



Article

Deep Reinforcement Learning for Attacking Wireless Sensor Networks

Juan Parras ^{1,*} , Maximilian Hüttenrauch ^{2,3}, Santiago Zazo ¹  and Gerhard Neumann ^{2,3}¹ Information Processing and Telecommunications Center, Universidad Politécnica de Madrid, 28040 Madrid, Spain; santiago.zazo@upm.es² Lincoln School of Computer Science, University of Lincoln, Lincoln LN6 7TS, UK; maximilian.huettentrauch@kit.edu (M.H.); gerhard.neumann@kit.edu (G.N.)³ Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany

* Correspondence: j.parras@upm.es

Abstract: Recent advances in Deep Reinforcement Learning allow solving increasingly complex problems. In this work, we show how current defense mechanisms in Wireless Sensor Networks are vulnerable to attacks that use these advances. We use a Deep Reinforcement Learning attacker architecture that allows having one or more attacking agents that can learn to attack using only partial observations. Then, we subject our architecture to a test-bench consisting of two defense mechanisms against a distributed spectrum sensing attack and a backoff attack. Our simulations show that our attacker learns to exploit these systems without having a priori information about the defense mechanism used nor its concrete parameters. Since our attacker requires minimal hyper-parameter tuning, scales with the number of attackers, and learns only by interacting with the defense mechanism, it poses a significant threat to current defense procedures.



Citation: Parras, J.; Hüttenrauch, M.; Zazo, S.; Neumann, G. Deep Reinforcement Learning for Attacking Wireless Sensor Networks. *Sensors* **2021**, *21*, 4060. <https://doi.org/10.3390/s21124060>

Academic Editors: Seokhoon Yoon and Jangyoung Kim

Received: 29 April 2021

Accepted: 10 June 2021

Published: 12 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: POMDP; Deep Reinforcement Learning; TRPO; SSDF attack; backoff attack

1. Introduction

Reinforcement learning (RL) is a field that has attracted great attention in the last twenty years in the community of artificial intelligence. It allows programming agents to learn by interacting with a complex, dynamic environment: the agent observes its state, chooses actions, obtains a reward and transitions to another state. It is important to remark that RL does not require indicating how the agent is to act; instead, the agent learns to choose its action by trial and error, such that its total reward is maximized. Introductory material for RL can be found in [1,2].

The recent advances in the field of Deep Learning [3] have produced a massive step forward in the RL field. A seminal work was the publication of the Deep Q-Networks (DQN) algorithm [4,5], which offered superhuman performance in a set of Atari games. Several other algorithms for Deep RL (DRL) followed and were applied successfully to complex problems [6–10].

These advances do affect many problems of interest in the area of Wireless Sensor Networks (WSNs). A survey notes that RL algorithms are used in several problems that arise in WSNs, such as routing, data latency, path determination, duty cycle management, QoS provisioning or resource management [11]. Another problem is the security in WSNs, where a great deal of research is being done at the moment [12–16], also taking advantage of these recent advances in deep learning [17,18]. The idea of applying RL to cyber security is not new [19,20]; to mention some examples, DRL tools are used in WSN security to detect spoofing attacks [21], for mobile off-loading [22,23], to avoid jamming [24,25] and to model DoS attacks [26] or data poisoning attacks [27].

However, many current defense mechanisms designed for WSNs are ad hoc, that is, designed against a specific attack [28–30]. However, the advances in DRL potentially

open the door to designing an attacker that learns to exploit a possibly unknown defense mechanism, as these methods follow a trial-and-error approach to learn an optimal behavior. Inspired by these advances, we apply DRL techniques for learning an optimal attack against WSN defense mechanisms that are unknown to the attacker. Thus, the question we pose is, given a WSN in which there are several Good Sensors (GSs) and one or more Attacking Sensors (ASs), how do current defense mechanisms perform against DRL-based attackers? Can we learn an attacker using DRL that exploits unknown defense mechanisms simply by interacting with them?

A previous work [31] showed that DRL techniques could be used to obtain quasi-optimal attacks in a Cooperative Spectrum Sensing network using a robust defense mechanism. This paper expands that work significantly, as we are interested in the multiple ASs case. Moreover, we study whether the ASs pose a larger threat to the defense mechanism if the ASs are able to communicate and cooperate on the task of attacking the defense mechanism. Compared to [31], our approach has several important differences. First, we do not assume that the state is observable for the ASs: they will only have partial information about the system, which is a far more realistic setting. This means that the ASs do not have access to the precise state of the defense mechanism (e.g., the reputation of each node), as they only have access to values that can be observed, such as the times when a node attempts to transmit. Second, we study two different defense mechanisms on the PHY and MAC layers: in the former, the action space is continuous, while in the latter, the action space is discrete. Our Deep Learning Attacker (DLA) is able to work with both types of action spaces, thus giving it more flexibility. Third, we focus on the case in which there are more than one AS, enabling the ASs to communicate their observations to other ASs in order to better exploit the defense mechanism. Since all the ASs have a common goal, which is to exploit the defense mechanism of the WSN, the ASs can be studied as a swarm [32], which is a group of homogeneous agents that cooperate to obtain a common goal in a decentralized fashion. While there is a large body of research devoted to the optimization of swarms from the evolutionary perspective, including [33–36], we use gradient-based optimization methods as in [37].

Our results show that our DLA architecture does pose a strong challenge to WSN security that current ad-hoc defense mechanisms fail to deal with. The threat that DRL methods pose to current defense mechanisms when ASs can communicate among them is surprisingly not well assessed in the current literature to the best of our knowledge.

This paper is structured as follows. We start by describing the mathematical foundations that our work is based on in Section 2. Section 3 introduces the defense mechanisms for the two attacks we study. Afterwards, Sections 4 and 5 introduce our DLA architecture and validate our approach via simulations, in which we test the effectiveness of our DLA. Finally, in Section 6, we discuss the results and draw some conclusions.

2. Background

We now introduce several key concepts to understand our model. We start by introducing Markov Decision Processes (MDPs), Partially Observable MDPs (POMDPs) and a swarm model. We also explain a DRL algorithm, Trust Region Policy Optimization (TRPO).

2.1. Deep Reinforcement Learning

Let us start by considering that there is a single AS. MDPs are a common framework to study optimal control problems and can be defined as [38]:

Definition 1. An MDP is a 5-tuple $\langle S, A, P, R, \gamma \rangle$, where:

- S is a finite set of states s .
- A is a finite set of actions a .
- $P_a(s^n, s^{n+1})$ is the probability that action a in state s^n and time step n will lead to state s^{n+1} in time step $n + 1$.

- $R_a(s^n, s^{n+1})$ is the reward received after transitioning from state s^n to state s^{n+1} due to action a .
- γ is a discount factor.

An MDP policy π is a mapping $\pi : S \rightarrow A$ that associates an action (or a probability distribution over actions) to each state. We can define π as a function of the current state because the Markov property holds: the probability to reach state s in time n depends only on the state in time $n - 1$.

There are several ways of solving an MDP. One approach is using Dynamic Programming methods [39], which require knowing P_a . Another approach is RL, which obtains an optimal policy without explicitly knowing the transition function P_a . There are several RL algorithms that could be used, such as Q-learning or SARSA [1]. However, these RL algorithms are impractical when the state and/or the action space are large or continuous. For these cases, there has been recently a lot of research that uses the advances in Deep Neural Networks (DNNs) to propose DRL algorithms, which are able to cope with continuous, large state spaces, such as DQN [4,5] or DRQN [6], and even with continuous action spaces, such as TRPO [9] or PPO [10].

We use TRPO [9] as a DRL algorithm in this work. TRPO uses a DNN of parameters θ to approximate the policy π_θ . The policy is updated by solving the following optimization problem:

$$\begin{aligned} & \underset{\theta}{\text{maximize}} \quad \mathbb{E} \left[\frac{\pi_\theta}{\pi_{\text{old}}} A(s^n, a) \right] \\ & \text{subject to} \quad \mathbb{E}[D_{\text{KL}}(\pi_\theta || \pi_{\text{old}})] \leq \delta, \end{aligned} \quad (1)$$

where \mathbb{E} is used to denote the mathematical expectation, π_{old} refers to the policy in the previous iteration, $A(s^n, a)$ is the advantage function, which gives an estimate of how good in terms of reward it is to use action a in state s^n . D_{KL} denotes the Kullback–Leibler divergence, and δ is a constant that forces the new policy to be closer than δ to the old policy.

