IEEE Open Journal of Vehicular Technology

Received 30 November 2022; accepted 5 December 2022. Date of publication 14 December 2022; date of current version 3 January 2023. The review of this article was coordinated by Editor Lina Mohjazi.

Digital Object Identifier 10.1109/OJVT.2022.3229229

How to Attack and Defend NextG Radio Access Network Slicing With Reinforcement Learning

YI SHI [1] (Senior Member, IEEE), YALIN E. SAGDUYU [2] (Senior Member, IEEE), TUGBA ERPEK [2] (Member, IEEE), AND M. CENK GURSOY [3] (Senior Member, IEEE)

[1]Commonwealth Cyber Initiative, Virginia Tech, Arlington, VA 22203 USA
[2]National Security Institute, Virginia Tech, Arlington, VA 22203 USA
[3]Syracuse University, Syracuse, NY 13244 USA

CORRESPONDING AUTHOR: YI SHI (e-mail: yshi@vt.edu)

ABSTRACT In this paper, reinforcement learning (RL) for network slicing is considered in next generation (NextG) radio access networks, where the base station (gNodeB) allocates resource blocks (RBs) to the requests of user equipments and aims to maximize the total reward of accepted requests over time. Based on adversarial machine learning, a novel over-the-air attack is introduced to manipulate the RL algorithm and disrupt NextG network slicing. The adversary observes the spectrum and builds its own RL based surrogate model that selects which RBs to jam subject to an energy budget with the objective of maximizing the number of failed requests due to jammed RBs. By jamming the RBs, the adversary reduces the RL algorithm's reward. As this reward is used as the input to update the RL algorithm, the performance does not recover even after the adversary stops jamming. This attack is evaluated in terms of both the recovery time and the (maximum and total) reward loss, and it is shown to be much more effective than benchmark (random and myopic) jamming attacks. Different reactive and proactive defense schemes such as suspending the RL algorithm's update once an attack is detected, introducing randomness to the decision process in RL to mislead the learning process of the adversary, or manipulating the feedback (NACK) mechanism such that the adversary may not obtain reliable information are introduced to show that it is viable to defend NextG network slicing against this attack, in terms of improving the RL algorithm's reward.

INDEX TERMS NextG security, network slicing, radio access network, reinforcement learning, adversarial machine learning, jamming, wireless attack, defense.

I. INTRODUCTION

A. MACHINE LEARNING FOR NEXTG RADIO ACCESS NETWORK SLICING

Next Generation (NextG) offers major enhancements to the performance of cellular communications to meet the data rate demands of emerging applications such as virtual/augmented reality and Internet of Things. One key component of NextG communications is *network slicing* in the *radio access network* (RAN), which splits communication resources into virtual resource blocks (RBs). These RBs can be allocated dynamically to support different types of user applications and transmissions in one RB do not interfere with other RBs. These applications are categorized as enhanced Mobile Broadband (eMBB), massive machine-type communications (mMTC) and ultra-reliable low-latency communications (URLLC) based on throughput and latency requirements. Efficient and fast resource allocation by RAN slicing is critical for near-real time RAN Intelligent Controller (Near-RT RIC). The details on resource allocation as part of RAN slicing are not defined yet in the 3GPP standards. To address this gap, research activities have focused on how the resources should be allocated as part of RAN slicing [1], [2], [3], [4], [5].

Machine learning provides automated means to learn from data and optimize decision making for complex tasks. Supported by recent algorithmic and computational advances, *deep learning* can operate on raw data without hand-crafted

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

feature extraction and learn the underlying complex data representations. Therefore, deep learning has found rich applications in wireless communications such as waveform design, spectrum situational awareness, and wireless security [6]. Related to network slicing, deep learning was studied in [7] for application and device specific identification and traffic classification problems, and in [8] for management of network load efficiency and network availability. Instead of relying on the availability of training data, *reinforcement learning* (RL) has emerged as a viable solution for NextG network slicing [9], [10], [11], [12], [13], [14], [15], [16], [17] such as learning from the NextG network performance and updating resource allocation decisions for network slicing.

In this paper, we consider a NextG base station, i.e., gNodeB, as the victim system that runs an RL algorithm (as an example, the *Q-learning* algorithm) to dynamically allocate resources for NextG network slicing, where RBs are allocated to support downlink communications from the gNodeB to the user equipments (UEs). One benefit of using RL algorithm is that it does not require a pre-trained model and thus there is no delay due to training. Each network slicing request from any UE is associated with user-centric priority (weight), throughput and latency (deadline) requirements (namely, the quality of experience (QoE)), and needs to be served for a specific duration.

B. ADVERSARIAL MACHINE LEARNING BASED ATTACK ON NEXTG RADIO ACCESS NETWORK SLICING WITH REINFORCEMENT LEARNING

Blockchain was applied to design a secure decentralized spectrum trading platform for network slicing [58], [59]. However, blockchain cannot protect the RL algorithms for network slicing from jamming attacks. Due to the broadcast nature of wireless communications, an adversary can overhear and jam transmissions. As a consequence, the adversary can launch a *jamming attack* on RBs. Security issues for machine learning based network slicing were discussed in [60], [61] but these work did not discuss security issues for the RL algorithms. Separate from NextG network slicing, attacks on RL algorithms have been considered in [18], [19], [20] for medium access with a jammer that can jam one channel over one time block only.

In this paper, we consider allocation of potentially multiple channels to different users over a time horizon for the NextG network slicing problem. If an RB is assigned to a network slicing request and is jammed by the adversary, this request cannot achieve the required QoE and is considered as a failure. The reward of this request becomes zero, i.e., the performance of the gNodeB is reduced under attack. Moreover, this reward is given as the input (along with the state) to the gNodeB's RL algorithm. Therefore, this algorithm is confused and will predict the existence of jamming attacks even if there is no attack. Thus, such a jamming attack not only affects the gNodeB's current performance but also affects its future performance even after the adversary stops jamming RBs. On the other hand, RL can recover from the attack over a period of time by collecting correct feedback once the attack stops and updating its algorithm. To measure the performance of this attack (in terms of its effect on NextG network slicing), we compute the *recovery time*, which is the time period from when the jamming attack stops to when the gNodeB's performance is back to normal (i.e., to the level before the attack starts), as well as the maximum and total reduction in the RL algorithm's reward during the recovery time.

We impose the practical constraint that the adversary has limited transmit power and thus cannot jam all RBs due to its *energy budget*. Then, the adversary needs to carefully select which RBs to jam with the objective of maximizing the impact of jamming on network slicing requests (namely, the number of failed network slicing requests). One potential attack strategy is *myopic*, which aims to jam some RBs to maximize the instantaneous impact of the attack without consideration of future impact. This strategy cannot work well as an online algorithm in general. Moreover, our results show that this rather simple strategy can be learned by the gNodeB's RL algorithm and thus its impact can be mitigated over time by the usual RL algorithm updates.

To maximize the impact of jamming the RBs, we pursue an *adversarial machine learning* approach. Different types of attacks built upon adversarial machine learning have been studied in wireless communications [21], [22] such as exploratory (inference) attacks [23], [24], evasion (adversarial) attacks [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39] and their extensions to secure and covert communications against eavesdroppers [40], [41], [42], causative (poisoning) attacks [43], [44], [45], membership inference attacks [46], [47], Trojan attacks [48], and spoofing attacks [49], [50], [51] that have been launched against various spectrum sensors and wireless signal (such as modulation) classifiers. Adversarial machine learning has also been considered for NextG by studying evasion and spoofing attacks on deep neural networks (without reinforcement learning) used for NextG spectrum sharing and NextG signal authentication [52]. In addition, flooding attacks have been considered for NextG network slicing with reinforcement learning [53].

In this paper, a jamming attack built upon adversarial machine learning is launched against the RL agent that performs resource allocation for NextG network slicing, and the attack exploits the unique properties that (i) the RL algorithm is affected by manipulated rewards and (ii) it takes a while for the RL algorithm to recover even after the attack stops.

