Learning to Defend: A Multi-Agent Reinforcement Learning Framework for Stackelberg Security Game in Mobile Edge Computing

Zihao Ding
EECS
South Dakota State University
Brookings, USA
zihao.ding@jacks.sdstate.edu

Jun Huang
EECS
South Dakota State University
Brookings, USA
jun.huang@sdstate.edu

Junjian Qi
EECS
South Dakota State University
Brookings, USA
junjian.qi@sdstate.edu

***Abstract*—This paper addresses the general security problem in mobile edge computing. The problem is modeled as a two-stage attacking-defending Stackelberg security game, and a multi-agent reinforcement learning framework based on independent proximal policy optimization is designed to solve the problem. The framework employs Markov decision processes for both the attacker and the defender to systematically represent network states, attack vectors, and resource allocation actions. The reward functions are designed to reflect the objectives of network disruption for the attacker and network protection for the defender. Experimental results demonstrate that our proposed framework achieves stable convergence and significantly reduces attack success rates compared to baseline methods in the randomly generated edge computing network.**

***Index Terms*—Mobile edge computing, Stackelberg security game, multi-agent reinforcement learning, proximal policy optimization, resource allocation**

I. Introduction

Mobile Edge Computing (MEC) infrastructure faces significant security challenges due to its distributed nature and resource constraints. Adversaries can launch attacks, including DDoS, data injection, and advanced persistent threats. The ultra-dense deployment of 5G/6G technologies opens new doors for attackers where compromising critical nodes triggers cascading failures [1]. Hardware-level vulnerabilities, such as timing fault-injection attacks, further worsen the situation [2].

Game-theoretic approaches provide powerful tools for modeling security interactions, among which Stackelberg games are widely adopted for MEC security. In a typical Stackelberg security game, defenders commit to strategies first and attackers respond after observation [3], [4]. Such a sequential structure reflects real-world scenarios where defenders deploy security mechanisms proactively while attackers conduct attacks [5].

However, obtaining the equilibrium for the Stackelberg security game is not a trivial task. The attacker and defender may evolve during the game, and when attackers deviate from the assumed payoff structure or switch to qualitatively different tactics, the equilibrium becomes meaningless and irrelevant [6]. Multi-Agent Reinforcement Learning (MARL) addresses these issues by allowing defenders and attackers to co-adapt through interaction. It handles nonstationarity and partial observability, and can discover effective countermeasures against previously unseen attack policies.

Recent advances in MARL for Stackelberg security games, including meta-MDP formulations [7], [8] and adaptively-regularized adversarial training [9], laid solid foundations for algorithm design. Existing studies, such as spatio-temporal decision-making [10], three-player games [11], federated UAV learning [12], peer-to-peer energy trading [13], and vehicle-signal control [14], have demonstrated that hierarchical game-theoretic models integrated with MARL are effective for strategic decision making. Meanwhile, research on distributed security systems has highlighted the value of spatial discounting and decentralized learning in scaling to large deployments [15], [16], while most works emphasize short-term performance in small-scale or centralized settings [11], [13]. However, current approaches often neglect the impact of critical nodes and the feature network connectivity, which are crucial in the MEC network because compromising key nodes can trigger cascading failures. Also, the existing work does not account for edge-specific defense mechanisms such as honeypot deployment, real-time scanning, and threat mitigation.

In this work, we model MEC security as a two-stage Stackelberg game and solve it with an MARL framework based on Independent Proximal Policy Optimization (IPPO). The game captures interactions between attackers targeting critical edge servers and defenders deploying honeypots, network scanning, and threat cleaning, while accounting for topology, node criticality, and budget limits. We design IPPO with crafted state, action, and reward so both players learn near-optimal strategies from sequential play. Simulation results on a randomly generated network with varying critical node configurations demonstrate that our proposed approach achieves stable convergence and significantly reduces the attack success rates compared to the baseline method.

The remainder of this paper is organized as follows. Section II presents the system model. Section III elaborates on the Stackelberg security game. Section IV introduces the IPPO-

based framework. Section V presents the simulation results and finally Section VI concludes the paper.

II. System Model

A. System Overview

Consider an MEC system deployed in an ultra-dense cellular network comprising N edge servers $\mathcal{N} = \{1, 2, \cdots, N\}$ with cellular base stations. Each edge server n maintains computational capacity $C_n \in \mathbb{R}_+$ and provides local processing services for mobile users. The edge servers connect through two types of communication links. Each server n connects to its local base station for serving mobile devices within its coverage area. Additionally, edge servers establish inter-server connections through the cellular backbone for coordination and load balancing.

