Machines and Algorithms
http://www.knovell.org/mna

Research Article

Software-Defined Network based Fog Computing for IoT Networks

Muhammad Tehseen Irshad[1, *]

[1]Department of Computer Science, Bahauddin Zakariya University, Multan, 60000, Pakistan
[*]Corresponding Author: Muhammad Tahseen Irshad. Email: tehseenirshad7370@gmail.com
Received: 25 October 2023; Revised: 1 November, 2023; Accepted: 28 December 2023; Published: 14 March 2024
AID: 003-01-000031

Abstract: The Internet of Things (IoT) connects smart gadgets. The IoT solution streamlines data collecting and processing. High-quality end-user services are making IoT systems appealing. Users can't get high-quality cloud services quickly. Fog computing computes quickly and provides excellent services. Fog computing is a novel processing layer between the cloud and consumer layers in the standard cloud computing concept. The fog layer uses distributed computing with tiny smart devices and access points. Use in diverse applications raises numerous major challenges. Challenges include network security, capacity, scalability, and latency. Security is a major concern for IoT applications. This paper introduces a novel Internet of Things architecture that blends software-defined networking with fog computing. We suggested access control management and trust evaluation algorithms. Our methodology allows fog computing systems to dynamically add fog nodes. Newly connected nodes get non-sensitive jobs. IoT devices and fog nodes may interact, exchange services, and report fog node activity to the Fog Manager node. The FMN measures fog node weight confidence based on behavior. This assessment checks for harmful devices that might compromise system or protection quality. Due to such diagnostics, the fog system filters away unreliable nodes and overweights confident nodes. We tested our technique in iFogSim using Java. The simulation results show that our system can recognize and eliminate harmful attacks/interactions among fog nodes in the fog environment.

Keywords: IoT; Fog Computing; SDN; Cloud; Access control; Weighted Trust Management Security; Dynamic behaviors;

1. Introduction

Smart devices in a networked environment form the Internet of Things (IoT) architecture. By 2025, more than 75 billion IoT devices will be operational, according to CISCO [1], [2]. Our application is more flexible with cloud computing since you can add and remove processing nodes during runtime. Cloud computing has been successful in many contexts. This is unsuitable for receiving reliable, low-latency inputs in smart grids, industrial automation, and ITS [3]. The Internet of Things has become an essential element of our daily lives and garnered attention in recent years. IoT is a network of smart systems, infrastructure, objects, and sensors [4]. Global networking of everyday products (refrigerators, fans) and smart city apps drive this growth. Wireless communications and electronics are also driving smart web-connected object adoption [5]. Creating scalable programs that serve many users at once is driving rapid innovation and cloud computing. Cloud computing allows service providers to run projects with little infrastructure without big proprietary data warehouses [3]. Not all IoT systems are perfect.

Edge computing—real-time device processing—is one of the hardest tasks. Tasks that need extra processing or storage are often moved to the cloud. Service delivery may be delayed. The device-cloud connection is difficult since there is no framework for data sharing and verification [2]. Fog computing is ideal for IoT networks and applications. Cisco launched fog computing as an extension of cellular edge computing. Several research examine IoT Fog Computing [6]. Fog computing is safer than cloud computing since data is temporarily stored and analyzed in local fog nodes near the data source. Instead of connecting to a central cloud, fog computing installs several cloud computing resources at the network's edge. End-users gain from fog computing layer latency reduction, QoS improvement, and entertainment [7]. End-user closeness to the cloud, geographic reach, and mobility support distinguish fog computing from cloud computing. By adding a layer of geographically dispersed fog nodes in the middle, fog improves cloud services [1].

Scalability, real-time data transport, and mobility are not supported by traditional network architectures. Most researchers use fog computing-based SDNs for real-time data transmission. Each network device's traditional communication method combines the control plane and data plane, whereas the SDN separates them [1]. SDN and network function virtualization have transformed network management. A logically central network control layer in SDN enables complex resource optimization algorithms [5]. These attempts rely on a central SDN-managed plane to handle fog computing infrastructure islands. This could severely damage dependability and performance due to increased traffic and failure. Create touchpoints. Dispersed control planes are closer to fog islands, making them more vulnerable to network events. Several researchers employ a distributed SDN control plane [8].

IoT networks struggle with reliability, access control, authentication, bandwidth, and latency. Additionally, IoT systems use device-to-device communication. Users benefit from this great functionality, but communicating data with other IoT devices puts their security and privacy at risk. Additionally, malevolent IoT devices may provide false data or use it for their own gain. These issues render IoT systems ineffective and potentially harmful [2], [7], [9]. The nearest IoT gateway connects several competing nodes. Portals connect to different countries via the Internet. Due of the need to disconnect and rejoin with the gateway, IoT nodes must be flexible [10]. Fog nodes are evaluated based on their network activity when they collaborate and exchange data to perform tasks. The malevolent fog nodes are removed from the network, and researchers struggle to keep them from rejoining.

Most researchers examined direct and indirect trust. Direct trust fog nodes evaluate fog node trust based on their own experience, while indirect trust is based on prior behavior. None of the models characterize the trust evaluator's honesty. Our model describes the trust evaluator's honesty and provides a novel framework for access restriction and dynamic weighted trust management. A centralized architecture for access control and trust management is proposed. The Fog Manager Node (FMN) controls access to newly connected fog nodes. The FMN gives new devices non-essential tasks. You can assign tasks to new shared devices using the FMN. Suggestions include calculating the reliability of newly connected devices. SDN controllers govern network infrastructure. It alerts all FMN about the malicious fog node so they can't allow it into their network.

The remainder of the study is laid out as follows. The literature review is described in Section II. The proposed framework is presented in Section III. Section IV contains the findings and discussion of the implemented model, and Section V contains the concluding remarks

2. Literature Review

Dynamically integrating malfunctioning fog nodes into trusted model and role-based access control computer systems was described in [3]. In appearance, the new dynamic nodes perform mathematical functions. Static, processing, and dynamic nodes in the system are faulty. Fog Nodes Manager handled all extra nodes. FNM looks for additional issues and assigns tasks based on trust. Several moving nodes will enter and leave the fog system. The fog manager recently sorted jobs into linked nodes and calculated confidence using confidence equations. Character classification using computer parameters determined accuracy. These include availability, data dependability, and inverse efficiency. Once any variable node

gains trust, the Fog Network extends its duty to the insured Fog nodes. System C builds the Fog computer platform. They test using Model A SystemC, which represents each processing component and may communicate with other operating systems over the SystemC channel. Each processor element operates at 500MHz–4GHz. The system registers applications based on random responsibility distribution over time. A suitable technique for Internet of Things networks is presented in [2]. Fog estimates underpin the proposed trust and dependability paradigm. The authors examine the framework and its integration into Fog Computing IoT. They use [9], Designed for the new frame. Trust and reputation addresses the previous framework's weaknesses. Each IoT device utilizes error codes to evaluate all IoT devices' dependability and connects to one that exceeds a certain level. This check ensures that no harmful apps are present that might harm the system or dissatisfy customers. The notion is defended against unfavorable exposure, on-off, and self-promotion. Several IoT devices are simulated in the testbed. The collection is on Epinions.com. The 500-testbed concept was modified to utilize IoT apps for various dangers. A Bad-Mouthing assault was repeated for hundreds of rounds from round 20 to 100. Bad-Mouthing attackers start at 10% and rise to 60%. the On-Off assault experiment with 30% and 60% attackers. When hostile gadgets attack at round 20 and persist until round 40, confidence drops. The data shows 60% even with various attacker counts. This dangerous gadget seeks to recover trust between 15 and 35. It began devious behavior in round 40 but was quickly defeated. Rebuilt its trust score between 85 and 100, and removed the device; joined at round 175.

Users can evaluate service providers and choose who to contact in mobile agent systems using an adopted trust and reputation model [9]. The review considers investigators' and witnesses' backgrounds. The credibility of witnesses was also checked to avoid dishonest reporting. Multiple tests are combined using novel, adjustable, and changeable weights depending on contact frequency, size, and witness accuracy. Discounts and punishments encourage witnesses to give accurate information. Another unique feature of the suggested security system is supplementary options for disrespectful witnesses to improve their reputations. The framework might manage detached agent behavior using this way. A testbed simulation assessed the trust and reputation paradigm. The matter is being examined by six auditors and 25 service providers. The user compared five service providers in this scenario. They presume the system user must connect with witnesses with a 70% trust rating (threshold conf = 0:7). If a service provider's Trust value exceeds 60% (threshold trust= 1:2), they are trustworthy. Threshold = 0:4 for average simulated application interaction. The exam is repeated 50 times.

According to [10], modern computing and the scale and variety of IoT devices should include distributed trust management. Trust between IoT entities was established via a multi-layered architecture. Identity, gateway, IoT, node, and server frameworks existed in the cloud. Their architecture considers all of these to provide reliable end-to-end IoT data flow. Identity creation is the first of four communication flows between endpoints, gateways, and identity providers. (2) Identifying the endpoint with the identity server to start the procedure. (3) Gates determine endpoints. Second gateway starts terminal device authentication.

Using a Hidden Markov Model (HMM), the authors developed a statistical, safe, and scalable rogue fog node detection method [11]. A trained HMM might spot dangerous fog nodes. Calculations are fast and accurate. Three steps comprise HMM-based detection. 1) instruction, 2) observation, 3) detection. A rogue Fog node in the network might compromise user data. Thus, connecting to a rogue Fog node was hazardous because attackers may steal sensitive user data. The problem may be pervasive if a larger network collaborated. To avoid malicious fog nodes, they recommend addressing security and privacy at every level of fog computing system architecture. Fog network systems must be secured and malicious fog nodes detected. A device that acts abnormally after being hacked by hostile users or hackers is called. MATLAB R2016a and Eclipse IDE were tried in Java to finish these operations. Badmouthing, self-promotion, on-off attack, and ballot stuffing assault results were recorded in a data file and loaded into MATLAB R2016a for Markov model predictions. Two devices were created to test the system model with different demands. The first fog device averages 0.75 attack chances, whereas the second averages 0.67. Individual device requests determine the aggregate attack probabilities, which are represented as Legitimate nodes (Chance 25%, 50%). 75% wanted 250, 500, and 750). An assault on nodes (37.5%, 25%, and 12.5% requests on 375, 250, and 125). incorrect node (37.5%, 25%, and 12.5% requests on 375, 250, and 125).

The authors examined the functional and non-functional aspects of an Intelligent Transportation Systems (ITS) model and its authorization issues in [12]. Privacy, interoperability, context awareness, resource restrictions, network coverage, selection delay, monitoring, and responsibility in fog computation are addressed. Their ITS architecture splits the ICT environment into four parts: cloud infrastructure (CI), roadside, cars and people, sensors, and enablers. Transportation infrastructure relies on mission-critical security and fog computing. Expect IoT-enabled devices on SA roads temporarily or permanently. Fixed nodes communicated with carriers using short-range radio communication [13]. Sensors like roadside actuators and smart traffic lights are mounted to the automobile. Cloud services were often used in CI to provide alternative routes during traffic bottlenecks. The affected location should have sensors deployed in traffic data systems to monitor traffic conditions. When using the service, the automobile must additionally provide its location and destination. Data will be examined and compared to other vehicle data via the cloud service. Fog computing might boost cloud capabilities and overcome these limits in RS and VH. Cloud services with fog computing might employ RS and VH field features to minimize latency, localize and decentralize apps, and link applications and users (vehicles). It was important to remember that automobiles and people may react unexpectedly. They outline the main traits of a good access control system. 1) Attribute-Based Access Control. In further research, they examined successful methods for establishing reference monitors, arguing for regulations, and supporting offline operations.

