Vol. 07, No. 01, pp. 47 – 66 (2026)
ISSN: 2708-0757

JOURNAL OF APPLIED SCIENCE AND TECHNOLOGY TRENDS

www.jastt.org

Trust-Aware and Adaptive Malicious Node Detection in Fog Network using Independent DQN with Centralized Training

Sheenu Singh[1*], Meetu Kandpal[1], Harshad B Bhadka [2]

[1]*L J University, Ahmedabad, Gujarat, India.* *sheenu.singh@ljku.edu.in*, *meetu.joshi@ljku.edu.in*
[2]*C.U. Shah University, Wadhwan, Gujarat, India.* *harshad.bhadka@gmail.com*
**Correspondence*: *sheenu.singh@ljku.edu.in*

Abstract

The proliferation of decentralized and dynamic networks, such as fog computing and the Internet of Things (IoT), has significantly raised the demand for network security and resilience solutions. This study presents a decentralized trust management framework for detecting malicious nodes in fog-to-fog networks using Multi-Agent Reinforcement Learning (MARL). Our approach utilizes Independent Deep Q-Networks (I-DQN) for decentralized decision making at each fog node with dynamic trust evaluation, allowing them to learn an optimal detection policy based on local observations. Importantly, we enhance this by centralized training controlled by a central orchestrator, which uses a shared global critic and parameter sharing. The proposed system was evaluated on a 10-15 nodes fog network under three distinct attack scenarios: aggressive, stealth, and gradual. Experimental results demonstrate superior performance with detection rates of 92.0% for aggressive attacks, 78.0% for stealth attacks, and 67.9% for gradual attacks. Security focused results demonstrate exceptional false negative performance with FNR values of 8.3% ± 2.0% for aggressive attacks (excellent performance), 22.0% ± 2.0% for stealth attacks (good performance), and 31.9% ± 1.9% for gradual attacks (acceptable performance), ensuring minimal malicious nodes remain undetected across all attack types. The proposed approach provides a highly secure and scalable solution for detecting malicious nodes in fog networks, offering superior threat detection through intelligent trust based decision making and coordinated multi-agent learning.

Keywords: *Fog Computing, Malicious Node, Multi-Agent Reinforcement Learning, Trust Management, Decentralized Network.*

Received: September 29th, 2025 / Revised: December 18th, 2025 / Accepted: December 31st, 2025 / Online: January 21st, 2026

I. INTRODUCTION

The rapid growth of fog computing is a vital solution to the limitations of centralized cloud infrastructures, particularly in latency sensitive applications like smart healthcare, autonomous vehicles, and industrial automation [1,29]. Real time data analysis and decision making are made possible by this localized processing, which drastically lowers latency and bandwidth usage to distant cloud data centers. This is essential for time sensitive IoT-based applications [2,30]. The dynamic and distributed nature of fog-to-fog communication raises security concerns about malicious node behavior that compromises data integrity, undermines service continuity, and erodes system confidence [3,4]. The key problem is detecting and isolating malicious nodes in the network, as they can compromise network reliability and data integrity through various attacks, such as selective packet dropping, bad-mouthing, and evasive "on-off" behaviors [5]. To overcome the problem of node's trustworthiness and automatic adjustment to shifting network conditions and hostile strategies, a strong trust management system is needed.

Existing trust management models in fog environments have limitations because trust value calculation relies on static rules, predefined threshold values and centralized decision making. Due to the dynamic and decentralized nature of fog networks, traditional reputation based or cryptographic based trust frameworks are less effective and typically not equipped to respond to real time malicious behavior. They are also not adaptive to rapidly evolving network states [6].

To overcome these challenges, there is a critical need for a decentralized, autonomous, and adaptive trust model that can operate effectively in uncertainty and dynamic conditions. One promising solution is MARL, which enables individual fog nodes modelled as intelligent agents to detect malicious nodes among their neighbors [7,8]. We employed an advanced Independent Deep Q-Network architecture, where every node is independent and can take its own decision. This model is also reinforced with centralized training to provide robust and cooperative learning in this demanding setting. This centralized critic offers a comprehensive global understanding of the network state and joint actions, delivering stable and consistent learning signals to the decentralized agents during training. The practical implementation demonstrates significant potential for deployment in smart city infrastructure and industrial IoT systems, where conventional security methods fall short due to the distributed architecture, limited computational resources, and stringent real time performance demands inherent in fog networks. The primary contributions and novelties of this study are as follows:

- **Integrated Trust MARL Framework**: We proposed a novel integration of trust management with multi-agent reinforcement learning by explicitly fusing direct and indirect trust metrics within an Independent Deep Q-Network (I-DQN) architecture for malicious node detection in fog computing networks.
- **Coordinated Learning Architecture**: We used the centralised training and decentralised execution (CTDE) paradigm, where a centralised coordinator guides learning while fog nodes retain independent decision making during execution.
- **Adaptive Trust Evaluation**: We design a dynamic trust management mechanism that combines local observations with reputation filtered neighbour recommendations and automatically adjusts a global trust threshold in response to changing network conditions.
- **Comprehensive Evaluation under Multiple Attack Complexities:** We conducted a detailed statistical evaluation under three black hole attack scenarios (aggressive, stealth and gradual), demonstrating improved robustness and generalization compared to state-of-the-art trust based and MARL baselines.

The remaining part of this paper is organized as follows: Section 2 provides a relevant literature review on trust based solutions for securing fog networks using Multi-Agent Reinforcement Learning. Section 3 details the research methodology, and Section 4 discusses our proposed trust-aware multi-agent DQN framework. Section 5 debates the experimental results. Lastly, Section 6 concludes the work and suggests directions for future research.

II. RELATED WORK

The rapid expansion of IoT technologies has created an urgent need for robust security mechanisms to protect distributed computing infrastructures. Trust management systems have emerged as critical components for maintaining network integrity in dynamic, decentralized environments, particularly within fog computing architectures. This growing importance has stimulated substantial research efforts across multiple technical domains. Our literature analysis examines three primary research streams: distributed trust management approaches, machine learning applications in network security, and the integration of MARL with trust mechanisms. A comprehensive comparison of existing methodologies against our proposed framework is detailed in Table 1, highlighting the unique contributions and advantages of our approach.

A. Trust Management in Distributed Systems

To address the scalability issues, single points of failure, and high communication overhead associated with centralized architectures, numerous distributed trust models have been proposed [9,10]. These models enable individual nodes to compute their trust scores based on local observations and interactions. For example, Al-Masri et al. [11] presented a subjective-based trust for fog-based IoT. This model provides a probabilistic foundation for uncertainty, it adapts via Bayesian inference and lacks the autonomous, data driven learning capabilities of reinforcement learning. It is critical for dealing with dynamic and innovative harmful behaviours. Likewise, Al-Tameemi et al.and Secure Comp. for fog-IIoT used direct and indirect trust measures in fog and industrial IoT to study distributed trust and reputation [19,20]. These methods frequently use implicit heuristic principles or threshold based procedures to identify actions such as disparaging others. Similarly, by considering various trust factors, TETES (MDPI, 2023) [21] focuses on trust based efficient task execution in smart cities. However, it is less resilient to unexpected attack patterns, as it also relies on preset thresholds and lacks autonomous learning mechanisms. [22]It robustly manages indirect trust in MANETs against suggestion attacks by addressing reputation based trust. This work is not specifically tailored for the unique dynamics of fog computing. Similarly, frameworks such as VANET-DDoSNet++ provided high accuracy, real time packet detection using a hybrid deep learning approach, with a primary focus on traffic analysis [35].

A more proactive security paradigm, zero-trust frameworks prioritise ongoing verification above implicit confidence. Al-Dubai [18] proposes a zero-trust architecture tailored for 6G Edge/Fog Networks that generates dynamic trust ratings and policies via AI-driven anomaly detection. This effort greatly aids continuous verification and adaptation in extremely highly dynamic contexts. One common issue across all trust management systems is their inability to adapt to the intelligence and adaptability of hostile entities in open, dynamic fog settings. They are frequently reactive, slow to converge or based on predefined rules.

B. Reinforcement Learning in Network Security

The demand for intelligent and adaptable network security solutions has significantly increased the applications of machine learning and, more recently, reinforcement learning (RL) in network security. Traditional machine learning methods for network security face several limitations. High false positive rates are frequently the consequence of unsupervised anomaly detection, particularly when new threats are detected. In contrast, supervised approaches require a continuous supply of updated labelled data, which can be costly and impractical. Moreover, the majority of these methods are reactive, detecting dangers only after they occur. They are not suitable for proactive trust management in extremely dynamic fog situations due to their intrinsic reactivity and inability to execute sequential decision making or adaptive policy learning. The advancement of artificial intelligence has enabled novel approaches in trust management in dynamic distributed systems through RL. RL enables agents to learn optimal trust assessment, allowing them to adapt their decision making processes based on environmental feedback and interaction outcomes. RL can be classified as single-agent and multi-agent RL.

Single-agent RL has demonstrated effectiveness in centralized cloud and Internet of Things (IoT) environments but in decentralized fog computing networks where multiple nodes need to collaborate and coordinate their trust evaluations for policy learning [14]. This collaborative learning can be addressed by multi-agent Reinforcement Learning which allows each agent to develop its own trust assessment capabilities while considering the collective behaviour and shared knowledge of the entire network.

TABLE I. COMPARISON OF TRUST MANAGEMENT MODEL IN DECENTRALISED NETWORK

Ref	Trust Management Approach	Network Environment	Trust Metrics	Learning Mechanism	Unique Contribution/ Limitations
[18]	Zero-Trust Framework	6G Edge/Fog Networks	Dynamic Trust Scores, Anomaly Detection	AI-driven Anomaly Detection, Dynamic Policies	Strengths: Emphasizes continuous verification, is adaptable to 6G dynamics, and achieves high accuracy. Limitations: Focuses on access control, not specific MARL for fog-to-fog interaction trust.
[15]	MARL for Resource Allocation	Telecommunication, Energy, Distributed Computing	_____	Multi-Agent Reinforcement Learning	Strengths: Comprehensive survey on MARL in decentralized systems, covers various paradigms. Limitations: Not focused on trust or malicious node detection specifically, but general MARL applicability.
[11]	Subjective Logic based Trust	Fog based IoT	Direct Observations, Recommendations	Bayesian Inference, Subjective Logic	Strengths: Provides a probabilistic approach to modelling trust related uncertainty. Limitations: Lacks adaptive, reinforcement learning for dynamic behavior; may not scale to diverse attacks.
[19]	Distributed Trust and Reputation	Fog Computing, IoT	Direct and Indirect Trust, Reputation	Threshold based (Fuzzy Logic often implied)	Strengths: Considers direct and indirect trust, attempts to detect bad-mouthing Limitations: Not truly adaptive learning; detection can be slow for on-off attacks.
[20]	Combined Direct and Indirect Trust	Industrial IoT (Fog enabled)	Direct Interaction, Indirect Recommendation	Rule based (implicit, often heuristic)	Strengths: Considers both trust types for task offloading Limitations: Static rules/heuristics limited adaptability; lacks a reinforcement learning approach
[21]	Trust based Efficient Task Execution	Fog Enabled Smart Cities	Direct (Packet Delivery), Indirect (Recommendations)	Threshold based filtering	Strengths: Improves task efficiency by accounting for multiple trust parameters. Limitations: Relies on fixed thresholds and lacks autonomous learning; less robust to unknown attacks
[22]	Reputation based Trust	MANET	Direct Observation, Recommendations	Reputation scoring, defense schemes	Strengths: Focuses on robust indirect trust against recommendation attacks. Limitations: Not explicitly designed for fog or MARL; may not scale to fog-to-fog dynamism.
[23]	Survey of Security Challenges & Fog Computing	Fog Computing	_____	_____	Strengths: Comprehensive overview of attacks and vulnerabilities in fog computing. Limitations: Does not propose a solution; highlights the threats.
[24]	Federated MARL for IDS	Wireless Sensor Networks (WSNs)	Agent reliability, IDS performance	Federated Learning and Multi-Agent DRL	Strengths: Privacy-preserving, distributed learning for intrusion detection. Limitations: Specific to WSNs and IDS, not general trust management; focuses on privacy.

C. Multi-Agent Reinforcement Learning(MARL) Paradigms

MARL enables multiple agents to learn and take decisions collaboratively within a shared environment [15]. This approach follows a centralized training and decentralized execution, where agents are self-trained with global information but operate independently during execution. MARL has shown significant potential across various domains, including resource allocation and traffic control. This makes RL suitable for trust management in a decentralized fog network [16, 17]. MARL approaches can be categorized based on agent learning and coordination strategies.

