shortest paths in the network topology. Given that our proposed acquisition framework routes all data to the cluster head node, our focus lies on evaluating node influence concerning data attainment between the cluster head and all the nodes.

Malicious Node Detection Methods

We have considered 4 methods to track Malicious Node Detection.

1. K-Means Clustering

K-Means Clustering is an unsupervised learning method utilized for clustering tasks, organizing unlabeled datasets into distinct clusters. The parameter 'K' determines the number of clusters to be generated; for instance, if K=2, there will be two clusters, and for K=3, there will be three clusters, and so forth. This iterative algorithm divides the dataset into K clusters in a manner where each data point belongs to only one group with similar characteristics. [29]

Steps of K-Means Algorithm:

1. Select the number K to determine the clusters.
2. Choose random K points or centroids (they can be different from the input dataset).
3. Assign each data point to its nearest centroid, forming K predefined clusters.
4. Calculate the variance and position a new centroid for each cluster.
5. Repeat the third step, reassigning each data point to the new nearest centroid of each cluster.
6. If any reassignment occurs, return to step 4; otherwise, proceed to FINISH.
7. The model is now ready.

2. **Hierarchical Clustering:**

Hierarchical clustering is another unsupervised algorithm used to cluster unlabeled datasets, constructing a cluster hierarchy in the form of a tree. Although the results of K-means clustering and hierarchical clustering may appear similar, they differ in their approach. Hierarchical clustering does not require the predetermined number of clusters as in K-Means; instead, it autonomously builds clusters. [30]

**Steps of Hierarchical Clustering:**

1. Initially, each data point is considered as a single cluster, resulting in N clusters for N data points.
2. Identify the two closest data points or clusters and merge them, reducing the number of clusters to N-1.
3. Repeat the merging process until only one cluster remains.
4. Once all clusters are combined into one, develop a dendrogram to divide the clusters based on the problem at hand.

3. **Local Outlier Probabilities**

Local Outlier Probabilities (LoOP) is a method for detecting outliers based on local density estimation, providing scores ranging from 0 to 1, representing the likelihood of a sample being an outlier. While LoOP may not detect complex anomalies occurring within the data distribution, it excels at identifying outliers lying far from the typical multivariate distribution. Its output scores are easily interpretable, making it suitable for various applications. [31]

**Steps for Local Outlier Probabilities:**

```python
import numpy as np
from sklearn.neighbors import LocalOutlierFactor

# Data can be Generated Randomly
X = np.random.randn(100, 10)

# LocalOutlierFactor estimator can be produced and requires it to be fit to the data
estimator = LocalOutlierFactor()
estimator.fit(X)
```

4. **Long Short-Term Memory (LSTM) Optimization**

Long Short-Term Memory (LSTM) Networks are a type of deep learning sequential neural network designed to address the vanishing gradient problem encountered by traditional Recurrent Neural Networks (RNNs). LSTMs enable the retention of information over long sequences, making them suitable for tasks involving temporal dependencies. Implemented using frameworks like Keras in Python, LSTMs excel at capturing long-term dependencies, such as remembering previous scenes while watching a video or recalling past events while reading a book. They are specifically designed to overcome the limitations of RNNs in handling long-term dependencies. [32]

IV. EXPERIMENTS AND RESULTS

**Experimental Outcomes for Cloud Vs. Fog:**

A comparative analysis between the cloud-based implementation and the proposed fog based scheme concerning network usage and latency will be provided in this section.

We can conclude through the experiments that there will be a lower network usage and lower latency if we use fog based implementation as compared to the cloud based approach. Table 1 represents all the results of the cloud-based
implementation and fog-based implementation alongside the measurements of latency and network usage in the fog environment.

Table 1. Simulation results with different number of Cameras for the Smart Car parking Architecture for Cloud and Fog

<table>
<thead>
<tr>
<th>Cameras</th>
<th>Fog Latency</th>
<th>Cloud Latency</th>
<th>Network Usage in Fog (kB)</th>
<th>Network Usage in Cloud (kB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>7.66</td>
<td>8.23</td>
<td>3210.3</td>
<td>27132</td>
</tr>
<tr>
<td>20</td>
<td>8.11</td>
<td>9.32</td>
<td>3901.2</td>
<td>35232</td>
</tr>
<tr>
<td>24</td>
<td>8.51</td>
<td>10.64</td>
<td>4703</td>
<td>44320.3</td>
</tr>
<tr>
<td>28</td>
<td>7.53</td>
<td>12.1</td>
<td>5502.1</td>
<td>54602.1</td>
</tr>
<tr>
<td>32</td>
<td>8.5</td>
<td>280.2</td>
<td>6202</td>
<td>61407</td>
</tr>
<tr>
<td>40</td>
<td>9.32</td>
<td>1885.5</td>
<td>7807.1</td>
<td>63101</td>
</tr>
<tr>
<td>44</td>
<td>10.11</td>
<td>2398</td>
<td>8695.4</td>
<td>64207</td>
</tr>
<tr>
<td>48</td>
<td>10.32</td>
<td>2801</td>
<td>9203.1</td>
<td>64787.3</td>
</tr>
</tbody>
</table>

![Figure 3 Latency Comparison of the Cloud and Fog](image-url)

![Figure 4 Network Usage Comparison of the Cloud and Fog](image-url)
To compute the results, we utilized iFogSim to run the scenarios outlined in Table 1, with a primary focus on measuring latency and network utilization. The latency comparison between the cloud and fog environments is depicted in Figure 3. The results indicate that as the number of cameras increases, the latency in the cloud environment experiences a substantial increase compared to the fog environment. This disparity can be attributed to the fact that in the fog environment, as the number of cameras grows, each fog node assigned to a specific area solely processes images for that area. Conversely, in cloud computing, as more cameras are added, latency increases significantly since the cloud server processes all parking lot photos.