The rationale for using TRPO is twofold. Firstly, it allows working with continuous and discrete action spaces [9], which provides flexibility to our DLA. Furthermore, TRPO is a powerful DRL algorithm, which has shown to benchmark very well in different sets of tasks [40].

2.2. Partially Observable Markov Decision Processes

In our case, the real state of the network is typically unknown and the agents can only obtain local observations such as their last actions and rewards. Thus, we need to introduce the concept of POMDP, which generalizes the MDP framework. A POMDP is defined as follows [38]:

Definition 2. A POMDP is a 7-tuple

$\langle S, A, P, R, \gamma, \Omega, O \rangle$, where:

- $S, A, P_a(s^n, s^{n+1}), R_a(s^n, s^{n+1})$ and γ are defined as in the MDP case.
- Ω is a finite set of observations o .
- $O_a(o^n, s^n)$ is the probability of observing o^n given that the actual state is s^n and the agent takes action a .

In an MDP, the observation corresponds to the state. For POMDPs, the state is not directly observable and hence the Markovian assumption does not hold: choosing an optimal action in time n in general requires knowing the whole history of past observations o^n for $n \in \{0, 1, \dots, n\}$; thus POMDPs are computationally complex to solve. If N and the sets S, A and O are finite, there exist algorithms that can obtain the optimal policy [38]. Another way of solving POMDPs consists in obtaining a belief, which is a sufficient statistic that collects all the pertinent past information and allows turning the POMDP

in an MDP over the belief space [38]. An alternative way of solving POMDPs is based on Predictive State Representation (PSR) [41,42]. Finally, there are methods that use DRL tools to obtain approximate solutions for POMDPs, which can be classified in two main branches according to [6]:

1. Using Recurrent Neural Networks (RNNs), which are able to store past information. This solution is proposed to learn POMDPs in [6].
2. Using a finite vector of past observations as input to the policy DNN. As [6] shows, this solution is an approximation for solving POMDPs, but it can provide very good results [5] while keeping a lower computational complexity. This motivates us to use this approach in this paper.

2.3. The Swarm Model

In this paper, we want to model more than one AS and study a potential performance increase if the ASs are able to communicate. If we have more than one AS, it is important to note that all ASs will share a common goal, which in this case is to exploit the defense mechanism of the network. Our case shares a lot of similarities with the swarm robotics literature [37]. A swarm is a group of homogeneous agents that cooperate to obtain a common goal in a decentralized fashion. In our case, the ASs need to cooperatively attack the defense mechanism. There is extensive ongoing research that tries to optimize the behavior of a swarm from an evolutionary perspective, taking inspiration from nature [36]. Some examples of these algorithms are Krill Herd algorithm [33,43], Cuckoo Search [44,45], Monarch Butterfly Optimization [35,46] or Elephant Herding Optimization [34,47]. Instead of using these models, we use a gradient-based approach as in [37] to train our policy DNN.

There are several ways of modeling a swarm from a dynamic systems perspective. A frequent model is the Dec-POMDP framework [48,49], Ch. 15, which, unfortunately, has a NEXP-Complete complexity in the worst case [50]. This problem affects even recent algorithms such as [51], which can be applied only in very limited time horizons. In order to alleviate the complexity that arises under the Dec-POMDP model, a particularization of this framework for swarms was proposed in [32], called the swarMDP:

Definition 3. A swarMDP is defined in two steps. First, we define $\mathbb{A} = \langle S, A, \Omega, \pi \rangle$, where the agent prototype \mathbb{A} is an instance of each agent of the swarm:

- S is the set of local states.
- A is the set of local actions.
- Ω is the set of local observations.
- π is the local policy.

A swarMDP is a 7-tuple $\langle I, \mathbb{A}, P, R, O, \gamma \rangle$ where:

- I is the index set of the agents, where $i = 1, 2, \dots, N$ indexes the agent.
- \mathbb{A} is the agent prototype defined before.
- $P_a(s^n, s^{n+1})$ is the transition probability function defined as in the POMDP. P depends on a , the joint action vector and s is the global state.
- $R_a(s^n, s^{n+1})$ is the reward function defined as in the POMDP case. R depends on the joint action vector a . In addition, all agents i share the same reward function.
- $O_a(o^n, s^n)$ is the observation model, defined as in the POMDP case. O depends on the joint action and observation vectors a and o , respectively.
- $\gamma \in [0, 1]$ is a discount factor as in the POMDP case.

The main difference between the swarMDP and the Dec-POMDP model lies in the fact that the swarMDP explicitly assumes that all agents are homogeneous. Whereas under the Dec-POMDP model, each agent could have a different action and/or observation set, under the swarMDP model, all agents share the same local state space, action space, observation space and policy. Due to this characteristic, which is called the homogeneity

property, the agents are interchangeable. In addition, a single agent swarMDP reduces to a POMDP.

The homogeneity property simplifies the problem of learning, i.e., the order of the agents does not matter. Note that all agents share the same policy due to the homogeneity property (i.e., each agent would act like the others if they observed the same observation o^n). Thus, we can use single-agent DRL algorithms and a centralized training/decentralized execution method to find a policy [37]. In our case, we make use of TRPO and train a single policy π_θ for all agents, which takes as input the local observation of each agent o and outputs a local action a . During training, the local observations of each agent are sent to the central learning algorithm for training. During execution, each agent uses a copy of the learned policy.

Finally, the observation vector of agent i may include not only information about agent i but also information about other agents if the agents are able to communicate. Let us denote by $o_{i,i}^n$ the information available to agent i about itself in time n , and $o_{i,j}^n$ the information available to agent i about agent j , $j \neq i$. A naive way of encoding this information is to build the total observation vector of agent i , o_i^n by simply concatenating $o_{i,i}^n$ and all the vectors $o_{i,j}^n$. However, this concatenation causes a large input space, as well as ignoring the permutation invariance inherent to a homogeneous swarm. A better option consists in using mean embeddings [37,52], which are based on the fact that a probability distribution $P(X)$ can be represented as an element in a reproducing kernel Hilbert space by its expected feature map $\mu_X = \mathbb{E}_X[\phi(X)]$, where $\phi(X)$ is a feature mapping. There are several possible mean embeddings that can be used, depending on the feature mapping choice. We choose to employ the following:

- **Neural Networks Mean Embedding (NNME):** In this approach, each $o_{i,j}^n$ is used as input to a DNN, which outputs $\phi(o_{i,j}^n)$, where ϕ denotes the transformation done by the DNN. The total observation vector of agent i , o_i^n is built by concatenating $o_{i,i}^n$ to the mean of the set of all $\phi(o_{i,j}^n)$.
- **Mean-based Mean Embeddings (MME):** Under this approach, we average $o_{i,j}^n$ and concatenate it to $o_{i,i}^n$. This vector is the input to the policy network.

Thus, mean embeddings are thus insensitive to the number of agents j and are also insensitive to their order. The NNME is trained at the same time as the policy network, allowing NNME to adapt to a concrete problem setup.

3. Defense Systems

We describe two defense mechanisms based on [30] for two different attack scenarios in a WSN: a Spectrum Sensing Data Falsification (SSDF) attack and a backoff attack. The former is a physical layer defense mechanism, while the latter is an MAC layer defense mechanism.

3.1. Spectrum Sensing Data Falsification Attack

The SSDF is an attack that affects the physical layer, addressed against Cooperative Spectrum Sensing (CSS), in which a WSN is used to determine whether a wireless channel is being used or not. A survey of different SSDF attacks can be found in [53].

We use a defense mechanism based on soft fusion: the reports sent by the sensors to a central entity known as Fusion Center (FC) are the energy levels E_m they sense. As shown by [54], if the channel is idle (i.e., only noise is in the channel), E_m follows a chi-square distribution, whereas if a signal is present, E_m follows a non-central chi-square distribution. We use a hypothesis test, where H_0 means that the channel is idle and H_1 means that the channel is busy, and hence:

$$E_m \sim \begin{cases} \chi_{2k}^2 & \text{if } H_0 \\ \chi_{2k}^2(2SNR_m) & \text{if } H_1 \end{cases} \quad (2)$$

where k is the time-bandwidth product and SNR_m is the signal-to-noise ratio in sensor m in natural units. An illustration of these probability density functions (pdfs) can be found in Figure 1.