Independent Learning (IL), also termed as Fully Decentralized Learning approach, treats each agent as an autonomous entity that learns independently while considering other agents as environmental factors. The Independent Deep Q-Network (I-DQN) exemplifies this approach, where individual agents implement their own DQN algorithms without direct inter-agent coordination [28].

The Centralized Training, Decentralized Execution (CTDE) paradigm represents a powerful approach that maintains decentralized decision making during execution but stabilizes learning through a global perspective during training. This method proves particularly effective in cooperative learning MARL scenarios. Foundational studies such as QMIX [26] and Value Decomposition Networks (VDN) [25] demonstrate the implementation of CTDE for cooperative discrete-action problems by decomposing or combining individual Q-values to estimate joint Q-value functions. Similarly, the Multi-Agent Deep Deterministic Policy Gradient approach (MADDPG) [27] applies to continuous action spaces utilizing a centralized critic. This provides stable gradients for decentralized agents. This approach addresses the fundamental issue of an agent's optimal action is contingent upon the behaviours of other agents, thus mitigating ongoing environmental changes that may result in unstable learning and suboptimal collective outcomes [15].

Although, prior studies have explored MARL-based network security and trust management in fog computing. But, no existing work integrates trust modelling with a hybrid I-DQN framework under centralized training and decentralized execution. Moreover, dynamic trust thresholding and evaluation under multiple black-hole attack complexities have not been jointly investigated. This study directly addresses these gaps.

III. METHODOLOGY

This study presents a comprehensive MARL simulation framework for detecting malicious nodes in the network by investigating the detection of black hole attacks in fog networks. This methodological section outlines the fog network architecture, the node characteristics, and the trust evaluation mechanisms, followed by the proposed MARL-based trust module and its implementation details. Additionally, also explains the attack model, data analysis process, and performance metrics used for evaluation.

A. Network Architecture and Topology

We consider fog network, a dynamic and decentralized network and it comprises a set of N heterogeneous fog nodes, denoted as $F=\{f_1,f_2,\cdots,f_N\}$. Every fog node $fi \in F$ have limited computational, storage, and communication capabilities. The network topology is a mesh architecture where nodes autonomously discover and maintain connections with neighboring devices within their communication range. The network topology is dynamically generated at the beginning of each simulation. The fog network with wireless nodes has the following characteristics:

Network Topology:

- Number of nodes: Variable (10-15 nodes)
- Simulation area: 1000m × 1000m
- Communication range: 200m radius per node

To get realistic packet-level communication, the Scapy library is used. The simulation leverages Scapy for realistic packet generation, creating authentic Ethernet frames with unique 48-bit MAC addresses and variable payload sizes (64-1500 bytes) while maintaining simplified network assumptions. These assumptions include perfect channel conditions with no packet corruption, deterministic 1-ms propagation delays, and no packet collisions through TDMA-like scheduling. The Key limitations include the absence of mobility modeling (static nodes), simplified routing restricted to direct neighbor communication, no cryptographic overhead or authentication delays, and the assumption of immediate trust updates and perfect state information sharing. This may not fully capture real-world network complexities, but it enables focused analysis of the core MARL learning dynamics and trust based detection mechanisms.

B. Fog Node Characteristics

Each fog node is modelled as an autonomous agent with the following distinct characteristics, which mirror those of real world devices.

1) Physical Attributes:

- Unique identifier (UUID) and MAC address for network identification.
- Battery capacity ranging from 80-100% of baseline (100 units) to simulate device heterogeneity.

2) Energy Management System:

- Multi-modal charging capabilities, including solar (0.8 units/second), station-based (3.0 units/second), and cooperative energy sharing (1.0 units/second)
- Dynamic energy consumption based on operational state: idle (0.01 units/second), communication (0.1 units/second), and trust evaluation (0.05 units/second)
- Intelligent power management with multiple operational modes: normal, power-saving (≤20% battery), critical (≤10% battery), and emergency (≤5% battery)

3) Communication Infrastructure:

- Thread-safe packet processing with separate send and receive queues
- Three-way handshake protocol implementation (SYN-ACK-CONFIRM) with timeout handling

- Packet buffering and transmission rate control (30% transmission probability per update cycle)

C. *Trust Evaluation Mechanisms*

Trust is the confidence an entity has in another entity, based on direct prior experience or feedback from other nodes in the network. Each fog node evaluates the trustworthiness of its neighbor nodes using the combined Trust (as shown in equation 4) that uses two metrics: direct trust(as shown in equation 2) and indirect trust(as shown in equation 3). The initial direct trust and indirect trust scores of each fog node are 0.7. These trust values are dynamically updated based on local observations and feedback that serve as input for the learning agent.

1) Direct Trust: Direct trust is calculated based on direct packet interactions between node *i* and node *j*. It shows how reliably node *j* has forwarded or acknowledged packets sent by node *i*, and it is computed as follows:

Packet Delivery Ratio is calculated as,

$$PDR = \frac{\text{Total Packets Received}}{\text{Total Packet Sent}} \times 100 \qquad (1)$$

The direct trust score is updated continuously using an exponentially weighted moving average (EWMA).

$$T_{ij}^{\text{direct}}(t) = \lambda \cdot T_{ij}^{\text{direct}}(t-1) + (1-\lambda) \cdot \text{PDR}_{ij}(t) \qquad (2)$$

Where,

- λ = forgetting factor ($0 < \lambda < 1$) that balances past and recent behavior.
- $T_{ij}^{direct}(t)$=Direct trust score of node i for node j at time(episode) *t*
- $PDR_{ij}\ (t)$ = Packet Delivery Ratio between nodes i and j at time t (as shown in equation 1)
- *t* = Interaction episodes/time steps

2) Indirect Trust: Indirect trust aggregation is a mechanism where a node (node *i*) evaluates the trustworthiness of another node (node *j*) based on recommendations from its trusted neighbors (nodes *k*) , rather than direct interactions. It is useful in cases where a new node enters the network and has an insufficient recent interaction history.

$$T_{ij}^{\text{indirect}} = \frac{1}{|N_i|} \sum_{k \in N_i} T_{kj}^{\text{direct}} \cdot T_{ik}^{\text{direct}} \qquad (3)$$

- $T_{ij}^{indirect}(t)$=Indirect trust score of node *i* for node *j* at time(episode) *t*
- Ni= Set of neighbor nodes trusted by node *i*
- $T_{ik}^{direct}(t)$= Direct trust of node *i* for neighbor *k*
- $T_{kj}^{direct}(t)$= Direct trust of neighbor k for node *j*

*3) Combined Trust Score:*The final trust score (as shown in equation 4) combines direct (an shown in equation 2) and indirect components (as shown in equation 3) using a weighted average (70% direct (α), 30% indirect (β)), providing a balanced trust assessment that prioritizes first hand observations.

$$Combined\ Trust = \alpha * T_{ij}^{direct}(t) + \beta * T_{ij}^{indirect}(t) \qquad (4)$$

D. *Data Collection and Analysis*

The simulation framework implements a comprehensive multi-layered data collection system designed to capture all aspects of network behaviour, security events, and learning dynamics for rigorous analysis. At the node level, each fog node maintains detailed logs capturing state information including position coordinates, battery levels, and energy consumption rates, communication metrics such as packet transmission/reception counts, and packet drop statistics, trust evolution data encompassing direct and indirect trust scores, trust variance, and interaction history, security events including malicious node detections and false positive/negative occurrences, and performance metrics including reward signals, action selections, and learning progress indicators.

The environment level data collection captures system wide metrics including dynamic network topology with neighbour relationships and connectivity matrices, attack progression tracking with initiation timing, malicious node selection patterns, and packet drop evolution, global performance indicators such as system wide detection rates, false positive/negative rates, and consensus accuracy, and training dynamics including hyperparameter optimization progress, convergence metrics, and learning stability indicators.

The logging system employs real time buffered streaming with thread safe buffers (10 entries for state data, 100 for interactions), automatic flushing mechanisms triggered by time based (2-5 seconds) and size based conditions, automatic log rotation every 10 minutes to manage storage and enable real time analysis, and cleanup mechanisms that automatically remove logs older than 3 hours to maintain storage efficiency.

Each experimental run generates a structured data hierarchy organized in scenario specific directories containing configuration files with complete experimental parameters and optimized hyperparameters, execution logs with aggregated performance metrics and episode summaries, environment logs capturing global network performance and attack progression, individual node logs with state evolution, communication events, trust progression, and security detection data, and analysis directories containing comprehensive performance metrics, temporal analysis data, comparative results, and statistical summaries.

The data collection system implements rigorous quality assurance measures, including range checking for all numerical values, consistency verification between related metrics, completeness assessment with the identification of missing data, and outlier detection using statistical methods. This comprehensive data collection framework enables detailed analysis of the MARL-based trust system's performance, providing a foundation for evaluating detection accuracy, learning convergence, and network resilience in fog computing security applications.

E. MARL Trust Module Architecture

The malicious node detection challenge is structured as a collaborative Multi-Agent Reinforcement Learning problem where fog nodes work together to enhance network security through accurate threat identification while reducing false alarms. The framework components are detailed as follows:

- Agent: Individual fog nodes function as autonomous Reinforcement Learning agents, with N concurrent agents operating within the fog network infrastructure.
- Environment: Each agent's environment encompasses thecomplete dynamic fog network, incorporating globally broadcast threshold values, updated reputation scores, behavioural patterns of all networkindirect trust indicators, current trust threshold values, and reputation scores of neighboring nodes.
- Action Space (a_n): Binary classification decisions (Malicious or Benign) made by each agent for every neighbor at each time step, with Q-values directly influencing action selection and contributing to the local malicious list.
- Reward Mechanism (r_n): Scalar rewards determined by the Central Orchestrator comparing global malicious node lists with local detection results, formulated as shown in equation 5:

$$r_{n=TP_n} - FP_n - 2FN_n \qquad (5)$$

Where *TPn* represents true positives, *FPn* denotes false positives, and *FNn* indicates false negatives. The penalty structure prioritizes security by applying stronger penalties for missed threats (-2) compared to false alarms (-1).

- Global State (S_{global}) comprehensive network snapshot for the centralized critic network, constructed by concatenating local observations from all agents, providing complete network visibility while scaling linearly with agent count.
- Joint Action (A_{joint}): Collective actions taken by all agents at each timestep, enabling the centralized critic to evaluate aggregate behavior and guide decentralized actors toward unified malicious node identification policies.

I-DQN CTDE Architecture: The proposed framework employs an Independent Deep Q-Network architecture integrated with a Centralized Training, Decentralized Execution methodology to address malicious node detection challenges in dynamic fog environments.The I-DQN implementation features individual fog nodes as independent learning agents, each equipped with a Deep Q-Network containing an input layer that processes a trust value (0.0-1.0 range), two fully connected hidden layers (128 and 64 neurons) with ReLU activation, and an output layer that provides binary classification. Target networks update every 80 steps or are triggered (e.g., topology changes, high false positive/negative rates) to maintain training stability, with hyperparameters optimized through tuning. This results in a learning rate of 0.0069, a discount factor of 0.961, and an exploration rate that decays from 1.0 to 0.026.

The CTDE paradigm addresses multi-agent non-stationarity through a two-phase approach. Firstly, computerized training enables nodes to share trust evaluations and detection results with a coordinator that aggregates decisions, distributes global feedback as rewards, and broadcasts optimized parameters. Secondly, decentralized execution enables autonomous operation, where every node utilizes trained policies for local trust evaluation, select actions through individual DQN policies, and maintain real time operation without incurring communication overhead.

This hybrid architecture delivers key advantages including scalability through decentralized execution for large scale networks, privacy preservation through local decision making, robustness ensuring continued functionality during partial failures, and learning efficiency through centralized training. This mechanism accelerates convergence and improves detection accuracy. The framework effectively balances the requirements for coordinated learning with the needs of practical distributed fog computing, enabling sophisticated trust evaluation capabilities while maintaining operational autonomy for real time security in dynamic network environments.

F. Implementation and Reproducibility

The simulation environment is developed in Python using the Gymnasium framework to provide a standardized interface for reinforcement learning. The learning components were implemented with PyTorch library and extended to support fog computing capabilities required by proposed model.