The network topology is modeled by an adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$. In the network, certain nodes are more important in providing services. We characterize node importance through network connectivity, where nodes with higher degrees serve as key hubs for distributed edge computing operations. Each edge server n has degree $k_n = \sum_{i=1}^{N} A_{n,i}$ representing the number of other edge servers it can directly communicate with. The degree is bounded by infrastructure constraints: $k_n \in [K_{\min}, K_{\max}]$ where $K_{\min}$ and $K_{\max}$ represent minimum and maximum connectivity limits.

Each server operates in state $s_n(t) \in \{0, 1\}$ where $s_n(t) = 1$ indicates normal operation and $s_n(t) = 0$ indicates compromised state. The distributed edge computing infrastructure enables local processing while maintaining connectivity for resource sharing and coordination among edge servers.

B. Problem Formulation

In a mobile edge computing system, there is a high chance that security threats target edge servers to disrupt local processing capabilities and compromise service quality for mobile users. The high-density deployment, the distributed nature of edge servers, and compromising key coordination nodes can lead to cascading failures throughout the network. To prevent it, we employ the tool of the Stackelberg game to analyze the strategic interactions between attackers and defenders.

We formulate the security problem as a two-stage Stackelberg game $\mathcal{G} = \{\mathcal{A}, \mathcal{D}, \mathbf{X}, \mathbf{Y}, U^{\mathrm{a}}, U^{\mathrm{d}}, \mathcal{C}^{\mathrm{a}}, \mathcal{C}^{\mathrm{d}}\}$ where the attacker $\mathcal{A}$ acts as the leader and the defender $\mathcal{D}$ as the follower. The game proceeds in a sequential way:

Stage 1 (Attacker's Action): The attacker selects attack strategy $\mathbf{x} = [x_1, x_2, \ldots, x_N]^{\mathrm{T}} \in \mathbf{X}$ where $x_n \in [0, 1]$ represents the attack intensity allocated to edge server n. The attack cost is $\mathcal{C}^{\mathrm{a}}(\mathbf{x}) = \sum_{n=1}^{N} c_n^{\mathrm{a}} \cdot x_n$ where $c_n^{\mathrm{a}} > 0$ denotes the unit attack cost for server n.

Stage 2 (Defender's Response): Observing the attack strategy, the defender responds with defense strategy $\mathbf{y} = [y_1, y_2, \ldots, y_N]^{\mathrm{T}} \in \mathbf{Y}$ where $y_n \in [0, 1]$ indicates the defense allocation to edge server n. The defense cost is $\mathcal{C}^{\mathrm{d}}(\mathbf{y}) = \sum_{n=1}^{N} c_n^{\mathrm{d}} \cdot y_n$ where $c_n^{\mathrm{d}} > 0$ represents the unit defense cost for server n.

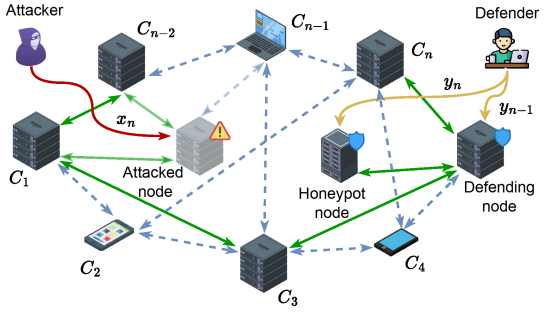


Fig. 1: Security modeling in MEC.

Both players operate under budget constraints: $\mathcal{C}^{\mathrm{a}}(\mathbf{x}) \leq B^{\mathrm{a}}$ and $\mathcal{C}^{\mathrm{d}}(\mathbf{y}) \leq B^{\mathrm{d}}$ where B^{a} and B^{d} are the available budgets. The game terminates when either player exhausts their budget or when the attacker successfully compromises a critical edge server with $x_n > \tau$ where $\tau \in (0, 1)$ is the compromise threshold.

The utility of attacker captures expected disruption to edge computing services:

$$U^{\mathrm{a}}(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{N} w_n \cdot P_n(\mathbf{x}, \mathbf{y}) - \mathcal{C}^{\mathrm{a}}(\mathbf{x}), \tag{1}$$

The utility of defender is reflected by the preservation of edge computing capabilities:

$$U^{\mathrm{d}}(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{N} w_n \cdot (1 - P_n(\mathbf{x}, \mathbf{y})) - \mathcal{C}^{\mathrm{d}}(\mathbf{y}), \tag{2}$$

where $w_n = k_n / \sum_{i=1}^{N} k_i$ represents edge server importance based on connectivity, and $P_n(\mathbf{x}, \mathbf{y}) = \frac{x_n}{x_n + y_n + \epsilon}$ denotes the attack success probability at edge server n with x_n the attack intensity, y_n the defense resource allocation, and $\epsilon > 0$ a constant preventing division by zero.