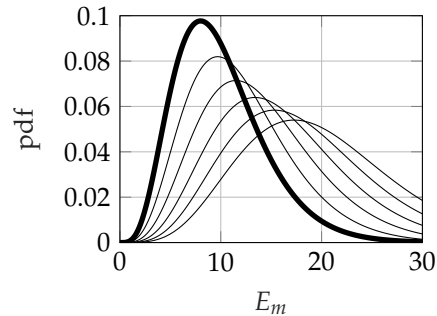


Figure 1. Illustration of the pdf of the chi-squared distributions from (2). The thick pdf corresponds to the H_0 case: the chi-squared χ_{2k}^2 distribution; and the thinner pdfs correspond to the H_1 case: the non-central chi-squared $\chi_{2k}^2(2SNR_m)$ distributions for SNR values $\{2, 4, 6, 8, 10\}$, from left to right in the plot. For all curves, $k = 5$ is the time-bandwidth parameter. Observe that, as the SNR increases, the pdf curves are more separated for H_0 and H_1 .

The SSDF attack proposed in [30] consists in reporting honestly E_m if $E_m > \xi$ and $E_m + \Delta$ if $E_m \leq \xi$, where Δ and ξ are attack parameters: Δ is the bias in the energy level introduced by the AS and ξ is the attack threshold. In other words, the AS reports that the channel is busy when it is actually idle if a certain threshold in the energy level is satisfied. We set Δ by using the means of the distributions (2), where the mean values μ of the distributions under H_0 and H_1 are:

$$\mu_0 = 2k, \quad \mu_1 = 2k + 2SNR_m. \quad (3)$$

We set $\Delta = \mu_1 - \mu_0 = 2SNR_m$, resulting in a defense mechanism that is tuned to detect a bias that tries to simulate the E_m values when the channel is busy. Graphically, according to Figure 1, this Δ value translates the H_0 pdf to the right Δ units. Depending on the value of ξ , E_m values that are actually produced under H_1 are also translated and some E_m measurements produced under H_0 are not. Observe that if the pdfs under H_0 and H_1 are close and overlap significantly (i.e., SNR_m is low for sensor m), attacking may be unnecessary in many cases.

We denote by G_0 the situation in which an AS does not attack and G_1 when it attacks. Under attack, the pdf from (2) can be written as:

$$E_m \sim \begin{cases} \chi_{2k}^2 & \text{if } G_0, H_0 \\ \chi_{2k}^2(2SNR_m) & \text{if } G_0, H_1 \\ \chi_{2k}^2 + \Delta & \text{if } G_1, H_0 \\ \chi_{2k}^2(2SNR_m) & \text{if } G_1, H_1 \end{cases}, \quad (4)$$

where we approximate the situation G_1, H_0 by a translation of the chi-squared pdf when G_0, H_0 . The accuracy of the approximation depends on the threshold ξ value. In addition, observe that (4) assumes that under H_1 hypothesis, there is no attack: again, this assumption is an approximation that depends on ξ .

The defense mechanism proposed in [30] is based on two Neyman-Pearson tests, which we summarize here for a better understanding. The first test decides whether sensor m senses a busy channel using reports from other sensors as:

$$\prod_{i=1, i \neq m}^M \frac{P(E_i = e_i | \chi_{2k}^2(2SNR_m))}{P(E_i = e_i | \chi_{2k}^2)} \underset{H_0}{\overset{H_1}{\gtrless}} \eta, \quad (5)$$

where H_1 and H_0 are the energies from (2) and η is the threshold of test 1.

The second test is used to individually detect which sensors are providing false reports (i.e., the sensors that are attacking). This test is only used for sensor m if H_0 was the result of test 1 (i.e., only noise detected) using the expressions from (4) as:

$$\frac{P(E_m = e_m - \Delta | \chi_{2k}^2)}{P(E_m = e_m | \chi_{2k}^2)} \underset{G_0}{\overset{G_1}{\geq}} \zeta, \quad (6)$$

where ζ is the threshold of test 2.

Test 2 allows detecting whether a sensor m is attacking or not. The defense mechanism keeps a reputation scheme, in which there are two values r and s for each sensor in the WSN, where s keeps track of how many times sensor m has attacked (s) and r counts how many times sensor m has not attacked. The reputation of each sensor t_{PHY} is computed as [30]:

$$t_{PHY} = \frac{r + 1}{r + s + 2}, \quad (7)$$

where, if t_{PHY} falls below a certain threshold λ_{PHY} , the sensor is considered to be an AS and is banned from the network. While in [30] the attack policy was fixed beforehand, we let our DLA choose the energy level E_m . Thus, it has to learn which value of E_m it should send and adapt it dynamically in order not to be detected by the defense mechanism. A scheme summarizing the defense mechanism is in Figure 2.

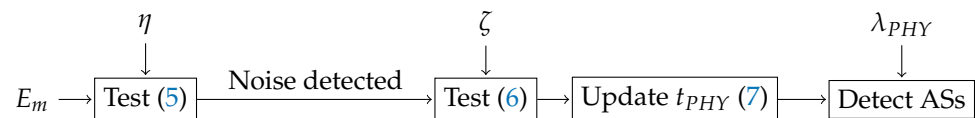


Figure 2. Block diagram for the defense mechanism against the SSDF attack. The inputs are the energy levels reported by the sensors E_m , the defense mechanism parameters are η , ζ and λ_{PHY} , and the result is an updated list of sensors detected as ASs.

3.2. Backoff Attack

The second defense mechanism used in our experiments is a backoff attack, which affects the MAC layer of any protocol that uses CSMA/CA mechanism (such as IEEE 802.11, [55] and most WSN proposed MAC protocols [56,57]). If the attack is successful, the ASs reach a higher share of the network throughput by not respecting the backoff procedure. A study of its effects can be found in [58].

As a defense method, we use a modified Cramer–von Mises (CM) statistical test [59] as in [30]. The CM test is fast to compute and allows deciding whether a stream of data is adjusting to a certain distribution, using the cumulative distribution function (CDF). We denote by x_m the observed backoff time, which is the estimated backoff window size that sensor m has used and is observed by the FC.

There are several parameters that need to be known in order to obtain x_m . According to the 802.11 standard [55], using the BA (Basic Access) mechanism, we can obtain the time that the channel is occupied when there is a transmission (T_t) or a collision (T_c) as [60]:

$$\begin{cases} T_c = H + T_p + DIFS + \delta \\ T_t = H + T_p + SIFS + T_\delta + ACK + DIFS + T_\delta \end{cases} \quad (8)$$

where H is the total header transmission time (adding PHY and MAC layers headers), $DIFS$ and $SIFS$ are interframe spacing defined in the standard, ACK is the transmission time of an acknowledgement frame (ACK), T_δ is the propagation delay and T_p is the time used to transmit payload bits. In this work, we use the values in Table 1. Using these values, the FC can obtain x_m as shown in [30].

Table 1. MAC network parameters.

Parameter	Value	Parameter	Value
MAC header	272 bits	PHY header	128 bits
ACK	112 bits + PHY header	Bit rate $R_{b,t}$	1 Mbps
SIFS	28 μ s	DIFS	128 μ s
T_δ	1 μ s	T_p	4096 μ s

In order to model the real distribution of the window backoff, we follow the mechanism used in [55]. After a successful transmission, the backoff window size is divided by 2, whereas after a collision, the backoff window size is doubled. The backoff window size is forced to be in the interval $[CW_{min}, CW_{max}]$, where these parameters are the minimum and maximum backoff window size, respectively. We consider that $CW_{min} = 32 = 2^5$ and $CW_{max} = 1024 = 2^{10}$ as in [55], with p_c being the collision probability, n_c the number of collisions and $U[\alpha, \beta]$ the random integer uniform distribution between α and β ; hence, the distribution of the window backoff if there is no attack, $f_0(x_m)$, is

$$f_0(x_m) = \begin{cases} \sum_{j=0}^{n_c} U[0, 2^{5+j}] & \text{w.p. } p_c^{n_c} (1 - p_c) \text{ if } n_c \leq 5 \\ \sum_{j=0}^5 U[0, 2^{5+j}] + \sum_{j=6}^{n_c} U[0, 2^{10}] & \text{w.p. } p_c^{n_c} (1 - p_c) \text{ if } n_c > 5 \end{cases}, \quad (9)$$

where w.p. stands for with probability. The collision probability p_c can be estimated by the FC by counting the number of successful transmissions and the number of collisions, and computing the proportion of collisions. Using (9) and the estimated p_c , we can obtain $F_0(x_m)$, the cumulative distribution of $f_0(x_m)$.