Hyperparameter optimization was performed using the Optuna framework with the Tree-structured Parzen Estimator (TPE) algorithm. A total of 80 optimization trials were conducted to tune model parameters and improve learning stability and performance systematically. The selected hyperparameters were kept fixed across all experimental scenarios. To obtain better performance, optuna framework with Tree-structured Parzen Estimator (TPE) optimization with 80 trials is used for systematic hyperparameter tuning.

To ensure both experimental determinism and reproducibility, a comprehensive seeding strategy is used across 30 multi-run simulations. A global random seed of 42 was used to initialize all experiments. In addition, all random number generators used with PyTorch and the Gymnasium environment were consistently seeded to ensure independent, verifiable trials.

Each attack scenario was executed for 500 simulation steps. The experimental design consisted of three black-hole attack scenarios, with 30 independent runs per scenario, resulting in a total of 90 simulation runs. To control computational overhead,

Each simulation run was executed with a maximum runtime limit of 2 hours on an Intel-class multi-core CPU with 8 GB memory allocation. This experimental configuration provides sufficient statistical power, achieving 80% power for robust ANOVA testing and enabling reliable estimation of 95%

confidence intervals across all performance metrics. Reported results represent the average performance across independent runs, thereby reducing the impact of stochastic variability.

G. Black Hole Attack detection

The proposed MARL trust based detection system is carefully tested against three different black hole attack scenarios in our assessment. Malicious behavior is activated at step 150 of the 500-step simulation (30% simulation progress), allowing sufficient time for normal network establishment and post attack recovery analysis to occur. The attack duration varies by scenario but typically spans 200 to 350 simulation steps.

The following parameters are common to all attack scenarios:

- Packet Dropping Approach: To target specific packet types, malicious nodes employ a hierarchical packet dropping approach. When the drop probability is fully configured, DATA packets (payload) are discarded. The SYN, ACK, and CONFIRM handshake packets, which are essential for preserving the appearance of network connectivity, are dropped at a rate 70% lower than the overall Drop Probability. The whole drop probability is used to drop unknown packets. In certain situations, this selective dropping process is crucial for achieving stealth.
- Attack Intensity: This parameter, which has a range of 0.0 to 1.0, regulates how active the harmful activity is generally and affects the Drop Probability.

Black hole attack scenarios are:

- Aggressive Attacks Scenario: As a baseline for basic detection capacity, this scenario is the most basic and, hence, the most detectable type of black hole attack. It has an attack with an intensity of 1.0, the highest level of aggression, with a drop rate of 90%. Every sort of incoming packet is discarded without distinction. The goal is to evaluate the system's baseline detection speed and accuracy under optimal (for the attacker) conditions by simulating overt network disruption, characterized by significant packet loss and throughput deterioration.
- Stealth Attack Scenario: This scenario employs a more complex strategy, striking a balance between evasive tactics and disruptive objectives to make discovery difficult. The goal is to assess the detection system's capacity to recognize increasingly complex attacks that seek to maintain a certain level of network presence while severely interfering with data flow, necessitating the use of more sophisticated detection logic. It has an attack intensity of 0.7, suggesting a conscious effort at nuance, with a 60% drop probability.
- Gradual Attack Scenario: This situation simulates the most undetectable and challenging type of black hole attack, which is marked by a continuous, slow onset danger. With a low initial drop chance of 0.3, the attack starts. After that, the drop probability gradually increases until it reaches 50%. The system's long term monitoring capabilities and resistance to stealthy, dynamic threats that frequently evade simpler detection methods designed for sudden changes are put to the test in this situation.

H. Performance Metrics

The evaluation of the proposed I-DQN-CTDE framework employs a comprehensive set of performance metrics designed to assess detection accuracy, learning efficiency, network resilience, and system robustness across multiple dimensions. The primary detection performance metrics include Detection Rate (DR), calculated as the proportion of malicious nodes correctly identified, False Positive Rate (FPR), representing the proportion of benign nodes incorrectly flagged as malicious, and False Negative Rate (FNR), indicating the proportion of malicious nodes missed by the detection system. They are calculated as per equations 6,7,8 as follows.

$$\mathrm{DR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{6}$$

$$\mathrm{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{TrueNegatives}} \tag{7}$$

$$\mathrm{FNR} = \frac{\text{False Negatives}}{\text{True Positives} + \text{False Negatives}} \tag{8}$$

Advanced detection metrics include detection time, which measures the temporal efficiency from attack initiation to the first detection, consensus accuracy, which measures the degree of agreement among detecting nodes and trust stability, measured as trust scores to evaluate the consistency of trust assessments. These comprehensive metrics demonstrate the effectiveness of detecting black hole attacks while maintaining network performance and ensuring robust operation in dynamic fog computing environments.

IV. PROPOSED TRUST-AWARE MULTI-AGENT DQN FRAMEWORK

Our proposed trust management model is designed to operate effectively in dynamic and decentralized fog networks by leveraging the adaptive capabilities of Multi-Agent Reinforcement Learning. This approach aims to combine the resilience of decentralized local decision making with the robustness of global aggregation and adaptive control.

In this model, each fog node serves as an independent smart agent. To overcome the limitations of handling large or complex state spaces inherent in traditional tabular Q-learning, an actor-critic architecture is utilized within the proposed framework. Under the CTDE paradigm, agents learn effective policies by approximating the optimal action-value function (critic) and the action selection strategy (actor) using neural networks. This enables the agents to dynamically identify malicious neighbours through continuous interaction with and observation of the environment. The overall framework is depicted in Fig 1, and its core algorithmic components are elaborated in Algorithms 1 and 2.

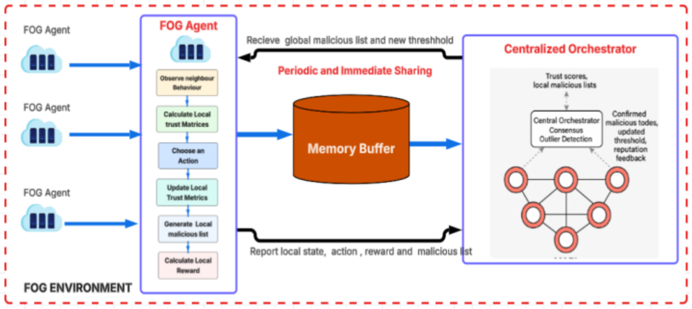
FOG Agent
FOG Agent
FOG Agent
FOG ENVIRONMENT
FOG Agent
Observe neighbour Behaviour
Calculate Local trust Matrices
Choose an Action
Update Local Trust Metrics
Generate Local malicious list
Calculate Local Reward
Recieve global malicious list and new threshhold
Periodic and Immediate Sharing
Memory Buffer
Report local state, action , reward and malicious list
Centralized Orchestrator
Trust scores, local malicious lists
Central Orchestrator Consensus Outlier Detection
Confirmed malicious todes, updated threshold, reputation feedback

Fig. 1. MARL Malicious Node Detection Architecture

A sequence diagram (Fig. 2) illustrates the operational flow of our proposed framework. This framework employs a centralized training, decentralized execution paradigm, coordinating interactions among the centralized orchestrator (O), global experience replay buffer (ERB), and individual fog Nodes (FN_n). The process initiates with system setup by the orchestrator, which initializes DRL networks and broadcasts initial policies to all fog Nodes. The initial Trust threshold for malicious node detection is 0.4. In each simulation step, decentralized fog nodes operate autonomously.

- Decision and Data Generation: FNn observes its local state, updates internal trust metrics (direct trust and indirect trust score), and selects a detection action (e.g., 'conservative' or 'aggressive') using its learned policy. Our research proposes a context-aware detection mechanism that shifts between conservative detection modes during stable periods and aggressive responses when threats emerge. The system resolves the security stability dilemma by continuously adjusting its sensitivity thresholds based on real time network assessments and attack patterns. This action guides the generation of its local malicious node list (L_n) by comparing aggregate trust matrices with the trust threshold value. Critically, FNn computes its reward (rn) by comparing the local malicious list (L_n) against the global malicious nodes broadcast by the central orchestrator and then penalizing inaccuracies.
- Reporting to Orchestrator: Each FNn reports its experience (S_n,a_n, $L_{n,}$ r_n) to the Orchestrator.
- Centralized Trust Adaptation & Orchestration: The orchestrator aggregates these reports. Periodically or in response to triggers (e.g., topology changes, high false positive/negative rates), it executes Algorithm 2. This module identifies reliable reporters, forms the network's consensus GlobalMaliciousList, dynamically adjusts the global trust threshold, and updates reporter Reputation Scores based on their consistency with the consensus list. The updated GlobalMaliciousList and global_trust_threshold are then disseminated back to the fog Nodes. This global_trust_threshhold is a dynamic threshold calculated as shown in equation 9.

$$\theta(t) = \theta_{\text{base}} + \beta \times (1 - T_{\text{avg}}(t)) \tag{9}$$

Where,

- $\theta(t)$ = Dynamic threshold at episode t, between θ_{min} and θ_{max}
- θ_{base} = Base threshold value
- $T_{avg}(t)$ =Average trust score across all nodes at episode t.

- DRL Policy Learning: The orchestrator constructs global experience tuples, which are stored in the ERB. When sufficient data accumulates, the orchestrator samples mini-batches and performs DQN-based gradient descent updates on the shared Q_actor and centralized Q_critic networks. Target networks are periodically updated. The improved Q_actor policy is then broadcast to the fog Nodes, enabling more accurate, adaptive malicious node detection in subsequent steps.
- This continuous cycle of decentralized action, centralized trust adaptation, and DRL-driven policy refinement forms the adaptive core of our framework. Our framework uses two major algorithms, as discussed below.

A. *MARL Training Algorithm for Trust Policy Learning*

The learning process for trust policy learning (Algorithm 1) addresses the dynamic nature of fog environments, enabling constant adaptability to changing network conditions and threats. Fundamentally, Algorithm 1 creates a global experience replay buffer for learning, initializes the shared neural networks (actor and critic), and defines key trust parameters, including a dynamic trust threshold and reputation scores for each fog node. The algorithm starts each episode by modeling genuine network changes, such as the addition of new nodes and the removal of old ones. The dynamic modification of reputation for rejoining nodes is a crucial component in this case.

Our system implements an intelligent node admission protocol that differentiates between returning participants and first time entrants as follows.

- Node Admission Framework: The algorithm evaluates each incoming node (FN_new) by first confirming its active and inactive state, then categorizing it based on historical presence in the ReputationScores registry.
- Returning Node Processing: Nodes with existing reputation data undergo a sophisticated decay calculation using effective_lambda based on several factors. The system begins by applying base_lambda to all nodes after they leave the network. If a node has a history of malicious behaviour, malicious_multiplier increases the decay factor to ensure its reputation drops faster. When the entire network is under threat (during high risk periods), the malicious_activity_boost factor further increases the decay