III. Design of Stackelberg Security Game

A. Attacker And Defender Behavior Analysis

As illustrated in Fig. 1, the attacker targets edge servers and disrupts their computing services. The attack success probability follows Eq. (1), where higher degree nodes k_n become more valuable targets. Successful attacks compromise the node and all its associated edges $\{(n, i) : i \in \mathcal{N}_n\}$, leading to the network breakdown.

To mitigate this, we assume that the defender uses three counter-strategies. *Honeypot deployment* involves placing fake nodes $\mathcal{H} \subset \mathcal{N}$ with (fake) degree $\hat{k}_h$ to trick attackers. The deception effectiveness is defined as:

$$P_h = \frac{y_h \cdot \hat{k}_h}{\hat{k}_h + x_h + \epsilon}, \tag{3}$$

where $h \in \mathcal{H}$ denotes honeypot nodes and $\hat{k}_h$ represents the fake degree of honeypot h.

Network scanning allows the defender to identify attack patterns and allocate defense resources according to the observed

attack strategy $\mathbf{x}$, and *threat cleaning* removes malicious components from compromised nodes with probability:

$$P_e = \min\left(1, \frac{y_n}{\psi \cdot x_n}\right), \tag{4}$$

where $\psi > 0$ represents the cleaning difficulty factor.

This give us a game where attacker control-and-cut actions compete against defender deception and cleaning responses.

Each edge server n has a base attack success probability $p_n^b \in (0,1)$ that reflects its vulnerabilities such as software configuration, hardware specifications, and deployment environment. The base probability is modified by the strategic resource allocation of both players, leading to the final attack success probability to be:

$$\tilde{P}_n(\mathbf{x}, \mathbf{y}) = p_n^b + (1 - p_n^b) \cdot P_n(\mathbf{x}, \mathbf{y}), \tag{5}$$

where p_n^b represents the base vulnerability, $P_n(\mathbf{x}, \mathbf{y})$ is the strategic component from Eq. (2).

B. Game-Theoretic Design

The attacker employs an attack maximization strategy to allocate resources to multiple edge servers. The attack optimization problem is formulated as:

$$\max_{\mathbf{x}} \quad \sum_{n=1}^{N} w_n \cdot \tilde{P}_n(\mathbf{x}, \mathbf{y}) - \sum_{n=1}^{N} c_n^{\mathrm{a}} \cdot x_n \tag{6a}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} c_n^{\mathrm{a}} \cdot x_n \le B^{\mathrm{a}}, \tag{6b}$$

$$x_n \in [0,1], \quad \forall n \in \{1, 2, \ldots, N\}, \tag{6c}$$

where w_n represents node importance based on connectivity, $c_n^{\mathrm{a}} > 0$ denotes the unit attack cost for server n, and B^{a} is the attacker's budget constraint.

In general, attacking follows a three-phase process: reconnaissance, infiltration, and disruption. During reconnaissance, the attacker identifies high-value targets with degree $k_n > \bar{k}$ where $\bar{k} = \frac{1}{N}\sum_{i=1}^{N} k_i$. The infiltration phase attempts to compromise target servers with success probability $\tilde{P}_n(\mathbf{x}, \mathbf{y})$ from Eq. (5). Upon successful compromise, the disruption phase blackouts all links associated with that server.

On the other hand, the defender employs a security preservation strategy to maintain edge computing capabilities against attacks. The defense optimization problem is formulated as:

$$\max_{\mathbf{y}} \quad \sum_{n=1}^{N} w_n \cdot (1 - \tilde{P}_n(\mathbf{x}, \mathbf{y})) - \sum_{n=1}^{N} c_n^{\mathrm{d}} \cdot y_n \tag{7a}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} c_n^{\mathrm{d}} \cdot y_n \le B^{\mathrm{d}}, \tag{7b}$$

$$y_n \in [0,1], \quad \forall n \in \{1, 2, \ldots, N\}, \tag{7c}$$

where $c_n^{\mathrm{d}} > 0$ represents the unit defense cost for server n, and B^{d} is the defender's budget constraint. This optimization maximizes network security preservation while managing defense expenditures efficiently.