The *states* of the *surrogate* RL model built by the adversary correspond to the availability of RBs, which are determined by passively sensing the RBs (since the RBs that are allocated to a user request are used for communications and thus are sensed as busy). Note that the adversary does not have access to the victim's RL model, namely it launches a *black-box attack*, and cannot obtain the availability of RBs by querying the model with inputs.

The *actions* of the adversary are the set of selected RBs to be jammed. We assume that the UEs send a negative acknowledgment (NACK) to confirm a failed transmission from

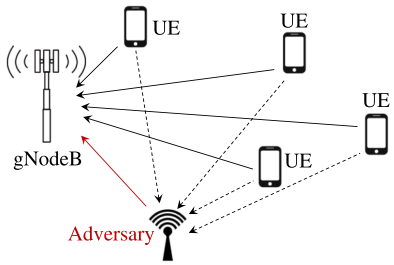

FIGURE 1. The interaction of the victim RL algorithm and the adversarial surrogate RL algorithm. The solid black lines from the UEs to the gNodeB represent the control messages sent for a failed network slicing request. The dashed lines from the UEs to the adversary represent the control messages for a failed network slicing request heard at the adversary. The adversary decides on its attack strategy based on its environment sensing results.

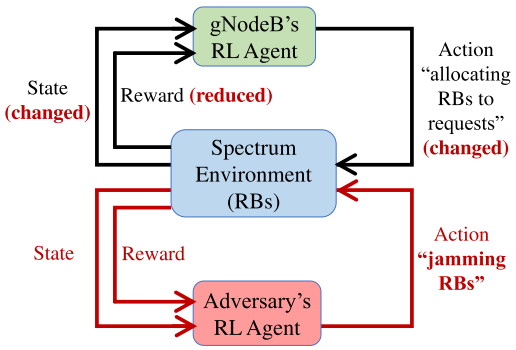

FIGURE 2. Illustration of how jamming built upon adversarial machine learning manipulates the RL process of the gNodeB to allocate the RBs to network slicing requests.

the gNodeB (so that it can be retransmitted later subject to its deadline for reliable communications) and the adversary needs to detect the presence of this feedback without decoding it, as shown in Fig. 1. Typically, the NACK message has a particular pattern: it has a short packet length and it follows data transmission after a fixed time lag. Therefore, it is not difficult to detect the presence of NACK transmissions.

The *reward* of the adversary's RL algorithm is the number of jammed and therefore failed requests. The RL algorithm at the adversary can learn the effect of its attack and update its RL model (in our example, the Q-table). Once the RL model is well trained, the adversary can make the optimal decision on selecting which RBs to jam by maximizing its expected jamming reward. Note that in this attack scenario, the adversary launches an *over-the-air attack* and indirectly manipulates the reward of the RL algorithm by jamming the RBs, as shown in Fig. 1. The interactions between the victim RL algorithm of the gNodeB and the surrogate RL algorithm of the adversary are illustrated in Fig. 2.

In performance evaluations, we compare the RL based attack with the *myopic* attack and *random jamming* (namely, jamming randomly selected RBs) subject to the same jamming budget constraint. We show that the RL based attack can achieve the largest reduction in the reward of the gNodeB's

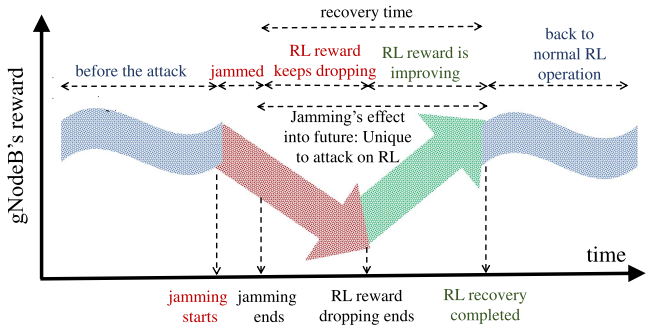

FIGURE 3. Adversarial machine learning for manipulating the RL process of the gNodeB when it allocates the RBs to network slicing requests.

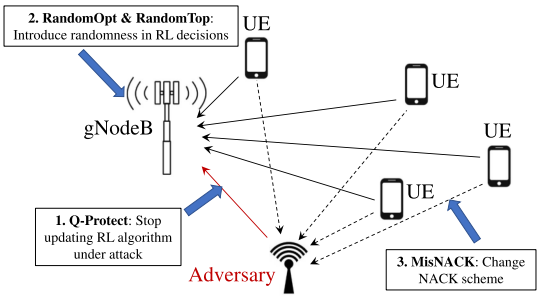

FIGURE 4. The defense schemes against RL based attack on NextG RAN slicing.

RL algorithm (under attack and after attack) and the longest recovery time from the attack (after the jamming attack stops). This result demonstrates the adversarial machine learning benefits of manipulating the RL process over a time horizon. As illustrated in Fig. 3, *the extension of the attack's impact beyond the time instant when the attack stops is a key capability of the RL based jamming attacks compared to conventional jamming attacks (on data transmissions) whose impact is typically limited to the duration of the attack* (see [54], [55] for examples on conventional jamming attacks on wireless communications).

C. DEFENSE AGAINST ADVERSARIAL MACHINE LEARNING BASED ATTACK ON NEXTG RADIO ACCESS NETWORK SLICING WITH REINFORCEMENT LEARNING

In this paper, we also investigate how to *defend* the network slicing operations against the RL based jamming attacks. For that purpose, we introduce three different defense schemes, Q-Protect, RandomOpt/RandomTop, and MisNACK, for the gNodeB or the UE to take (illustrated in Fig. 4):

1) Q-Protect protects the RL algorithm itself by suspending the RL algorithm (i.e., Q-table) update once an attack is detected to avoid the impact of the attack on the RL algorithm;
2) RandomOpt and RandomTop introduce randomness to the decision process in RL (in particular, add perturbations in the Q-table updates) to mislead the learning process of the adversary;

3) MisNACK manipulates the feedback (NACK) mechanism such that the adversary may not obtain reliable information to build its attack strategy.

We show that the second defense schemes is more effective than others and can be combined with others to help network slicing operations sustain its performance relative to the case without an attack.

D. CONTRIBUTIONS AND PAPER ORGANIZATION

The contributions of this paper are summarized as follows.

- For an RL based NextG RAN slicing algorithm, we design a novel RL based attack scheme built upon adversarial machine learning that selectively jams the available RBs so that the RL algorithm for network slicing receives incorrect reward (feedback) and updates itself in a wrong way, thereby leading to a significant performance loss of resource allocation for NextG RAN slicing.
- We design novel defense schemes by considering various characteristics of RL algorithms. The Q-Protect scheme stops the RL algorithm if the reward is unexpected. The RandomOpt and RandomTop schemes make it more challenging for an adversary to learn. The MisNACK scheme provides incorrect information to the adversary.
- We show the effectiveness of the designed attack and defense schemes using different benchmarks in numerical results. Our results show that the RL based attack scheme achieves better attack performance than benchmark attack schemes, and a combined scheme with multiple defense schemes achieves the best protection.

The rest of the paper is organized as follows. Section II describes resource allocation for network slicing via RL. Section III presents the RL based jamming attack that aims to maximize the impact on the gNodeB's performance under the attack and after the attack. Section IV introduces defense schemes to protect the network slicing operations from RL based jamming attacks. Section V evaluates the attack and defense performances. Section VI concludes this paper.

II. THE VICTIM SYSTEM TO ATTACK: REINFORCEMENT LEARNING BASED RESOURCE ALLOCATION FOR NETWORK SLICING

In this section, we summarize the NextG RAN slicing setting that an adversary aims to attack. We follow the RL formulation of [14] for network slicing as an example, while the attack and defense schemes that we consider in the next two sections apply to other RL based NextG RAN slicing settings (e.g., [9], [57]), as well. As depicted in Fig. 5, we consider a general scenario in which multiple NextG UEs send requests over time with different QoE requirements, i.e., rate, latency (deadline) and lifetime demands and priority weights, and the gNodeB needs to allocate the RBs to selected requests such that the total weight of served requests over a time period can be maximized. If a request is not granted, it will be kept in a waiting list until its deadline expires. There is also an adversary that we will describe in Section III.