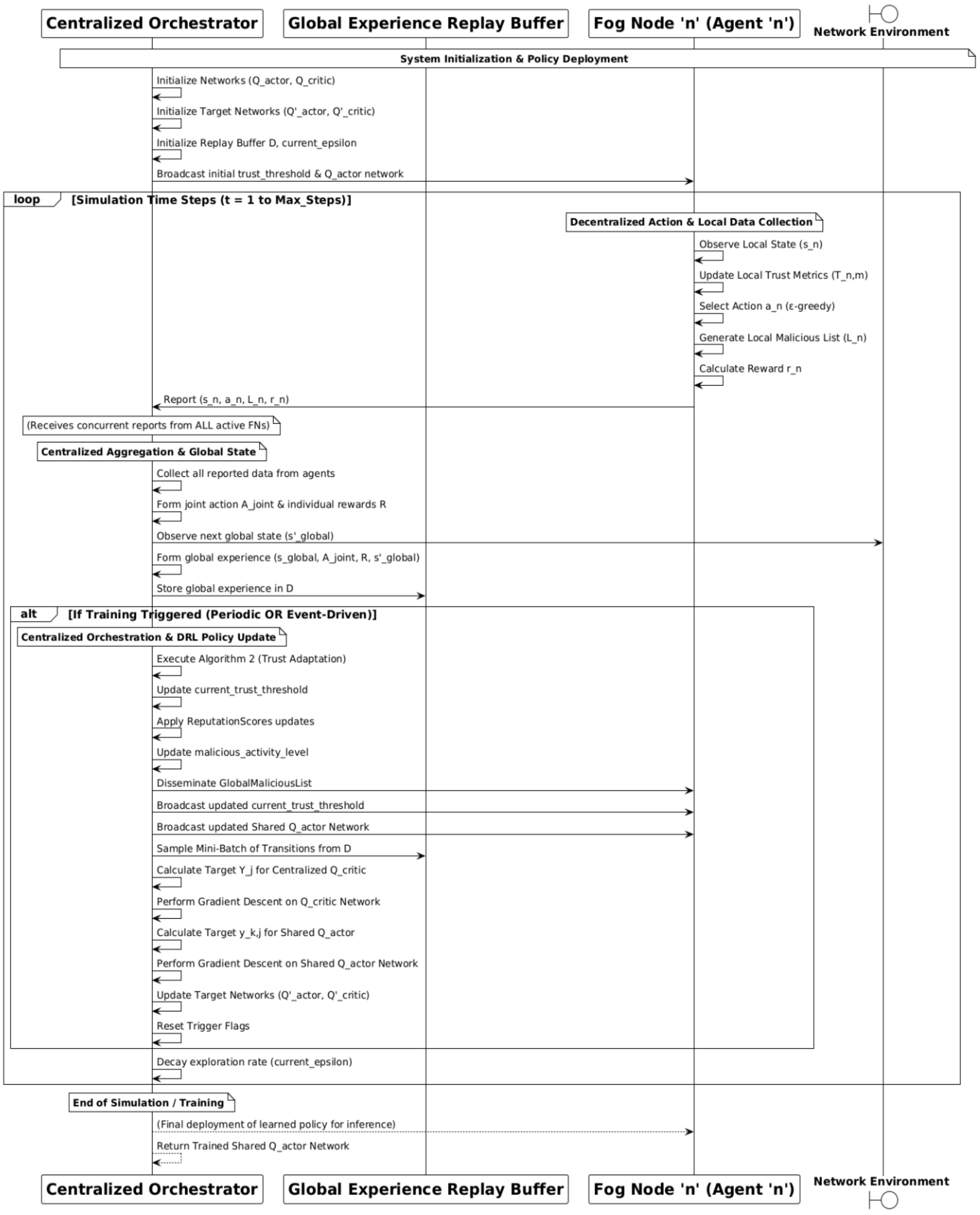
Centralized Orchestrator
Global Experience Replay Buffer
Fog Node 'n' (Agent 'n')
Network Environment
System Initialization & Policy Deployment
Initialize Networks (Q_actor, Q_critic)
Initialize Target Networks (Q'_actor, Q'_critic)
Initialize Replay Buffer D, current_epsilon
Broadcast initial trust_threshold & Q_actor network
loop [Simulation Time Steps (t = 1 to Max_Steps)]
Decentralized Action & Local Data Collection
Observe Local State (s_n)
Update Local Trust Metrics (T_n,m)
Select Action a_n (ε-greedy)
Generate Local Malicious List (L_n)
Calculate Reward r_n
Report (s_n, a_n, L_n, r_n)
(Receives concurrent reports from ALL active FNs)
Centralized Aggregation & Global State
Collect all reported data from agents
Form joint action A_joint & individual rewards R
Observe next global state (s'_global)
Form global experience (s_global, A_joint, R, s'_global)
Store global experience in D
alt [If Training Triggered (Periodic OR Event-Driven)]
Centralized Orchestration & DRL Policy Update
Execute Algorithm 2 (Trust Adaptation)
Update current_trust_threshold
Apply ReputationScores updates
Update malicious_activity_level
Disseminate GlobalMaliciousList
Broadcast updated current_trust_threshold
Broadcast updated Shared Q_actor Network
Sample Mini-Batch of Transitions from D
Calculate Target Y_j for Centralized Q_critic
Perform Gradient Descent on Q_critic Network
Calculate Target y_k,j for Shared Q_actor
Perform Gradient Descent on Shared Q_actor Network
Update Target Networks (Q'_actor, Q'_critic)
Reset Trigger Flags
Decay exploration rate (current_epsilon)
End of Simulation / Training
(Final deployment of learned policy for inference)
Return Trained Shared Q_actor Network
Centralized Orchestrator
Global Experience Replay Buffer
Fog Node 'n' (Agent 'n')
Network Environment

Fig. 2. MARL Working

rate for problematic nodes, serving as a proactive security measure. A fairness component addresses legitimate disconnections. For pre-arranged departures, effective_lambda is reset to 0.0, eliminating unwarranted reputation penalties. The final score is computed using exponential decay methodology and constrained by initial_reputation_for_new_nodes to prevent excessive score reduction.

- New Node Processing: Those Incoming Nodes having no reputation records are classified as newcomers. They receive a neutral initial reputation as new nodes, and their malicious history indicator is set to False.

- Integration Protocol: Both returning and new nodes are activated and equipped with current network trust parameters and policy configurations, enabling immediate participation within the existing security framework. This dual path methodology ensures intelligent initial trust allocation that considers both individual history and current network security conditions.

Once the network is ready, all fog nodes start observing their neighbors and managing their local states in response to network dynamics. These states contain the orchestrator displayed current_trust_threshold as well as local trust metrics for their neighbors. Based on observation, each agent decides whether to take an "aggressive" or "conservative" approach. An "aggressive" action can identify potentially malicious nodes by using a lower trust threshold, which could result in more detections but also more false positives. Conversely, a "conservative" action can use a higher threshold, which results in fewer detections but higher confidence. These decisions affect the local malicious lists (Ln) they disclose and their local trust criteria. Then the central orchestrator collects local malicious lists along with other matrices from all active nodes. These matrices will be used to update the global trust threshold and collectively generate a global malicious list as explained in Algorithm 2. Algorithm 2 is called periodically (50 time steps) or when an event such as a new node added, a node deleted, etc., occurs. Otherwise, the previous trust threshold and global malicious list will be considered. To continually improve the agents' capacity to identify malicious activity, the orchestrator lastly refreshes the comprehensive reputation scores, gathers global experiences for the replay buffer, and trains the shared actor-critic networks using this data. This flexible strategy provides training and adaptation to the dynamic fog environment.

B. Trust based Malicious Node Detection and Threshold Adaptation by Central Orchestrator

Algorithm 2 describes the process of collaborative identification of malicious nodes in the network and calculation of dynamic trust threshold in the framework. The centralized orchestrator governs a robust approach that incorporates four critical stages.

- The coordinator's first process begins with global state aggregation. It receives local malicious node reports (L_n) and local trust evaluations (T_{scores}) from all active functioning nodes. The coordinator conducts an initial screening phase to identify trustworthy reporters, ensuring the integrity of the consensus that follows. A node is classified as a ReliableReporter only if its local report is sufficiently consistent with the previous global state and its reputation score meets a minimum threshold. This mitigates the risk of low quality or adversarial reports skewing the global decision.
- The second stage is crucial for establishing agreement across the network. In this stage, reports from the identified ReliableReporters are aggregated to form the global malicious nodes list $(M_{current})$. Specifically, a node is designated as malicious if it receives votes from more than two-thirds of the reliable reporters, thereby effectively implementing a simple majority voting protocol. By resolving disagreements between divergent local detection findings, this centralized consensus method creates a single, reliable source of truth for the network as a whole.
- In this stage the coordinator maintains the model's adaptive capabilities by managing the dynamic trust threshold. It utilizes the validated trust scores (T_{scores}) to calculate a new detection threshold (new_trust_threshhold) based on statistical analysis and a smoothing factor derived from the Exponential Moving Average (EMA). This continuous adaptation is vital, as it ensures the detection mechanism remains sensitive to evolving, non-stationary malicious behaviour patterns within the multi-agent environment.
- The final stage is to update the reporter's reputation score.

The centralized coordination is essential between multi-agents for critical network decisions during training phase but may lead to single point of failure but decentralized execution offers temporary resilience by maintaining functionality even if the coordinator is unavailable.

C. Theoretical Scalability Analysis

The decentralized execution paradigm of the proposed framework offers inherent scalability benefits, enabling deployment in large scale fog computing networks. While our current empirical validation focuses on networks with 10-15 nodes, theoretical analysis demonstrates the framework's potential for deployments of significantly larger scale. This section presents a comprehensive scalability analysis addressing computational complexity and communication overhead across varying network sizes.

1) Computational Complexity Analysis: The computational overhead can be analysed through three critical dimensions: per node operations, centralized training coordination, and overall system scalability. In our model, each fog node maintains an independent Deep Q-Network with an input dimension of d = *2*, representing direct and indirect trust values. The time complexity per decision operation can be expressed as shown in equation 10.

$$O(d \times h1 + h1 \times h2 + h2 \times |A|) \qquad (10)$$

where, $h1=128$, $h2=64$ represent the hidden layer neuron counts
$|A|=2$ represents the binary action space