Algorithm 1 IPPO based Learning Framework for Stackelberg Security Game in MEC

Input: Network topology $\mathbf{A}$, node degrees $\{k_n\}_{n=1}^{N}$, budgets $B^{\mathrm{a}}, B^{\mathrm{d}}$, episodes E;
Initialize attacker policy $\pi_{\theta^{\mathrm{a}}}$, defender policy $\pi_{\theta^{\mathrm{d}}}$, value functions $V_{\phi^{\mathrm{a}}}, V_{\phi^{\mathrm{d}}}$;
Initialize buffers $\mathcal{B}^{\mathrm{a}} \leftarrow \emptyset$, $\mathcal{B}^{\mathrm{d}} \leftarrow \emptyset$;
for $episode = 1, 2, \ldots, E$ **do**
  Phase I: Attacker Learning
  for $t = 1, 2, \ldots, T^{\mathrm{a}}$ **do**
    Observe attacker state $\mathcal{S}^{\mathrm{a}}[t]$;
    Sample attack allocation $\{\rho_n[t]\}_{n=1}^{N} \sim \pi_{\theta^{\mathrm{a}}}$;
    Compute attack intensities and simulate defender response;
    Update server states based on success probabilities ▷ Eq. (5)
    Compute attacker reward $R^{\mathrm{a}}[t]$ ▷ Eq. (10)
    Store experience in $\mathcal{B}^{\mathrm{a}}$;
  end for
  Update attacker policy using IPPO if buffer ready;
  Phase II: Defender Learning
  for $t = 1, 2, \ldots, T^{\mathrm{d}}$ **do**
    Observe defender state $\mathcal{S}^{\mathrm{d}}[t]$ with attacker strategy;
    Sample defense allocation and honeypot decisions $\sim \pi_{\theta^{\mathrm{d}}}$;
    Compute defense intensities and honeypot deployment;
    Compute defender reward $R^{\mathrm{d}}[t]$ ▷ Eq. (13)
    Store experience in $\mathcal{B}^{\mathrm{d}}$;
  end for
  Update defender policy using IPPO if buffer ready;
end for
return $\pi^*_{\theta^{\mathrm{a}}}, \pi^*_{\theta^{\mathrm{d}}}$;

In response to the attacking process, the defender takes two counter-strategies. Honeypot deployment builds fake edge servers $h \in \mathcal{H}$ with artificial connectivity $\hat{k}_h$ to mislead attackers. The deception effectiveness is given by Eq. (3), where higher fake degrees $\hat{k}_h$ make honeypots more attractive targets. As such, attackers can waste resources on non-critical nodes; and thus reducing their effective attacks on real servers.

Network scanning allows the defender to detect compromised servers and initiate recovery operations. The scanning mechanism identifies attack patterns $\mathbf{x}$ and allocates recovery resources accordingly. Detected compromised servers undergo cleaning procedures with probability P_n^c from Eq. (4).

IV. A Reinforcement Learning-Based Framework

To obtain the Stackelberg equilibrium, we design an MARL framework based on IPPO. IPPO effectively handles the multi-agent interactions and high-dimensional strategy spaces inherent in the formulated security game, while its clipped objective function prevents destabilizing policy updates in adversarial environments. We develop two interrelated *Markov Decision Processes* (MDPs) to model the attacker and defender decision processes.

A. MDP for Attacker

1) State Space: At time step t, the attacker's state space $\mathcal{S}_a$ captures network characteristics and historical defense patterns:

$$\mathcal{S}^{\mathrm{a}}[t] = \{\bar{k}, k_{\max}, N_c[t-1], \bar{y}[t-1], B_r^{\mathrm{a}}[t]\}, \tag{8}$$

where $\bar{k} = \frac{1}{N}\sum_{n=1}^{N} k_n$ is the average node degree in the network, $k_{\max}$ is the highest node degree indicating the most

TABLE I: Parameter settings.

Parameter	Symbol	Value
Number of edge nodes	N	30
Number of critical nodes	N_c	5, 10, 15
Number of agents	-	2 (attacker, defender)
10-node Episodes	E_1	400
100-node Episodes	E_2	1000
Max episode steps	T_{step}	500
Buffer size per agent	-	8192
Batch size per agent	B	512
Learning rate	η	1.0×10^{-6}
PPO epochs	K	8
Clip range	ϵ	0.1
Discount factor	γ	0.995
GAE lambda	λ	0.98
Entropy coefficient	β_e	0.005

connected server, $N_c[t-1]$ is the total count of successfully compromised servers from the previous time step, $\bar{y}[t-1] = \frac{1}{N}\sum_{n=1}^{N} y_n[t-1]$ is the average defense resource allocation observed in the previous round, and $B_r^{\text{a}}[t]$ is the attacker's remaining budget available for allocation at time t.

2) Action Space: The action space for the attacker $\mathcal{A}_a$ is consisted of the normalized attack intensity allocation:

$$\mathcal{A}^{\text{a}}[t] = \{\rho_n[t]\}_{n=1}^{N}, \quad \rho_n[t] \in [0,1], \quad \sum_{n=1}^{N} \rho_n[t] = 1, \tag{9}$$

where $\rho_n[t]$ is the proportion of total remaining budget allocated to attack server n, ensuring actual attack intensity $x_n[t] = \rho_n[t] \cdot B_r^{\text{a}}[t]/c_n^{\text{a}}$.