The modified CM test proposed in [30], which we summarize here for the sake of clarity, requires K observations x_1, x_2, \dots, x_K from sensor m , which are used to obtain F_1 , the empirical CDF of the window size from sensor m . The test also requires L samples y_1, y_2, \dots, y_L generated from the real distribution when there is no attack. The test statistic for each sensor θ is obtained as

$$\begin{aligned} \theta_1 &= \sum_{j=1}^K \text{sgn}(F_0(x_m) - F_1(x_m)) [F_0(x_m) - F_1(x_m)]^2, \\ \theta_2 &= \sum_{j=1}^L \text{sgn}(F_0(y_m) - F_1(y_m)) [F_0(y_m) - F_1(y_m)]^2, \\ \theta &= \frac{KL}{(K+L)^2} (\theta_1 + \theta_2), \end{aligned} \quad (10)$$

where $\text{sgn}(x)$ is the sign function. The value of θ gives a measurement of differences between the two CDFs. The magnitude of θ depends on the difference between the cumulative distributions; i.e., if F_0 and F_1 differ significantly, $|\theta|$ will be large. Moreover, observe that the sign information is crucial to determine whether F_1 is above or below F_0 , a positive θ indicates that F_0 is mostly above F_1 , which means that the backoff window values for sensor m are larger than expected, and hence, sensor i is not doing a backoff attack. The opposite happens when θ sign is negative, indicating that F_0 is mostly under F_1 , which means that the observed values of backoff window for the sensor m are smaller than expected, and hence, a backoff attack is being detected. Hence, the reputation in the MAC layer of each sensor m is determined as follows [30]:

$$t_{MAC} = e^{-D^2}, \quad D = \min\{\theta, 0\}. \quad (11)$$

In (11), t_{MAC} will be 1 (i.e., sensor m is completely trusted) when θ is positive, which is the case in which there is no attack. As θ becomes negative, t_{MAC} decreases, indicating that sensor m is less trusted because it might be performing a backoff attack. If t_{MAC} falls

below a certain threshold λ_{MAC} , the sensor m is considered to be an AS and it is banned from the network. A scheme summarizing the defense mechanism is in Figure 3.

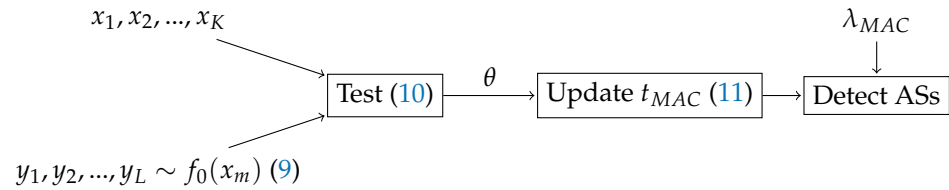


Figure 3. Block diagram for the defense mechanism against the backoff attack. The inputs are a set of K backoff windows used by a sensor and a set of L backoff window sampled from $f_0(x_m)$ (9). In this case, the defense mechanism parameters are K , L and λ_{MAC} , and the result is an updated list of sensors detected as ASs.

4. Deep Reinforcement Learning Attacker Architecture

We start our discussion of the DLA architecture by relating the attack description from Section 3 with the swarm model presented in Section 2. First, let us note that the state of the defense mechanism is the reputation value for each sensor (t_{PHY} , t_{MAC}). If the ASs had access to this value and knew the reputation threshold λ_{PHY} or λ_{MAC} , we would have an MDP setting where ASs could choose what to do in order not to be discovered following similar procedures to [61]. However, this is not a realistic assumption, as in many cases the ASs may not know the concrete parameters of the defense mechanism, or even the defense technique implemented by the defense mechanism. However, we can assume that ASs have access to values that can be observed, such as the time from last transmission attempt, the energy levels reported or whether the node has been banned from the network. This means that we shift from an MDP model to a POMDP, as ASs only have access to observations rather than the state of the defense mechanism. This also means that ASs do not have a priori knowledge about the defense mechanism. Another important remark is that we may assume that cooperation among ASs may increase the attack performance. As we have mentioned in Section 2, we can model this situation by using the swarm model. Concretely, we can use the swarMDP model, as we can assume that all ASs are homogeneous and interchangeable (i.e., they share a common goal and the set of states, actions, observations and policy), and we may benefit from the computational advantage that this model brings over the Dec-POMDP model.

Thus, let us model our attack situation by making use of the swarMDP model. We assume that we have a swarm of ASs trying to attack a WSN, where GSs are all sensors of the WSN except the ASs. At each timestep n , the i th AS has a certain observation o_i^n (which, as we will see shortly, may include information from other sensors), and it uses its policy function, $\pi_\theta(o_i^n)$, to select a certain action a_i^n . After this action is executed, the i th AS receives the common reward r^n and the next observation, o_i^{n+1} . Under this scheme, we have the following characteristics for the SSDF attack:

- A continuous set of actions in the range $[0, 1]$, where the action indicates the normalized energy that the sensor reports to the FC.
- We consider that the reward to each AS is +1 if the FC decides that there is a primary transmitting, whereas the reward to each AS is 0 if the FC decides that there is no primary transmitting. We use a maximum number of timesteps for each episode and if all ASs are discovered we terminate the episode. The DLA must therefore learn to maximize the number of timesteps without being discovered in order to maximize its reward. The punishment for being detected consists of being banned from the network, which means that the agent stops receiving rewards. Furthermore, the reward also tries to increase the probability of misdetection of the WSN; that is, it makes the FC believe that the primary is transmitting more often than it really does.
- In order to build the observations vector, each agent stores its last five actions (i.e., energy reported) and a binary flag indicating whether the agent has been already

discovered by the defense mechanism (and hence, banned from the network) or not. We assume that agent i can also access the observations of other sensors $j \neq i$, i.e., ASs can communicate their local observations to other sensors (which can be implemented either by direct communication among nodes or by observing the behavior of other sensors).

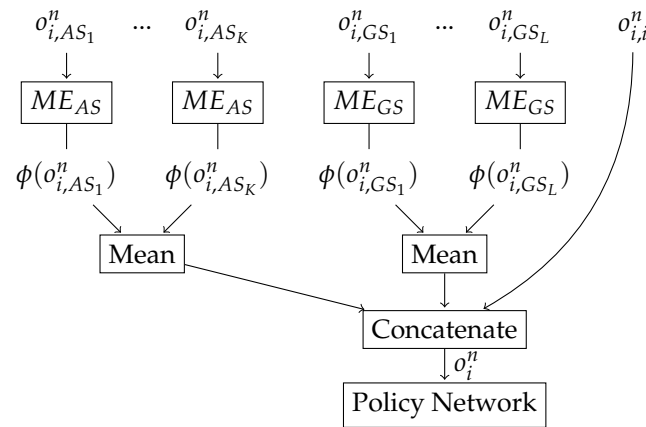
For the backoff attack, we have the following:

- The action space is composed by two discrete actions that indicate whether the sensor starts transmitting or not. That is, the actions start transmitting in the current time slot or not. Hence, two consecutive actions may be separated by several physical time slots if no sensor starts transmitting.
- The reward is -1 in case that a GS starts transmitting and 0 otherwise. Our choice of the rewards is different as the attacks have different targets: in case of the SSDF attack, we want to detect the primary as often as possible. However, in the backoff attack, we want GSs to transmit as little as possible. We set a fixed simulation time, which is completed regardless of whether the ASs are discovered. Thus, the DLA needs to learn not to be discovered while preventing GSs from transmitting.
- In order to build the observations vector, we use the time difference between the current timestep and the last K transmissions (i.e., this indicates the frequency of attempted transmission of the sensor). This difference is normalized by the maximum number of timesteps. We also add a binary flag indicating whether the agent has been discovered by the defense mechanism or not, as in the SSDF case.

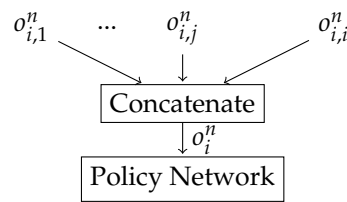
We remark that, in both attacks, the observation does not match the system state, which are the reputation (t_{PHY} and t_{MAC}) of each sensor. Rather, each ASs only has access to a limited set of information about its past actions and information about whether a sensor has been banned or not by the defense mechanism. In addition, since we use a model-free DRL approach, and we do not require a model of the transition probabilities for the attacks, which is also a significant advantage over methods that require explicit models of these probabilities, as this allows us to significantly ease the required computational load [31]. Furthermore, we aim to study whether communicating the local observations can be exploited by the DLA agent to learn better attack mechanisms. Therefore, we also compare with the non-communication case.

