# How to Attack and Defend 5G Radio Access Network Slicing with Reinforcement Learning

Yi Shi[1,2], Yalin E. Sagduyu[1], Tugba Erpek[1,2], and M. Cenk Gursoy[3]

[1]Intelligent Automation, Inc., Rockville, MD 20855, USA
[2]Virginia Tech, Blacksburg, VA 24061, USA
[3]Syracuse University, Syracuse, NY 13244, USA



***Abstract*—Reinforcement learning (RL) for network slicing is considered in the 5G radio access network, where the base station, gNodeB, allocates resource blocks (RBs) to the requests of user equipments and maximizes 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 5G network slicing. Subject to an energy budget, the adversary observes the spectrum and builds its own RL-based surrogate model that selects which RBs to jam with the objective of maximizing the number of failed network slicing 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 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 mechanisms (protecting the RL algorithm's updates or misleading the adversary's learning process) are introduced to show that it is viable to defend 5G network slicing against this attack.**

***Index Terms*—5G security, network slicing, radio access network, reinforcement learning, adversarial machine learning, jamming, wireless attack, defense.**

## I. Introduction

5G 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 5G communications is *network slicing* in the *radio access network* (RAN), which splits communication resources into virtual resource blocks (RBs) and allocates RBs dynamically to support different types of user applications. 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]–[5].

This effort is supported 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.

*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 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 5G network slicing [9]–[15] such as learning from the 5G network performance and updating resource allocation decisions for network slicing.

In this paper, we consider a 5G base station, gNodeB, as the victim system that runs an RL algorithm (as an example, the *Q-learning* algorithm) to dynamically allocate resources for 5G network slicing, where resource blocks (RBs) are allocated to support downlink communications from gNodeB to the user equipments (UEs). Each network slicing request from any of UEs 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.

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. Separate from 5G network slicing, attacks on RL algorithms have been considered in [16]–[18] 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 5G network slicing problem. If an RB is assigned to a 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 the performance of the updated RL algorithm continues to drop. Thus, such a jamming attack not only affects the gNodeB's current performance but also will affect 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 (namely, its effect on 5G network slicing), we compute the *recovery time*, namely 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 may not work well since there is no need to jam multiple RBs assigned to the same request. Moreover, our results show that this rather simple strategy can be learned by 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 RBs, we pursue an *adversarial machine learning* approach and let the adversary build its surrogate model as an RL algorithm (Q-learning) in the form of an *exploratory (inference) attack* to select the RBs to be jammed. Different types of attacks built upon adversarial machine learning have been studied in wireless communications such as exploratory (inference) attacks [19], [20], evasion (adversarial) attacks [21]–[30] and their extensions to secure and covert communications against eavesdroppers [31]–[33], causative (poisoning) attacks [34], [35], membership inference attacks [36], Trojan attacks [37], and spoofing attacks [38]–[40] that have been launched against various spectrum sensors and wireless signal (such as modulation) classifiers. Adversarial machine learning has been also considered for 5G by studying evasion and spoofing attacks on deep neural networks (without reinforcement learning) used for 5G spectrum sharing and 5G signal authentication [41].

In this paper, a jamming attack built upon adversarial machine learning is launched against the RL agent that performs resource allocation for 5G network slicing, and the attack exploits the unique properties that the RL algorithm is affected by manipulating rewards and it takes a while for the RL algorithm to recover even after the attack stops. 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 *states* of the *surrogate* RL model built by the adversary correspond to the availability of RBs, which are determined by passively sensing RBs (since the adversary does not have access to the victim RL model, namely it launches a *black-box attack*, and cannot query it with inputs). The *actions*

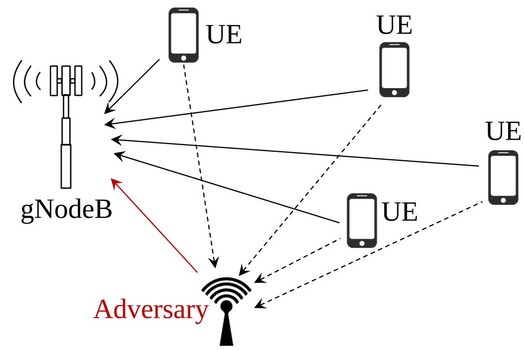


Fig. 1: The interaction of the victim RL algorithm and the adversarial surrogate RL algorithm.