Algorithm 1 Trust-Aware Multi-Agent DQN Training

```
Input:
  • $\mathcal{F}_{initial}$: Set of initial Fog Nodes
  • $\gamma$: Discount Factor; $\alpha_{actor}, \alpha_{critic}$: Learning Rates
  • periodic_training_interval: Time steps/duration for periodic Algorithm 2 call        ▷ NEW Input
  • event_driven_threshold_maliciousness: Threshold for event-driven trigger        ▷ NEW Input
  • $C_{actor\_update}, C_{critic\_update}$: Target Network Update Frequencies
  • $\epsilon_{\text{initial}}$: Initial Exploration Rate decay_rate: Exploration Decay Factor
  • $D_{\text{min\_size}}$: Minimum Replay Buffer size for training
  • $C$: Max capacity of Global Experience Replay Buffer $D$
  • $\Delta_{\text{aggressive}}$: Trust adjustment for ”aggressive” action
  • $\Delta_{\text{conservative}}$: Trust adjustment for ”conservative” action
  • initial_reputation_for_new_nodes: Default reputation for brand new nodes
  • $\lambda_{base}$: Base decay constant for node reputation        ▷ For rejoining nodes
  • $\lambda_{malicious\_multiplier}$: Multiplier for $\lambda$ if node has malicious history        ▷ Differentiated Decay
  • $\lambda_{malicious\_activity\_boost}$: Additional $\lambda$ for high network maliciousness        ▷ Contextual Decay
  • network_maliciousness_threshold: Threshold for current_malicious_activity_level
  • $\alpha_{recovery}$: Learning rate for reputation recovery of active nodes        ▷ Explicit recovery term
Output: Shared $Q_{actor}$ network $\theta_{actor}$
Initialize: (on Centralized Orchestrator)
  • Initialize Shared $Q_{actor}$ network with random weights $\theta_{actor}$
  • Initialize Shared Target $Q'_{actor}$ network with weights $\theta'_{actor} \leftarrow \theta_{actor}$
  • Initialize Centralized $Q_{critic}$ network with random weights $\phi_{critic}$
  • Initialize Centralized Target $Q'_{critic}$ network with weights $\phi'_{critic} \leftarrow \phi_{critic}$
  • Initialize Global Experience Replay Buffer $D$
  • Initialize ReputationScores = {} (a dictionary mapping $FN_{id}$ to ‘’score’, ’has_malicious_history’‘)
  • Initialize DepartureTimes = {} (a dictionary to store last departure time for nodes)
  • Initialize TrustHistory
  • Initialize current_trust_threshold with 0.5.
  • Initialize ActiveFNs = $\emptyset$
  • Initialize current_malicious_activity_level = 0.0        ▷ Tracks network-wide maliciousness
  • Initialize planned_departures_log = {} (a dictionary: $FN_{id} \rightarrow$ boolean for planned departure)
  • Initialize GlobalMaliciousList = $\emptyset$        ▷ NEW: Default for initial state
  • Initialize Algorithm2_ReputationUpdates = {}        ▷ NEW: Default for initial state
  • Initialize last_algorithm2_run_time = 0        ▷ NEW: To track periodic trigger
  • Initialize next_periodic_training_time = periodic_training_interval        ▷ NEW: First periodic trigger
1: for each $FN_i$ in $\mathcal{F}_{initial}$ do
2:     ‘ReputationScores[FN$_i$] = {′score′ : initial_reputation_for_new_nodes,′ has_malicious_history′ : False}‘.
3:     Add $FN_i$ to ‘ActiveFNs‘.
4: $current\_\epsilon = \epsilon_{\text{initial}}$

5: for each episode do
6:     Reset Environment: Initialize fog network, agent states, malicious node distribution.
7:     ‘current_simulation_time‘ ← Get current overall simulation time.
8:         ▷ Simulate/Obtain network changes and planned departures for this episode
9:     ‘newly_joined_nodes‘, ‘departed_nodes‘ ← Simulate network changes.
10:
11:    for each $FN_{departed}$ in ‘departed_nodes‘ do
12:        if $FN_{departed}$ in ‘ActiveFNs‘ then
13:            Remove $FN_{departed}$ from ‘ActiveFNs‘.
14:            ‘DepartureTimes[FN$_{departed}$] = $current\_simulation\_time$‘.
               ▷ Initial call to Algorithm 2 for episode start
15:    Collect all $L_n$ from $FN_n$ in ActiveFNs.        ▷ Will be empty initially or based on reset
16:    (GlobalMaliciousList, new_trust_threshold, Algorithm2_ReputationUpdates) ←
           Execute Algorithm 2($L_{ns}$, GroundTruthMaliciousSet, ReputationScores, current_trust_threshold, TrustHistory,
       ActiveFNs).
17:    GlobalMaliciousList_last = GlobalMaliciousList        ▷ Store for next
       timesteps
18:    current_trust_threshold ← new_trust_threshold
19:    Algorithm2_ReputationUpdates_last = Algorithm2_ReputationUpdates        ▷ Store for next
       timesteps
20:    last_algorithm2_run_time = current_simulation_time
21:    next_periodic_training_time = current_simulation_time + periodic_training_interval
```

```
22: for each $FN_{\text{new}}$ in newly_joined_nodes do
23:     if $FN_{\text{new}}$ not in ActiveFNs then
24:         if $FN_{\text{new}}$ in ReputationScores then                                   ▷ Rejoining node
25:             time_since_departure = current_simulation_time - DepartureTimes.get(FN_new,
                0)
26:             previous_reputation = ReputationScores[FN_new]['score']
27:             effective_lambda = $\lambda_{\text{base}}$
28:             if ReputationScores[FN_new]['has_malicious_history'] is True then
29:                 effective_lambda = effective_lambda $\cdot\lambda_{\text{malicious\_multiplier}}$.
30:             if current_malicious_activity_level > network_maliciousness_threshold then
31:                 effective_lambda = effective_lambda + $\lambda_{\text{malicious\_activity\_boost}}$.
32:             if planned_departures_log.get(FN_new, False) is True then
33:                 effective_lambda = 0.0.
34:             ReputationScores[FN_new]['score'] = $\max(previous\_reputation \cdot$
                $e^{-\text{effective\_lambda}\cdot\text{time\_since\_departure}}, initial\_reputation\_for\_new\_nodes)$
35:             Remove $FN_{\text{new}}$ from DepartureTimes.
36:             Remove $FN_{\text{new}}$ from planned_departures_log if present.
37:         else                                                                            ▷ Brand-new node
38:             ReputationScores[FN_new] = {'score': initial_reputation_for_new_nodes,
                'has_malicious_history': False}.
39:         Add $FN_{\text{new}}$ to ActiveFNs.
40:         $FN_{\text{new}}$ receives current_trust_threshold and shared $\theta_{\text{actor}}$.

41: for each time step $t$ in the current episode do              ▷ Decentralized Action Selection (Execution)
42:     for each agent $FN_n \in$ ActiveFNs do
43:     Observe local state $s_n$
        (includes direct/indirect trust metrics for active neighbors, current_trust_threshold).
44:     Update local trust scores $T_{n,m}$ (e.g., based on recent interactions with $FN_m \in$ ActiveFNs, trust
        decay).
45:     With probability $current\_\epsilon$: select random overall action $a_n \in \{0, 1\}$.
46:     Else: select $a_n = \arg\max_{a'} Q_{\text{actor}}(s_n, a'; \theta_{\text{actor}})$.
47:     Add $a_n$ to $A_{\text{joint}}$ (corresponding to $FN_n$).
48:     Initialize $L_n = \emptyset$.
49:     for each active neighbor $FN_m$ of $FN_n$ where $FN_m \in$ ActiveFNs do
50:         Let $T_{n,m}$ be $FN_n$'s trust score for $FN_m$.
51:         if $a_n = 1$ then                                              ▷ "Aggressive" identification
52:             $Threshold_{\text{local}}$ = current_trust_threshold $- \Delta_{\text{aggressive}}$
53:         else($a_n = 0$)                                                ▷ "Conservative" identification
54:             $Threshold_{\text{local}}$ = current_trust_threshold $+ \Delta_{\text{conservative}}$
55:         if $T_{n,m} < Threshold_{\text{local}}$ then
56:             Add $FN_m$ to $L_n$.
57: for each time step do
    Central Orchestrator Role
58:     Collect all $L_n$ from $FN_n$ in ActiveFNs.
59:     current_simulation_time ← Get current overall simulation time.
                                                                       ▷ Check for periodic trigger
60:     is_periodic_due = (current_simulation_time ≥ next_periodic_training_time)
                                                                       ▷ Check for event-driven trigger
61:     is_event_driven_due = (current_malicious_activity_level ≥ threshold AND changed significantly)
62:     if is_periodic_due OR is_event_driven_due then
63:         (GlobalMaliciousList_temp, new_trust_threshold_temp, Algorithm2_ReputationUpdates_temp) ←
            Execute Algorithm 2(...).
64:         GlobalMaliciousList = GlobalMaliciousList_temp
65:         current_trust_threshold ← new_trust_threshold_temp
66:         Algorithm2_ReputationUpdates = Algorithm2_ReputationUpdates_temp
67:         GlobalMaliciousList_last = GlobalMaliciousList_temp
68:         Algorithm2_ReputationUpdates_last = Algorithm2_ReputationUpdates_temp
69:         last_algorithm2_run_time = current_simulation_time
70:         if is_periodic_due then
71:             next_periodic_training_time = current_simulation_time + periodic_training_interval
72:     else
73:         GlobalMaliciousList = GlobalMaliciousList_last
74:         Algorithm2_ReputationUpdates = Algorithm2_ReputationUpdates_last
75:     Disseminate GlobalMaliciousList to all $FN_n$ in ActiveFNs.
```

```
Reward Calculation (Individual Nodes, collected by Orchestrator)
for each $FN_n$ in ActiveFNs do
    $r_n \leftarrow$ CalculateReward(...)
Global Data Collection and Storage
    Collect and store global state, actions, rewards, next states into buffer $D$.
Update Reputation Scores
for each $FN_{id}$ in ActiveFNs do
    if $FN_{id}$ in Algorithm2_ReputationUpdates_last AND not empty then
        Update reputation score with score_change
    if $FN_{id}$ in GroundTruthMaliciousSet then
        Mark as malicious history
    Update malicious activity level
if $|D| \geq D_{min\_size}$ then
    Sample mini-batch
    Update critic and actor using targets $Y_j$, $y_{k,j}$
if $t \pmod{C_{actor\_update}} == 0$ then
    Update actor target
if $t \pmod{C_{critic\_update}} == 0$ then
    Update critic target
Update exploration rate
```

This formulation yields constant operations per decision, independent of network size n. This constant computational requirement per node represents a fundamental scalability advantage, as additional nodes do not increase individual processing load. The computing load on the centralised coordinator grows almost linearly with the network's size. The coordinator uses $O(N \times d)$ operations for global state aggregation and $O(N \times log(N))$ operations for consensus building by creating a sorted malicious node list during the training phase. When these elements are combined, the overall training overhead is $O(N \times log(N))$, exhibiting effective scaling properties. During the training phase, each node transmits 16 bytes of trust data, along with detection information, every training interval, resulting in minimal communication overhead.

Overall scalability projections are promising where the per node computation effort remains constant, indicating favourable $O(k)$ complexity. However, the total training overhead increases significantly across the tested network range

2) Communication Overhead Growth Analysis: Communication overhead in the proposed system follows known linear scaling characteristics, principally driven by the training phase coordination needs. Total_Communication = $N \times M \times S$ is the theoretical model for total communication, where N is the number of nodes, M is the number of messages in each training interval, and S is the message size in bytes.

A critical advantage of the proposed architecture is the complete elimination of communication overhead during execution. Unlike fully centralised approaches that require constant communication between coordinators, our decentralised execution paradigm enables autonomous node operation with zero communication cost. This fundamental design choice separates learning coordination from runtime decision making, ensuring that network growth does not impact critical path performance during execution.

V. RESULTS AND DISCUSSION

This section evaluates the performance of proposed model. A comprehensive comparison against a diverse set of baselines and a depth ablation study is also discussed in this section to validate the contribution of the model's components.

A. Proposed Model Performance

1) Packet Delivery Ratio Performance: Our system shows significant performance improvement with the analysis of Packet Delivery Ratio (PDR) across all attack scenarios as shown in Fig. 3. Under normal conditions, the network maintains an excellent PDR of 95-97%, demonstrating baseline network efficiency. During aggressive attacks, the PDR experiences an immediate degradation to 83-84% but after detection recovers to 89-90%. In stealth attacks, the PDR drops up to 78-79% during the undetected phase, and after detection, it is set to 85-86%. The most challenging scenario is gradual attacks, where there is severe degradation of PDR up to 66-67% but our system successfully recovers it to 82-83% after detection, which is a remarkable improvement. This performance demonstrates the system's ability to adaptively respond to varying attack intensities and restore network functionality effectively.

2) Detection Rate and False Rate Analysis: Fig 4(a), Fig 4(b), and Fig 4(c) show the effectiveness of the model by analyzing the DR, FNR, and FPR in all attack scenarios. The model performs exceptionally at the time of aggressive attacks with the average detection rate of 92.0% along with an exceptional low FNR of 8.3% and FPR of 5.0%.

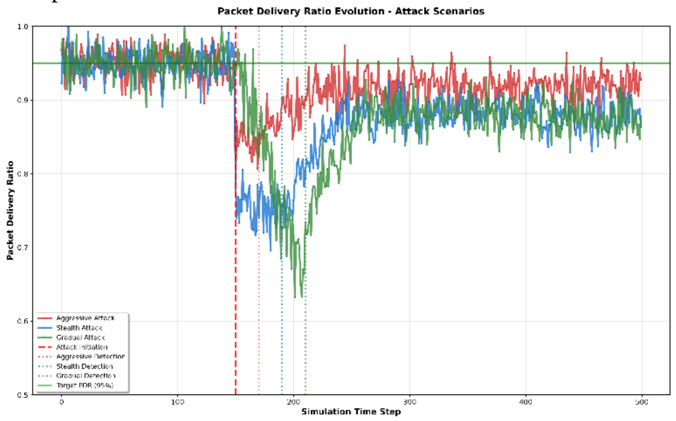

Fig. 3. Packet Delivery Ratio Evolution Under Aggressive, Stealth, and Gradual Attacks

This suggests that false alarms are accurately identified and minimized. On the other hand, the model performs good in case of stealth attack and acceptable in gradual attacks. The DR falls to 78.1% for the stealth attack, but the FPR (12.0%) and FNR (22.0%) rise. In a gradual attack, the model is getting the greatest FNR (31.8%) and FPR (17.6%), and a DR of 67.7%. These results indicate that our model maintains a favourable trade-off between sensitivity and specificity, with higher detection rates correlating with lower false rates as attack intensity increases.