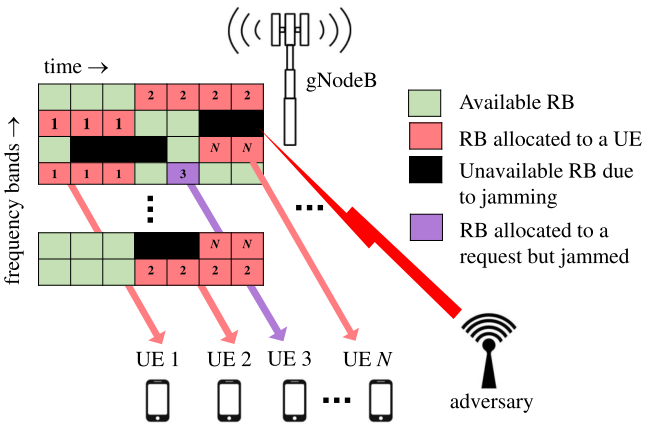

FIGURE 5. System model for NextG network slicing in the presence of an adversary.

At time slot t, there are a set of active requests $A(t)$ (requests that have just arrived or are in the waiting list). UE i's QoE requirement of rate for its request j is given by

$$D_{ij} \geq d_{ij}, \quad (i, j) \in A(t), \tag{1}$$

where D_{ij} is the achieved downlink data rate and d_{ij} is the minimum required rate. D_{ij} is determined by the assigned bandwidth F_{ij} in an RB and the modulation/coding scheme used for communications between the gNodeB and UE i. The data rate (bps) is approximated as [56]:

$$D_{ij} = c\, K_{ij}\, (1 - BER_{ij}), \quad (i, j) \in A(t), \tag{2}$$

where K_{ij} is the number of aggregate component carriers in a band combination and BER_{ij} is the bit error rate of UE i for its request j (which depends on the signal-to-noise ratio (SNR) and is computed for additive white Gaussian noise (AWGN) channel with low-density parity-check coding), and constant c is approximately 12.59×10^6 when a single-antenna UE uses quadrature phase shift keying (QPSK) modulation, 60 kHz subcarrier spacing and 10 MHz bandwidth. The data rate equation provided in [56] can be modified accordingly if different configuration parameters are used.

The constraints of resource assignments to network slices are given by

$$\sum_{(i,j)\in A(t)} F_{ij}\, x_{ij}(t) \leq F(t), \tag{3}$$

where F_{ij} is the assigned bandwidth and $F(t)$ represents the available communication resources (RBs) of the gNodeB at time t (resources that are assigned previously to some requests and not terminated yet become temporarily unavailable) and $x_{ij}(t)$ is the binary indicator on whether UE i's request j is satisfied at time t.

By considering the optimization problem for a time horizon, the resources are updated from time $t-1$ to time t as

$$F(t) = F(t-1) + F_r(t-1) - F_a(t-1), \tag{4}$$

where $F_r(t-1)$ and $F_a(t-1)$ are the released and allocated resources on frequency at time $t-1$, respectively. Each request has a lifetime l_{ij} and if it is satisfied at time slot t (namely, the service starts in time slot t), this request will end at the end of time slot $t+l_{ij}-1$. The released and allocated resources at time t are given by

$$F_r(t) = \sum_{(i,j)\in R(t)} F_{ij}, \tag{5}$$

$$F_a(t) = \sum_{(i,j)\in A(t)} F_{ij} x_{ij}(t), \tag{6}$$

where $R(t)$ denotes the set of requests ending (completed or expired) at time t. Then, the optimization problem is given by

$$\max_{x_{ij}(t)} \sum_{t} \sum_{(i,j)\in A(t)} w_{ij}\, x_{ij}(t), \tag{7}$$

subject to (1)–(6), where w_{ij} is the weight for UE i's request j to reflect its priority.

As a model-free RL algorithm, we use Q-learning to learn the policy that determines which action (resource assignment) to take under a given state (available resources and requests) for the gNodeB. The gNodeB applies Q-learning to compute the function $Q: S \times A \to \mathbb{R}$ (maintained as the Q-table) to evaluate the quality of action A producing reward R at state S. At each time t, the gNodeB selects an action a_t, observes a reward r_t, and transitions from the current state s_t to a new state s_{t+1} (this transition depends on current state s_t and action a_t), and updates Q.

Initializing Q as a random matrix and using the weighted average of the old value and the new information, Q-learning performs the value iteration update for Q as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right), \tag{8}$$

where α is the learning rate ($0 < \alpha \leq 1$) and γ is the discount factor ($0 \leq \gamma \leq 1$) for rewards over time. As the size of the states increases, it becomes computationally more efficient to approximate the Q-function by training a deep neural network, leading to a deep Q-network formulation.

In dynamic resource allocation to network slices, the reward at time t is w_{ij} if UE i's request j is satisfied at time t, i.e., $x_{ij}(t) = 1$. Note that the reward measures the satisfied QoE demands of network slices and therefore it indirectly reflects the achieved QoE performance such as throughput and delay.

An action is to assign resources to a request at time t. Multiple actions can be taken at the same time instance. The states at t are F binary variables on the availability of F RBs and (F_{ij}, w_{ij}) for a request under consideration. The state transition at time t is driven by allocating resources for requests granted at time t and releasing resources after lifetimes of some active services expire at time t. In particular, the state transitions are given by (4)–(6). The states, actions, and rewards of the RL algorithm for network slicing are summarized in Table 1. The standard Q-learning algorithm of (8) is considered. The Q-table size mainly depends on the number of RBs. For a large number of RBs, the Q-table size can be very large and thus a deep Q-network (DQN) can be applied to reduce the algorithm complexity. The notation used in this paper is shown in Table 2.

TABLE 1 RL Algorithm for Network Slicing

RL term	Specification (at any given time instant)
State	Availability of RBs, an active request
Action	Assign RBs if the request is selected
Reward	The weight of the request if it is selected, 0 otherwise

TABLE 2 Notation Table

Symbol	Definition
a_t	Action at time slot t
$A(t)$	Set of active requests at time t
B	Maximum number of RBs that the adversary can jam at any given time
BER_{ij}	Bit error rate for UE i's request j
d_{ij}	Minimum required rate for UE i's request j
D_{ij}	Achieved downlink data rate for UE i's request j
F	Number of all RBs
F_{ij}	Assigned RBs for UE i's request j
$F(t)$	Available RBs of the gNodeB at time t
$F_r(t)$	Released RBs on frequency at time t
$F_a(t)$	Allocated RBs on frequency at time t
l_{ij}	Lifetime of UE i's request j
$r(t)$	Reward at time t
r_{top}	Percentage to determine whether a reward is considered as top reward or not
$R(t)$	Set of requests ending at time t
$s(t)$	State at time t
w_{ij}	Weight for UE i's request j to reflect its priority
$x_{ij}(t)$	Binary indicator on whether UE i's request j is satisfied at time t
α	Learning rate of the Q-learning algorithm
γ	Discount factor for rewards over time

III. ATTACK ON REINFORCEMENT LEARNING FOR NEXTG NETWORK SLICING

We now consider an adversary that attacks an RL algorithm for NextG RAN slicing, e.g., the one discussed in Section II. Other example victim systems include the RL based network slicing schemes in [9], [57].

A. REINFORCEMENT LEARNING BASED ATTACK

Since RL keeps collecting data and updating itself, it has *two unique properties* that we leverage to build and evaluate attacks on RL.

1) If an adversary changes the state or the reward, it can affect the RL algorithm.
2) On the other hand, if the adversary stops attacking, the RL algorithm will recover by itself.