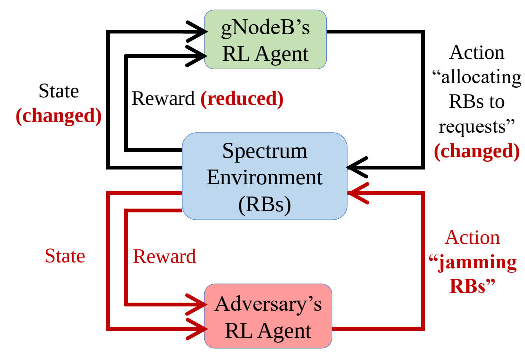


Fig. 2: Illustration of how jamming built upon adversarial machine learning manipulates the RL process of the gNodeB to allocate RBs to network slicing requests.

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 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. 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 due to jamming. 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. The interaction of the victim RL algorithm of the gNodeB and the surrogate RL algorithm of the adversary is illustrated in Fig. 2.

In performance evaluation, we compare the RL-based attack with the *myopic* attack (namely, jamming RBs to maximize the adversary's instantaneous reward) 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 gNodeB's RL algorithm (under attack and after attack) and the longest recovery time from the attack (after the jamming

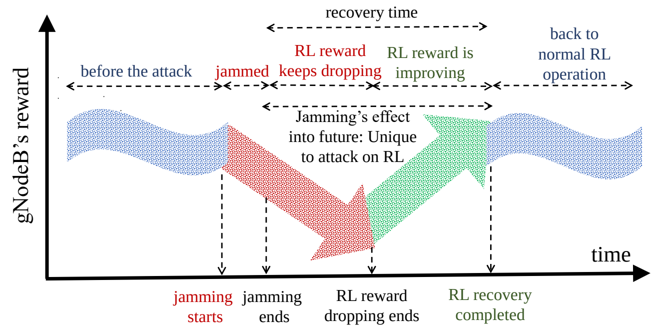


Fig. 3: Adversarial machine learning for manipulating the RL process of the gNodeB when it allocates RBs to network slicing requests.

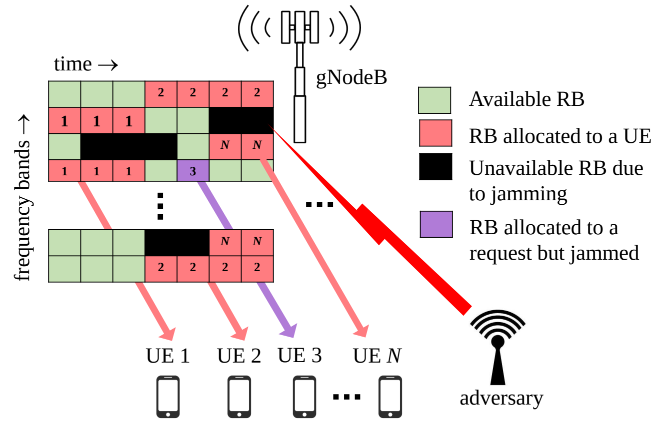


Fig. 5: System model for 5G network slicing in the presence of an adversary.

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 attack compared to conventional jamming attacks (on data transmissions) whose impact is typically limited to the duration of the attack* (see [42], [43] for examples of conventional jamming attacks on wireless communications).

Next, we present how to *defend* the network slicing operations against the RL-based jamming attacks. For that purpose, we introduce three different defense approaches for the 5G gNodeB or the UE to take (illustrated in Fig. 4): (i) protect the RL algorithm itself by stopping the RL algorithm (i.e., Q-table) update once an attack is detected to avoid the impact of the attack on the RL algorithm; (ii) 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; or (iii) manipulate the feedback (NACK) mechanism such that the adversary may not obtain reliable information to build its attack strategy. We show that the second defense approach 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.