In [14], the authors advise assessing the trust value and rating of each fog IoT and fog node/device based on their interactions to ensure routing and handoff. A trust manager between the fog and IoT layers tracks all fog nodes in its lookup table and identifies malicious fog and IoT nodes. Fog nodes also define the IoT layer's capacity and provide services via the most dependable pathways. They examined one-way links to understand the paradigm. In the first SITO approach, nodes generate trust levels between 0 and 1 and assign them to neighbors based on recent interactions. All FN ratings and TV/TF data will be added to the TM lookup table. The Tidal Trust method starts with a randomly picked FN and estimates its nearest nodes' TVs at various stages of the dialog. The tidal confidence technique calculates step confidence scores. In general, the tidal Trust technique calculates trust and ratings depending on each BS level in two steps. First layer fog node was used to compute i+1 layer. Fog nodes determine end-user dependability and then pick the most reliable intermediary nodes to transmit services in the second phase. The testbed held this experiment. Microsoft Azure Cloud DS2 provisioned three virtual machines. Each fog environment started with 2,000 nodes and added 50 every minute to evaluate the architecture's scalability. The three NS2 configurations use the suggested method to increase fog node reliability. The recommended approach allows fog and IoT node combinations and changes.

The authors propose in [15] that fog node authentication verifies data owners and requesters without third parties. Create a fog node-smart contract system to validate user access to IoT devices using smart contract tokens. The procedure was also used to study backups. The model works properly when the smart contract is deployed, boosting security and search result trust. This scenario managed IoT devices, fog nodes, and end users using Ethereum smart contracts. Five primary pieces of the system model access Ethereum smart contracts online. Each participant utilizes cloud and fog nodes to engage with smart contracts using the Ethereum client and has an Ethereum address (EA). Decentralized cloud storage solutions should use this authentication method. The suggested approach comprises five phases: device registration, mapping, authentication, token production, and data sharing. Admin, Smart contract, End-user, Fog nodes, and IoT devices comprise the decentralized storage system. The organization handles donor accounting and exchange. A proposed authentication technique was supplied for Python testing. How expanding the group tail from 50 to 200 bits affects shader node authentication. The tale is valid for 4,009,083 minutes, ranging from 76,915 seconds to 76 times, but the validation approach is superior.

For fog computing, the author introduces access control Ciphertext-policy attribute-based encryption (CP-ABE) with outsourcing and attribute modification [16]. Determining the encrypted data's owner and decryption user is independent of the key's access structure and other factors. Updates to attributes are cheap since they just update the variable value's ciphertext. The proposed system has five agencies. Cloud service providers, fog nodes, data owners, end-users, and key authorities are examples. A effective fuzzy computing

access approach involved outsourcing and theme adjustments in five stages. to test java, run the system, produce keys, encrypt and decrypt data, and alter attributes The CPAB Toolkit and Java Pairing Cryptographic Library implemented these laws. Java on Android phones with quad-core CPUs and 3GB RAM encrypts and decrypts data owners and users. Their key generation computational cost was half that of other models. Consumers found it handy that the data owner generated and sent the ciphertext to the fog node in 0.615 seconds and the user decrypted it in 0.459 seconds. Smart low-resource devices.

The persistent memory leakage model (CML) was one of the most powerful models for allowing continuing loss of user and master secret keys [17]. Their fog computing technology protects against third-party dangers, allows privacy, and controls access. The two settings have distinct threats, thus using the same technology might generate issues. Fog nodes are close to users and widely utilized in public spaces, making physical attacks (side-channel attacks) conceivable. With a smart gateway, the attacker may see device power usage and operating time. Therefore, typical functional encryption may not provide enough security in a foggy computing environment. Basically, fog computing access control is best solved using functional encryption. Due to new dangers, the previous definition may not be enough. Their technique converts LR-FE modeling and design to conditional coding. It has two designs: double encryption with paired encryption leak prevention. From a sealed pair encoder, make a sealed FE encoder. They offer equations and assertions to substantiate their stance.

A security architecture based on IoT and fog collaboration is suggested in [18]. Secure cooperation across resources and operational components is achieved by combining efficient access control with monitoring. Their security approach includes a thorough resource scheduling and allocation method to maximize system performance. Fog computing systems are divided into cells with various fog nodes in their study. The FNM FMN leads the cluster and manages each unit. Different service classes are found in fog nodes[19]. As long as the service was available, IoT users received it. Its major function was to handle new IoT users. Their second FNM assignment classifies resources. Provide an algorithm that boosts credibility and rating for high- or medium-access devices. Devices with lower service class will have mediocre access. They developed his network architecture (iFogSim) and a Java application to test his concept. The fog node service and the proposed trust access control and fog computation algorithm (TACRM) have the most resource management mechanisms and the greatest response, whereas the cloud server service takes longer and has a range of at least 90ms. Group Task Scheduler (GTS) 12.46 ms, 5.0328ms interval.

SDN-based public infrastructure (PKI) trust management solutions are sluggish to adopt in the cloud. TRUFL, a distributed trust mechanism, was devised to build and validate SDN trust [20]. Distributed trust management is consistent, resulting in faster transfers than centralized trust management. Unlike previous studies, the TRUFL system scales effectively as OpenFlow rules increase. Their OpenStack control node had an SDN controller. The Open Day Light Controller, a management vessel, has various uses, including a flow rule conflict checker and one or more Certificate Authorities (CAs) to verify node computer science dependability. A network-based intrusion detection system (NIDS) mirrors data plane traffic between numerous data plane virtual machines via port mirroring. NIDS uses a Neutron API (OpenStack Network Manager). TRUFL transfer verification time has grown from 7 to 28 milliseconds, while OpenFlow rules have increased from 10,000 to 50,000. SDPA had 130-145ms latency for 50k rules, while Net-Syn had 65ms. Their testing findings made SDPA delay data difficult to determine. Hassle delays 50k regulations by 6 seconds. The DPDK's distributed trust management reduces packet processing and authentication to milliseconds, making TRUFL's authentication quicker.

In [21], the authors present a model for assessing node trustworthiness that accounts for detrimental node behaviour. The suggested trust and reputation model addresses the security issues with delegation mechanisms in distributed systems. A distributed architecture with N nodes was investigated. These N home gateways (HG) can connect to the central server via one or more hops. Nodes were rated dependable or unreliable based on their past activity. Use selfish and accusing nodes to test his trust modeling. Trust-based trust management systems struggled with trust measurements and dependability data. The WSN agent-based trust and credit management (ATRM) strategy is explained. Our reputation is the backbone of

their trust model. In this example, nodes support each other make honest evaluations. Two layers make up this model. A trust model protects nodes against selfishness on the first layer. This implies that selfish nodes will be identified, penalized, and maybe banned from participating. To interrupt network activity, hostile nodes falsely accuse other nodes of being untrustworthy. The Trust Modeling layer defends against this. Four nodes (3, 9, 5, and 8) wish to send packets. Checks confidence matrix before transmitting. This 8 knots selfishly. They also employ 4 load nodes (2, 6, 12, and 14). The research shows that nodes 2, 6, 12, and 14 have low confidence levels and the number 2 is wrong.

The LoRaWAN method for connecting end devices to the network server is analyzed for serious security flaws in [22]. They uncovered protocol flaws, including the use of random integers in join packets to thwart replay attacks. They transmit two communications between the end device and the web server to join and accept the application. Terminals send network server connection requests. The web server will permit the receiver if the end device is authorized to access the network. No answer will be delivered to the last device if the membership request is denied. They then create different quantities of concealed visible data, which the attacker must retrieve using Entropy, the average voltage. This technique solves several security difficulties. Start by creating a single DevNonce for the device's random generators, which assess this event's likelihood and the network server's response. Second, DevNonce struggles to provide real randomness, especially for low-cost devices. SX1272 transceivers add the least significant bit of the received signal strength (RSSI) signal to form the random bit pattern. They study key generator quality factors and specific attacks that might reduce number unpredictability for long-term number production. This WiMOD SK-iM880A test shows how the DevNonce generator works without attack and with jammers. They discovered $\rho = 320.6$ without interference, identical to the comparative article, proving attacker-free random number generators' applicability. The first experiment, in which the jammer transmitted random signals, yielded findings identical to the 1m and no jammers. The second experiment's minimal entropy, which is lower than the others, is most essential. The minimal entropy of 12.66 means the maximum value is created every 6451.6 operations.

In [23], the authors provide a cloud resource provider's credentials and capabilities-based trust model. They demonstrate how to construct SLAs that integrate customer experience with cloud service provider capabilities. They expand the trust model and provide a cloud resource trust value equation based on QoS needs including dependability, availability, processing time, and data integrity. They also explain how to pick reliable and useful cloud resources. Architecture revolves around the system manager. It cooperates with the system. SLA managers negotiate and compromise on user QoS demands. The provisioning service component links system administrators with middleware agents. Provisioning solutions virtualize cloud customers' work environments and separate IaaS, PaaS, and SaaS systems from the cloud, affecting SLAs. The three main services were monitoring, metering, and billing. Control subsystems distributed and used resources. The middleware agent manages VM creation, modification, sharing, administration, and network deployment. Trustworthy repositories may be trusted. Adds a trust manager that stores and processes trust values from the trust store. A trust store contains cloud resource trust values. They also control trust management algorithms. They simulated their infrastructure with CloudSim. Revenue efficiency, dependability, availability, and data integrity were examined separately for each QOS configuration. They compared their model to two others using 5000 tasks per feature. QoS trust outperformed others.

They provide a detailed overview of fog computing data access control [24]. They evaluated fog computing access control needs latency, efficiency, generality, aggregation, privacy, resource limitation, and policy administration. They also discussed access control models. The DAC model allows the data owner to decide and determine access to others based on the identity of the group members. The MAC Paradigm resource is designed with users in mind for matching. Thus, it suits distributed systems better than DAC. Role-based access control (RBAC) prioritizes article responsibility above authorship. RBAC-assigned roles allow users to access system elements. Attribute-based access control (ABAC) and attribute-based encryption (ABE) are ideal cloud computing access control solutions because they preserve data privacy and let data owners manage access. Usage-control-based access control (UCON), RMAC, and

proxy re-encryption. They concluded by discussing exporting expensive computation, regulating access policies, and fog computing access control research.