In this section, we exploit the first property to design the attack on the RL algorithm of the NextG network slicing. As this attack can still affect the RL significantly even after the attack stops for a while, we measure the impact due to the second property in Section V.

To launch an attack, the adversary can change either the state or the reward of the RL agent. For the RL algorithm presented in Section II, the state includes the RB availability and a request under consideration. Both are maintained by the gNodeB. Therefore, they cannot be changed by the wireless adversary that is physically separated from the gNodeB and does not have direct access to the gNodeB's RL algorithm. On the other hand, the adversary can affect the reward if it jams an RB to be allocated to a request. In that case, the request will not be successful even if resources are allocated by the RL algorithm and there is no reward gained by the RL algorithm.

We assume a practical constraint that the adversary has *limited jamming capability* (typically due to limited energy budget) and thus cannot jam all RBs to maximize its impact. We denote B as the maximum number of RBs that the adversary can jam at any given time. Due to this constraint, it is important for the adversary to select the RBs that are available and likely to be allocated such that jamming these RBs can affect network slicing requests to be selected by the RL algorithm.

The ideal case is that the adversary can build a *surrogate model* (another RL algorithm) that can predict which RBs will be allocated and then use the predicted results to decide which RBs should be jammed. However, this case is impractical since (i) the request under consideration is a part of the state, which is unknown to the adversary, and (ii) the reward is the request's weight, which is unknown to the adversary. Therefore, the adversary builds a different RL model (as an approximate surrogate model). Although the RL algorithm is the same as that discussed in Section II, this RL model of the adversary has different state, action, and reward properties given as follows.

- The *state* is the set of binary variables that indicate the availability of all RBs.
- An *action* corresponds to selecting the set of $\min\{B, F(t)\}$ RBs from $F(t)$ available RBs, and jamming those selected RBs. Note that there is also the action of not jamming any RB. Thus, the number of possible actions is $C^B_{F(t)} + 1$ (where $C^B_{F(t)}$ is the number of B-combinations from a set of $F(t)$ elements, i.e., the number of possibilities in picking B out of $F(t)$) if $F(t) > B$, or 2 (jam or not) if $F(t) \leq B$.
- The *reward* is the number of jammed requests at a given time. We assume that there is a NACK transmitted from a NextG UE at the end of a time slot if the transmission is not successful. If the adversary jams an RB and later observes the NACK, the reward on this channel is 1. Note that the adversary does not need to decode the NACK. It needs to detect the presence of NACK only, which is possible by distinguishing the NACK from data transmissions (as the NACK is shorter than data portion and has the structure of appearing between requests and data transmissions).

To initialize the Q-table, we set entries in the column of no jamming to zeros and entries in other columns to the number of jammed RBs.

TABLE 3 The Adversary's RL Algorithm

RL term	Specification (at any given time instant)
State	Availability of RBs
Action	Jam selected RBs
Reward	Number of jammed requests

The adversary applies RL to update its Q-table by (8) and to take actions based on its Q-table. The states, actions, and rewards of the adversary's RL algorithm are summarized in Table 3. The Q-table size depends on the number of RBs. If the number of RBs is large, we can apply DQN instead to reduce the algorithm complexity.

B. PERFORMANCE METRICS AND BENCHMARK ATTACK SCHEMES

When the adversary launches its attack, we can observe the performance reduction of the gNodeB by comparing it with the case of no attack. The reason for the performance loss is that some requests fail due to jamming and thus their weights are not counted in the reward of the gNodeB.

More interestingly, since some rewards are changed by jamming the RBs and the gNodeB's RL algorithm is updated based on these changed rewards, the attack also affects the RL algorithm itself. As a result, even if the adversary stops jamming the RBs, the performance of NextG network slicing cannot return to previous levels (before the attack) right away. Instead, it takes some time for the gNodeB to collect sufficient data to correct its algorithm and then finally its performance can go back to the case when there is no attack. To measure this impact after the attack stops, we consider the following metrics.

- *Recovery time:* The time it takes (after the attack, namely jamming, stops) for the network slicing performance (namely, the reward) to go back to "normal" (the level before the attack). The recovery time is an important metric since if it is long, the adversary can stop its attack to avoid being detected or to save energy and then start its attack again before the recovery time.
- *Maximum performance reduction:* The maximum gap in performance compared to the normal (before-the-attack) value during the recovery time. The performance is measured as the running averaged reward. The maximum performance reduction describes the maximum impact during the recovery time.
- *Total performance reduction:* The accumulated performance gap to the normal value during the recovery time. The total performance reduction is a more robust metric than the above two, since it is not affected by small performance reduction (comparing with recovery time) or single extreme point (comparing with maximum performance reduction).

In addition to this attack, we also consider the case of no attack and two benchmark attacks, namely random attack and myopic attack, for performance evaluation:

- *Random attack:* The adversary randomly jams some RBs (that are uniformly randomly selected from all RBs) subject to the jamming budget.
- *Myopic attack:* The adversary selects which RBs to jam (subject to the jamming budget) with the objective of maximizing the instantaneous reward without the consideration of future rewards.

Note that the proposed RL based attack takes time to improve its attack actions as its RL algorithm learns how to attack NextG RAN slicing. Other attacks schemes do not have this of process of gradual improvement. As we measure the recovery time, maximum and total performance reduction over the same period of time (including the warm-up time) for all attack schemes, we provide a fair comparison of RL based attacks with random and myopic attack. The performance of these attacks is evaluated in Section V.

IV. DEFENSE AGAINST ATTACKS ON REINFORCEMENT LEARNING FOR NEXTG NETWORK SLICING

To protect the RL based resource allocation for NextG RAN slicing (e.g., [9], [14], [57]) from the RL based jamming attacks, we present different defense schemes (illustrated in Fig. 4) for the gNodeB or the UE to take.

1) *Q-Protect:* One *reactive* defense scheme is based on protecting the RL algorithm itself. Note that if there is no attack, once a network slicing request is served, some reward is expected. However, if the RBs that are allocated to this request are jammed, this request cannot be satisfied and therefore its reward is reduced to zero. Thus, the gNodeB can detect the jamming attack by checking the changes in the reward. For numerical results, we assume that the attack is detected if the running average of the rewards drops by 10%. Hence, the gNodeB suspends the Q-table update once an attack is detected to avoid the impact of the attack on the RL based network slicing algorithm. We call this defense scheme "Q-Protect," which can be applied to any RL algorithm. The adversary cannot force the gNodeB to update its RL algorithm and thus cannot circumvent this defense.
2) *RandomOpt* and *RandomTop:* A *proactive* defense scheme aims to manipulate the adversary's learning process (namely, its surrogate model). This defense scheme can be effective against any learning-based attack. However, it cannot protect network slicing from random jamming attacks. The gNodeB can proactively introduce randomness to the resource allocation actions in its RL algorithm such that an adversary cannot easily learn how to build its RL algorithm. We propose two defense schemes, with and without performance loss when there is no attack
   a) Note that there may be multiple best actions with the same reward in the Q-table. Then, the gNodeB can randomly select any action without any performance loss.[1] We call this defense scheme "RandomOpt," which can be applied to any algorithm that can find multiple optimal actions.
   b) The randomness among best actions may not be sufficient to mitigate the performance loss due to the attack. Another defense scheme is to randomly select an action from top actions (those with rewards that are close to the best reward). An action is considered as "Top" if its reward is at least r_{top} percentage of the maximum reward. This defense scheme introduces more randomness but may incur performance loss even if there is no attack. We call this defense scheme "RandomTop," which can be applied to any algorithm that can find multiple near-optimal actions. The adversary cannot remove the randomness introduced by the defender and thus cannot circumvent these two defense schemes.
3) *MisNACK:* Another *proactive* defense scheme aims to manipulate the feedback (NACK) mechanism such that the adversary may not obtain reliable information to build its attack strategy. We note that the UE sends a NACK over any jammed RB if some of its RBs are jammed. That is, there is one NACK transmitted for each failed request. The adversary monitors the jammed RBs to detect the presence of NACK transmission and thus defines the reward of its action. As a defense, each UE can send the NACK over an unjammed RB (if any) such that no NACK can be detected by the adversary that monitors only the channel that it has jammed. If all its RBs are jammed, the UE can send multiple NACKs over these RBs such that the adversary will overestimate the effect of its attack. This way, the adversary reduces the reliability of NACK for the adversary. We call this defense scheme "MisNACK," which can be applied to any algorithm that uses NACK. The adversary cannot force UEs not to send misleading NACKs and thus cannot circumvent this defense.