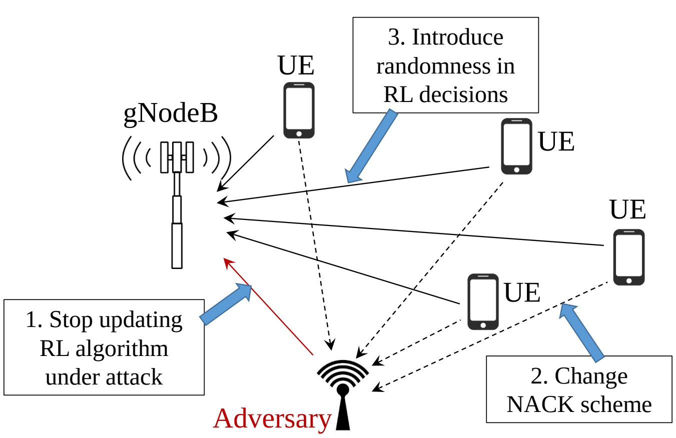


Fig. 4: The defense approaches.

The rest of the paper is organized as follows. Section II describes the resource allocation for network slicing via RL. Section III presents the RL-based jamming attack that aims to maximize the impact on gNodeB's performance under the attack and after the attack. Section IV presents defense approaches 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. Reinforcement Learning based Resource Allocation for Network Slicing

In this section, we summarize the 5G 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 mechanisms that we consider in Section III applies to other RL-based 5G RAN slicing settings, as well. As shown in Fig. 5, where multiple 5G UEs send requests over time with different QoE, i.e., rate, latency (deadline) and lifetime, requirements and priority weights, and the 5G 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 (more formally, in a queue) until its deadline expires. There is also an adversary that we will describe in Section III.

At time $t$, 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 $F_{ij}$ and the modulation/coding scheme used for communications between gNodeB and UE $i$. The data rate (bps) is approximated as [44]:

$$D_{ij} = c \cdot K_{ij} \cdot (1 - BER_{ij}), \tag{2}$$

where $K_{ij}$ is the number of aggregated component carriers (CCs) in a band combination and $BER_{ij}$ is the bit error rate (BER) of UE $i$ for its request $j$ (it depends on the signal-to-noise ratio (SNR) and is computed for AWGN channel with low-density parity-check (LDPC) coding), and constant $c$ is approximately $12.59 \cdot 10^6$ when a single-antenna UE uses QPSK modulation, 60 kHz subcarrier spacing and 10 MHz bandwidth.

The constraints of resource assignments to network slices are given by

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

where $F(t)$ 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 $t$ (namely, the service starts at time $t$), this request will end at time $t + l_{ij}$. The released and allocated resources at time $t$ are given by

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

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

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

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

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

As the 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:

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

where $\alpha$ is the learning rate ($0 < \alpha \leq 1$) and $\gamma$ 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.

TABLE I: RL algorithm for network slicing.

| RL term | Specification (at any given time instant) |
|---|---|
| State | Availability of RBs, active requests |
| Action | Assign RBs to selected network slicing requests |
| Reward | Total weights of satisfied network slicing requests |

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 I.

## III. Attack on Reinforcement Learning for 5G Network Slicing

We now consider an adversary that attacks the RL algorithm discussed in Section II. 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 can change 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 5G 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 for RL. For the RL algorithm presented in Section II, the state includes the RB availability and a request under consideration. Both are maintained by the 5G 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 energy budget) and thus cannot jam all RBs to maximize its impact. We denote $A$ as the maximum number of RBs that the adversary can jam at any given time. Due to this constraint, it is important for

TABLE II: 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 to select 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). This RL model has the following state, action, and reward properties.