The authors of [25] compare Fog security approaches using IoT security criteria. They evaluate fog-based strong authentication for IoT devices and how it meets security goals. No standard fuzzy computing architecture to handle trust, privacy, and other obstacles can lead to IoT security difficulties, as shown in this article. IoT devices and fog visibility are also goals of the new approach. Traditional fog and IoT hardware manufacturers still report to many cloud service providers. Many issues require decentralized design. Authentication methods with the lowest ratings on architectural elements including security, usability, and productivity were investigated. They then addressed how they want to enlarge this pool and establish a qualitative and quantitative diagnostic framework for IoT authentication systems. In [26], the authors establish a model to assess reputation and trust management and set the scene for integrating trust and reputation into a security architecture to ensure WSN data security, dependability, integrity, and trustworthiness. The authors examined fog computing topologies and authentication methods. They also reveal IoT security innovations. IoT device security and privacy risks are growing, he says. Thus, they must build security mechanisms against a range of threats to give the system a convincing foundation for detecting an intruder, even if the threat changes. Fog computing has several risks, thus uniform standards may help solve some of them. Since IoT network security depends on fog computing architectures, they must be developed with security in mind. Communication is wireless between N sensor nodes. Multi-hop routing sends regulated data from sensor nodes to other networks or the Internet via a group leader or gateway. Every member node has an environment-wide reputation value. A packet that gets such a signal keeps just neighboring data and ignores other node data. Through experimentation and research, their algorithm found a solution. This optimisation approach improved communication dependability and cost-efficiency.

Researchers [7] describes a fog computing trust management (COMITMENT) strategy that uses quality of service and quality of protection experience metrics from earlier direct and indirect fog node encounters to assess and manage node trust. They built a model approach to assist foggers make the optimal judgments during degassing by working with them. The authors study a distributed fog topology with nodes effectively spread over many sites and linked by a communication mechanism that assigns each node an IP address. Fog nodes may interact without a central console (mesh network), making resource distribution and function transparent. Since there is no central authority to designate trusted nodes in the network, Fog nodes regularly generate trust scores and a local list of trusted nodes for their neighbors. They evaluated the suggested technique in MATLAB (2018b) and found that it outperformed random wax discharge (RWO) and proximity fog discharge (NFO) using competing conventional methods. Every node without 75% fog increases the network's fog to 5%. The average number of successful and failed contacts during the test showed that network abuse raised the proportion of unsuccessful interactions and lowered the percentage of successful interactions.

The Ciphertext Policy Attribute-Based Encryption (CP-ABE) algorithm and blockchain technology are used in [27] to remove rogue fog nodes and minimize network latency between the cloud service provider (CSP) and fog nodes. The blockchain stores on-chain tracking tables, and the smart contract verifies fog nodes' identities before they can access the CSP's encrypted data. FNs that purposefully change the on-chain tracking table are tagged as rogue fog nodes due to blockchain immutability. Transferring data from cloud storage to end-user devices is expensive and latency-prone. To address these challenges, Fog Computing was invented. However, BSs are vulnerable to malicious attacks that compromise user data. The blockchain uses and integrates the contemporary cryptographic primer CP-ABE algorithm to secure communication between the BS and the CSP with secrecy, integrity, and access to end-user data. Blockchain may allow CSP FN to spread access control authority.

A cloud fog control middleware that manages service requests with limits is proposed in [28] to merge cloud and fog computing. Cloud fog management middleware may be efficiently maintained without additional design components if one wants to join a new fog node, group, or node goes down. Their fog node manager (FNM) only works in fog. A fog group maintains all fog nodes or related fog nodes. They

think their model will save energy use and service times. Cloud-IoT-Fog interaction generates massive data that may need change and integration. Attacks might happen in sequence. Ineffective resource rules and user activity monitoring might lead to such assaults. Basically, poor security implementation might cause security vulnerabilities. Fog systems must be protected from resource exploitation, malware, and other threats [3]. They must provide anti-fogging and resource management to speed up turnaround. Fog systems analyze vast amounts of data well.

3. Proposed Methodology

This study introduces safe fog computing using software-defined fog computing. The fog computing platform protects end-user and fog node processing. Fog management nodes monitor fog nodes. The fog management node (FMN) validates IDs and assigns non-sensitive jobs to fog nodes entering the network in our idea. FMN watches the new fog node after access and estimates trust based on activity. Trust between fog-2-fog nodes depends on recent activity. FMNs control fog nodes. A harmful fog node is removed from the network and designated malicious. All FMNs get malicious node status and SDN controller alert. This section covers FMN. Three subsections comprise this section. Part 1 introduces our SDN-enabled fog computing architecture. The second and third components involve access control and weighted trust evaluation.

3.1. Software-Defined network-based fog computing

This section describes our SDN-enabled fog computing architecture's layers. The IoT devices, Fog, SDN controller, and cloud layers have different methods. Figure 1 shows these layers.

3.1.1. IoT devices layer

IoT devices employ sensors, actuators, microcontrollers, RFID tags, and transceivers to process final commands and deliver applications. Internet of Things isn't one technology. Instead, it's a combination of technologies. People interact with their surroundings via sensors and actuators. Users must store and interpret sensor data wisely to gain insights. We define "sensor" generally; a microwave oven or cell phone can be a sensor provided it delivers current state information. An actuator, like an air conditioner temperature controller, affects the surroundings.

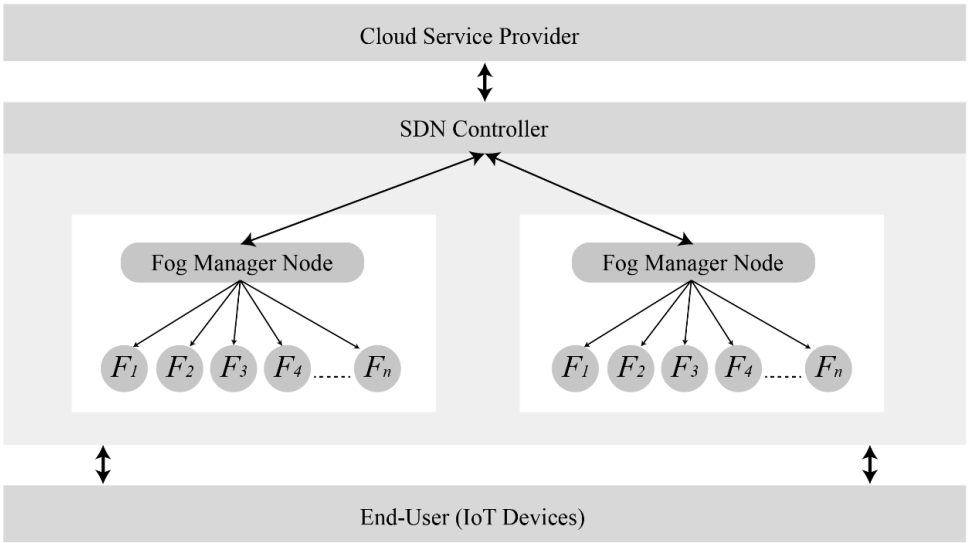

Figure 1: Proposed SDN-Based Fog Computing Framework

3.1.2. Fog computing layer

Cisco said edge computing penetrates deeper than fog computing, which includes smart doors and sensors. This model implies motors, pumps, and lights can process data intelligently. Network edge devices should preprocess as much data as feasible.

Our fog computing paradigm contains parent and child nodes. Fog management nodes, or fog heads, are parents, whereas fog nodes are children.

Fog nodes: Software on IoT devices is called cloud nodes. These nodes connect with IoT devices using CoAP and SNMP. Less devices than computer resources are required. A router, access point, switch, gateway, firewall, and dedicated server can be cloud nodes. Cloud nodes can be linked directly to an SDN device like a switch or router or configured with one. The following operational modules make up each cloud node. This server accepts the Cloud Manager node's request. Monitors IT services implementation includes this module. Database - This module saves the received request in the database, updates the current state of the cloud node system, and available resources, and monitors the readiness of the data.

Fog manager node: The central console of the fog network is the node of the fog manager. The node must be connected to the FMN every time it joins the network. If the connected node is an edge node, the FMN will send the addresses of the active fog nodes based on their proximity. The edge node will later establish a connection with the nearest fog node and start transmitting data. If a fog node goes down, it will send the address of the next available nodes to connect to.

When a smart device tries to join the fog network, the FMN sets the lowest level of trust connecting the node. If a node needs to be removed from the network (due to an outage or permanent disconnection), FMN will revoke that node's access rights and update the list of active fuzzy nodes for the edge sensor, SDN, and cloud layer. SDN will notify other fog management nodes of unauthorized nodes.

Fog Manager Node performs the following operations.

1. Access control management.
2. Monitoring of fog nodes.

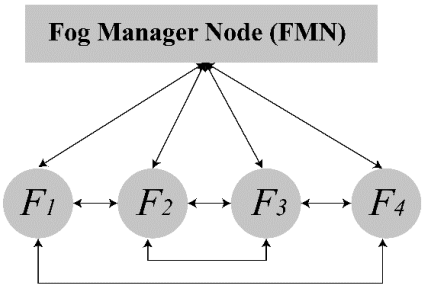

Figure 2: Fog Manager Node and Fog Node

3.1.3. Software-defined network (SDN) Controller

Software-Defined Networking (SDN) is a network development in which the data layer is separated from the console level and all actions, management, and control are concentrated on a single console. Because of its administrative features, it finds applications in other countries, such as cloud computing and cloud computing, to manage asymmetric communication between nodes, thereby increasing security performance.

SDN manages the network in our framework by updating the fog manager nodes. It notifies other hostile nodes of all activities so that they might be spared from the harmful nodes. If a fog node engages in

malicious behavior, a fog manager node estimates its trust and notifies the software-defined network controller, which subsequently notifies the offending node's ID.

3.1.4. Cloud Service Provider (CSP)

Because it won't be calculated, unnecessary data gets transferred to the cloud. Fog computing cannot compete with cloud computing for data computation. Fog computing enhances the computation workflow by reducing latency. When a fog node gains trust, it may connect with the cloud and receive and send data without a fog manager node.

3.2. Access Control Management

The first function here is the Fog Manager node to control access to the new IoT user. When user requests are added, the trust level is calculated at the time. The accuracy of each user is determined by their actions. The authorization assigned to each user is done on a proxy-defined Fog node (FMN) level basis. Delegation is used by resource allocation method. Once the data in this cell is authenticated, the user does not need to access the cloud node method.

New fog nodes (Fns) or user devices must obtain permission from the fog manager node or network head to join the network. This enables the fog node to switch between services. The fog manager node (FMN) will match the new fog node's ID to the malicious fog list. We need the ID verification phase to prevent malevolent nodes from joining another fog network after removal. If FMN decides the new fog is malicious, he will deny the fog node's request to join the network. If the fog node lacks an ID, the fog head will verify and issue one. All network nodes will get the new fog node's ID from the fog management node (FMN). All neighbor fog nodes (NFN) will exchange services with the newly joined node. After that, the fog manager node will assign a non-sensitive task to avoid affecting the surrounding fog node. One algorithm describes the whole procedure.