The set of observations, actions and rewards of each agent in each timestep is used to update the common policy π_θ using TRPO. We use as policy a feed-forward DNN, which takes as input the observation vector o_i^n and outputs the action. The two first layers of the network have 256 neurons and use rectified linear activations. As TRPO can be used for continuous and discrete actions, we use it for both of our attacks, as the SSDF attack has a continuous action set, while the backoff attack has two discrete actions.

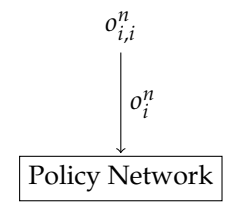
Finally, as mentioned before, we want to test the influence of communication in the attack performance. As mentioned before, a possible option consists of concatenating the local observation of each sensor, although this approach means that the policy input depends on the number of sensors, hence complicating the learning process when there are several sensors. Moreover, this method is not invariant to the number and order of sensors. A better alternative is MEs, which we use to combine the observations from the agents in the case that there is communication in a meaningful way. We emphasize that MEs allow a combination that is invariant to the number and order of the agents. We remark that there are two different kind of sensors: ASs and GSs. In order to combine the sensor observations meaningfully, we concatenate one mean embedding for the observations of the ASs and another with the observations of the GSs. A graphical description of all the architectures that we use are in Figure 4.



(a) Communication and ME



(b) Communication without ME



(c) No communication

Figure 4. Sketch of the different DLA architectures. The difference in the architectures lies in how the observation o_i^n is obtained. In (a,b), there is communication among the swarm agents, and hence, each agent i has access to the local observations of the rest of the agents. (a) shows the architecture when a Mean Embedding is used: we use separate Mean Embeddings for ASs and GSs, and we assume that there are $K + 1$ ASs (agent i is also an AS), L GSs and that o_i^n is the concatenation of the mean values of the Mean Embeddings and the local information of the agent i . (b) shows the architecture when there is communication but we do not use any Mean Embedding: in this case, o_i^n is the concatenation of the observations. (c) shows the no-communication case in which only the local observation is available.

5. Results

We evaluate our DLA architectures on the SSDF and the backoff attack on a WSN that contains 10 GSs and $\{1, 3, 10\}$ ASs. For each attack, we tested four different setups:

- Communication and Neural Networks Mean Embedding (CNNME) setting: in this case, ASs communicate among them and use MEs to aggregate information (situation (a) in Figure 4). Concretely, we use NNME, which is based on using a DNN as ME where the weights of the DNN are trained together with the TRPO policy.
- Communication and Mean-based Mean Embedding (CMME) setting: in this case, ASs communicate among them and use MEs to aggregate information (situation (a) in Figure 4). Concretely, we use MME, which consists of using the mean operation as ME.
- Communication without using mean embeddings (C) setting: in this case, ASs communicate among them without using any ME to aggregate information (situation (b) in Figure 4). In this case, the input size of the policy network is equal to the dimension of o_i^n and thus increases with the number of ASs, while it remains invariant for all other cases.
- Without any communication among ASs (NC) setting (situation (c) in Figure 4). In this case, each AS only uses its local observations in order to obtain their local policy.

For CNNME, CMME and C settings, the reward that each agent maximizes is the sum of the rewards of all ASs, by following the swarMDP model (i.e., they have a common goal). We train the policy network until convergence, using 500 TRPO iterations for both attacks. In each iteration, a batch with 10^4 timesteps is used to optimize the policy. For each combination of number of ASs, DLA setup and attack type, we repeat the training

using 10 different seeds, as the results of DRL methods are known to be dependent on initial conditions [62]. The code used is available at <https://github.com/jparras/dla> (last accessed: 11 June 2021).

5.1. Baselines

We remark that we do not know the optimal solution to the underlying POMDP that models the attack. In order to evaluate the quality of the results of the DLA agent, we compare the results obtained to three baselines policies based on the always-false, always-busy and always-free attack policies from [63]:

- Random policy (RP), which samples the actions uniformly from the action space; i.e., in the SSDF attack, it means that ASs report a random energy level, and in the backoff attack, ASs transmit randomly.
- Always High (AH), which selects the highest action possible; i.e., in the SSDF attack, it means that ASs always report the maximum energy level, and in the backoff attack, ASs always transmit.
- Always Low (AL), which selects the lowest action possible; i.e., in the SSDF attack, it means that ASs always report the minimum energy level, and in the backoff attack, ASs never transmit.

The advantages of these baselines used in the literature are that they are simple to implement, they do not require knowing the defense mechanism and they have a low complexity. However, they are unable to adapt in order to exploit a certain defense mechanism, although, as shown in [63], they can be successful attack policies in some cases. More complex attacks could be devised by having knowledge of the defense mechanism [31], although we do not use them in this paper to have a fair comparison with DLA: none of the policies we test relies on an a priori knowledge of the defense mechanism.

5.2. Spectrum Sensing Data Falsification Attack

We use finite horizon episodes; i.e., in each episode, the FC asks the sensors up to 250 times to report the energy level they measure. We consider that the duty cycle is 0.2; i.e., the probability that the channel is actually occupied by a primary transmitter is 0.2. If an AS is detected, the episode ends for this AS, since the FC does not ask it to send more reports. We implement the defense mechanism explained in Section 3.1, with $\eta = 1$, $\zeta = 1.6$ [30] and $\lambda_{PHY} = 0.5$. If the reputation of a sensor t_{PHY} falls below λ_{PHY} , the sensor is detected as an AS and the episode ends. If the AS attacks indiscriminately, the episode will end early and its reward will be low. At each timestep, the defense mechanism is invoked and the sensor reputation is updated.

At the beginning of each episode, we pick the sensor distances to the FC, d_m , from a uniform random distribution in the range [800, 1000] m. We consider that the transmitter power is $P_{tx} = 23$ dBm and use the following path loss expression:

$$P_m = P_{tx} - (35 + 3 \cdot 10 \log_{10}(d_m)), \quad (12)$$

where P_m is the received power in the sensor m in dBm. This expression allows us to obtain $SNR_m = 10^{\frac{P_m - NP}{10}}$, where we consider the noise power to be $NP = -110$ dBm. We consider the time-bandwidth product to be $k = 5$. Finally, we generate the E_m values sampling from the distributions in (2), depending on whether the primary is transmitting or not.

5.3. Backoff Attack

The backoff attack is simulated for $5 \cdot 10^5$ μ s. In each timestep, an AS decides whether to start transmitting or stay idle. Hence, timesteps are related to backoff steps, not to physical time (i.e., if an agent starts transmitting in timestep n , timestep $n + 1$ will take place when that agent finishes transmitting). We do not penalize collisions. The defense system explained in Section 3.2 is used, with $K = 5$ and $L = 1000$. If the reputation of a sensor t_{MAC} falls below $\lambda_{MAC} = 0.5$, the episode ends for this sensor. We note that

if the AS attacks indiscriminately, the episode ends early with a low reward: recall that a final reward is given to each AS that corresponds to the remaining reward of the episode; i.e., if the AS is caught, it does not have the opportunity any more to hinder the GSs to transmit, yielding a lower reward. We use the network parameters from Table 1 to simulate the backoff attack. The defense mechanism is run once every five timesteps in order to ease the computational load.

5.4. Results

The results for both attacks can be observed in Figure 5 and Tables 2–4. First, in Figure 5, we observe that ASs do learn to successfully exploit the defense mechanism in both attacks, as the reward increases with the TRPO iterations. There is a higher variability among seeds in the SSDF attack, which is a known problem that arises when using DRL methods [62]. The remarkable fact is that this attack improvement is achieved simply by interacting with the defense mechanism, as the ASs do not have a priori knowledge about it.