- The state is the set of binary variables that indicate the availability of all RBs.
- An action corresponds to the set of $\min\{A, n_a\}$ RBs selected from available RBs, where $n_a$ is the number of available RBs. Note that there is another action of not jamming any RB. Thus, the number of possible actions is $C_{n_a}^{A} + 1$ (where $C_{n_a}^{A}$ is the combination for $n_a$ and $A$, i.e., number of possibilities picking $A$ out of $n_a$) if $n_a > A$ or 2 (jam or not) if $n_a \leq A$.
- The reward is the number of jammed requests at a given time. We assume that there is a NACK transmitted from 5G UEs at the end of each time slot to indicate whether a UE's request is successful or not. If the adversary jams an RB and later observes the NACK, the reward on this channel is one. 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 NACK from data transmissions (as NACK is shorter and has the structure of appearing between requests and data transmissions).

To initialize the Q-table, we let the column of no jamming as zeros and other entry as the number of jammed RBs. 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 II.

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 is that some requests fail due to jamming and thus their weights are not counted in the reward of 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 5G 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 5G gNodeB's network slicing performance to go back to "normal" (the case when there is no attack).
- *Maximum performance reduction*: The maximum performance gap to the normal value during the recovery time (the performance is measured as the running averaged reward).
- *Total performance reduction*: The accumulated performance gap to the normal value during the recovery time.

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.

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 consideration of future reward.

The performance of these attacks is evaluated in Section V.

## IV. Defense against Attacks on Reinforcement Learning for 5G Network Slicing

To protect the network slicing operations from the RL-based jamming attacks, we present different defense approaches (illustrated in Fig. 4) for the 5G gNodeB or the UE to take.

- *Q-Protect*: One reactive approach 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 RBs allocated to this request are jammed, this request cannot be satisfied and therefore the reward is reduced to zero. Thus, the gNodeB can detect the jamming attack by checking the changes in the reward. In particular, the gNodeB checks if the reward drops by a certain amount. For numerical results, we assume that the change is detected if the running average of the rewards drops by $10\%$. Without consideration of attacks, the gNodeB always updates its Q-table. Hence, a reactive defense approach is designed to stop 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 approach "Q-Protect".
- *RandomOpt* and *RandomTop*: A proactive defense approach aims to manipulate the adversary's learning process (namely, its surrogate model). This approach 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. There are two policies, with and without performance loss when there is no attack. 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 approach "RandomOpt". However, the randomness among best actions may not be sufficient to mitigate the performance loss due to the attack. The second approach 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 $r$ percentage of the maximum reward. This approach introduces more randomness but may incur performance loss even if there is no attack. We call this approach "RandomTop".

- *MisNACK*: Another proactive defense approach 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 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 approach "MisNACK".

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

## V. Performance Evaluation

Suppose that the gNodeB receives requests from three UEs. For each UE, requests arrive with the rate of $0.5$ 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 and is split into 11 bands, i.e., there are 11 RBs. We also consider a scenario with a smaller number of RBs, namely $5$ RBs.

[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, Q-table may not be perfect and thus there can still be performance loss under this policy.

TABLE III: Performance comparison of Q-learning and other attacks for 11 RBs.

| Algorithm | Maximum jammed RBs | Recovery time | Maximum reduction in reward | Total reduction |
|---|---|---|---|---|
| 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 | 1186 | 0.994 | 502.166 |
| | 2 | 1114 | 1.486 | 745.885 |
| | 3 | 1112 | 1.835 | 918.069 |
| | 4 | 1097 | 2.072 | 1038.146 |
| | 5 | 1139 | 2.254 | 1124.589 |

### 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 $10,000$ slots. The benchmark of no attack case is also run over $10,000$ slots in total and the achieved reward is measured as $3.032$ over the first $1000$ slots (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, 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 III, where results for random jamming are averaged over 20 runs. We can see that the Q-learning based attack has longer and larger impact on the 5G network slicing performance than other attacks. We show in Fig. 6 how the reward is changing over time after the attack stops. Time unit corresponds to $0.23$ ms of time blocks, as described earlier in this section.