Algorithm 1 discusses sending network access requests to fog management nodes. Consider this scenario. fog F_n seeks network membership. F_n queries the fog management node in line 1. If the ID and maliciousness of F_n match those on the malicious fog list in lines 2-4, the fog management node will reject their request. Lines 5-7 show the fog management node (FMN) assigning the ID and job to fog node F_n after verification. The wireless sensor network assigns roles based on reputation to facilitate role-based access control.

- **Algorithm 1:** Fog node joining into the network and assigning a task.

1 ***Input:*** *FogManagerNode (FMN); NewFogNode (F_n); Maliciousfognode(ML_f)*
2 ***Parameters:*** *FogList (F_L);*
3 ***Result:*** *Give Access control and assign a task to NewFogNode (F_n)*
4 *NewFogNode (F_n)* will request *FogManagerNode (FMN)* for joining the system.
5 **If** $F_n \epsilon\, ML_F$ **then** ▷ *FMN* will check the ID in MF_L for verification F_n
6 **Declined** ▷ *FMN* will remove the untrusted node
7 **else**
8 $F_n \leftarrow ID$ ▷ *FMN* will assign Id and task to the new fog node
9 $F_L \leftarrow ID(F_n)$ ▷ *FMN* will update the list & send the *Id* of F_n in the network
10 **End**
11 **End**
12 return;
13 **End**

3.3. Weighted Trust Management

Monitoring continues after user authorization. Additionally, reporting unusual behaviors activates a deactivation feature. The Fog Manager Node (FMN) monitors the system to track user activities. The FMN tracks all cloud-fog computing traffic. The FMN notifies multiple fog nodes within a small cell when connected users engage in harmful behavior. Weighted trust management has two parts: FMN evaluates fog node trust, and FMN updates fog node honesty by informing about harmful and genuine nodes. The following subsections summarize the entire scenario.

3.3.1. Trust Evaluation

This procedure involves fog nodes sharing services based on quality of protection. If one fog node (F_1) requests service from another (F_2). If F_2 behaved maliciously or positively with F1, the fog management node (FMN) created a list of F_2's neighbors (NFn) for verification and trust evaluation. FMN selected dynamically weighted witness neighbors. The fog manager node (FMN) tests F_2's trust using eq1.

Let's say the neighbor's fog F_1 gives F_2 a favorable assessment, F_3 negative, F_4 positive, and so on. F_1, F_3, F_4, and F_n have dynamic weights w_1, w_3, w_4, and w_n, respectively, therefore service provider fog node trust is computed in eq1.

Let

$$\{ \propto_1 = (F_2, +),\ \propto_3 = (F_2, -),\ \propto_4 = (F_2, +), \cdots,\ \propto_n = (F_2, n)\}$$

Then trust evaluation is calculated as

$$EV'(F_2) = \sum_{n\in\{1,\cdots,N\}\setminus 2}^{N} w_n * \propto_n \tag{1}$$

Where N is the total number of fog nodes, n is the neighbor nodes interacting with F_2, w is the dynamic weight, and $\propto$ is F_2's positive or negative behavior with n. We add the freshly assessed F_2 trust to the existing trust. Eq2 calculates F_2's trust.

$$Trust\ EV(F_2) = \ EV\ (F_2) + EV'\ (F_2) \tag{2}$$

The trust evaluation for fog node F_2 is EV(F_2). The table now displays the updated final trust evaluation for fog node F_2. Two instances will follow. In the first scenario, fog node F_2 is considered valid if its trust evaluation exceeds the threshold and updates its ultimate trust score. In example 2, the fog node F_2 is malevolent if its trust evaluation is below the threshold. FMN removes harmful nodes from the table/list and adds them to the malicious fog list. FMN notifies both the network fog node and the SDN about the presence of the malicious node. The parent fog manager node will get the malicious node's ID via SDN. Therefore, the second Fog management node will not permit the presence of malicious nodes in the future. We calculate all fog nodes' trusts, enabling FMN to update them and remove malicious nodes.

3.3.2. Honesty updating

In this subsection, the honesty of informer is updated based on their honesty. We see in previous scenario that the fog node *F_1* inform the FMN about the behavior of fog node *F_2*. So based on their behavior with neighbor FMN node evaluate their trust. But in this scenario the honesty of *F_1* is increase or decrease based on their honesty. Let's suppose the *F_1* inform the behavior of the *F_2* to FMN. Then there will be two scenarios.

First scenario: fog node F_1 sends F_2 negative input, FMN recalls neighbor as witness. The witnesses also critique. Positive fog node F_1 feeds F_2. Even witnesses spoke well. Therefore, F_1's claim about F_2 is true. Equation 4 rewards F_1's honesty.

$$H(\ F_1)' = H\ (F_1) + \ H(F_1) * \ \beta \tag{3}$$

In this case, β represents the reward threshold. Multiply by H(F_1) and add to current honesty. A node's existing honesty from past honest feedback. Fog node F_1's updated honesty is H(F_1)'.

Second scenario: fog node F_1 sends F_2 negative feedback, but the nearby witness offers positive or opposite input. Or fog node F_1 feeds F_2 positively. And witnesses criticize. Therefore, F_1 is erroneous about his

assertion with F_2. F_1's honesty will suffer as a punishment. F_1 will be removed from the network and tagged as malicious in eq5 if its honesty is below threshold.

$$H(F_1)' = H(F_1) - H(F_1) * \beta \quad (4)$$

In this case, β represents the penalty threshold. This value multiplies and subtracts existing honesty.

- **Algorithm 2:** Weighted Trust Management
  Input: *NeighborFogNode (F_n), TotleNumberOfFogNode (N); FogManagerNode (FMN); SoftwareDefinedNetwork (SDN); Maliciousfognode(ML_f)*

  Parameters: *HonestyEvaluation; WeightedTrustEvaluation; FogList (F_L);*

  Initialization: *Honesty = 1; Trust = {Ø};* $\propto = \pm n; \beta = n;$ *w = 1; Thr = n;* $F_L = \{f_1, \cdots, f_n\}$
  Result: Weighted neighbor trust and honesty evaluations *' F_1 toward F_2*

1. **Procedure 1:** Weighted Trust Calculation;
2. $F_1 \rightarrow F_2$ ▷ *F_1* will interact with *F_2*
3. $FMN \leftarrow F_1(F_2, Comment)$ ▷ *F_2* will report the behavior of *F_2* to *FMN*
4. $FMN \rightarrow NF_2 = [F_3, \cdots, F_n]$ ▷ *FMN* will create witness neighbor fog node list
5. $$EV'(F_2) = \sum_{n \in \{1,\cdots,N\} \setminus 2}^{N} w_n * \propto_n$$ ▷ Evaluate the trust of *F_2*
6. $Trust\ EV(F_2) = EV(F_2) + EV'(F_2)$ ▷ Calculate the overall trust of *F_2*
7. **if** $Trust\ EV(F_2) < Thr$ **then**
8. $FMN = F_L - F_2$ ▷ Remove untrusted node and marked as malicious
9. $SDN \leftarrow FMN(F_2, update)$ ▷ *FMN* will Update the status of *F_2* to *SDN*
10. $FMN' \leftarrow SDN(F_2, informID)$ ▷*SDN* will inform the ID of *F_2* to all parent *FMN'*
11. **Else**
12. $F_L = update(F_2)$ ▷ update list
13. **End**
14. **End**
15. return;
16. **End**

Procedure 2: Update the Honesty of *F_1* by;

17. **If** $(Trust\ EV(F_2) < 0\ \&\ \propto_1 < 0)\ OR$ $(Trust\ EV(F_2) > 0\ \&\ \propto_1 > 0)$ **then** ▷ In both cases F_1 was true about F_2

18. $H(F_1)' = H(F_1) + H(F_1) * \beta$

19. $F_L = updated(F_1)$ ▷ update honesty of F_1 in list

20. **else if** $(Trust\ EV(F_2) > 0\ \&\ \propto_1 < 0)\ OR\ (Trust\ EV(F_2) < 0\ \&\ \propto_1 > 0)$ **then** ▷ In both cases F_1 was false about F_2

21. $H(F_1)' = H(F_1) - H(F_1) * \beta$

22. $F_L = updated(F_1)$ ▷ update honesty of F_1 in list

23. **End**

24. return;

25. **End**

Algorithm 2 explains how to get a suggestion from nearby fog nodes and calculate weighted trust. If fog F_1 interacts with fog F_2, F_1 follows. Procedures 1 and 2. In the procedure 1, F_1 interacts with fog F_2 and gives feedback based on its behavior to the fog manager node from lines 1-3. The fog management node obtains the neighbor list and assesses the trustworthiness of fog F_2 in lines 4 through 6. Based on the final trust score, the fog management node erases or updates the fog F_2 trust and informs the SDN, preventing any network contact with the malicious node in lines 7-13. In the procedure 2, FMN assesses Fog F_1 honesty and updates. F_1's honesty increases in lines 17-20, while their dishonesty decreases in lines 21-23.

3.3.3. Experimental Setup

Table 1: System Setup and Simulation Settings

Parameter	Value
Operating system	Win 10
Processor	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz
RAM	8.00 GB
System Type	64-bit operating system, x64-based processor
Simulation environment	iFogSim, Java jdk-8u241 and Eclipse-IDE
Number of fog nodes	10
Number of IoT devices	10
$\boldsymbol{Thr_{trust}}$	1.5
$\boldsymbol{Thr_{honesty}}$	1.5
$\propto$	0.5
$\boldsymbol{\beta}$	0.5

In this part, we evaluate the safe Fog-2-Fog collaboration paradigm to enable secure fog service requests. We used Java to sketch our network architecture, testing the suggested access control and trust evaluation technique. We simulate using iFogSim [29]. iFogSim is used to manage IoT services within a fog infrastructure. Figure 3 shows how iFogSim visualizes the proposed design.

3.4. Simulation Settings

Table 2 shows the system functions that were used in the simulation. We provide simulation parameters in terms of network topology, propagation and transmission delay, uplink and downlink capabilities, and Fog interaction.

3.4.1. Network Topology

Fogs are represented as a mesh network when the network topology is characterized as an indirect graph. The simulation comprises ten fog nodes (i.e., Fn = 10) that are connected via an internal communication network. In our simulation, we used iFogSim default settings, which include a 100 ms delay for cloud and proxy connections and a 2 ms latency for SDN controller and fog-to-end devices.

3.4.2. Network Bandwidth

The link bandwidth depends on the type of service's request; hence, heavy-request will require more bandwidth than light request. Table 2 shows the default entity settings in iFogSim. It shows the fog node and IoT device default specifications in terms of MIPS and RAM in gigabytes.

Table 2: Default entity configurations in iFogSim

Attribute	**F1, F2, F3, F4 F5, F6, F7, F8, F9, F10**	**FMN1, FMN2**	**SDN**	**IoT devices**
MIPS	2000	10000	10000	1500
RAM (GB)	10	4	10	2
Uplink Bw	10000	100	200	10000
Downlink Bw	270	10000	20000	10000

3.4.3. Fogs interactions

We allow fog nodes to interact and request services. To calculate fog node trust, we used a mesh network to allow fogs to communicate and report their service providers to the fog management node. Binary confidence score values representing excellent or poor interactions are stored locally as 1 and 0. Using all prior node interactions/collaborations, a weighted trust is generated to assess the partner fog node's trustworthiness.