3) Reward Function: The attacker's reward function balances immediate damage and future opportunities:

$$R^{\text{a}}[t] = \alpha_1 \sum_{n=1}^{N} w_n \cdot \Delta s_n[t] - \alpha_2 \sum_{n=1}^{N} c_n^{\text{a}} x_n[t], \tag{10}$$

where $\alpha_1, \alpha_2 > 0$ are weight parameters that balance damage rewards against attack costs, and $\Delta s_n[t] = s_n[t-1] - s_n[t]$ is the binary indicator of whether server n was newly compromised in the current round, taking value 1 when compromised and 0 otherwise.

B. MDP for Defender

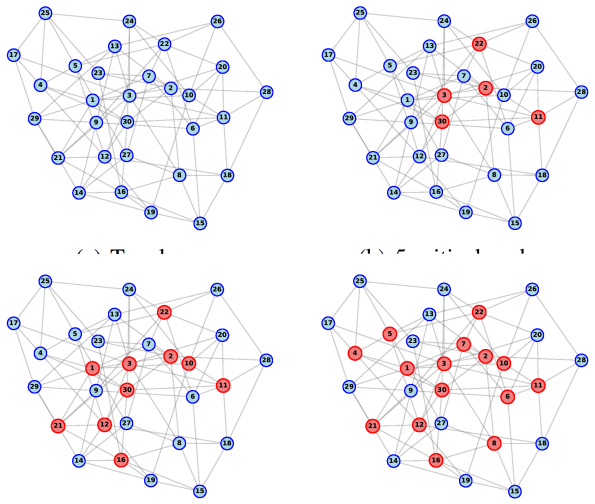
1) State Space: The state space of the defender $\mathcal{S}_d$ includes current attack observations and network status:

$$\mathcal{S}^{\text{d}}[t] = \{x_n[t], s_n[t], k_n, N_h[t-1], B_r^{\text{d}}[t]\}_{n=1}^{N}, \tag{11}$$

where $x_n[t]$ is the observed attack intensity currently targeting server n, $s_n[t] \in \{0,1\}$ is the current operational status of server n with value 1 for operational and 0 for compromised, k_n is the connectivity degree of server n, $N_h[t-1]$ is the total number of honeypot nodes that were active in the previous round, and $B_r^{\text{d}}[t]$ is the defender's remaining budget available for allocation at time t.

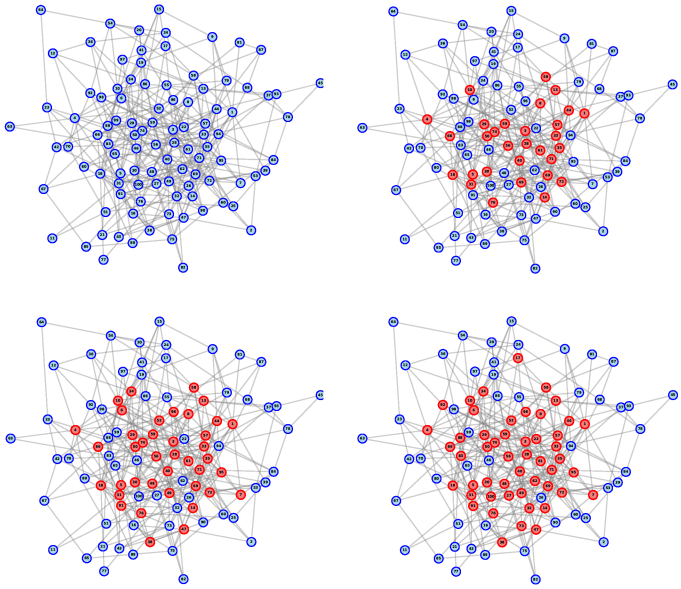
2) Action Space: The defender's action space includes resource allocation, honeypot deployment, and threat cleaning decisions:

$$\mathcal{A}^{\text{d}}[t] = \{\phi_n[t], h_n[t], c_n[t]\}_{n=1}^{N}, \tag{12}$$



(c) 10 critical nodes. (d) 15 critical nodes.

Fig. 2: A randomly generated 10-node network topology.