The RL algorithm's performance under different attacks is shown in Fig. 7. Since we show the average reward over past 1000 slots, the performance is high at the beginning and decreases fast. Then, the performance under random jamming or Q-learning based jamming remains still small while the performance under myopic jamming keeps going up. This is because the myopic algorithm is deterministic and thus it is easy to learn and mitigate it by RL.

Next, we measure the performance by changing the number of RBs from 11 to 5 (other parameters remain the same). Results are shown in Table IV. As before, the RL-based attack has longer and larger impact on the performance than benchmark attacks. Fig. 8 and Fig. 9 show the reward over time after the attack stops and under the attack, respectively.

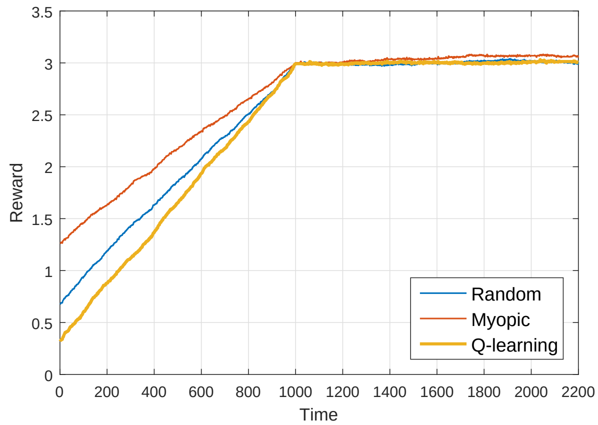


Fig. 6: Algorithm performance after the attack stops when there are 11 RBs.

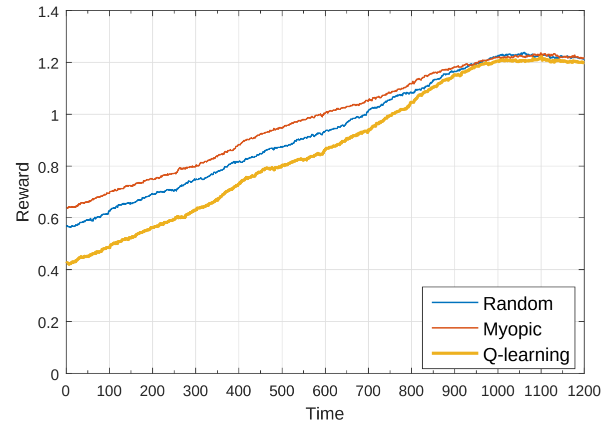


Fig. 8: Algorithm performance after the attack stops when there are 5 RBs.

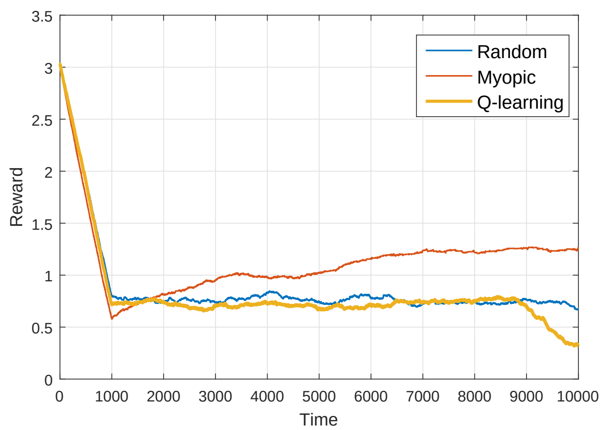


Fig. 7: Algorithm performance under the attack when there are 11 RBs.

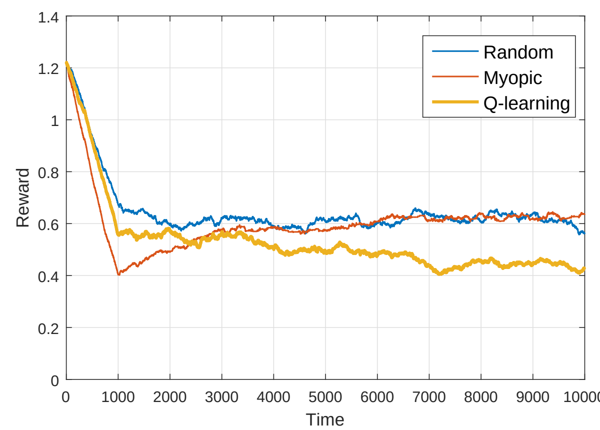


Fig. 9: Algorithm performance under the attack when there are 5 RBs.