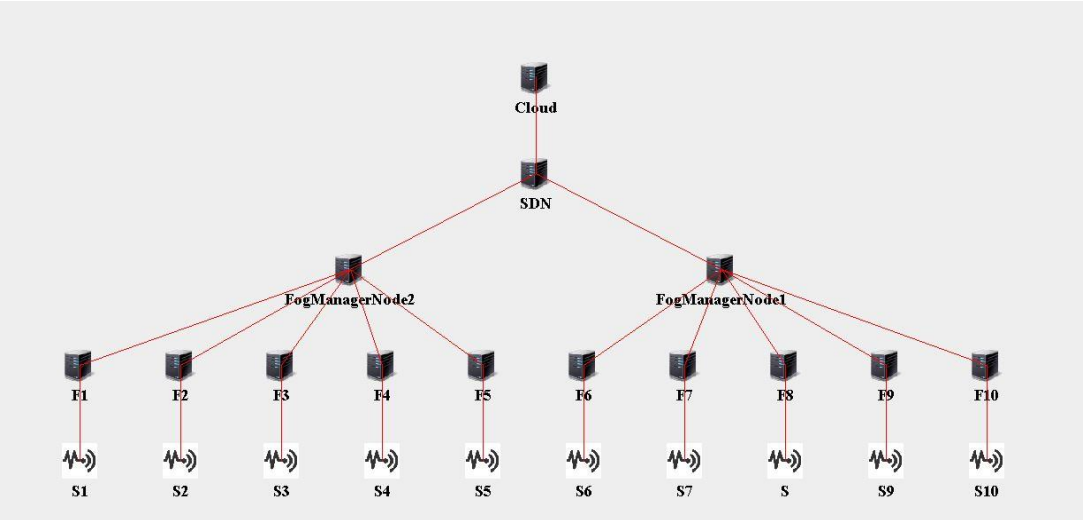

Figure 3 Overall Simulated Architecture in iFogSim

As you can see that we planted to FMN(FMN) and each FMN have five child nodes. These nodes can communicate with each other and also with their parent's head node. Fog manager nodes are connected with SDN controller. When any node leaves the network due to their negative behavior are informed to SDN so that he could inform other fog manger node so they will be aware of malicious node and did not give permission to these nodes to enter in the network.

4. Results and discussion

This section shows the numerical results of the experimentations on the proposed model to validate the accuracy of our secure Access control based on the weighted trust model. We first evaluate the performance of the Proposed weighted trust algorithm and then the performance of the access control algorithm. We compare our proposed trust algorithm on different parameter and did two experiments to gets the results

4.1. Experiment I

At the first experiment, we set the α and β respectively 0.5 and 0.5. Initial trust value of all fog nodes *Trust* = 1 and *Honesty value* = 1. and we keep a 1.5 threshold. The results of simulation analysis demonstrate a favorable impact on network usage and trust in our proposed algorithm, where Figs. 6.1, 6.2, and 6.3 show that the proposed algorithm detects malicious nodes very precisely.

4.1.1. Trust Evaluation

Fig 6.1 shows how the F_1 and F_5 maintain their trust value and achieve the highest Trust and how the F_2, F_3, and F_4 did the malicious activity and these are removed from the network.

When the trust value of the fog node decreases from this threshold then that fog node is removed from the network. Trust and honesty increase with the passage of time and their behavior. Trust value increase and decrease by 0.5 after each transaction.

In fig 6.1 we can see that when F_1 and F_5 did positive behavior then its value increased from 1 to 1.5 and likewise when F_2, F_3, and F_4 misbehave then their values decrease 0.5. and at the end the value of these fog nodes becomes 0 and these nodes are removed from the network.

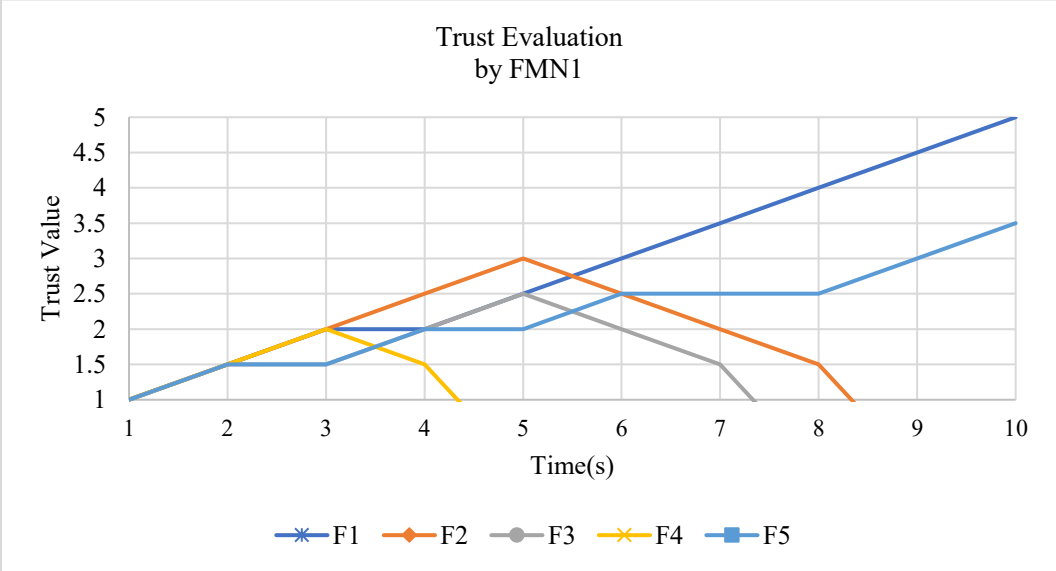

Figure 4: Trust Evaluation of Fog Nodes by Fog Manager Node (FMN1)

Fig 6.2 shows how the F_6, F_8, and F_9 maintain their trust value and achieve the highest Trust and how the F_7 and F_9 do the malicious activity and were removed from the network.

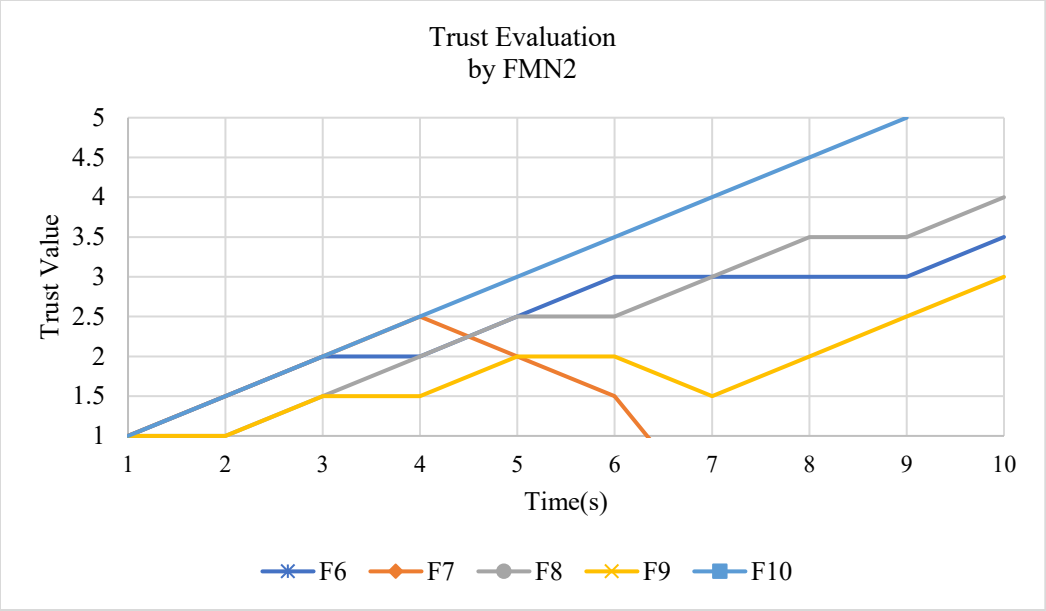

Figure 5: Trust Evaluation of Fog nodes by Fog Manager Node (FMN2)

In above fig 6.2 we can see that when the F_6, F_8, and F_9 did positive behavior then their values increased from 1 to 1.5 and finally reached they're to the highest value. and likewise, when F_2, F_3, and F_4 misbehave than their values decrease by 0.5. and at the end, the value of these fog nodes becomes 0 and finally these nodes are removed from the network.

In fig. 6.3 below we can see the overall simulation result of all fog nodes in the network. F_2, F_3, F_7, F_2, and F_9 are removed from the network due their malicious activity.

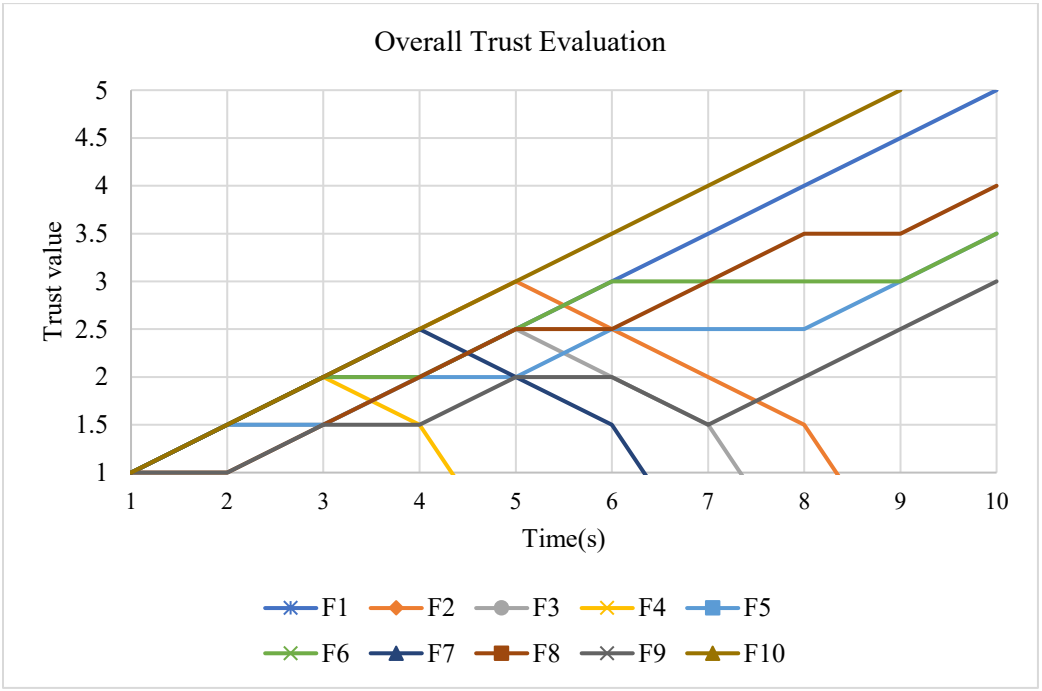

Figure 6: Trust Evaluation of all Fog nodes by their Fog Manager nodes (FMN)

4.1.2. Honesty updating

After the trust evaluation of service provider fog nodes, the FMN updates the honesty of the informer. Informer is the fog node that interacts with the service provider. It gives feedback about the service provider. The honesty of the informer is updated based on their honest review of the service provider.