Algorithm 2 Trust-Based Malicious Node Detection and Threshold Adaptation

```
Input:
  $\mathcal{T}_{\text{scores}}$: Set of current trust scores $\{T_{n,m}\}$ reported by all active $FN_n$.
  $\mathcal{L}_{\text{all}}$: Set of all local malicious lists $\{L_n\}$ reported by active $FN_n$.
  $M_{\text{previous}}$: Global malicious list from the previous iteration.
  $current\_trust\_threshold$: The currently active global trust threshold.
  ReputationScores: Global dictionary of reputation scores for all $FN_{\text{id}}$.
  $\rho$: Acceptance ratio for reliable reporter identification (e.g., 0.8).
  $\beta$: Smoothing factor for Exponential Moving Average (EMA) (e.g., 0.1).
  outlier_threshold: Threshold for outlier detection in trust scores (e.g., 3 standard deviations).
  $\alpha_R$: Reputation update rate (e.g., 0.01).            ▷ Used in Algorithm 1, implicitly updated here
Output:
  $M_{\text{current}}$: Updated Global malicious list.
  $new\_trust\_threshold$: Adapted global trust threshold.
  Algorithm2_ReputationUpdates: Dictionary of reputation changes suggested by Algorithm 2.
Initialize:
  $M_{\text{current}} \leftarrow \emptyset$.
  ReliableReporters $\leftarrow \emptyset$.
  NodeVotes $\leftarrow$ Dictionary of counts, initialized to 0 for all nodes.
  Algorithm2_ReputationUpdates $\leftarrow$ Empty Dictionary.
                                                        ▷ Step 1: Identify Reliable Reporters
for each $FN_n \in$ ActiveFNs do
    $N_{\text{consistent\_with\_Mprev}} \leftarrow 0$.
    $N_{\text{total\_in\_Ln}} \leftarrow |L_n|$.
    for each node $m$ in $L_n$ do
        if $m \in M_{\text{previous}}$ then $N_{\text{consistent\_with\_Mprev}} \leftarrow N_{\text{consistent\_with\_Mprev}} + 1$.
    if $N_{\text{total\_in\_Ln}} > 0$ AND $\frac{N_{\text{consistent\_with\_Mprev}}}{N_{\text{total\_in\_Ln}}} \geq \rho$ AND ReputationScores$[FN_n]['score'] \geq$
    ReputationThresholdForReliableReporter then
        Add $FN_n$ to ReliableReporters.
                                         ▷ Step 2: Consensus-Based Global Malicious List Formation
if ReliableReporters is empty then
    $M_{\text{current}} \leftarrow M_{\text{previous}}$.          ▷ Default to previous list if no reliable reporters
else
    for each $FN_n \in$ ReliableReporters do
        for each node $m$ in $L_n$ do NodeVotes$[m] \leftarrow$ NodeVotes$[m] + 1$.
    for each node $m$ in NodeVotes.keys() do
        if NodeVotes$[m] \geq \lceil 2 * \text{num\_reliable\_reporters}/3 \rceil$ then       ▷ Simple majority
        voteAdd $m$ to $M_{\text{current}}$.
                                                ▷ Step 3: Adaptive Trust Threshold Calculation
Collect all valid trust scores from $\mathcal{T}_{\text{scores}}$ into a list aggregated_trust_values.
if aggregated_trust_values is empty then
    $new\_trust\_threshold \leftarrow current\_trust\_threshold$. Else
    $\mu_{\text{trust}} \leftarrow$ mean(aggregated_trust_values).
    $\sigma_{\text{trust}} \leftarrow$ std_dev(aggregated_trust_values).
    dynamic_threshold_component $\leftarrow \mu_{\text{trust}} -$ outlier_threshold $\cdot \sigma_{\text{trust}}$.
    $new\_trust\_threshold \leftarrow (1 - \beta) \cdot current\_trust\_threshold + \beta \cdot$ dynamic_threshold_component.
                                                ▷ Step 4: Update Reporter Reputation Scores
for each $FN_{\text{reporter}} \in$ ActiveFNs do
    $TruePositiveCount \leftarrow 0$.
    $FalsePositiveCount \leftarrow 0$.
    $FalseNegativeCount \leftarrow 0$.
    $TotalReported \leftarrow |L_{\text{reporter}}|$.
    $TotalActualMalicious \leftarrow |M_{\text{current}}|$.
    for each node $m$ in $L_{\text{reporter}}$ do
        if $m \in M_{\text{current}}$ then $TruePositiveCount \leftarrow TruePositiveCount + 1$. Else $FalsePositiveCount \leftarrow$
        $FalsePositiveCount + 1$.
    for each node $m$ in $M_{\text{current}}$ do
        if $m \notin L_{\text{reporter}}$ then $FalseNegativeCount \leftarrow FalseNegativeCount + 1$.
    $accuracy_{\text{reporter}} \leftarrow 0$.
    if $TruePositiveCount + FalsePositiveCount + FalseNegativeCount > 0$ then ▷ Avoid division by
    zero
        $accuracy_{\text{reporter}} \leftarrow \frac{TruePositiveCount}{TruePositiveCount+FalsePositiveCount+FalseNegativeCount}$.
    $reputation\_change \leftarrow 0$.
    if $accuracy_{\text{reporter}} \geq$ ReputationAccuracyThreshold then
        $reputation\_change \leftarrow +\alpha_R \cdot (accuracy_{\text{reporter}} -$ ReputationAccuracyThreshold). Else
        $reputation\_change \leftarrow -\alpha_R \cdot$ (ReputationAccuracyThreshold $- accuracy_{\text{reporter}}$).
    Algorithm2_ReputationUpdates$[FN_{\text{reporter}}] \leftarrow$ {'score_change': $reputation\_change$}.
return $M_{\text{current}}$, $new\_trust\_threshold$, Algorithm2_ReputationUpdates.
```

3) Comprehensive Stress Testing against Black Hole Attack Scenarios: To evaluate the robustness of proposed trust model under diverse adversarial conditions, we conducted comprehensive stress testing across three attack categories in fog computing environment that consists of 5 nodes. Our experimental design and attack patterns are shown in Table II to assess the framework's adaptability and resilience against sophisticated threats.

TABLE II. BLACK HOLE ATTACK SCENARIO SPECIFICATION

Attack Scenario	Attack Intensity	Drop Probability	Attack Pattern	Detection Challenge
Aggressive attack	1.0 (Highest)	90%	Immediate, indiscriminate	Baseline detection
Stealth attack	0.7 (Moderate)	60%	Selective, evasive	Complex detection
Gradual attack	0.3-0.5 (Progressive)	30% to 50%	Slow-onset, continuous	Long-term monitoring

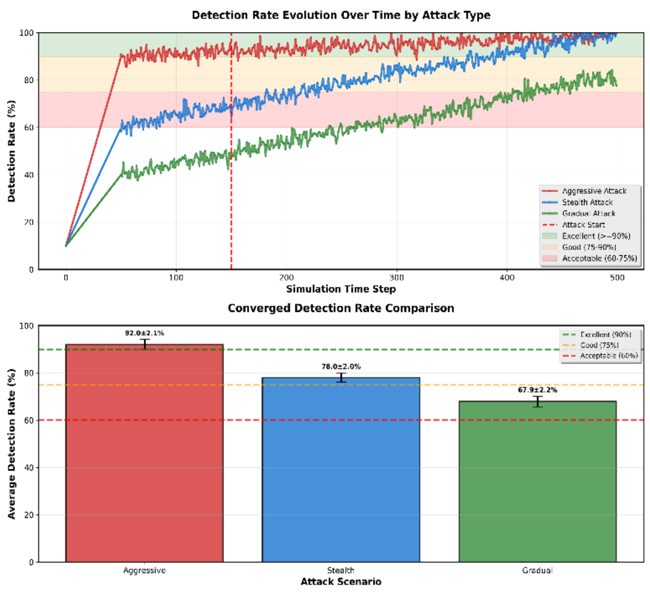

Fig. 4. (a) Detection Rate Analysis

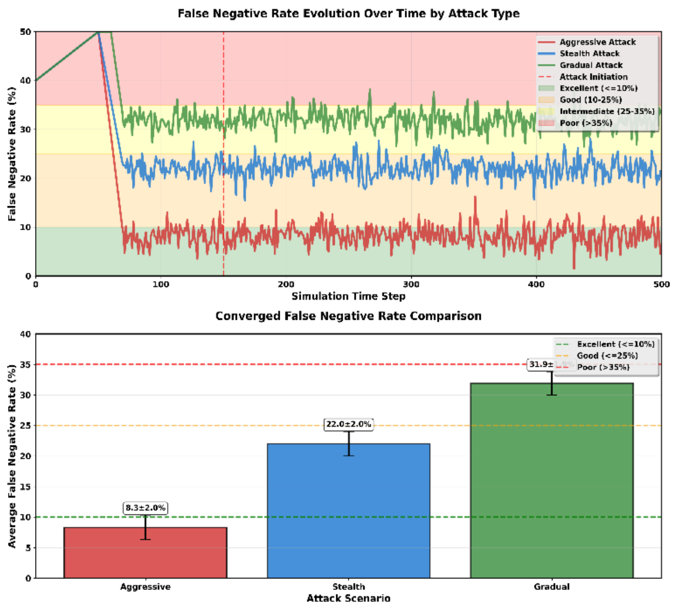

Fig. 4. (b) False Negative Rate Analysis

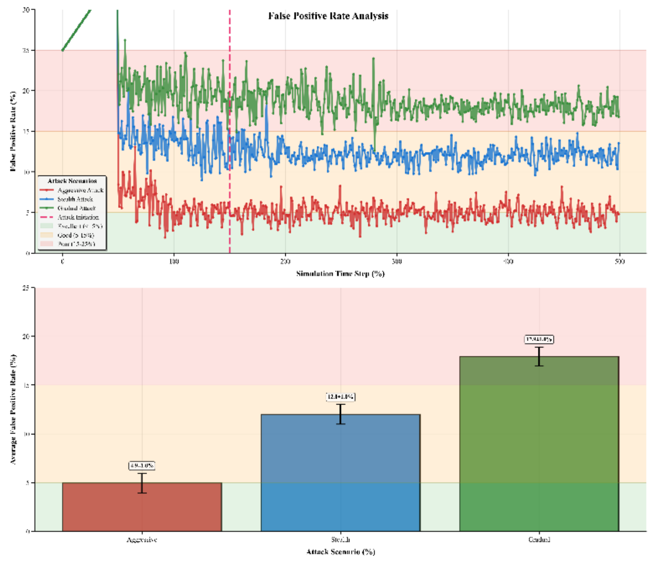

Fig. 4. (c) False Positive Rate Analysis

The aggressive attack scenario represents the most basic and detectable type of black hole attack, serving as a baseline for basic detection capacity. The stealth attack scenario employs a more complex strategy, balancing evasive tactics with disruptive objectives to make discovery difficult. The gradual attack scenario simulates the most difficult and undetectable type of black hole attack, marked by a continuous, slow onset threat that tests the system's long term monitoring capabilities.

Our multi-run statistical analysis in Table III revealed distinct performance characteristics across attack scenarios. The aggressive attack achieved the highest detection rate of 92.0% ± 2.1%, demonstrating the framework's effectiveness against basic attack patterns The stealth attack scenario achieved a detection rate of 78.0% ± 2.0%, reflecting the increased complexity of detecting sophisticated attacks. The gradual attack achieved a detection rate of 67.9% ± 2.2%, demonstrating the framework's capability to identify slow onset, continuous threats despite inherent detection challenges.

TABLE III. MULTI-RUN PERFORMANCE STATISTICS

Attack Scenario	Detection Rate	False Positive Rate	False Negative Rate	Packet Delivery Ratio
Aggressive attack	92.0± 2.1%	4.9 ± 1.0%	8.3 ± 2.0%	0.92-0.96
Stealth attack	78.0 ± 2.0%	12.0 ± 1.0%	22.0 ± 2.0%	0.88-0.92
Gradual attack	67.9 ± 2.2%	17.9 ± 1.0%	31.9 ± 1.9%	0.85-0.90

The FPR analysis reveals that aggressive attacks demonstrate the lowest FPR of 4.9% ± 1.0%, indicating high precision in detection decisions. Stealth attacks show a moderate false positive rate of 12.0% ± 1.0%, reflecting the increased complexity of distinguishing between legitimate and malicious behaviour. Gradual attacks exhibit the highest FPR of 17.9% ± 1.0%, indicating challenges in maintaining precision when detecting slowly evolving threats.

We compared the proposed model against established baselines using paired statistical tests to isolate and confirm the significance of our performance, as mentioned in Table 4. The results of the paired t-test (for mean comparison) and the Wilcoxon signed-rank test (non-parametric comparison) were entirely consistent, both showing highly significant superiority for the I-DQN CTDE framework over all baselines ($p < 0.001$). The large Cohen's d values (ranging from 1.41 to 3.03) indicate the practical significance of our framework's advantage. Additionally, one-way Analysis of Variance (ANOVA) revealed significant differences across attack scenarios for all key metrics. Detection rates showed highly significant differences ($F = 15.847$, $p < 0.001$), FPR demonstrated significant variations ($F = 8.234$, $p = 0.002$), and FNR exhibited highly significant differences ($F = 12.456$, $p < 0.001$), indicating that attack complexity significantly impacts all aspects of detection performance.

TABLE IV BASELINE MODEL COMPARISON WITH STATISTICAL SIGNIFICANCE

Method	ΔDR vs. Proposed	t-stat	p-value	Cohen's d
IDQN- NT	-5.8 %	8.45	< 0.001***	1.41
MAAC-NT	-8.1 %	9.87	< 0.001***	1.69
EMAT	-17.9 %	12.34	< 0.001***	2.88
HTM	-20.6 %	13.21	< 0.001***	3.03

The comprehensive stress testing validates the proposed framework's effectiveness across different black hole attack complexities, demonstrating excellent performance against aggressive attacks, good performance against stealth attacks, and acceptable performance against gradual attacks. Unlike existing approaches that rely on static detection mechanisms, our trust based MARL framework employs adaptive learning to handle diverse attack patterns, providing a novel solution to malicious node detection in fog computing environments. The statistical analysis confirms significant performance differences across attack types, with the framework's detection capability appropriately scaling with attack complexity. These results establish the framework's readiness for deployment in real world fog computing environments where diverse black hole attack patterns are expected.