The trends in Fig. 8 and Fig. 9 are the same as the trends observed in Fig. 6 and Fig. 7 when there are 11 RBs.

### *B. Defense Performance Evaluation*

We now present the performance of different defense approaches against the RL-based attack. The results for the "Q-Protect" approach are shown in Table V. Comparing with results in Table III for the attack only case (when there is no defense), "Q-Protect" cannot achieve consistently better results for all instances.

TABLE IV: Performance comparison of RL-based and other attacks when there are 5 RBs.

| 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 | 993.45 | 0.404 | 199.604 |
| | 2 | 993.00 | 0.613 | 300.781 |

The results for the "RandomOpt" approach are shown in Table VI. Comparing with those in Table III, "RandomOpt" can improve the performance for most instances and in terms of most performance metrics. Therefore, "RandomOpt" emerges as a more viable defense approach than "Q-Protect".

The results for the "RandomTop" approach are shown in Table VII, when an action is considered as "Top" if its reward is at least $r = 50\%$ of the maximum reward. When the value of $r$ is varied for the "Top" action selection, similar results are obtained. Comparing with those in Table VI, we note that "RandomTop" can also improve the performance for most

TABLE V: Performance by the "Q-Protect" approach under RL-based attack when there are 11 RBs.

| Maximum jammed RBs | Recovery time | Maximum reduction in reward | Total reduction |
|---|---|---|---|
| 1 | 1113 | 1.343 | 662.253 |
| 2 | 1081 | 1.927 | 971.749 |
| 3 | 1018 | 2.216 | 1094.859 |
| 4 | 998 | 2.276 | 1123.483 |
| 5 | 998 | 2.589 | 1288.492 |

TABLE VI: Performance by the "RandomOpt" approach under RL-based attack when there are 11 RBs.

| Maximum jammed RBs | Recovery time | Maximum reduction in reward | Total reduction |
|---|---|---|---|
| 1 | 994 | 1.140 | 571.151 |
| 2 | 2668 | 1.526 | 817.817 |
| 3 | 999 | 1.770 | 882.487 |
| 4 | 1095 | 2.049 | 1019.578 |
| 5 | 992 | 2.198 | 1077.678 |

TABLE VII: Performance by the "RandomTop" approach under RL-based attack when there are 11 RBs.

| Maximum jammed RBs | Recovery time | Maximum reduction in reward | Total reduction |
|---|---|---|---|
| 1 | 1039 | 1.105 | 578.227 |
| 2 | 1452 | 1.531 | 802.987 |
| 3 | 1172 | 1.834 | 964.558 |
| 4 | 998 | 2.116 | 1065.831 |
| 5 | 1037 | 2.316 | 1169.106 |

instances and in terms of most performance metrics. There is no significant improvement compared to "RandomOpt".

The results for the "MisNACK" approach are shown in Table VIII. Comparing with those in Table III, "MisNACK" cannot achieve better results than the attack only case for all instances.

In summary, the best defense approach for results presented so far is "RandomOpt". Therefore, we evaluate the performance of "RandomOpt" in further detail. The results for a gNodeB with 5 RBs are shown in Table IX. Comparing with those in Table IV, "RandomOpt" can improve the performance (this observation holds for both cases with 11 and 5 RBs).

We can also combine different defense approaches and apply them jointly to strengthen the overall defense against the RL-based attack on 5G 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. The results under the combined defense are shown in Table X. Comparing with those in Table VII, this combined defense approach can further improve the performance for some instances and provide an additional design choice for the defense of 5G network slicing depending on the attack strength (maximum of number jammed RBs) and the preference of the performance metric that the defense seeks to improve.