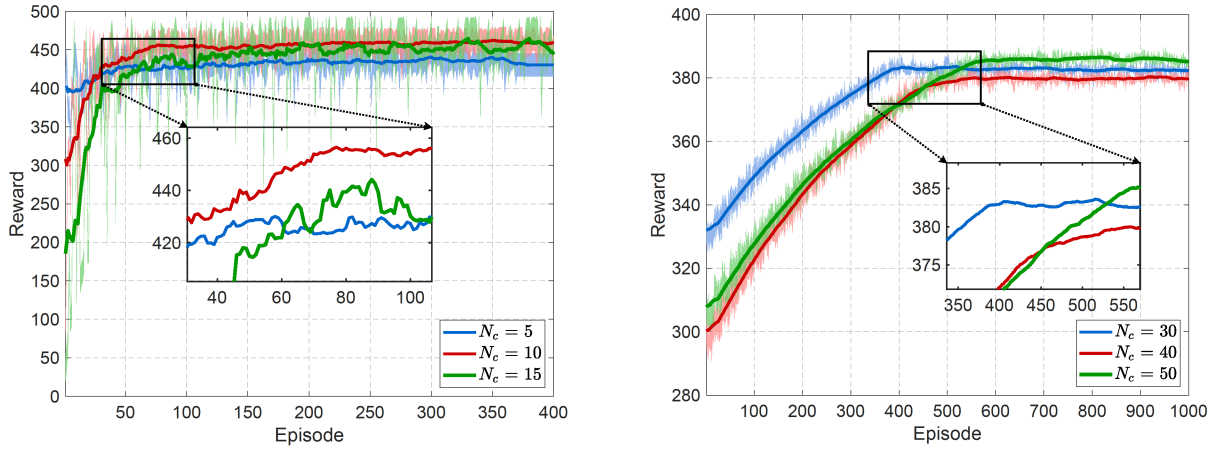
(c) 40 critical nodes. (d) 50 critical nodes.

Fig. 3: A randomly generated 100-node network topology.

where $\phi_n[t] \in [0,1]$ is the proportion of total remaining budget allocated to defend server n with constraint $\sum_{n=1}^{N} \phi_n[t] = 1$, $h_n[t] \in \{0,1\}$ is the binary decision to activate a honeypot at location n, and $c_n[t] \in \{0,1\}$ is the binary decision to initiate threat cleaning on server n, ensuring actual defense intensity $y_n[t] = \phi_n[t] \cdot B_r^{\text{d}}[t]/c_n^{\text{d}}$.

3) Reward Function: The reward function of the defender is designed based on network preservation:

$$R^{\text{d}}[t] = \beta_1 \sum_{n=1}^{N} w_n \cdot s_n[t] + \beta_2 \sum_{n=1}^{N} h_n[t] \cdot \mathbf{1}_{x_n[t]>0} + \beta_3 \sum_{n=1}^{N} c_n[t] \cdot P_e[t] - \beta_4 \sum_{n=1}^{N} c_n^{\text{d}} y_n[t], \tag{13}$$



(a) Defender's reward under 5, 10, and 15 critical nodes.

(b) Defender's reward under 30, 40, and 50 critical nodes.

Fig. 4: Defender's reward convergence under different critical node configurations.

where $\beta_1, \beta_2, \beta_3, \beta_4 > 0$ are weight parameters that balance network protection rewards, honeypot attraction benefits, threat cleaning rewards, and defense costs, $w_n = k_n / \sum_{i=1}^{N} k_i$ is the normalized importance weight based on server connectivity, $s_n[t] \in \{0, 1\}$ indicates whether server n remains operational with value 1 for operational and 0 for compromised, $h_n[t] \in \{0, 1\}$ indicates honeypot activation status, $c_n[t] \in \{0, 1\}$ denotes threat cleaning activation, $\mathbf{1}_{x_n[t]>0}$ is the indicator function for active attacks, and $P_e[t]$ is the cleaning success probability from Eq. (4).

The proposed MARL-based framework is given in Algorithm 1, which implements a sequential Stackelberg game through alternating training phases. The time complexity per episode is $O((T^{\mathrm{a}} + T^{\mathrm{d}}) \cdot N + K_{\text{epoch}} \cdot |\mathcal{B}|)$, scaling linearly with network size N and making it suitable for large-scale mobile edge computing systems.

V. Performance Evaluation

The evaluation of the proposed IPPO-based framework is conducted on two network scales: (1) a 30-node edge computing network with critical node counts set to 5, 10, and 15, and (2) a 100-node network with critical node counts of 30, 40, and 50. Training is performed for 400 episodes in the 30-node scenario and 1000 episodes in the 100-node scenario. The network topologies and configurations are illustrated in Fig. 2 and Fig. 3. Our IPPO framework is benchmarked against five baseline strategies: HAPPO [17], Independent SAC (ISAC), Greedy, Random, and Fixed allocation. Performance is assessed by analyzing the defender's reward during learning and the attack success counts from the attacker's perspective. The evaluation settings are summarized in Table I.

Fig. 4a shows the convergence of the defender's reward for the 30-node network under three configurations: 5, 10, and 15 critical nodes. In all cases, the reward curves converge stably. The configuration with 5 critical nodes achieves the highest reward, approximately 195, while the 10 and 15 critical node cases stabilize at around 185 and 180, respectively. These results demonstrate the inherent trade-off between maximizing protection coverage and maintaining resource allocation efficiency, as characterized by Eq. (13).