Figure 7, 8, and 9 show how the honesty of the fog node is updated. Honesty is the reward and penalty of honest or dishonest interactor. In below fig F_1, F_3, and F_5 fog node become dishonest by providing wrong information about the service provider.

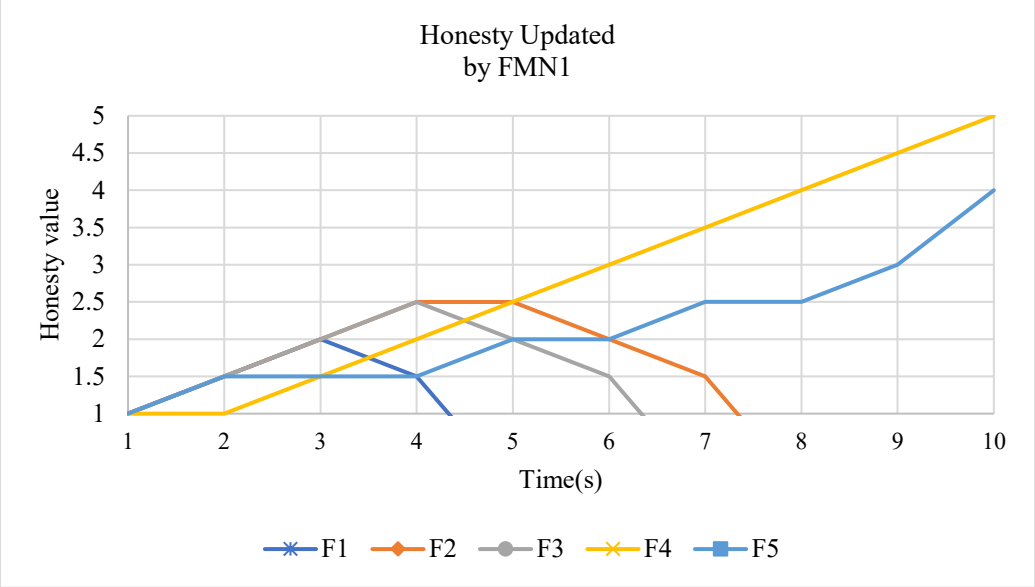

Figure 7: Honesty Updating of Fog Nodes by Fog Manager node (FMN1)

In Fig 7 we can see that F_1, F_3, and F_5 interact with service provider fog nodes and give honest feedback about them. So, the Fog manager node increases their honesty by 0.5. and that's how by the passage of time they get the highest honesty on every honest feedback about their service provider in the network. On another hand, F_2, F_3, and F_4 misbehave and give dishonest feedback about their service provider and that's why they lose their honesty and are removed from the network.

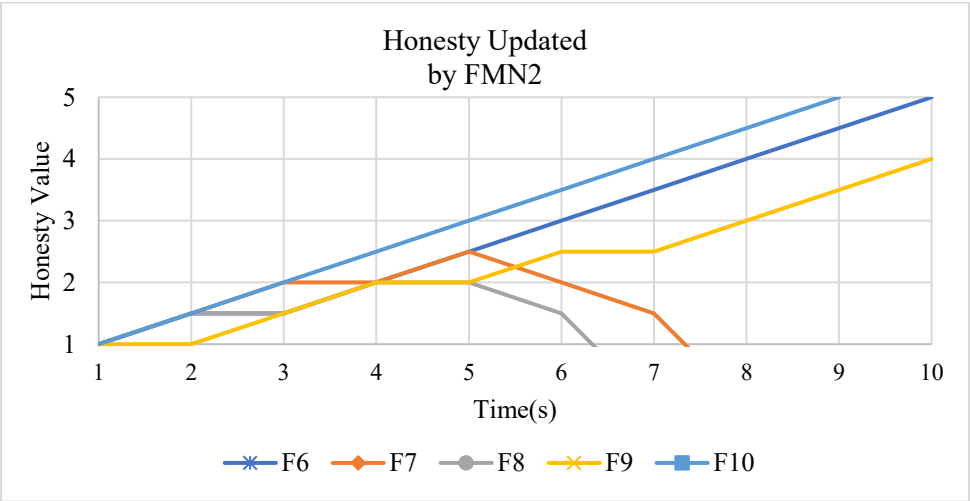

Figure 8: Honesty updating of Fog Nodes by Fog Manager Node (FMN2)

In figure 8 fog node F_6, F_8, and F_9 get highest trust value by giving honest reports to the Fog manager node of the service provider. And F_7 and F_{10} send the wrong report to the fog manager node and loss their honesty.

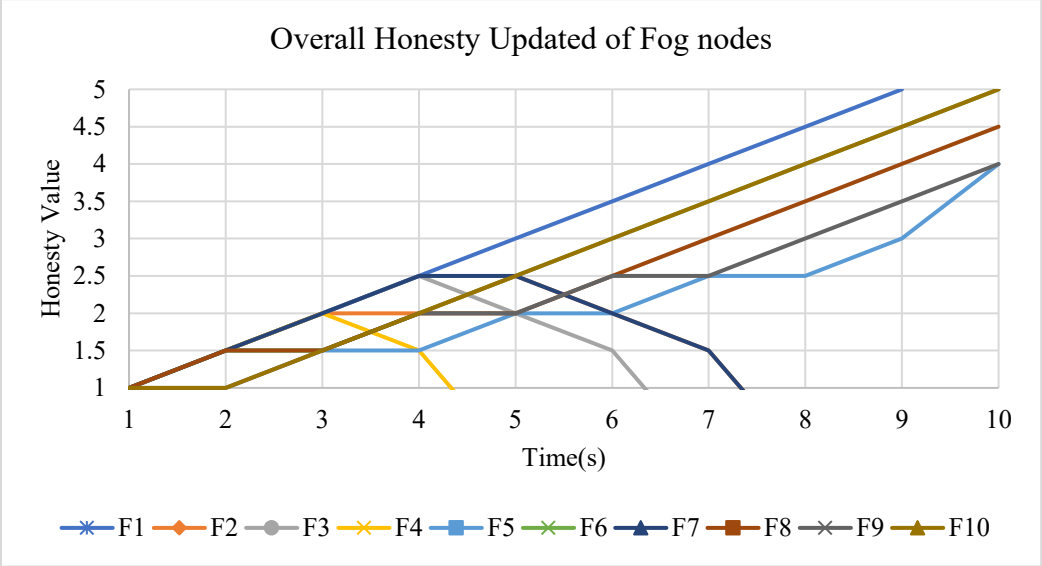

Figure 9: Honesty Updating of all Fog Nodes by their Fog Manager Nodes (FMN)

In figure 9 we can see the overall honesty updated of fog nodes in their networks. Fog manager node remove the dishonest reporter and give an update to the SDN controller so that he could report all fog manager nodes in the network.

The final result we can see in table 3 and 4.

Table 3: Fog Nodes Present in Network with their ID, Honesty, and Trust level

Node ID	***Node Trust***	***Node Honesty***
F_1	5.0	5.0
F_2	-	-
F_3	-	-
F_4	-	-
F_5	4.0	5.0
F_6	3.5	5.0
F_7	-	-
F_8	4.0	4.5
F_9	3.0	4.0
F_{10}	5.0	5.0

Malicious fog node ID are kept in Malicious fog node list so when malicious node tries to re-enter in the network and request the fog manager node then FMN match their ID in fog list and declined their request.

Table 4: Malicious Fog Node List (ML_F)

Malicious Node ID	Value
F_2	0.0
F_3	0.0
F_4	0.0
F_7	0.0
F_{10}	0.0

4.2. Experiment II

At the first experiment, we set the α and β respectively 0.3 and 0.3. Initial trust value of all fog nodes *Trust* = 1 and *Honesty value* = 1. and we keep a 1.3 threshold. we can see the result in Fig 6.7, 6.8, and 6.9.

4.2.1. Trust Evaluation

In figure 10 F_1, and F_3 try to misbehave and are removed from the network. on other hand F_2 F_4 and F_5 become most weighted and trusted fog node of network.

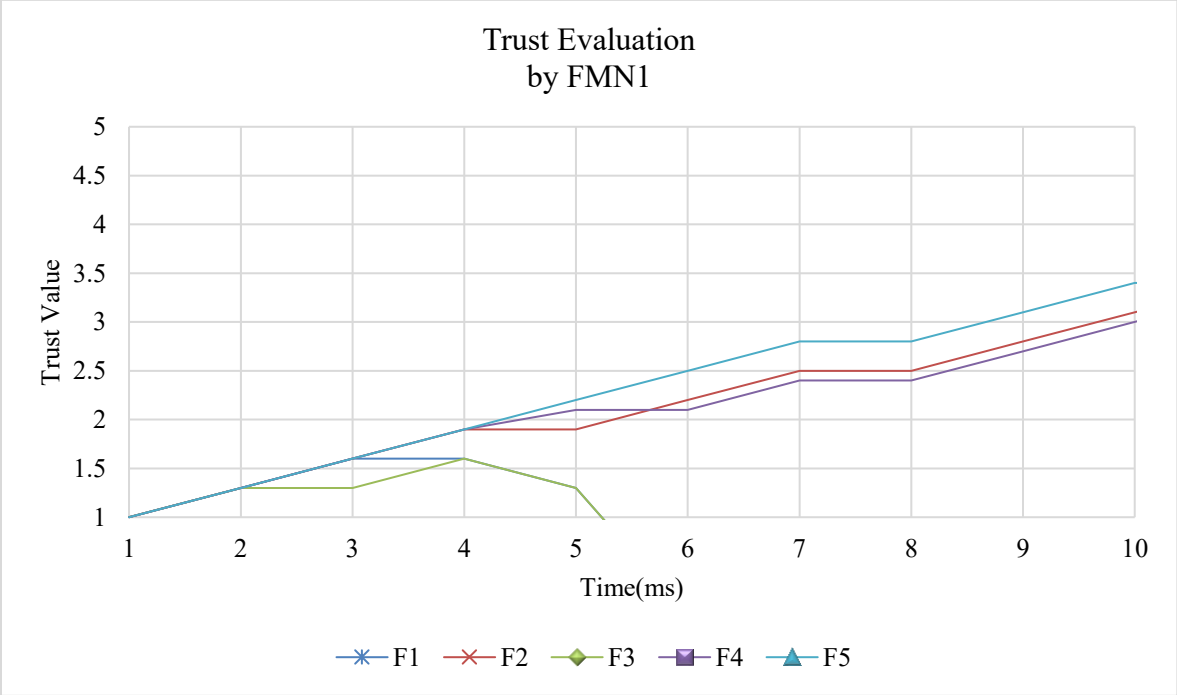

Figure 10: Trust Evaluation by FMN1

In figure 11 we can see that F_9 and F_7 did malicious activity and removed from the network. F_9 loss their trust at 3ms after two transactions. and F_7 after six transactions. F_6, F_8, and F_{10} reached at highest trust level.