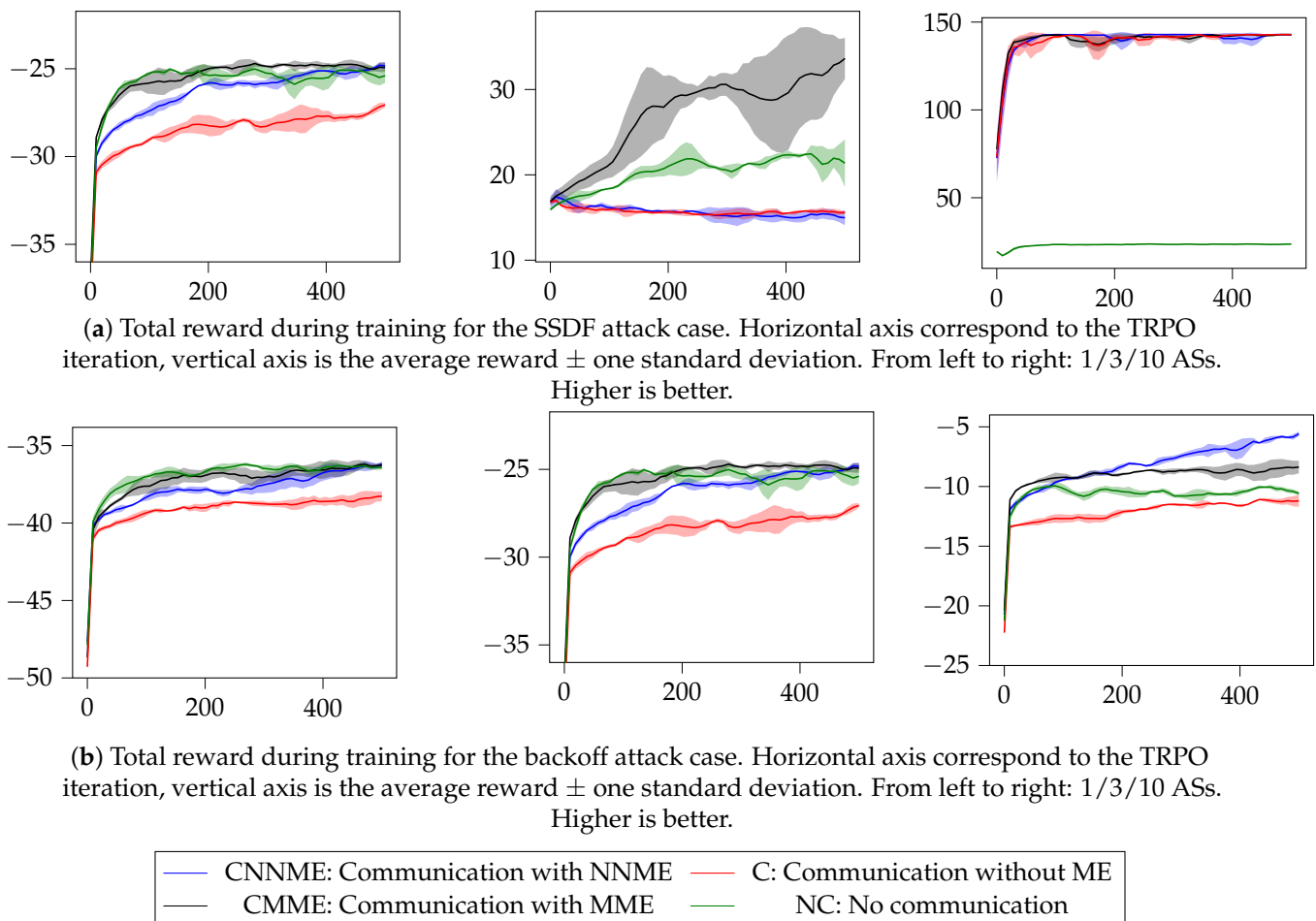


Figure 5. Training results for the SSDF and backoff attacks for the best three seeds. We observe that the variability between seeds is higher in the SSDF attack. Also, note how DLAs improve their reward as training progresses, and specifically, observe that they do so simply by interacting with the defense mechanism.

Table 2. Final rewards obtained for each combination of attack, number of ASs and setup. The values were obtained averaging 50 episodes for the best seed of each attack policy after training. We show the average reward \pm one standard deviation. Bold entries show the best results, where a Welch test is used to detect whether means are significantly different for a significance level $\alpha = 0.01$. Higher is better.

	ASs	CNNME	CMME	C	NC	RP	AH	AL
SSDF	1	22.66 \pm 3.36	22.07 \pm 4.54	19.93 \pm 4.92	22.79 \pm 4.87	5.35	0.30	11.66
	3	18.02 \pm 6.02	36.22 \pm 10.51	17.4 \pm 6.03	22.95 \pm 3.33	14.47	1.42	0.53
	10	142.88 \pm 0.00	142.88 \pm 0.00	142.88 \pm 0.00	24.97 \pm 6.78	23.06	142.88	0.00
Backoff	1	−35.86 \pm 4.97	−35.54 \pm 3.96	−37.51 \pm 3.46	−34.90 \pm 3.30	−75.88	−77.81	−44.62
	3	−24.48 \pm 3.74	−24.36 \pm 3.47	−26.72 \pm 3.68	−24.7 \pm 3.21	−71.40	−78.84	−43.67
	10	−4.38 \pm 3.10	−7.54 \pm 2.49	−10.39 \pm 3.10	−9.12 \pm 2.92	−66.43	−78.16	−43.99

Table 3. Proportion of agents detected as ASs by the defense mechanism for each combination of attack, number of ASs and setup. The values were obtained averaging 50 episodes for the best seed of each policy after training. We show the average proportion \pm one standard deviation.

	ASs	CNNME	CMME	C	NC	RP	AH	AL
SSDF	1	0.00 \pm 0.00	2.00 \pm 14.00	2.00 \pm 14.00	2.00 \pm 14.00	82.00	100.00	0.00
	3	53.33 \pm 25.82	0.00 \pm 0.00	54.00 \pm 24.85	22.00 \pm 20.67	77.33	100.00	0.00
	10	0.00 \pm 0.00	0.00 \pm 0.00	0.00 \pm 0.00	61.00 \pm 21.93	80.60	0.00	0.00
Backoff	1	22.00 \pm 41.42	4.00 \pm 19.60	42.00 \pm 49.36	8.00 \pm 27.13	100.00	100.00	0.00
	3	47.33 \pm 35.96	30.00 \pm 34.80	41.33 \pm 28.72	21.33 \pm 21.87	100.00	100.00	0.00
	10	66.80 \pm 18.16	59.20 \pm 26.97	74.00 \pm 18.00	40.60 \pm 21.58	100.00	100.00	0.00

Table 4. Final results in terms of primary detection percentage (SSDF) and kbits that GSs transmit (backoff), obtained for each combination of attack, number of ASs and setup. The values were obtained averaging 50 episodes for the best seed of each attack policy after training. We show the average value \pm one standard deviation. Higher is better in SSDF; lower is better in backoff attack.

	ASs	CNNME	CMME	C	NC	RP	AH	AL	No ASs
SSDF	1	15.91 \pm 2.22	15.34 \pm 3.16	14.65 \pm 3.40	16.06 \pm 3.26	9.84	10.80	8.24	8.00
	3	14.61 \pm 12.68	25.75 \pm 7.36	12.03 \pm 4.14	16.58 \pm 2.23	15.78	45.80	0.35	28.00
	10	100.00 \pm 0.00	100.00 \pm 0.00	100.00 \pm 0.00	17.16 \pm 4.42	18.09	100.00	0.00	22.00
Backoff	1	258.79 \pm 19.43	258.46 \pm 17.36	275.01 \pm 15.78	252.81 \pm 16.83	328.83	326.78	350.13	255.4
	3	167.03 \pm 24.65	160.65 \pm 19.23	182.85 \pm 26.77	161.46 \pm 17.51	312.20	331.12	348.00	211.87
	10	27.53 \pm 23.69	43.50 \pm 16.54	60.87 \pm 17.84	54.07 \pm 20.50	295.65	328.25	348.73	130.57

In Table 2, we show the final rewards obtained by using all the DLA architectures and compare them with the baseline policies. The baselines provide clearly worse results than the DLA architectures proposed. If we focus on the results of the SSDF attack, we first see that DLA with communication and ME have an advantage over the other methods. With a single AS, the performance of all DLA is similar. When there are 10 ASs, the defense mechanism can be overpowered by continuously reporting high energy levels: thus, not only all communication-based DLAs, but also the AH baseline, provide the same rewards. In all cases, all DLAs provide a significantly better reward than the baselines (except for the commented case of AH baseline with 10 ASs).