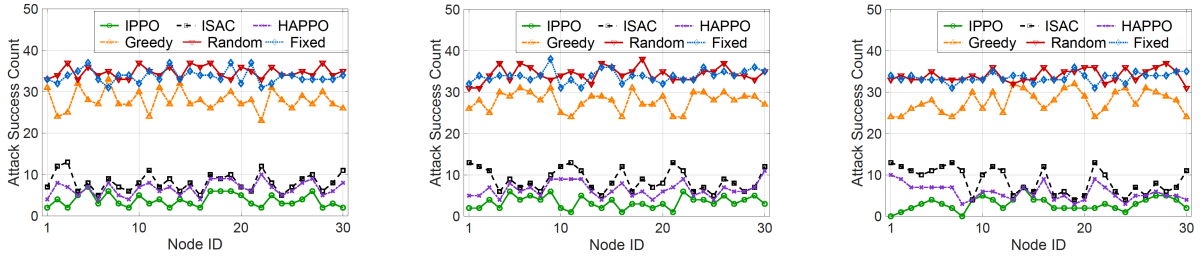
Fig. 5 presents the Attack Success Count (ASC) for the three critical node configurations. The IPPO framework consistently achieves ASC below 7, outperforming all baseline methods. Specifically, HAPPO yields ASC values between 12 and 18, ISAC between 15 and 20, Greedy between 25 and 35, and both Random and Fixed strategies between 35 and 45. Throughout the training process, the attacker adapts to the defender's strategies, yet the defender utilizing the IPPO framework maintains robust defense on critical nodes. These results indicate that the IPPO-based defender effectively adapts to evolving attack strategies and sustains a high level of protection.

For large-scale validation, Fig. 4b illustrates the defender's reward convergence in the 100-node network. Across all three critical node configurations, the reward stabilizes between 380 and 385, demonstrating that the learning framework scales effectively to larger networks. The inset window shows that equilibrium is reached after approximately 450 episodes, confirming that the defender can efficiently allocate resources across varying critical node densities in large-scale edge computing environments.

Fig. 6 shows the ASC distribution across all 100 nodes. The IPPO-based approach consistently achieves the lowest ASC, remaining below 12, and significantly outperforms the baseline methods: HAPPO (ASC 12 to 20), ISAC (ASC 17 to 22), Greedy (ASC 28 to 36), and Random and Fixed (ASC 37 to 45). In comparison to the 30-node scenario, the large-scale deployment demonstrates more stable ASC patterns with reduced variance, and the performance advantage of IPPO becomes more pronounced. The IPPO framework achieves approximately 40 to 50 percent lower ASC than HAPPO, confirming its scalability and robust defense effectiveness in complex, large-scale edge computing networks.

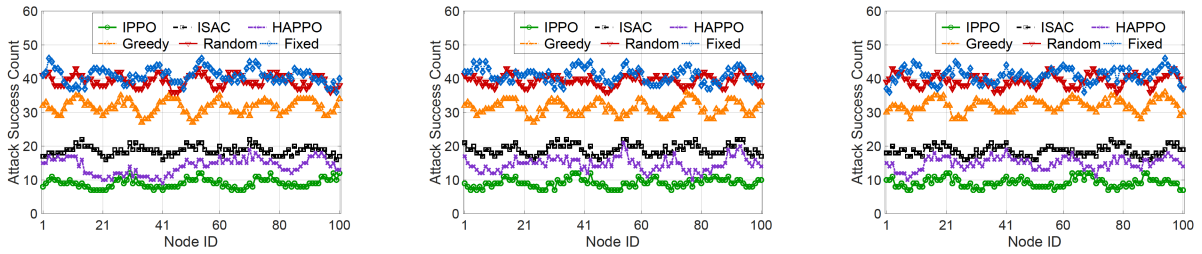
VI. Conclusion

In this paper, we have investigated security challenges in mobile edge computing networks. We have formulated a two-



(a) ASC with 5 critical nodes. (b) ASC with 10 critical nodes. (c) ASC with 15 critical nodes.

Fig. 5: Attack Success Count (ASC) under different critical node configurations (5, 10, and 15).



(a) ASC with 30 critical nodes. (b) ASC with 40 critical nodes. (c) ASC with 50 critical nodes.

Fig. 6: Attack Success Count (ASC) under different critical node configurations (30, 40, and 50).

stage Stackelberg security game that captures the strategic interactions between attackers targeting critical edge servers and defenders employing protection mechanisms, including honeypot deployment and threat cleaning. To solve this game, we have designed a multi-agent reinforcement learning framework that enables both players to learn optimal strategies through sequential decision-making processes. Our experimental results demonstrate that the IPPO-based framework effectively converges to stable policies and significantly reduces attack success counts compared to baseline methods.