The performance of these defense schemes is evaluated in Section V.

V. PERFORMANCE EVALUATION

Suppose that the gNodeB receives requests from 30 UEs. For each UE, requests arrive with the rate of 0.05 per slot. Here, a slot corresponds to each time block which is 0.23 ms long with 60 kHz subcarrier spacing. For each request, the weight of a request is assigned (uniformly) randomly in [1,5], the lifetime is assigned randomly in [1,10] slots, and the deadline is assigned randomly in [1,20] slots. The maximum received SNR is selected randomly from [1.5,3]. The total frequency is 10 MHz including guard bands and is split into 11 bands, i.e.,

[1]To simplify discussion, we assume that the Q-table is perfect and thus the same reward in the Q-table means the same long-term reward in the objective. In reality, the Q-table may not be perfect and thus there can still be performance loss under this policy.

TABLE 4 Performance Comparison of Q-Learning and Other Attacks When There are 11 RBs

Attack scheme	Maximum jammed RBs	Recovery time	Maximum reduction in reward	Total reduction in reward
Q-learning	1	1038	1.447	736.216
	2	1191	1.801	911.604
	3	1548	1.957	1006.174
	4	2086	2.014	1038.988
	5	2038	2.714	1410.069
Myopic	1	1035	1.343	670.071
	2	1060	1.587	788.289
	3	1028	1.684	836.998
	4	1207	1.775	894.721
	5	1365	1.772	889.113
Random	1	1197	1.000	506.947
	2	1233	1.489	750.976
	3	1170	1.813	907.546
	4	1180	2.061	1032.088
	5	1202	2.273	1141.359

there are 11 RBs. We also consider a scenario with a smaller number of RBs, namely 5 RBs.

A. ATTACK PERFORMANCE EVALUATION

The same scenario over 1000 time slots is repeated to evaluate these attacks. For Q-learning, we set the discount factor as $\gamma = 0.95$ and the learning rate as $\alpha = 0.1$.

We assume that the adversary launches its attack over 10000 slots. The benchmark of no attack case is also run over 10000 slots in total and the achieved reward is measured as 3.032 over the first 1000 slots (and this is used as the benchmark for recovery). Then, we measure the average reward over the past 1000 slots after the attack stops and once this average reward reaches 3.032, namely when the system performance is assumed to recover from the attack. We also measure the performance gap to the benchmark and present results on the maximum gap and the total gap during the recovery time.

For comparison purposes, we obtain results for attacks by random and myopic jamming attacks in Table 4, where results for random jamming are averaged over 20 runs. The RL based attack (introduced in Section III-A) has longer and larger impact on the NextG network slicing performance than other attacks, which means that the RL based attack has better performance. Depending on the maximum number of jammed RBs, Q-learned based attack increases the recovery time by up to 77%, increases the maximum reduction in reward up by to 53%, and increases the total reduction in reward by up to 59% compared to benchmark attack schemes. We show in Fig. 6 how the reward changes over time after the attack stops when the maximum number of jammed RBs is 5. The advantage of RL based attack comes from the smallest reward when the attack stops. Thus, we also check the RL algorithm's reward under different attacks (see Fig. 7). Since we show the average reward over the past 1000 slots, the performance is high at the beginning and decreases fast. Then, the performance under random jamming remains still small while the performance under myopic jamming keeps increasing. This is because the myopic algorithm is deterministic and thus it is

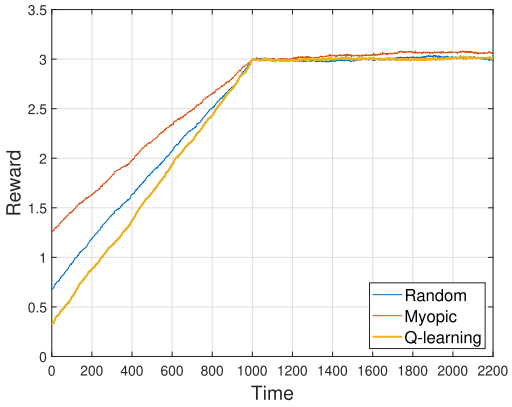

FIGURE 6. The reward of RL algorithm for NextG RAN slicing after the attack stops when there are 11 RBs.

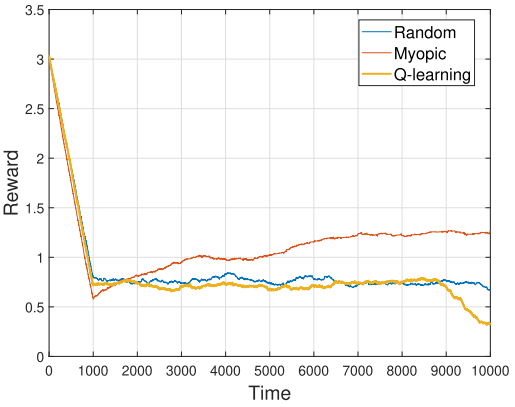

FIGURE 7. The reward of RL algorithm for NextG RAN slicing under the attack when there are 11 RBs.

easy to learn and mitigate it by the RL algorithm for NextG RAN slicing. There is another decrease for the performance under Q-learning based jamming at time slot 9000, where in addition to failed requests due to jamming, the RL algorithm for network slicing also starts making errors in selecting requests. That is, the RL algorithm for network slicing receives the wrong reward due to attack and updates itself incorrectly. Then, it starts to make wrong decisions on whether to select a request or not.

Next, we evaluate the performance when the number of RBs is reduced from 11 to 5 (other parameters remain the same). Results are shown in Table 5. As before, the Q-learning based attack has longer and larger impact on the performance than benchmark attacks. Figs. 8 and 9 show the reward over time after the attack stops and under the attack, respectively. The trends in Figs. 8 and 9 are the same as the trends observed in Figs. 6 and 7 when there are 11 RBs. Note that due to the smaller problem size, the almost flat period between two decreases of network slicing performance under the RL based attack is much shorter, i.e., the start of the second decrease is at about time slot 3200. On the other hand, we may increase the number of RBs. We find that the Q-table size will be

TABLE 5 Performance Comparison of RL Based and Other Attacks When There are 5 RBs

Attack algorithm	Maximum jammed RBs	Recovery time	Maximum reduction in reward	Total reduction
Q-learning	1	990	0.476	253.221
	2	1100	0.799	418.045
Myopic	1	925	0.370	175.576
	2	992	0.583	283.019
Random	1	1029	0.403	199.924
	2	1003	0.620	308.266

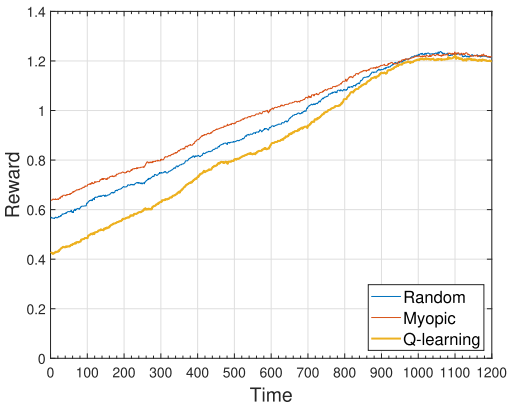

FIGURE 8. The reward of RL algorithm for NextG RAN slicing after the attack stops when there are 5 RBs.