TABLE VIII: Performance by the "MisNACK" approach under RL-based attack when there are 11 RBs.

| Maximum jammed RBs | Recovery time | Maximum reduction in reward | Total reduction |
|---|---|---|---|
| 1 | 993 | 1.190 | 598.569 |
| 2 | 1021 | 1.850 | 945.604 |
| 3 | 992 | 2.169 | 1079.212 |
| 4 | 993 | 2.486 | 1232.382 |
| 5 | 993 | 2.471 | 1205.585 |

TABLE IX: Performance by the "RandomTop" approach under Q-learning based attack when there are 5 RBs.

| Maximum jammed RBs | Recovery time | Maximum reduction in reward | Total reduction |
|---|---|---|---|
| 1 | 919 | 0.509 | 253.189 |
| 2 | 942 | 0.845 | 390.464 |

TABLE X: Performance by the combined defense of "Q-Protect", "RandomTop", and "MisNACK" approaches under RL-based attack when there are 11 RBs.

| Maximum jammed RBs | Recovery time | Maximum reduction in reward | Total reduction |
|---|---|---|---|
| 1 | 1059 | 1.067 | 539.256 |
| 2 | 1312 | 1.523 | 775.195 |
| 3 | 990 | 1.952 | 959.317 |
| 4 | 996 | 2.089 | 1039.144 |
| 5 | 1178 | 2.378 | 1160.997 |

## VI. Conclusion

In this paper, we studied the security vulnerability of 5G 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 5G gNodeB) for communication requests from 5G UEs, the broadcast nature of wireless communications makes the 5G 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. Moreover, this reward is used as input to the gNodeB's RL algorithm and thus the performance algorithm 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 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 RL-based jamming attack is more effective than random or myopic jamming.

To protect network slicing against RL-based jamming attacks, we introduced several defense mechanisms such as stopping 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 it is possible to effectively defend network slicing by fooling the adversary into making wrong decisions and reducing its impact.

# REFERENCES


[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, 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 Communications Magazine*, vol. 55, no. 5, May 2017.

[3] P. Rost, C. Mannweiler, D. S. Michalopoulos, C.Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications Magazine*, vol. 55, May 2017.

[4] A. Kaloxylos, "A survey and an analysis of network slicing in 5G networks," *IEEE Communications Standards Magazine*, vol. 2, no. 1, Apr. 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," *IEEE INFOCOM*, 2019.

[6] T. Erpek, T. O'Shea, Y. E. Sagduyu, Y. Shi, and T. C. Clancy, "Deep Learning for Wireless Communications," *Development and Analysis of Deep Learning Architectures, Springer*, 2019.

[7] A. Nakao, and P. Du, "Toward in-network deep machine learning for identifying mobile applications and enabling application specific network slicing," *IEICE Transactions on Communications*, 2018.

[8] A. Thantharate, R. Paropkari, V. Walunj, C. Beard, "DeepSlice: A deep learning approach towards an efficient and reliable network slicing in 5G networks," *IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, 2019.

[9] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *arXiv preprint* arXiv:1805.06591, 2020.

[10] J. Koo, M. R. Rahman, V. B. Mendiratta, and A. Walid, "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics," *arXiv preprint* arXiv:1908.03242, 2019.

[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," *Information Sciences*, 2019.

[12] Q. Liu and T. Han, "When network slicing meets deep reinforcement learning," *International Conference on Emerging Networking Experiments and Technologies*, 2019.

[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 International Conference on Communications (ICC)*, 2017.

[14] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Reinforcement learning for dynamic resource optimization in 5G radio access network slicing," *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020.

[15] A. Nassar and Y. Yilmaz, "Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities," *arXiv preprint* arXiv:2010.09916, 2020.

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

[17] C. Zhong, F. Wang, M. C. Gursoy, and S. Velipasalar, "Adversarial jamming attacks on deep reinforcement learning based dynamic multichannel access," *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[18] F. Wang, C. Zhong, M. C. Gursoy, and S. Velipasalar, "Defense strategies against adversarial jamming attacks via deep reinforcement learning," *IEEE Conference on Information Sciences and Systems (CISS)*, 2020.