4) Trust Score Evolution and Dynamic Threshold Adaptation: In Fig 5, all legitimate nodes demonstrate consistent improvement in their trust score, converging toward 1.0 across all scenarios, while malicious nodes exhibit distinct degradation patterns. In aggressive attacks, malicious nodes experience a rapid decline in their trust score, dropping below the 0.5 threshold within 20-30 time steps and converging toward 0.0, enabling quick detection. Stealth attacks exhibit a more gradual degradation of trust, with malicious nodes crossing the threshold between 50 and 100time steps, reflecting the system's ability to adapt to subtle attack patterns. Gradual attacks present the most challenging scenario, with malicious nodes maintaining higher trust scores for longer periods, crossing the threshold around 150–200 time steps, yet still achieving eventual detection. The dynamic threshold mechanism in Fig 6 demonstrates intelligent adaptation, with thresholds decreasing during attack phases to enhance sensitivity and increasing post detection to reduce false positives, achieving an optimal balance between detection effectiveness and system stability.

B. Comparative Performance against Baseline

As research on trust based security in fog computing is in its early stages, there are few reinforcement learning based models. There is no trust model based on I-DQN and CTDE. As a result, the results are compared with the baseline model. A comprehensive and diverse set of baseline methods is essential for establishing the validity and significance of any proposed machine learning approach, particularly in network security, where multiple competing paradigms exist. In trust based malicious node detection in fog networks, a robust baseline comparison must include both trust-centric approaches that demonstrate the value of incorporating trust mechanisms and MARL-based approaches that validate the effectiveness of our specific architectural choices.

1) Trust Only Baselines

- Heuristic Trust Model (HTM): This baseline implements a traditional rule based trust management system that calculates trust scores using predefined heuristics without learning capability. The HTM employs a simple weighted average of direct trust (based on packet delivery success rates) and indirect trust (based on neighbor recommendations) with fixed weights of 0.7 and 0.3 respectively. Trust scores are updated using a static learning rate of 0.1, and malicious nodes are identified using a fixed threshold of 0.5. The HTM represents conventional trust based security approaches that rely on predetermined rules and static parameters, providing a baseline to demonstrate the value of dynamic learning capabilities in trust management.
- Exponential Moving Average Trust (EMAT): This baseline extends the HTM by incorporating temporal smoothing through exponential moving averages for trust score updates. The EMAT employs a decay factor of 0.9 to weight recent interactions more heavily than historical data, using the equation 11:

  $$\text{Trust}(t) = \alpha \times \text{Current\_Trust} + (1-\alpha) \times \text{Trust}(t-1) \tag{11}$$

  where $\alpha = 0.1$.

  This approach represents a more sophisticated version of traditional trust management, accounting for temporal dynamics, but lacks the adaptive learning capabilities of MARL systems.

2) MARL Only Baselines

- Independent DQN without Trust (IDQN-NT): This baseline implements a standard Independent Deep Q-

Network approach where each node learns to detect malicious neighbors based solely on network metrics (packet drop rates, response times, communication patterns) without any explicit trust mechanism. The IDQN-NT utilizes the same neural network architecture as our proposed method, but replaces trust based state representation with raw network metrics. It employs a 4-dimensional state space comprising packet drop rate, average response time, communication frequency, and neighbour count.

- Multi-Agent Actor-Critic without Trust (MAAC-NT): This baseline implements a multi-agent actor-critic framework where agents learn coordinated policies for malicious node detection without trust based state representation. This model is similar to our CTDE approach, but uses actor-critic architecture instead of DQN focuses on network level metrics rather than trust values.

3) Comparative Performance Against Baseline

Table 5 shows the comprehensive evaluation across all baseline methods, where the proposed model demonstrates superior performance. Our proposed approach uses trust based behavioural analysis to effectively filter network noise and capture attack patterns over extended periods. Unlike

alternative methods that rely on raw network measurements or isolated learning processes, our framework integrates centralized coordination with explicit trust modelling to achieve superior accuracy in the detection of malicious nodes.

TABLE V BASELINE MODEL COMPARISON

Method	Category	Detection (%)	FPR (%)	FNR (%)	F1-Score	Trust stability
I-DQN CTDE (proposed)	Trust MARL	87.3 ± 3.2	6.8 ± 2.1	12.7 ± 3.2	0.85 ± 0.03	0.82 ± 0.04
IDQN-NT	MARL only	81.5 ± 4.1	9.2 ± 2.8	18.5 ± 4.1	0.78 ± 0.04	0.71 ± 0.06
MAAC-NT	MARL only	79.2 ± 4.8	11.4 ± 3.2	20.8 ± 4.8	0.75 ± 0.05	0.67 ± 0.07
EMAT	Trust only	69.4 ± 6.2	23.1 ± 4.8	30.6 ± 6.2	0.64 ± 0.07	0.57 ± 0.07
HTM	Trust only	66.7 ± 6.8	26.3 ± 5.1	33.3 ± 6.8	0.61 ± 0.08	0.52 ± 0.12

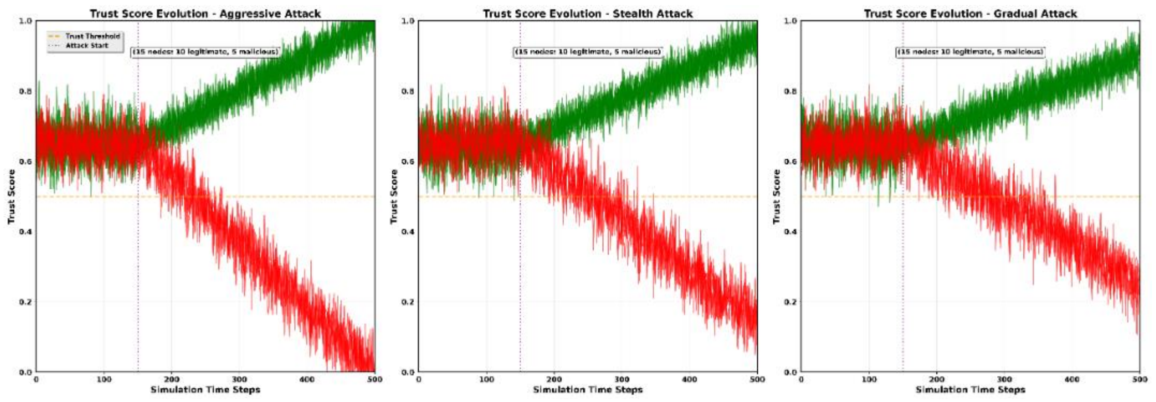

Fig. 5. Trust Score Evolution

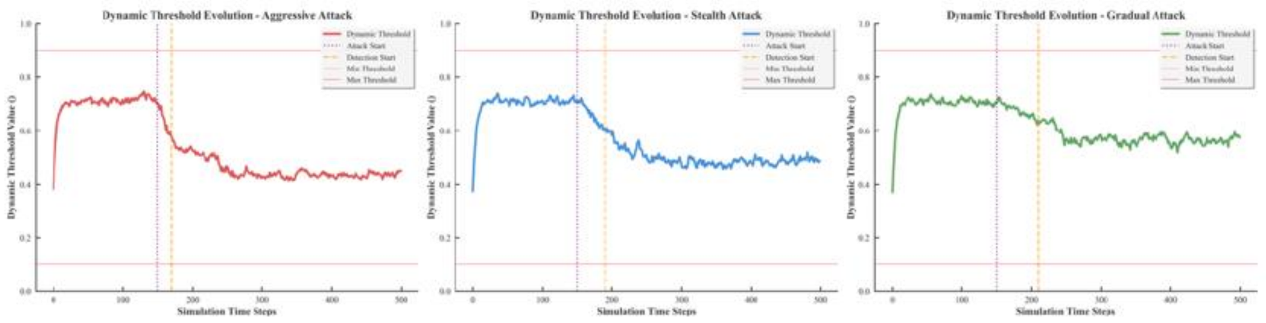

Fig. 6. Dynamic Threshold

C. Ablation Study

Our proposed model ablation study shows the algorithmic advances for the contribution of each component to overall system performance.

1) Centralized vs. Decentralized Critic Architecture

This ablation study examines the significance of centralized training in proposed framework by comparing the complete model with a fully decentralized training where individual agents learn autonomously without global coordination. In our proposed architecture nodes exchange their local trust assessment and detection outcomes with a central coordinator that consolidates all fog node decisions, and provides global feedback. An individual node cannot work and take decision independently, they require collaboration for decision making. The decentralized version eliminates all centralized coordination, with each node functioning as an autonomous I-DQN agent making decisions exclusively based on local observations without any global information exchange or consensus mechanisms. The centralized critic variant demonstrates 5.8% superior detection accuracy (87.3% vs 81.5%), 2.4% lower FPR (6.8% vs 9.2%), and 5.8% lower FNR (12.7% vs 18.5%) compared to the purely decentralized IDQN-NT approach, confirming the significance value of global coordination in our framework.

2) Static vs. Dynamic Threshold Management

This ablation compares the effectiveness of our adaptive threshold system with that of a model with static, hard-coded thresholds (EMAT). In a static threshold trust system, every node employs a fixed trust threshold value of 0.5. But in a dynamic threshold trust system, the trust threshold value is adjusted based on network conditions, attack intensity, performance metrics, and temporal factors. A Trust management system with an adaptive threshold provides superior performance compared to a static approach, particularly in dynamic network environments where attack patterns and network conditions vary over time. Dynamic threshold achieves 17.9% higher detection accuracy (87.3% vs 69.4% compared to EMAT), 16.3% lower FPR (6.8% vs 23.1% compared to EMAT), and 17.9% lower FNR(12.7% vs 30.6% compared to EMAT), demonstrating the advantage of adaptive threshold in fog network security applications.

3) Fixed vs. Learned Trust Parameters

This ablation evaluates the contribution of our MARL-based learning approach by comparing the proposed model with learned trust parameters against fixed, heuristic based parameters. The fixed parameters are selected using standard defaults from recent literature, including Adam learning rate (0.001) from Stable Baseline [31], DQN discount factor (0.99) from [32], and trust management defaults (recovery rate: 0.1, decay factor: 0.95) from [33,34] representing typical non-optimised configurations found in recent frameworks. The learned parameter variant is systematically tuned through Optuna optimization, resulting in optimized values like learning parameters and trust parameters displayed in Table 3. The optimized trust recovery rate of 0.042 is lower than literature defaults due to specific attack patterns and network dynamics in our fog computing environment where stability is prioritized over rapid recovery.

Learned parameters achieve 20.6% higher detection accuracy (87.3% vs 66.7% compared to HTM), 19.5% lower FPR (6.8% vs 26.3% compared to HTM), and 20.6% lower FNR (12.7% vs 33.3% compared to HTM) as shown in Table 6, validating the effectiveness of our MARL based learning approach in optimizing trust management for fog network security. The significant performance improvement demonstrates that systematic parameter optimization provides substantial advantages over standard default configurations, making our approach particularly valuable for critical security applications where performance optimization is justified.

TABLE VI. FIXED VS LEARNED PARAMETERS

Paramateres	Fixed	Optimized
Learning_rate	0.001	0.0069
Gamma	0.99	0.961
epsilon_decay	0.995	0.998
Epsilon_min	0.01	0.026
Trust recovery rate	0.1	0.042
Trust_decay _factor	0.95	0.967

4) Results Interpretation

The proposed framework exhibits varying detection effectiveness across different attack complexities, with aggressive attacks achieving a 92.0% ± 2.1% detection rate due to their behavioral signatures. In comparison, stealth and gradual attacks show reduced performance at 78.0% ± 2.0% and 67.9% ± 2.2% respectively, reflecting the increased challenge of identifying subtle malicious patterns. These findings indicate that the trust based approach excels at detecting aggressive and stealth attacks. It faces limitations against sophisticated attack strategies that require long term behavioral monitoring and pattern recognition.