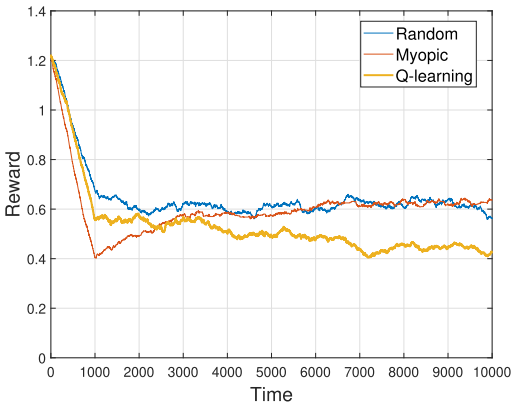

FIGURE 9. The reward of RL algorithm for NextG RAN slicing under the attack when there are 5 RBs.

increased by the second order of the number of RBs. A large Q-table requires both long training time and large memory usage. It would be better to design a solution using deep Q-learning instead.

B. DEFENSE PERFORMANCE EVALUATION

We now present the performance of different defense schemes (described in Section IV) against the RL based attack. For RandomTop, an action is considered as "Top" if its reward is at least $r_{\text{top}} = 50\%$ of the maximum reward. The recovery time for the no defense case and all defense schemes is

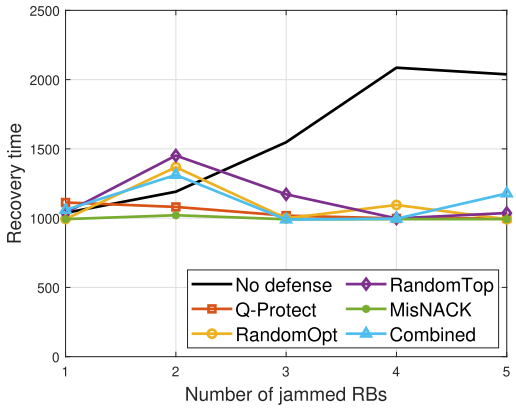

FIGURE 10. Recovery time of RL algorithm for NextG RAN slicing under different attacks.

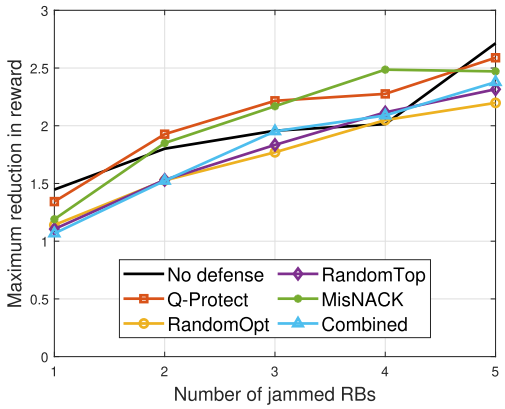

FIGURE 11. Maximum reduction in the reward of RL algorithm for NextG RAN slicing under different attacks.

shown in Fig. 10, where the "Combined" scheme combines different defense schemes ("Q-Protect," "RandomTop," and "MisNACK") and apply them jointly to strengthen the overall defense against the RL based attack on NextG network slicing. In particular, "Q-Protect" aims to protect the Q-table while both "RandomTop" and "MisNACK" aim to attack the adversary's learning process, and thus they all can be combined. Compared with the no attack case, all the defense schemes reduce the recovery time if the number of jammed RBs is at least three. The improvement when there is one jammed RB is not significant. The random effect in "RandomOpt" and "RandomTop" makes them worse than the no defense case if the number of jammed RBs is 2. In fact, although it takes long time to recover, the amount of reduction in reward is not large. Thus, we further study the reduction in reward, in terms of the maximum reduction and the total reduction, during the recovery period.

The maximum reduction in reward for the no defense case and all defense schemes are shown in Fig. 11. Compared with the no attack case, the "RandomOpt," "RandomTop," and "Combined" schemes achieve smaller reduction in most of the cases.

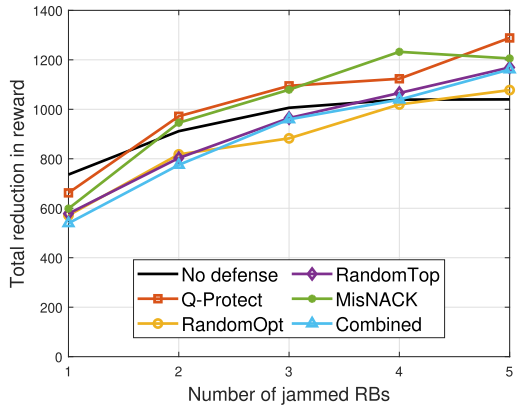

FIGURE 12. Total reduction in the reward of RL algorithm for NextG RAN slicing under different attacks.

TABLE 6 Performance by the Combined Defense of "q-Protect," "Random Top," and "MisNACK" Schemes Under the RL Based Attack When There are 5 RBs

Maximum jammed RBs	Recovery time	Maximum reduction in reward	Total reduction
1	986	0.536	266.581
2	918	0.679	320.966

The total reduction in reward for the no defense case and defense schemes is shown in Fig. 12. Compared with the no attack case, the "RandomOpt," "RandomTop," and "Combined" schemes achieve smaller reduction for most cases.

In summary, the "Combined" scheme achieves better defense performance than other defense schemes for most of cases. Therefore, we evaluate the performance of the "Combined" scheme in further detail. The performance when there are 5 RBs is shown in Table 6. Compared with the results in in Table 5, we note that the "Combined" scheme can improve the performance for NextG RAN slicing, and this observation holds for both cases with 5 and 11 RBs.

VI. CONCLUSION

In this paper, we studied the security vulnerability of NextG network slicing by designing a jamming attack on the underlying RL operations for resource allocation. Although RL is an efficient solution to optimally allocate network resources (RBs at the NextG gNodeB) for communication requests from NextG UEs, the broadcast nature of wireless communications makes the NextG RAN vulnerable to jamming attacks. In particular, if an RB is assigned to a request and is jammed by an adversary, that request cannot be satisfied and the associated reward becomes zero. This reward is used as input to the gNodeB's RL algorithm and thus its performance starts deteriorating. Even after the adversary stops jamming, the gNodeB's performance cannot be recovered until its algorithm is updated by a sufficient number of feedback messages.

To select the RBs for jamming, the adversary builds a surrogate RL model to maximize the number of jammed requests over time subject to an energy budget (namely, a constraint on the number of channels that can be jammed simultaneously). We showed that such an algorithm is highly effective to reduce the gNodeB's performance, even after the adversary stops attacking. We compared this attack with other attack benchmarks such as random jamming and myopic jamming (that aims to maximize the instantaneous number of jammed RBs) and showed that the RL based jamming attack is more effective than both random or myopic jamming.

To protect network slicing against RL based jamming attacks, we introduced several defense schemes such as suspending the Q-table updates when an attack is detected, introducing randomness into network slicing decisions or manipulating the feedback mechanism in network slicing to mislead the learning process of the adversary. We showed that these defense schemes can be effectively combined to defend network slicing by fooling the adversary into making wrong decisions and reducing its impact.

ACKNOWLEDGMENT

This effort is supported in part by the U.S. Army Research Office under contract W911NF-17-C-0090. The content of the information does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred. This effort is also supported in part by the Commonwealth Cyber Initiative, an investment in the advancement of cyber RD, innovation, and workforce development. For more information about CCI, visit www.cyberinitiative.org.

REFERENCES

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[2] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, May 2017.

[3] P. Rost et al., "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017.

[4] A. Kaloxylos, "A survey and an analysis of network slicing in 5G networks," *IEEE Commun. Standards Mag.*, vol. 2, no. 1, pp. 60–65, Mar. 2018.

[5] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia, "The slice is served: Enforcing radio access network slicing in virtualized 5G systems," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 442–450.