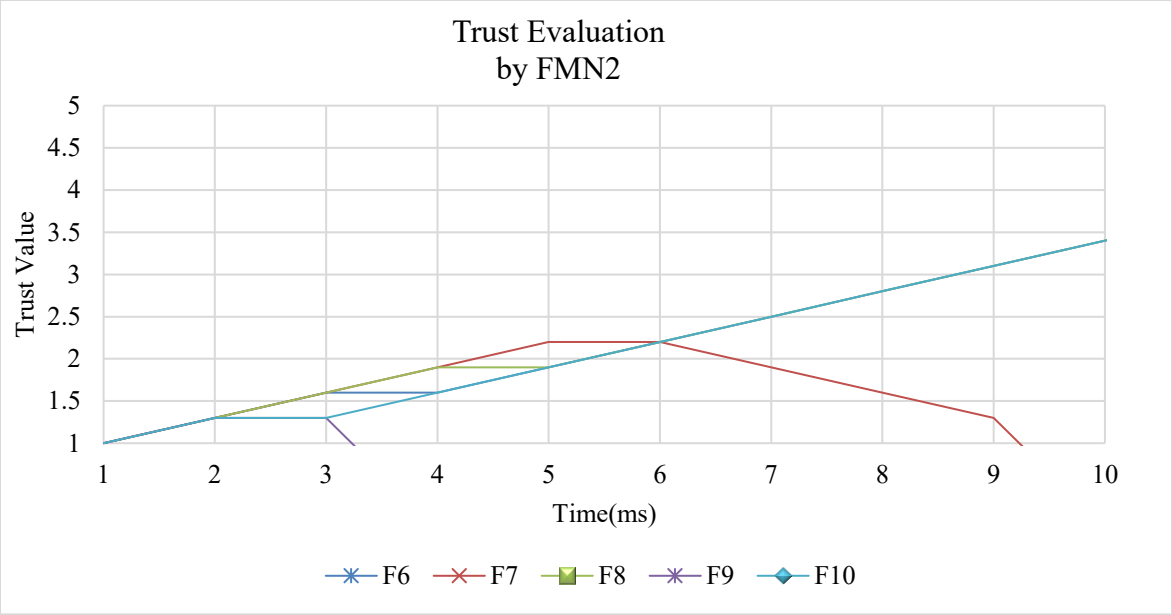

Figure 11: Trust Evaluation by FMN2

In figure 12 we can see overall trust evaluated for all fog nodes.

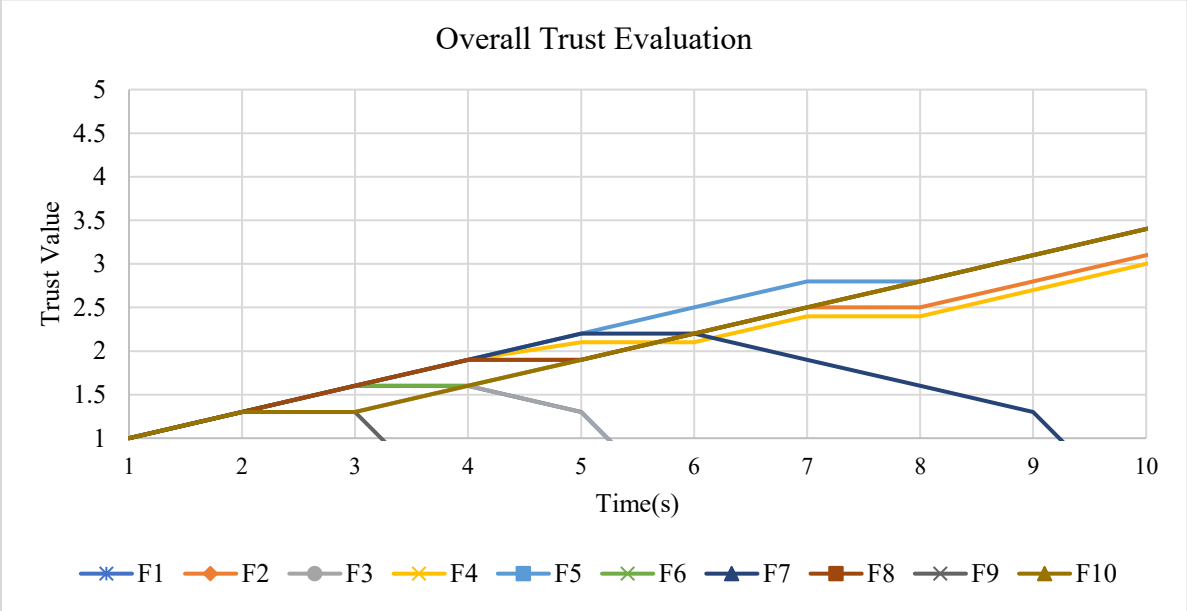

Figure 12: Overall Trust Evaluation

4.2.2. Honesty updating

In figure 13 F_3 and F_1 interact with service provider and report negative to the fog manager node. While the service provider is not malicious node. So, F_3 and F_1 is malicious and loss their honesty. F_2, F_4, and F_5 behave positive and increase their honesty.

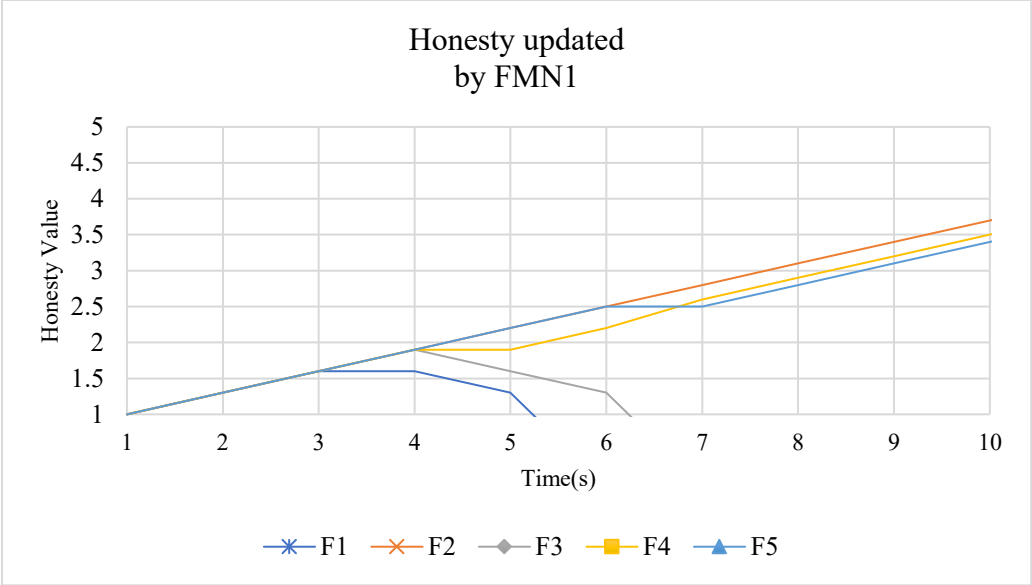

Figure 13: Honesty Updated by FMN1

In figure 14 we can see that F_6, F_8, and F_{10} are increases their honesty by providing honest report to fog manager about service provider. While on the other hand F_7 and F_9 provide dishonest report and loss their honesty.

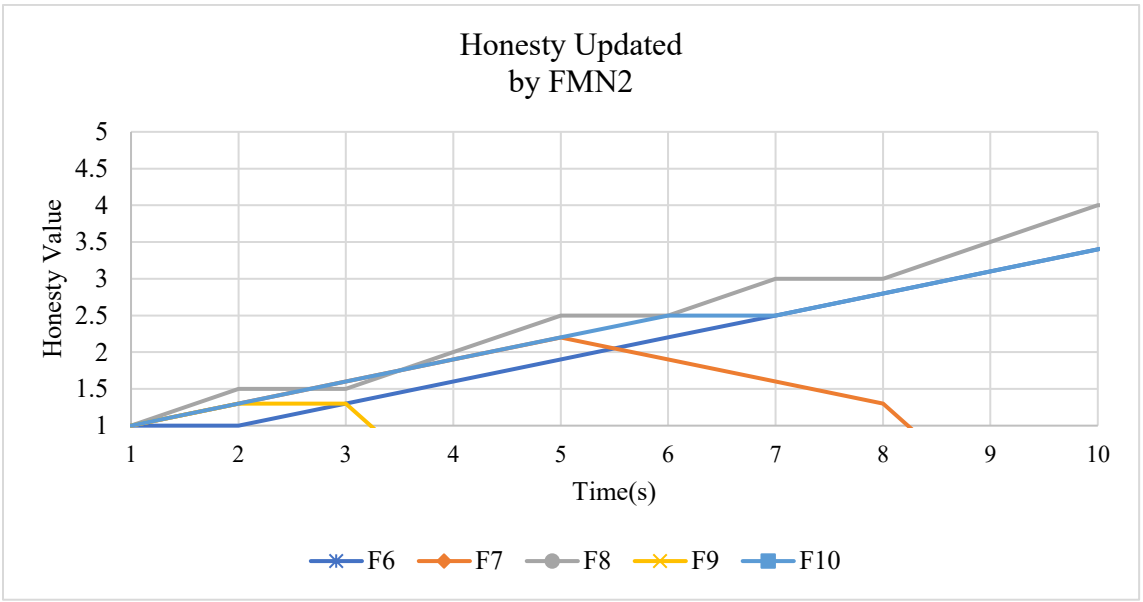

Figure 14: Honesty Updated by FMN2

In figure 15 we can see overall activity of fog nodes in network.

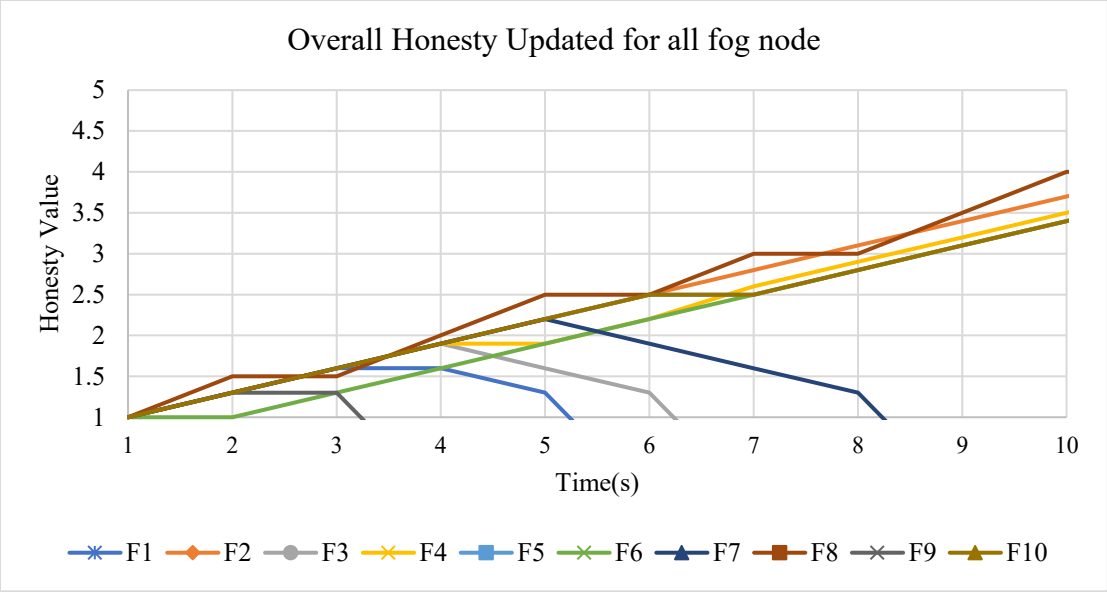

Figure 15: Overall Honesty Updated for all Fog Nodes

We can see the Final output in table 5 is obtained from our proposed weighted trust evaluation algorithm. Untrusted fog nodes are removed from the list and added in malicious fog list.