The proposed system outperforms other methods, achieving 87.3% detection accuracy compared to 81.5% for basic I-DQN and 66.7% for simple trust models, demonstrating particular strength in detecting complex stealth and gradual attacks through its advanced trust evaluation approach. While the system achieves better results, it requires more computing power for trust calculations. It needs a sufficient interaction history to function correctly, which could limit its use in fast changing or resource limited networks.

The system faces several key weaknesses in different situations. The framework struggles to detect gradual attacks at their early stages because it requires sufficient evidence before deciding a node is malicious, leading to delays in identification. As our model depends on centralized coordination during training, it makes the model susceptible to collusion attacks where malicious nodes can strategically manipulate the local malicious list to overwhelm the consensus based formation of the global malicious list. Still, the model uses only reliable reporters based on reputation and consistency scores, and it utilizes byzantine fault tolerance (BFT) compliant threshold. Specifically, a node is confirmed malicious only if it is reported by more than two-thirds of the reliable agents. This can prevent only 51% attacks from collusion attacks, but it still needs improvement, as they can erode the network. The system also performs poorly in networks with few connections, as it relies heavily on neighbor recommendations that may not be sufficient. Additionally, smart attackers could potentially learn how the detection system works and adjust their behaviors to evade detection, requiring the system to adapt and update its detection methods continually.

VI. CONCLUSION AND FUTURE ENHANCEMENT

This research presents a novel approach for the detection of malicious nodes in dynamic network like fog network with the integration of MARL approach with an adaptive trust management system. Our comprehensive experimental evaluation demonstrates the effectiveness of the proposed model across three attack scenarios - aggressive, stealth, and gradual. One of the main contributions is the creation of a dynamic trust threshold adaptation mechanism, where in case of network condition change, the intelligent node alternates between aggressive and gradual detection approaches. The experimental results demonstrate that our MARL-based approach achieves superior detection rates compared to traditional baseline methods, with detection accuracies of 92%, 78%, and 68% for aggressive, stealth, and gradual attacks, respectively. The system demonstrates remarkable resilience in maintaining network performance, with packet delivery ratios recovering to 88-92% after detection and mitigation of the attack. Our adaptive consensus algorithm (Algorithm 2) successfully creates a robust global malicious node registry through weighted evidence aggregation. The reputation based node management system effectively balances security requirements with fairness, ensuring that legitimate nodes are not unjustly penalized while maintaining strict oversight of previously malicious entities. The dynamic threshold adaptation mechanism proves particularly effective in optimizing the trade-off between detection sensitivity and FPR. By employing exponential moving averages and contextual factors such as network maturity and threat levels, the system maintains stable operation while remaining responsive to emerging security challenges. The suggested MARL framework provides notable improvements for further study.

- Secure Trust Data Management: To ensure the integrity, immutability and auditability of shared reputation scores and global malicious lists, secure trust data management is required, such as integration of Distributed Ledger Technologies (DLT), such as blockchain or verifiable computation approaches. As a result, trust data would no longer be dependent on a centralized, perhaps weak orchestrator.
- Communication Efficiency Optimization: To drastically cut down on the communication overhead between fog nodes and the orchestrator, a crucial factor in dense and scalable fog deployments, experiment with sophisticated data aggregation techniques, hierarchical reporting structures, or reinforcement learning driven sparse communication policies.
- Enhanced Learning Algorithms: Future research could explore advanced deep reinforcement learning architectures, including transformer based attention mechanisms and graph neural networks, to capture complex spatial temporal dependencies in fog network topologies. The integration of meta-learning approaches could enable faster adaptation to novel attack patterns that were not encountered during training.
- Federated Trust Management: Implementing federated learning principles within the trust management framework could enable privacy preserving collaboration between multiple fog domains. This approach would allow trusted information sharing without exposing sensitive network details.
- Behavioral Pattern Analytics: The integration of advanced behavioral analytics using unsupervised learning techniques could enhance the system's efficiency to detect zero-day attacks and sophisticated adversarial behaviors. Anomaly detection based on network flow patterns and node interaction frequencies could complement the existing trust based approach.
- In case of central orchestrator compromise scenario, it may lead to false information injection, poisoned learning and configuration hijacking, which distorts the learning signal and potentially causes cascading policy degradation across the network. We can integrate Cryptographic Authentication, Byzantine Fault Tolerance (BFT), and distributed log storage to protect the coordinator's resources and maintain operational continuity.

REFERENCES

[1] Burhan, M., Alam, H., Arsalan, A., Rehman, R. A., Anwar, M., Faheem, M., & Ashraf, M. W. (2023). A comprehensive survey on the cooperation of fog computing paradigm-based IoT applications: layered architecture, real-time security issues, and solutions. IEEE Access, 11, 73303-73329. DOI: 10.1109/ACCESS.2023.3294479

[2] Jin, J., Pang, Z., Kua, J., Zhu, Q., Johansson, K. H., Marchenko, N., & Cavalcanti, D. (2025). Cloud-Fog Automation: The New Paradigm towards Autonomous Industrial Cyber-Physical Systems. IEEE Journal on Selected Areas in Communications. DOI: 10.1109/JSAC.2025.3574587

[3] Liang, F., Zhang, Z., Lu, H., Li, C., Leung, V., Guo, Y., & Hu, X. (2024). Resource allocation and workload scheduling for large-scale distributed deep learning: A survey. arXiv preprint arXiv:2406.08115.

[4] Wang, X., Wang, B., Wu, Y., Ning, Z., Guo, S., & Yu, F. R. (2024). A survey on trustworthy edge intelligence: From security and reliability to transparency and sustainability. IEEE Communications Surveys & Tutorials. DOI: 10.1109/COMST.2024.3446585

[5] Saidi, A. (2024). An adaptive trust system for misbehavior detection in wireless sensor networks. Wireless Networks, 30(4), 2589-2615. https://doi.org/10.1007/s11276-024-03687-4

[6] Abughazalah, M., Alsaggaf, W., Saifuddin, S., & Sarhan, S. (2024). Centralized vs. Decentralized Cloud Computing in Healthcare. Applied Sciences, 14(17), 7765 https://doi.org/10.3390/app14177765

[7] Hu, K., Li, M., Song, Z., Xu, K., Xia, Q., Sun, N., ... & Xia, M. (2024). A review of research on reinforcement learning algorithms for multi-agents. Neurocomputing,128068. https://doi.org/10.1016/j.neucom.2024.128068

[8] Zhu, C., Dastani, M., & Wang, S. (2024). A survey of multi-agent deep reinforcement learning with communication. Autonomous Agents and Multi-Agent Systems, 38(1), 4. https://doi.org/10.1007/s10458-024-09644-x

[9] Al-Khafajiy, M., Baker, T., Asim, M., Guo, Z., Ranjan, R., Longo, A., ... & Taylor, M. (2020). COMITMENT: A fog computing trust management approach. Journal of Parallel and Distributed Computing, 137, 1-16. https://doi.org/10.1016/j.jpdc.2019.10.006

[10] Rehman, A., Awan, K. A., Ud Din, I., Almogren, A., & Alabdulkareem, M. (2023). FogTrust: fog-integrated multi-leveled trust management mechanism for internet of things. Technologies, 11(1), 27. https://doi.org/10.3390/technologies11010027

[11] Ghaleb, M., & Azzedin, F. (2023). Trust-aware fog-based iot environments: Artificial reasoning approach. Applied Sciences, 13(6), 3665. https://doi.org/10.3390/app13063665

[12] P. K. R. Goel, J. Jain, S. Sahu, and S. Singh, "PREPUTATION BASED TRUST MANAGEMENT SYSTEM FOR MALICIOUS FOG NODE

DETECTION," Journal of Theoretical and Applied Information Technology, vol. 100, no. 24, pp. 6463-6472, 2022.

[13] Ogundoyin, S. O., & Kamil, I. A. (2021). A trust management system for fog computing services. Internet of Things, 14, 100382 https://doi.org/10.1016/j.iot.2021.100382

[14] Alruwaythi, M., Kambhampaty, K., & Nygard, K. E. (2019). User Behavior and Trust Evaluation in Cloud Computing. In CATA (pp. 378-386).

[15] Hady, M. A., Hu, S., Pratama, M., Cao, J., & Kowalczyk, R. (2025). Multi-Agent Reinforcement Learning for Resources Allocation Optimization: A Survey. arXiv preprint arXiv:2504.21048.

[16] We, X., Farooq, J., & Chen, J. (2024, November). Multi-Agent Distributed Decentralized Dynamic Resource Orchestration in 5G Edge-Cloud Networks. In 2024 IEEE 13th International Conference on Cloud Networking (CloudNet) (pp. 1-8). IEEE. DOI: 10.1109/CloudNet62863.2024.10815780

[17] Lu, J., Hong, C., & Wang, R. (2024). MAGT-toll: A multi-agent reinforcement learning approach to dynamic traffic congestion pricing. PloS one, 19(11), e0313828 https://doi.org/10.1371/journal.pone.0313828

[18] Alnaim, A. K., & Alwakeel, A. M. (2025). Zero-Trust Mechanisms for Securing Distributed Edge and Fog Computing in 6G Networks. Mathematics (2227-7390), 13(8). DOI 10.3390/math13081239

[19] Shehada, D., Gawanmeh, A., Yeun, C. Y., & Zemerly, M. J. (2022). Fog-based distributed trust and reputation management system for internet of things. Journal of King Saud University-Computer and Information Sciences, 34(10), 8637-8646. https://doi.org/10.1016/j.jksuci.2021.10.006

[20] Alvi, A. N., Ali, B., Saleh, M. S., Alkhathami, M., Alsadie, D., & Alghamdi, B. (2024). Secure computing for fog-enabled industrial IoT. Sensors, 24(7), 2098. https://doi.org/10.3390/s24072098

[21] Alvi, A. N., Ali, B., Saleh, M. S., Alkhathami, M., Alsadie, D., & Alghamdi, B. (2023). TETES: trust based efficient task execution scheme for fog enabled smart cities. Applied Sciences, 13(23), 12799 https://doi.org/10.3390/app132312799

[22] Shabut, A. M., Dahal, K. P., Bista, S. K., & Awan, I. U. (2014). Recommendation based trust model with an effective defence scheme for MANETs. IEEE Transactions on mobile computing, 14(10), 2101-2115. DOI: 10.1109/TMC.2014.2374154

[23] ALGESHARI, W., RAMAZAN, M. S., ALOTAIBI, F., & ALYOUBI, K. (2024). AN EXTENSIVE REVIEW OF SECURITY ISSUES AND CHALLENGES IN FOG COMPUTING ENVIRONMENT. Journal of Theoretical and Applied Information Technology, 102(22).

[24] Moudoud, H., Abou El Houda, Z., & Brik, B. (2024). Advancing security and trust in wsns: A federated multi-agent deep reinforcement learning approach. IEEE Transactions on Consumer Electronics. DOI: 10.1109/TCE.2024.3440178

[25] Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., ... & Graepel, T. (2017). Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296.

[26] Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. Journal of Machine Learning Research, 21(178), 1-51

[27] Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems, 30.

[28] Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the tenth international conference on machine learning (pp. 330-337).

[29] Samann, F. E. F., Zeebaree, S. R., & Askar, S. (2021). IoT provisioning QoS based on cloud and fog computing. *Journal of Applied Science and Technology Trends*, *2*(01), 29-40 https://doi.org/10.38094/jastt20190.

[30] Hasan, R. T. (2022). Internet of things and big data analytic: A state of the art review. *Journal of Applied Science and Technology Trends*, *3*(02), 93-100. https://doi.org/10.38094/jastt302135

[31] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of machine learning research*, *22*(268), 1-8.

[32] Zhang, Y., & Ross, K. W. (2021, July). On-policy deep reinforcement learning for the average-reward criterion. In *International Conference on Machine Learning* (pp. 12535-12545). PMLR.

[33] Alshehri, M. D., & Hussain, F. K. (2019). A fuzzy security protocol for trust management in the internet of things (Fuzzy-IoT). *Computing*, *101*(7), 791-818.

[34] Alghofaili, Y., & Rassam, M. A. (2022). A trust management model for IoT devices and services based on the multi-criteria decision-making approach and deep long short-term memory technique. *Sensors*, *22*(2), 634. https://doi.org/10.3390/s22020634

[35] Naramalli Jayakrishna & N. Narayanan Prasanth, A hybrid deep learning model for detection and mitigation of DDoS attacks in VANETs, Scientific Reports, (2025) 15:34170 https://doi.org/10.1038/s41598-025-15215-1