Figure 4 illustrates the network consumption results for the fog scenario. It demonstrates how network usage escalates with the number of fog nodes and cameras. In a setup where each camera is linked to a cloud server, each tuple is processed sequentially by a single cloud server, resulting in increased network consumption in the cloud environment. Conversely, in a fog-based architecture, multiple cameras are connected to several fog nodes, with each fog node assigned to a specific parking space. Consequently, one fog node exclusively processes tuples from cameras within its assigned area, leading to more efficient network usage.

Implementing a fog-based architecture for smart parking can reduce the time, fuel, and carbon dioxide emissions associated with searching for parking spaces, while also providing timely information on available spots in specific parking areas. Additionally, these findings underscore the potential of fog computing in Internet of Things scenarios where rapid response times are crucial. In summary, the architecture that is based on fog emerges as more realistic for the real time applications due to its low latency and network utilization.

**Experimental Results using Malicious Node Detection Scheme:**

In all the experiments of Malicious Node Detection, we have used the sample Dataset.

**Pseudo Code:**

1. **K-Means Clustering**

   ```python
   def Kmeans_clustering(number_of_nodes, positions, number_clusters):
       km = KMeans(n_clusters=number_clusters)
       km.fit(positions)
       labels = km.labels_
       clustering = {k: [] for k in range(number_clusters)}
       print("labels are: ", labels)
       for i in range(0,len(labels)):
           clustering[labels[i]].append(i)
       return clustering
   ```

2. **Hierarchical Clustering**

   ```python
   def hierarchical_clustering(number_of_nodes, positions, number_clusters):
       cluster = AgglomerativeClustering(n_clusters=number_clusters, affinity='euclidean', linkage='ward')
       cluster.fit_predict(positions)
       labels = cluster.labels_
       clustering = {k: [] for k in range(number_clusters)}
       for i in range(0,len(labels)):
           clustering[labels[i]].append(i)
       return clustering
   ```
Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic algorithm inspired by the foraging behavior of ants. It aims to find optimal solutions to various optimization problems, such as the Travelling Salesman Problem (TSP), by mimicking the collective behavior of ants in finding the shortest path between their nest and food sources. [33]

Procedure for Ant Colony Optimization (ACO):

1. Initialize the necessary parameters and pheromone concentration.
2. While termination condition is not met:
   - Generate an initial population of ants.
   - Calculate the fitness values for each ant in the colony based on the problem objective.
   - Utilize selection methods to find an optimal solution.
   - Update the pheromone concentration on paths based on the solutions found by the ants.
3. End while loop.

ACO iteratively explores the solution space by simulating ants' movement and pheromone deposition, gradually converging towards an optimal solution over multiple iterations.

Pseudo Code:

```python
import math
import random
import turtle

s = turtle.getscreen()
tag = turtle.Turtle()
tag.shape("circle")
tag.shapesize(0.9,0.9,0)
tagSpeed = "normal"
totalPopulation = 0 # Later set to number of ants times number of rounds
def drawCities(cityMatrix): # Draw all the city as circles which include index number
    style = ("Courier", 30)
tag.pensize(4)
```

Figure 5. Experimental Results
for i in range(len(cityMatrix)):
    ant.up()
    ant.goto(cityMatrix[i][0],cityMatrix[i][1])
    ant.down()
    ant.circle(30)
    ant.write(i,font=style)

**Figure 6: Ant Colony Optimization Results detect the Malicious Node**

V. CONCLUSION:

In our proposed work, First we implemented both the Fog Based and Cloud based system to know which one will provide the better results whenever Network Usage and Latency components are there. As we got the better results for Fog based system, we introduced a novel fog computing-based malicious node detection scheme, utilizing reputation calculation to evaluate suspicious nodes' behavior based on data correlation and network topology. Our approach includes calculating node influence within the network topology and employing outlier detection to identify unusual continuous data. By correlating outlier detection results with node influence, we propose a reputation mechanism to score questionable/Suspect nodes. Experimental results exhibit the efficiency of our scheme in detecting the system nodes which are malicious, enhancing the fog server's data reliability. However, our scheme has limitations. It needs real time data and a small historic data set, challenging rapid malicious node detection. Additionally, simultaneous data and dynamic network topology gathering by the fog server incur additional costs on nodes for data and network position transmission. Addressing these drawbacks requires further research in future work.

REFERENCES