REFERENCES

[1] T. Pathirana, R. F. Olimid, and G. Nencioni, "Joint Security and Dependability Modeling: An Updated Overview," *IEEE Access*, vol. 13, pp. 93 523–93 550, 2025.

[2] O. G. Martins, H. Akesson, M. Gomes, D. P. M. Osorio, P. Sen, and J. P. Vilela, "Delving Into Security and Privacy of Joint Communication and Sensing: A Survey," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 4978–5004, 2025.

[3] J. Chen, Z. Kuang, Y. Zhang, S. Lin, and A. Liu, "Blockchain-Enabled Computing Offloading and Resource Allocation in Multi-UAVs MEC Network: A Stackelberg Game Learning Approach," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 3632–3645, 2025.

[4] J. Huang, Q. Duan, C.-C. Xing, B. Gu, G. Wang, S. Zeadally, and E. Baker, "A Fine-Grained Video Traffic Control Mechanism in Software-Defined Networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3501–3515, 2022.

[5] Y. Ma, L. Liu, Z. Liu, F. Li, Q. Xie, K. Chen, C. Lv, Y. He, and F. Li, "A Survey of DDoS Attack and Defense Technologies in Multiaccess Edge Computing," *IEEE Internet of Things Journal*, vol. 12, no. 2, pp. 1428–1452, 2025.

[6] J. Huang, B. Wu, Q. Duan, L. Dong, and S. Yu, "A Fast UAV Trajectory Planning Framework in RIS-assisted Communication Systems with Accelerated Learning via Multithreading and Federating," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2025.

[7] J. McMahan, Y. Wu, X. Zhu, and Q. Xie, "Optimal Attack and Defense for Reinforcement Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 13, 2024, pp. 14 332–14 340.

[8] L. Hong, S. Pan, F. Feng, and C. Jiao, "Collaborative Communication for Edge LLM Servicing in Adversarial Networks: An MARL Empowered Stackelberg Game Approach," *IEEE Internet of Things Journal*, pp. 1–1, 2025.

[9] P. Huang, M. Xu, F. Fang, and D. Zhao, "Robust Reinforcement Learning as a Stackelberg Game via Adaptively-Regularized Adversarial Training," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, 2022, pp. 3099–3106.

[10] B. Zhang, L. Li, Z. Xu, D. Li, and G. Fan, "Inducing Stackelberg Equilibrium through Spatio-Temporal Sequential Decision-Making in Multi-Agent Reinforcement Learning," arXiv preprint arXiv:2304.10351, 2023, available: https://arxiv.org/abs/2304.10351.

[11] G. Xu, G. Chen, Z. Cheng, Y. Hong, and H. Qi, "Consistency of Stackelberg and Nash Equilibria in Three-Player Leader-Follower Games," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 5330–5344, 2024.

[12] W. He, H. Yao, T. Mai, F. Wang, and M. Guizani, "Three-Stage Stackelberg Game Enabled Clustered Federated Learning in Heterogeneous UAV Swarms," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 9366–9380, 2023.

[13] P. Zhao, J. Wu, F. Shi, L. Li, B. Li, and Y. Wang, "Stackelberg game and multi-agent deep reinforcement learning based peer to peer energy trading for multi-microgrids," *CSEE Journal of Power and Energy Systems*, pp. 1–12, 2023.

[14] X. Zhang, Z. He, Y. Zhu, and W. Huang, "Coupled vehicle-signal control based on Stackelberg Game Enabled Multi-agent Reinforcement Learning in mixed traffic environment," *Physica A: Statistical Mechanics and its Applications*, vol. 658, p. 130289, 2025.

[15] D. Chen, K. Chen, Z. Li, T. Chu, R. Yao, F. Qiu, and K. Lin, "PowerNet: Multi-Agent Deep Reinforcement Learning for Scalable Powergrid Control," *IEEE Transactions on Power Systems*, vol. 37, no. 2, pp. 1007–1017, 2022.

[16] J. Huang, C. Huang, C.-C. Xing, Z. Chang, Y. Zhao, and Q. Zhao, "An Energy-Efficient Communication Scheme for Collaborative Mobile Clouds in Content Sharing: Design and Optimization," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5700–5707, 2019.

[17] Yifan Zhong and Jakub Grudzien Kuba and Xidong Feng and Siyi Hu and Jiaming Ji and Yaodong Yang, "Heterogeneous-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 25, no. 32, pp. 1–67, 2024. [Online]. Available: http://jmlr.org/papers/v25/23-0488.html