[6] T. Erpek, T. O'Shea, Y. E. Sagduyu, Y. Shi, and T. C. Clancy, "Deep learning for wireless communications," in *Development and Analysis of Deep Learning Architectures*, Cham, Switzerland: Springer, 2020, pp. 223–266.

[7] A. Nakao and P. Du, "Toward in-network deep machine learning for identifying mobile applications and enabling application specific network slicing," *IEICE Trans. Commun.*, vol. 101, pp. 1536–1543, 2018.

[8] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "DeepSlice: A deep learning approach towards an efficient and reliable network slicing in 5G networks," in *Proc. IEEE 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf.*, 2019, pp. 0762–0767.

[9] R. Li et al., "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018.

[10] J. Koo, V. B. Mendiratta, M. R. Rahman, and A. Walid, "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics," in *Proc. 15th Int. Conf. Netw. Serv. Manage*, 2019, pp. 1–5.

[11] H. Wang, Y. Wu, G. Mina, J. Xu, and P. Tang, "Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach," *Inf. Sci.*, vol. 498, pp. 106–116, 2019.

[12] Q. Liu and T. Han, "When network slicing meets deep reinforcement learning," in *Proc. Int. Conf. Emerg. Netw. Exp. Technol.*, 2019, pp. 29–30.

[13] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs," *IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[14] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Reinforcement learning for dynamic resource optimization in 5G radio access network slicing," in *Proc. IEEE 25th Int. Workshop Comput. Aided Model. Des. Commun. Links Netw.*, 2020, pp. 1–6.

[15] Y. Shi, P. Rahimzadeh, M. Costa, T. Erpek, and Y. E. Sagduyu, "Deep reinforcement learning for 5G radio access network slicing with spectrum coexistence," 2021. [Online]. Available: https://doi.org/10.36227/techrxiv.16632526.v1

[16] A. Nassar and Y. Yilmaz, "Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 222–235, 2021.

[17] K. Suh, S. Kim, Y. Ahn, S. Kim, H. Ju, and B. Shim, "Deep reinforcement learning-based network slicing for beyond 5G," *IEEE Access*, vol. 10, pp. 7384–7395, 2022.

[18] F. Wang, C. Zhong, M. C. Gursoy, and S. Velipasalar, "Adversarial jamming attacks and defense strategies via adaptive deep reinforcement learning," 2020, *arXiv:2007.06055*.

[19] C. Zhong, F. Wang, M. C. Gursoy, and S. Velipasalar, "Adversarial jamming attacks on deep reinforcement learning based dynamic multichannel access," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2020, pp. 1–6.

[20] F. Wang, C. Zhong, M. C. Gursoy, and S. Velipasalar, "Defense strategies against adversarial jamming attacks via deep reinforcement learning," in *Proc. IEEE 54th Annu. Conf. Inf. Sci. Syst.*, 2020, pp. 1–6.

[21] Y. E. Sagduyu et al., "When wireless security meets machine learning: Motivation, challenges, and research directions," 2020, *arXiv:2001.08883*.

[22] D. Adesina, C.-C. Hsieh, Y. E. Sagduyu, and L. Qian, "Adversarial machine learning in wireless communications using RF data: A review," *IEEE Commun. Surveys Tut.*, 2022.

[23] Y. Shi, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, Z. Lu, and J. Li, "Adversarial deep learning for cognitive radio security: Jamming attack and defense strategies," *IEEE Int. Conf. Commun. Workshop Promises Challenges Mach. Learn. Commun. Netw.*, 2018, pp. 1–6.

[24] T. Erpek, Y. E. Sagduyu, and Y. Shi, "Deep learning for launching and mitigating wireless jamming attacks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 1, pp. 2–14, Mar. 2019.

[25] M. Sadeghi and E. G. Larsson, "Adversarial attacks on deep-learning based radio signal classification," *IEEE Commun. Lett.*, vol. 8, no. 1, pp. 213–216, Feb. 2019.

[26] Y. Shi, T. Erpek, Y. E. Sagduyu, and J. Li, "Spectrum data poisoning with adversarial deep learning," in *Proc. IEEE Mil. Commun. Conf.*, 2018, pp. 407–412.

[27] S. Bair, M. DelVecchio, B. Flowers, A. J. Michaels, and W. C. Headley, "On the limitations of targeted adversarial evasion attacks against deep learning enabled modulation recognition," in *Proc. ACM Workshop Wireless Secur. Mach. Learn.*, 2019, pp. 25–30.

[28] B. Flowers, R. M. Buehrer, and W. C. Headley, "Evaluating adversarial evasion attacks in the context of wireless communications," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1102–1113, 2020.

[29] S. Kokalj-Filipovic, R. Miller, and G. Vanhoy, "Adversarial examples in RF deep learning: Detection of the attack and its physical robustness," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, 2019, pp. 1–5.

[30] S. Kokalj-Filipovic, R. Miller, and J. Morman, "Targeted adversarial examples against RF deep classifiers," in *Proc. ACM Workshop Wireless Secur. Mach. Learn.*, 2019, pp. 6–11.

[31] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Over-the-air adversarial attacks on deep learning based modulation classifier over wireless channels," in *Proc. IEEE 54th Annu. Conf. Inf. Sci. Syst.*, 2020, pp. 1–6.

[32] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Channel-aware adversarial attacks against deep learning-based wireless signal classifiers," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 3868–3880, Jun. 2020.

[33] B. Kim, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, and S. Ulukus, "Adversarial attacks with multiple antennas against deep learning-based modulation classifiers," in *Proc. IEEE GLOBECOM Open Workshop Mach. Learn. Commun.*, 2020, pp. 1–6.

[34] B. Kim, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, and S. Ulukus, "Channel effects on surrogate models of adversarial attacks against wireless signal classifiers," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[35] B. Kim, Y. E. Sagduyu, T. Erpek, and S. Ulukus, "Adversarial attacks on deep learning based mmWave beam prediction in 5G and beyond," in *Proc. IEEE Stat. Signal Process. Workshop*, 2021, pp. 590–594.

[36] Y. Lin, H. Zhao, Y. Tu, S. Mao, and Z. Dou, "Threats of adversarial attacks in DNN based modulation recognition," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2469–2478.

[37] M. Sadeghi and E. G. Larsson, "Physical adversarial attacks against end-to-end autoencoder communication systems," *IEEE Commun. Lett.*, vol. 23, no. 5, pp. 847–850, May 2019.

[38] B. R. Manoj, M. Sadeghi, and E. G. Larsson, "Adversarial attacks on deep learning based power allocation in a massive MIMO network," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[39] F. Restuccia et al., "Hacking the waveform: Generalized wireless adversarial deep learning," in *Proc. ACM Workshop Wireless Secur. Mach. Learn.*, 2020.

[40] M. Z. Hameed, A. Gyorgy, and D. Gunduz, "Communication without interception: Defense against deep-learning-based modulation detection," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, 2019, pp. 1–5.

[41] M. Z. Hameed, A. Gyorgy, and D. Gunduz, "The best defense is a good offense: Adversarial attacks to avoid modulation detection," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1074–1087, 2021.

[42] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "How to make 5G communications "invisible" adversarial machine learning for wireless privacy," in *Proc. IEEE 54th Asilomar Conf. Signals, Syst., Comput.*, 2020, pp. 763–767.

[43] Y. E. Sagduyu, Y. Shi, and T. Erpek, "Adversarial deep learning for over-the-air spectrum poisoning attacks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 306–319, Feb. 2021.

[44] Z. Luo, S. Zhao, Z. Lu, J. Xu, and Y. E. Sagduyu, "When attackers meet AI: Learning-empowered attacks in cooperative spectrum sensing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 5, pp. 1892–1908, May 2022.

[45] Z. Luo, S. Zhao, Z. Lu, Y. E. Sagduyu, and J. Xu, "Adversarial machine learning based partial-model attack in IoT," in *Proc. 2nd ACM Workshop Wireless Secur. Mach. Learn.*, 2020, pp. 13–18.