[19] 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 International Conference on Communications (ICC) Workshop on Promises and Challenges of Machine Learning in Communication Networks*, 2018.

[20] T. Erpek, Y. E. Sagduyu, and Y. Shi, "Deep learning for launching and mitigating wireless jamming attacks, *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 1, pp. 2–14, Mar. 2019.

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

[22] Y. Shi, T. Erpek, Y. E. Sagduyu, and J. Li, "Spectrum data poisoning with adversarial deep learning," *IEEE Military Communications Conference (MILCOM)*, 2018.

[23] 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," *ACM Workshop on Wireless Security and Machine Learning (WiseML)*, 2019.

[24] B. Flowers, R. M. Buehrer, and W. C. Headley, "Evaluating adversarial evasion attacks in the context of wireless communications," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1102–1113, 2020.

[25] S. Kokalj-Filipovic and R. Miller, "Adversarial examples in RF deep learning: Detection of the attack and its physical robustness," *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2019.

[26] S. Kokalj-Filipovic, R. Miller, and J. Morman, "Targeted Adversarial Examples against RF Deep Classifiers," *ACM Workshop on Wireless Security and Machine Learning (WiseML)*, 2019.

[27] 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," *Conference on Information Sciences and Systems (CISS)*, 2020.

[28] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Channel-aware adversarial attacks against deep learning-based wireless signal classifiers," *arXiv preprint arXiv:2005.05321*, 2020.

[29] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Adversarial attacks with multiple antennas against deep learning-based modulation classifiers," *arXiv preprint* arXiv:2007.16204, 2020.

[30] B. Kim, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, and S. Ulukus, "Channel Effects on Surrogate Models of Adversarial Attacks against Wireless Signal Classifiers," *arXiv preprint* arXiv:2012.02160, 2020.

[31] M. Z. Hameed, A. Gyorgy, and D. Gunduz, "Communication without interception: Defense against deep-learning-based modulation detection," *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2019.

[32] M. Z. Hameed, A. Gyorgy, and D. Gunduz, "The best defense is a good offense: Adversarial attacks to avoid modulation detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, 2021

[33] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "How to make 5G communications "invisible" adversarial machine learning for wireless privacy," *arXiv preprint arXiv:2005.07675*, 2020.

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

[35] Z. Luo, S. Zhao, Z. Lu, J. Xu, and Y. E. Sagduyu, "When Attackers Meet AI: Learning-empowered Attacks in Cooperative Spectrum Sensing," *IEEE Transactions on Mobile Computing*, 2020.

[36] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Over-the-air membership inference attacks as privacy threats for deep learning-based wireless signal classifiers," *ACM WiSec Workshop on Wireless Security and Machine Learning (WiseML)*, 2020.

[37] K. Davaslioglu and Y. E. Sagduyu, "Trojan attacks on wireless signal classification with adversarial machine learning," *IEEE DySPAN Workshop on Data-Driven Dynamic Spectrum Sharing*, 2019.

[38] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative Adversarial Network in the Air: Deep Adversarial Learning for Wireless Signal Spoofing," *IEEE Transactions on Cognitive Communications and Networking*.

[39] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative Adversarial Network for Wireless Signal Spoofing," *ACM Workshop on Wireless Security and Machine Learning (WiseML)*, 2019.

[40] S. Karunaratne, E. Krijestorac, and D. Cabric, "Penetrating RF fingerprinting-based authentication with a generative adversarial attack," *arXiv preprint* arXiv:2011.01538, 2020.

[41] Y. E. Sagduyu, T. Erpek, and Y. Shi, "Adversarial machine learning for 5G communications security," *arXiv preprint* arXiv:2101.02656, 2021.

[42] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005.

[43] Y. E. Sagduyu, R. Berry, and A. Ephremides, "Jamming games in wireless networks with incomplete information," *IEEE Communications Magazine*, 2011.

[44] 3GPP TS 38.306: "NR; User Equipment (UE) radio access capabilities".