Table 5: Fog nodes present in network with their ID, Honesty, and Trust level

Node ID	*Node Trust*	*Node Honesty*
F_1	-	-
F_2	3.1	3.7
F_3	-	-
F_4	3.0	3.5
F_5	3.4	3.4
F_6	3.4	3.4
F_7	-	-
F_8	3.4	4.0
F_9	-	-
F_{10}	3.4	3.4

Table 6: Malicious Fog Node List (MLF)

Malicious Node ID	*Value*
F_1	0.0
F_3	0.0
F_7	0.0
F_9	0.0

4.3. Access Control Management

Table 7 shows iFogSim data for fog nodes joining networks. Performance measurements include access control method implementation time in milliseconds. Some malicious fog nodes are rejected by the fog management. Fog nodes cannot connect to the network due to access control. The maximum fog node

access duration has raised from 2.3 to 3.4 milliseconds. When malicious fog attempts to enter the network, FMN matches their ID with the malicious fog node list and denies their request.

Table 7: Example of the Various Metrics Reported by iFogSim

Fog ID	STATUS	Fog Manager Node	Time
F_1	FAILURE	-	-
F_2	ACCEPT	1	2.5
F_3	FAILURE	-	-
F_4	ACCEPT	1	3.5
F_5	ACCEPT	1	3
F_6	ACCEPT	2	3.4
F_7	FAILURE	-	-
F_8	ACCEPT	2	2.3
F_9	FAILURE	-	-
F_{10}	ACCEPT	2	3
F_{11}	ACCEPT	1	3.6
F_{12}	FAILURE	-	-
F_{13}	FAILURE	-	-
F_{14}	ACCEPT	2	2.6
F_{15}	ACCEPT	1	3.4

5. Conclusion

Many modern IoT devices need communication to other IoT devices, gateways, or network components due to their processing power, mobility, and increased discovery. Thus, an untrusted fog node may get illegal access to resources, compromising IoT network functionality. Malicious nodes can also alter data. Malware can also hinder system performance. To enable global growth and end-user migration, fog computing brings cloud computing and services to the network edge. SDN-based fog computing infrastructure security architecture was described in this work. To boost fog layer performance, we recommended using adjacent smart devices like smartphones and tablets' idle resources. This approach raises security challenges. We offer access control and trust evaluation model. We showed how our design can solve fog network security issues. Our approach calculates trust levels from node behavior. We provided algorithms for applying our hypothesis. Our implementation showed that the fog system could distinguish trusted and untrusted dynamic nodes. In iFogSim, we showed the fog network access control and trust evaluation algorithms' results. Simulation results showed the importance of integrating our technique within the Fog paradigm to resolve the trust and access control issue of newly joining nodes and IoT devices. We tested our algorithms on different parameters and thresholds and found and eliminated malicious fog nodes using FMN in fog computing.

References

[1] Tomovic, Slavica, Kenji Yoshigoe, Ivo Maljevic, and Igor Radusinovic. "Software-defined fog network architecture for IoT." *Wireless Personal Communications* 92 (2017): 181-196.

[2] Shehada, Dina, Amjad Gawanmeh, Chan Yeob Yeun, and M. Jamal Zemerly. "Fog-based distributed trust and reputation management system for internet of things." *Journal of King Saud University-Computer and Information Sciences* 34, no. 10 (2022): 8637-8646.

[3] Hosseinpour, Farhoud, Ali Shuja Siddiqui, Juha Plosila, and Hannu Tenhunen. "A security framework for fog networks based on role-based access control and trust models." In *Research and Practical*

Issues of Enterprise Information Systems: 11th IFIP WG 8.9 Working Conference, CONFENIS 2017, Shanghai, China, October 18-20, 2017, Revised Selected Papers 11, pp. 168-180. Springer International Publishing, 2018.

[4] Kumar, Sachin, Prayag Tiwari, and Mikhail Zymbler. "Internet of Things is a revolutionary approach for future technology enhancement: a review." *Journal of Big data* 6, no. 1 (2019): 1-21.

[5] Diro, Abebe Abeshu, Haftu Tasew Reda, and Naveen Chilamkurti. "Differential flow space allocation scheme in SDN based fog computing for IoT applications." *Journal of Ambient Intelligence and Humanized Computing* (2024): 1-11.

[6] Khakimov, Abdukodir, Abdelhamied A. Ateya, Ammar Muthanna, Irina Gudkova, Ekaterina Markova, and Andrey Koucheryavy. "IoT-fog based system structure with SDN enabled." In *Proceedings of the 2nd international conference on future networks and distributed systems*, pp. 1-6. 2018.

[7] Al-Khafajiy, Mohammed, Thar Baker, Muhammad Asim, Zehua Guo, Rajiv Ranjan, Antonella Longo, Deepak Puthal, and Mark Taylor. "COMITMENT: A fog computing trust management approach." *Journal of Parallel and Distributed Computing* 137 (2020): 1-16.

[8] Hakiri, Akram, Bassem Sellami, Prithviraj Patil, Pascal Berthou, and Aniruddha Gokhale. "Managing wireless fog networks using software-defined networking." In *2017 ieee/acs 14th international conference on computer systems and applications (aiccsa)*, pp. 1149-1156. IEEE, 2017.

[9] Shehada, Dina, Chan Yeob Yeun, M. Jamal Zemerly, Mahmoud Al-Qutayri, Yousof Al-Hammadi, and Jiankun Hu. "A new adaptive trust and reputation model for mobile agent systems." *Journal of Network and Computer Applications* 124 (2018): 33-43.

[10] Sadique, Kazi Masum, Rahim Rahmani, and Paul Johannesson. "Trust in Internet of Things: An architecture for the future IoT network." In *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, pp. 1-5. IEEE, 2018.

[11] Patwary, Abdullah Al-Noman, Nishat Hossain, and Mohammad Aslam Sami. "A detection approach for finding rogue fog node in fog computing environments." *Am. J. Eng. Res* 8 (2019): 2320-0847.

[12] Salonikias, Stavros, Ioannis Mavridis, and Dimitris Gritzalis. "Access control issues in utilizing fog computing for transport infrastructure." In *Critical Information Infrastructures Security: 10th International Conference, CRITIS 2015, Berlin, Germany, October 5-7, 2015, Revised Selected Papers 10*, pp. 15-26. Springer International Publishing, 2016.

[13] Panja, Biswajit, Sanjay Kumar Madria, and Bharat Bhargava. "A role-based access in a hierarchical sensor network architecture to provide multilevel security." *Computer Communications* 31, no. 4 (2008): 793-806.

[14] Rathee, Geetanjali, Rajinder Sandhu, Hemraj Saini, M. Sivaram, and Vigneswaran Dhasarathan. "A trust computed framework for IoT devices and fog computing environment." *Wireless Networks* 26 (2020): 2339-2351.

[15] Pallavi, K. N., and V. Ravi Kumar. "Authentication-based access control and data exchanging mechanism of IoT devices in fog computing environment." *Wireless Personal Communications* 116 (2021): 3039-3060.

[16] Zhang, Peng, Zehong Chen, Joseph K. Liu, Kaitai Liang, and Hongwei Liu. "An efficient access control scheme with outsourcing capability and attribute update for fog computing." *Future Generation Computer Systems* 78 (2018): 753-762.

[17] Yu, Zuoxia, Man Ho Au, Qiuliang Xu, Rupeng Yang, and Jinguang Han. "Towards leakage-resilient fine-grained access control in fog computing." *Future Generation Computer Systems* 78 (2018): 763-777.

[18] Daoud, Wided Ben, Mohammad S. Obaidat, Amel Meddeb-Makhlouf, Faouzi Zarai, and Kuei-Fang Hsiao. "TACRM: trust access control and resource management mechanism in fog computing." *Human-centric Computing and Information Sciences* 9 (2019): 1-18.

[19] Misra, Sudip, and Ankur Vaish. "Reputation-based role assignment for role-based access control in wireless sensor networks." *Computer Communications* 34, no. 3 (2011): 281-294.

[20] Chowdhary, Ankur, Dijiang Huang, Adel Alshamrani, Myong Kang, Anya Kim, and Alexander Velazquez. "TRUFL: distributed trust management framework in SDN." In *ICC 2019-2019 IEEE*

International Conference on Communications (ICC), pp. 1-6. IEEE, 2019.

[21] Ukil, Arijit. "Secure trust management in distributed computing systems." In *2011 Sixth IEEE International Symposium on Electronic Design, Test and Application*, pp. 116-121. IEEE, 2011.

[22] Tomasin, Stefano, Simone Zulian, and Lorenzo Vangelista. "Security analysis of lorawan join procedure for internet of things networks." In *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1-6. IEEE, 2017.

[23] Manuel, Paul. "A trust model of cloud computing based on Quality of Service." *Annals of Operations Research* 233 (2015): 281-292.

[24] Zhang, Peng, Joseph K. Liu, F. Richard Yu, Mehdi Sookhak, Man Ho Au, and Xiapu Luo. "A survey on access control in fog computing." *IEEE Communications Magazine* 56, no. 2 (2018): 144-149.

[25] Al Harbi, Saud, Talal Halabi, and Martine Bellaiche. "Fog computing security assessment for device authentication in the internet of things." In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1219-1224. IEEE, 2020.

[26] Ukil, Arijit. "Trust and reputation based collaborating computing in wireless sensor networks." In *2010 Second International Conference on Computational Intelligence, Modelling and Simulation*, pp. 464-469. IEEE, 2010.

[27] Alshehri, Mohammed, and Brajendra Panda. "A blockchain-encryption-based approach to protect fog federations from rogue nodes." In *2019 3rd Cyber Security in Networking Conference (CSNet)*, pp. 6-13. IEEE, 2019.

[28] Apat, Hemant Kumar, Bibhudatta Sahoo, and Prasenjit Maiti. "Service placement in fog computing environment." In *2018 International Conference on Information Technology (ICIT)*, pp. 272-277. IEEE, 2018.

[29] Awaisi, Kamran Sattar, Assad Abbas, Samee U. Khan, Redowan Mahmud, and Rajkumar Buyya. "Simulating fog computing applications using iFogSim toolkit." *Mobile edge computing* (2021): 565-590.