[46] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Over-the-air membership inference attacks as privacy threats for deep learning-based wireless signal classifiers," in *Proc. 2nd ACM Workshop Wireless Secur. Mach. Learn.*, 2020, pp. 61–66.

[47] Y. Shi and Y. E. Sagduyu, "Membership inference attack and defense for wireless signal classifiers with deep learning," *IEEE Trans. Mobile Comput.*, to be published.

[48] K. Davaslioglu and Y. E. Sagduyu, "Trojan attacks on wireless signal classification with adversarial machine learning," in *Proc. IEEE DySPAN Workshop Data-Driven Dyn. Spectr. Sharing*, 2019, pp. 1–6.

[49] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative adversarial network in the air: Deep adversarial learning for wireless signal spoofing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 294–303, Mar. 2021.

[50] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative adversarial network for wireless signal spoofing," in *Proc. ACM Workshop Wireless Secur. Mach. Learn.*, 2019, pp. 55–60.

[51] S. Karunaratne, E. Krijestorac, and D. Cabric, "Penetrating RF fingerprinting-based authentication with a generative adversarial attack," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[52] Y. E. Sagduyu, T. Erpek, and Y. Shi, "Adversarial machine learning for 5G communications security," in *Game Theory and Machine Learning for Cyber Security*, New York, NY, USA: Wiley, 2021, pp. 270–288.

[53] Y. Shi and Y. E. Sagduyu, "Adversarial machine learning for flooding attacks on 5G radio access network slicing," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2021, pp. 1–6.

[54] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2005, pp. 46–57.

[55] Y. E. Sagduyu, R. A. Berry, and A. Ephremides, "Jamming games in wireless networks with incomplete information," *IEEE Commun. Mag.*, vol. 49, no. 8, pp. 112–118, Aug. 2011.
[56] 3rd Generation Partnership Project (3GPP), "NR: User Equipment (UE) radio access capabilities," 3GPP, Sophia Antipolis, France, Tech Specification 38.306.
[57] J. Koo, V. B. Mendiratta, M. R. Rahman, and A. Walid, "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics," in *Proc. IEEE 15th Int. Conf. Netw. Serv. Manage.*, 2019, pp. 1–5.
[58] G. O. Boateng, D. Ayepah-Mensah, D. M. Doe, A. Mohammed, G. Sun, and G. Liu, "Blockchain-enabled resource trading and deep reinforcement learning-based autonomous RAN slicing in 5G," *IEEE Trans. Netw. Serv. Manage.* vol. 19, no. 1, pp. 216–227, Mar. 2022.
[59] G. O. Boateng, G. Sun, D. Ayepah-Mensah, D. M. Doe, R. Ou, and G. Liu, "Consortium blockchain-based spectrum trading for network slicing in 5G RAN: A multi-agent deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, early access, Jul. 19, 2022, doi: 10.1109/TMC.2022.3190449.
[60] R. Dangi, A. Jadhav, G. Choudhary, N. Dragoni, M. K. Mishra, and P. Lalwani, "ML-Based 5G network slicing security: A comprehensive survey," *Future Internet*, vol. 14, no. 4, pp. 116, 2022.
[61] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and T. Miyazawa, "Consideration on Automation of 5G Network Slicing With Machine Learning," in *Proc. IEEE ITU Kaleidoscope: Mach. Learn. A 5G Future*, 2018, pp. 1–8.

YI SHI (Senior Member, IEEE) is currently a Research Associate Professor with Commonwealth Cyber Initiative, Virginia Tech, Arlington, VA, USA. He is also a Research Associate Professor of Electrical and Computer Engineering (by courtesy), Virginia Tech. Prior to joining Virginia Tech, he was a Senior Lead Research Scientist with BlueHalo Inc., Rockville, MD, USA. His research interests include algorithm design, optimization, and machine learning for NextG wireless networks. He was the recipient of Best Paper Award at IEEE INFOCOM 2008, only Best Paper Award Runner-Up at IEEE INFOCOM 2011, Best Student Paper Award at ACM WUWNet 2014, and Best Paper Award at IEEE HST 2018. He was the Exemplary Editor of IEEE COMMUNICATIONS SURVEYS AND TUTORIALS in 2014 and a Distinguished TPC Member of IEEE INFOCOM in 2021. Dr. Shi is the Editor of IEEE COMMUNICATIONS SURVEYS AND TUTORIALS and TPC Chair of IEEE and ACM Symposiums, Tracks, and Workshops.

YALIN E. SAGDUYU (Senior Member, IEEE) received the B.S. degree in electrical and electronics engineering from Bogazici University, Istanbul, Turkey, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA. He is currently a Research Professor with Virginia Tech National Security Institute, Arlington, VA, USA. Prior to that, he was the Director of networks and security with Intelligent Automation, Inc./BlueHalo, Rockville, MD, USA. He is also a Visiting Research Professor with the Department of Electrical and Computer Engineering, University of Maryland. His research interests include wireless communications, networks, security, and machine learning. He is the Editor of IEEE TRANSACTIONS ON COMMUNICATIONS. He chaired workshops at ACM MobiCom, ACM WiSec, IEEE CNS, and IEEE ICNP. He was the Track Chair at IEEE PIMRC, IEEE GlobalSIP, and IEEE MILCOM, and served in the Organizing Committee of IEEE GLOBECOM and IEEE MILCOM. He was the recipient of IEEE HST 2018 Best Paper Award.

TUGBA ERPEK (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Virginia Tech, Arlington, VA, USA. She is currently a Research Associate Professor with the Intelligent Systems Division, Virginia Tech National Security Institute, Arlington, VA, USA. Prior to joining to Virginia Tech, she was a Lead Scientist and Network Communications Technical Area Lead with Intelligent Automation, a BlueHalo Company, Rockville, MD, USA, and a Senior Communications Systems Engineer with Shared Spectrum Company, Vienna, VA, USA. She has authored or coauthored extensively in her research fields which include wireless communications and networks, 5G and beyond, wireless security, machine learning, and resource allocation. She is a TPC Member and reviewer of major IEEE conferences and journals.

M. CENK GURSOY (Senior Member, IEEE) is currently a Professor with EECS Department, Syracuse University, Syracuse, NY, USA. His research interest include the general areas of wireless communications, information theory, communication networks, signal processing, and machine learning. He is a Member of the Editorial Boards of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON COMMUNICATIONS, and the Area Editor of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He was also the Editor of IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING between 2016 and 2021, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS between 2010 and 2015, IEEE COMMUNICATIONS LETTERS between 2012 and 2014, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS - Series on Green Communications and Networking (JSAC-SGCN) between 2015 and 2016, *Physical Communication* (Elsevier) between 2010 and 2017, and IEEE TRANSACTIONS ON COMMUNICATIONS between 2013 and 2018. He has been the Co-Chair of the 2017 International Conference on Computing, Networking and Communications (ICNC) - Communication QoS and System Modeling Symposium, 2019 IEEE Global Communications Conference (Globecom) - Wireless Communications Symposium, 2019 IEEE Vehicular Technology Conference Fall - Green Communications and Networks Track, and 2021 IEEE Global Communications Conference (Globecom), Signal Processing for Communications Symposium. He was the recipient of an NSF CAREER Award in 2006 EURASIP Journal of Wireless Communications and Networking Best Paper Award, 2020 IEEE Region 1 Technological Innovation (Academic) Award, 2019 The 38th AIAA/IEEE Digital Avionics Systems Conference Best of Session (UTM-4) Award, 2017 IEEE PIMRC Best Paper Award, 2017 IEEE Green Communications & Computing Technical Committee Best Journal Paper Award, UNL College Distinguished Teaching Award, and the Maude Hammond Fling Faculty Research Fellowship. He is the Aerospace/Communications/Signal Processing Chapter Co-Chair of IEEE Syracuse Section.