If we focus on the results of the backoff attack, which has a statistically complex defense mechanism (see Section 3.2), we note, again, that the baseline results are considerably improved by all DLAs. This is due to the fact that DLAs are able to learn to exploit the defense mechanism successfully, while the baselines fail to provide good results. In this

case, having communication between agents using an ME consistently provides the best results: we believe that this is due to the fact that the swarMDP model is specially suited for the context of network attacks, as it facilitates exploiting the defense mechanism by using the local observations of the other nodes.

One of the main targets in our DLA was to be able to attack without being discovered, that is, to camouflage. Hence, the rewards presented in Section 4 were designed to exploit the defense mechanism and not be detected. Thus, the reward scheme implicitly induces a trade-off between camouflage and exploit of the defense mechanism that can be observed in Tables 3 and 4. First, Table 3 shows the detection proportion for each DLA and baseline: in the SSDF case, a low detection is usually related to a high reward (i.e., low detection and attack come together) except for the AL baseline, which we remark consisted of always reporting a low energy level (and hence, ASs are never detected). However, in the backoff attack, the ranges of detection proportion are larger: this may be due to a cooperative behavior among ASs, such that some ASs are detected for the sake of getting a better reward (which, recall, is shared among all ASs).

Table 4 shows the attack results in terms directly related to the attack target. In the SSDF case, we show the proportion of times that the primary is detected, which is always in the 10 ASs case: as we mentioned before, 10 ASs are enough to overpower the defense mechanism in this attack. In the backoff case, we show the bits transmitted by GSs, and there are two conclusions from the results that are important. First, the baselines actually provide GSs with more resources: either the ASs are detected fast (RP, AH baselines in Table 3) and hence banned from the network, or they do not attack (AL baseline), and both situations lead to the network to have less sensors to split the bandwidth, as mostly GSs are transmitting. For instance, in the 10 ASs case, there are 10 GSs to access the bandwidth, as 10 ASs are either banned or not transmitting, compared to 20 sensors if all the sensors were GSs. Second, the DLA may significantly reduce the transmission rates of GSs, up to an 80% in the case of the CNNME with 10 ASs, which is a very notable decrease in the system throughput.

6. Conclusions

In this work, we propose using DRL tools in order to create an attacker architecture able to challenge several defense mechanisms used in WSN. We considered attacks on the PHY and MAC layers, and show that our approach poses strong challenges to current defense mechanisms:

1. We do not need to know the state of the system (i.e., the reputations), as DLA relies only on partial observations (i.e., information about how the sensors have interacted in the past that can be observed by other sensors). Even though we have only partial observations, they provide very good results, especially when the ASs are able to communicate their observations—in Table 2, the results using communication and ME consistently are the best. However, even when this communication is not considered—i.e., the NC setup—the results are still better than the baselines. Even though the underlying model of the attack is a POMDP, DLA learns to attack having only a limited amount of past observations. Thus, when attacking a network, if communication among ASs is possible, our results point towards using ME to aggregate the local information in a meaningful way. We test using two different ME types: NNME, which may extract better features at the cost of extra training complexity, and MMEs, which provide a lower training computational cost but obtain more limited features for the policy. The results shown in Table 2 show that both MEs provide good results, and hence, choosing between them may depend on the concrete problem to solve and the training restrictions we may have.
2. Since DLA does not need to know a priori which defense mechanism it is facing, it is a very flexible approach. Thus, DLA could be the base for a universal attacker, successful in exploiting many defense mechanisms. Indeed, DRL methods have been used to provide human-like performance in many Atari games [5], so DRL could also be applied to exploit defense mechanisms in a universal fashion.

3. It is a method with balanced computational requirements. The training process is the most computationally expensive part of the system. However, most of this cost was used in generating samples from the defense mechanisms: training the DNNs using gradient techniques was fast. This low training cost appears because we use simple DNNs, which, however, are enough to exploit the defense mechanisms. Once the DNN is trained, the policy is quick to execute and can be deployed in devices with low computational capabilities.
4. We remark that we have used the same set of hyper-parameters for all of our simulations. We have done no fine-tuning of these hyper-parameters, and thus, our approach may suit very different attack situations with minimal tuning. Equivalently, the results obtained could be improved by doing a fine-tuning for each situation.

However, there are also several drawbacks that arise from the results of this paper and that are future lines of research that can further advance this work:

1. The reward scheme has a strong influence on the attack that is learned. There is a tradeoff between attacking and not being discovered, and hence, modifying the reward scheme will cause the DLA to learn a different attack strategy. In other words, we must carefully design the reward depending on the attack result desired. Note that this is not something specific to our problem but a general problem that arises in the RL field.
2. Our approach relies on the DLA being able to interact with a network continuously, episode after episode. As we considered that ASs could be banned during episodes, this means that the banning must be temporal. If it is permanent, then the DLA could not learn. Note, however, that if ASs had access to a simulator of the defense mechanism, this problem would be overcome.
3. Our DLA is not sample-efficient, as it requires many samples to learn; in addition, the results are dependent on the initial parameters of the policy DNN. These are known problems of DRL methods, which are subject to current research [62]. A very promising research field that could address these problems is few-shot learning, focused on learning from a few samples. There are several works in this area that may be used to improve the sample efficiency of DRL methods, such as [64,65], and hence could further improve the results of attacks to WSN.
4. A related challenge comes from the fact that we have assumed a static defense mechanism, but it could be dynamic and change after a series of attack attempts have been detected. DLA may be currently vulnerable to this defense strategy due to its low sample efficiency, and hence, we expect that the research on increasing the DRL sample efficiency would also be useful to prevent and/or adapt quickly to changes in the defense mechanism. Another alternative could consist in replacing a banned AS with a new one that is initialized based on the experience of the banned AS (i.e., by having the same policy as the banned one).

Hence, the attacking approach that we propose in this work presents strong challenges to current WSN defense mechanisms. First, because of the growing computational capabilities of current hardware, there could soon, if not already, be sensors with enough computational capabilities to implement a DLA [66]. An alternative approach could be based on the use of evolutionary techniques to perform the optimization, as there are many swarm algorithms based on these ideas such as [33–36]. Second, because DLA is adaptive and flexible, not requiring an a priori modeling of the defense mechanism nor knowledge of their parameters, it can learn to exploit a wide range of defense mechanisms. Thus, it is of capital importance to research defense mechanisms against such attack mechanisms, in order to minimize the threat they pose. A promising defense mechanism could be one in which the defense mechanism also uses RL tools for learning how to defend, which could mean entering the field of Multi Agent Competitive Learning [67,68], which still poses strong challenges.

Author Contributions: Conceptualization, J.P., M.H., S.Z. and G.N.; methodology, J.P., M.H., S.Z. and G.N.; software, J.P. and M.H.; validation, J.P., M.H., S.Z. and G.N.; formal analysis, J.P., M.H., S.Z. and G.N.; investigation, J.P., M.H., S.Z. and G.N.; resources, J.P., M.H., S.Z. and G.N.; writing—original draft preparation, J.P.; writing—review and editing, J.P., M.H., S.Z. and G.N.; visualization, J.P.; supervision, S.Z. and G.N.; project administration, S.Z. and G.N.; funding acquisition, S.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a Ph.D. grant given to the first author by Universidad Politécnica de Madrid, as well as by the Spanish Ministry of Science and Innovation under the grant TEC2016-76038-C3-1-R (HERAKLES).

Acknowledgments: We would like to thank Riccardo Polvara for his hardware support and expertise, which significantly helped while testing the software used in this work.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

A3C	Asynchronous Advantage Actor Critic
ACER	Actor Critic with Experience Replay
ACK	Acknowledgement
AH	Always High
AL	Always Low
AS	Attacking Sensor
CDF	Cumulative Distribution Function
CM	Cramer - von Mises
CNNME	Communication and Neural Networks Mean Embedding
CMME	Communication and Mean-based Mean Embedding
DLA	Deep Learning Attacker
DNN	Deep Neural Network
DoS	Denial of Service
DRL	Deep Reinforcement Learning
DRQN	Deep Recurrent Q-Networks
DQN	Deep Q-Networks
FC	Fusion Center
GS	Good Sensor
MAC	Medium Access Control
MDP	Markov Decision Process
ME	Mean Embedding
MME	Mean-based Mean Embedding
NNME	Neural Networks Mean Embedding
PHY	Physical
POMDP	Partially Observable Markov Decision Process
PPO	Proximal Policy Optimization
QoS	Quality of Service
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RP	Random Policy
SNR	Signal-to-Noise Ratio
SSDF	Spectrum Sensing Data Falsification
TRPO	Trust Region Policy Optimization
WSN	Wireless Sensor Network

References

1. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998; Volume 1.
2. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285.
3. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.

4. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
5. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; others. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.
6. Hausknecht, M.; Stone, P. Deep recurrent q-learning for partially observable mdps. *arXiv* **2015**, arXiv:1507.06527.
7. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
8. Wang, Z.; Bapst, V.; Heess, N.; Mnih, V.; Munos, R.; Kavukcuoglu, K.; de Freitas, N. Sample efficient actor-critic with experience replay. *arXiv* **2016**, arXiv:1611.01224.
9. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, Lille, France 6–11 July 2015; pp. 1889–1897.
10. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
11. Alsheikh, M.A.; Lin, S.; Niyato, D.; Tan, H.P. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Commun. Surv. Tutorials* **2014**, *16*, 1996–2018.
12. Curiac, D.; Volosencu, C.; Doboli, A.; Dranga, O.; Bednarz, T. Neural network based approach for malicious node detection in wireless sensor networks. In Proceedings of the WSEAS International Conference on Circuits, Systems, Signal and Telecommunications, Gold Coast, QLD, Australia, 17–19 January 2007; pp. 17–19.
13. Curiac, D.I.; Plastoi, M.; Baniyas, O.; Volosencu, C.; Tudoroiu, R.; Doboli, A. Combined malicious node discovery and self-destruction technique for wireless sensor networks. In Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications, Athens, Greece, 18–23 June 2009; pp. 436–441.
14. Yang, K. *Wireless Sensor Networks*; Springer: Berlin, Germany, 2014.
15. Rawat, P.; Singh, K.D.; Chaouchi, H.; Bonnin, J.M. Wireless sensor networks: A survey on recent developments and potential synergies. *J. Supercomput.* **2014**, *68*, 1–48.
16. Ndiaye, M.; Hancke, G.P.; Abu-Mahfouz, A.M. Software defined networking for improved wireless sensor network management: A survey. *Sensors* **2017**, *17*, 1031.
17. Shi, Y.; Sagduyu, Y.E.; Erpek, T.; Davaslioglu, K.; Lu, Z.; Li, J.H. Adversarial deep learning for cognitive radio security: Jamming attack and defense strategies. In Proceedings of the 2018 IEEE International Conference on Communications Workshops (ICC Workshops), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6.
18. Xiao, L.; Wan, X.; Lu, X.; Zhang, Y.; Wu, D. IoT Security Techniques Based on Machine Learning. *arXiv* **2018**, arXiv:1801.06275.
19. Cannady, J. Next generation intrusion detection: Autonomous reinforcement learning of network attacks. In Proceedings of the 23rd National Information Systems Security Conference, Baltimore, MD, USA, 16–19 October 2000; pp. 1–12.
20. Gwon, Y.; Dastangoo, S.; Fossa, C.; Kung, H. Competing mobile network game: Embracing antijamming and jamming strategies with reinforcement learning. In Proceedings of the 2013 IEEE Conference on Communications and Network Security (CNS), National Harbor, MD, USA, 14–16 October 2013; pp. 28–36.
21. Xiao, L.; Li, Y.; Liu, G.; Li, Q.; Zhuang, W. Spoofing detection with reinforcement learning in wireless networks. In Proceedings of the Global Communications Conference (GLOBECOM), 2015 IEEE, San Diego, CA, USA, 6–10 December 2015; pp. 1–5.
22. Xiao, L.; Xie, C.; Chen, T.; Dai, H.; Poor, H.V. A mobile offloading game against smart attacks. *IEEE Access* **2016**, *4*, 2281–2291.
23. Xiao, L.; Li, Y.; Huang, X.; Du, X. Cloud-based malware detection game for mobile devices with offloading. *IEEE Trans. Mob. Comput.* **2017**, *16*, 2742–2750.
24. Aref, M.A.; Jayaweera, S.K.; Machuzak, S. Multi-agent reinforcement learning based cognitive anti-jamming. In Proceedings of the Wireless Communications and Networking Conference (WCNC), 2017 IEEE, San Francisco, CA, USA, 19–22 March 2017; pp. 1–6.
25. Han, G.; Xiao, L.; Poor, H.V. Two-dimensional anti-jamming communication based on deep reinforcement learning. In Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing, New Orleans, LA, USA, 5–9 March 2017.
26. Li, Y.; Quevedo, D.E.; Dey, S.; Shi, L. SINR-based DoS attack on remote state estimation: A game-theoretic approach. *IEEE Trans. Control Netw. Syst.* **2017**, *4*, 632–642.
27. Li, M.; Sun, Y.; Lu, H.; Maharjan, S.; Tian, Z. Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems. *IEEE Internet Things J.* **2019**, *7*, 6266–6278.
28. Fragkiadakis, A.G.; Tragos, E.Z.; Askoxylakis, I.G. A survey on security threats and detection techniques in cognitive radio networks. *IEEE Commun. Surv. Tutorials* **2013**, *15*, 428–445.
29. Sokullu, R.; Dagdeviren, O.; Korkmaz, I. On the IEEE 802.15. 4 MAC layer attacks: GTS attack. In Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008), Cap Esterel, France, 25–31 August 2008; pp. 673–678.
30. Wang, W.; Sun, Y.; Li, H.; Han, Z. Cross-layer attack and defense in cognitive radio networks. In 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), Miami, FL, USA, 6–10 Dec. 2010; pp. 1–6.
31. Parras, J.; Zazo, S. Learning attack mechanisms in Wireless Sensor Networks using Markov Decision Processes. *Expert Syst. Appl* **2019**, *122*, 376–387.

32. Šošić, A.; KhudaBukhsh, W.R.; Zoubir, A.M.; Koepl, H. Inverse reinforcement learning in swarm systems. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS 17), São Paulo, Brazil, 8–12 May 2017; pp. 1413–1421.
33. Wang, G.G.; Gandomi, A.H.; Alavi, A.H.; Gong, D. A comprehensive review of krill herd algorithm: Variants, hybrids and applications. *Artif. Intell. Rev.* **2019**, *51*, 119–148.
34. Li, J.; Lei, H.; Alavi, A.H.; Wang, G.G. Elephant herding optimization: Variants, hybrids, and applications. *Mathematics* **2020**, *8*, 1415.
35. Feng, Y.; Deb, S.; Wang, G.G.; Alavi, A.H. Monarch butterfly optimization: A comprehensive review. *Expert Syst. Appl.* **2020**, *168*, 114418.
36. Li, W.; Wang, G.G.; Gandomi, A.H. A survey of learning-based intelligent optimization algorithms. *Arch. Comput. Methods Eng.* **2021**, <https://doi.org/10.1007/s11831-021-09562-1>, 1–19.
37. Hüttenrauch, M.; Šošić, A.; Neumann, G. Deep Reinforcement Learning for Swarm Systems. *J. Mach. Learn. Res.* **2019**, *20*, 1–31.
38. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; MIT Press: Cambridge, MA, USA, 2005.
39. Bertsekas, D.P. *Dynamic Programming and Optimal Control*; Athena Scientific: Belmont, MA, USA, 1995; Volume 1.
40. Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June, 2016; pp. 1329–1338.
41. Littman, M.L.; Sutton, R.S.; Singh, S.P. Predictive representations of state. In *Advances in Neural Information Processing Systems (NIPS)*; MIT Press: Cambridge, MA, USA, 2001; Volume 14, p. 30.
42. Singh, S.P.; Littman, M.L.; Jong, N.K.; Pardoe, D.; Stone, P. Learning predictive state representations. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), Washington, DC, USA, 21–24 August 2003; pp. 712–719.
43. Wang, G.G.; Deb, S.; Gandomi, A.H.; Alavi, A.H. Opposition-based krill herd algorithm with Cauchy mutation and position clamping. *Neurocomputing* **2016**, *177*, 147–157.
44. Li, J.; Li, Y.X.; Tian, S.S.; Xia, J.L. An improved cuckoo search algorithm with self-adaptive knowledge learning. *Neural Comput. Appl.* **2019**, *32*, 11967–11997.
45. Li, J.; Yang, Y.H.; Lei, H.; Wang, G.G. Solving Logistics Distribution Center Location with Improved Cuckoo Search Algorithm. *Int. J. Comput. Intell. Syst.* **2020**, *14*, 676–692.
46. Feng, Y.; Wang, G.G.; Dong, J.; Wang, L. Opposition-based learning monarch butterfly optimization with Gaussian perturbation for large-scale 0-1 knapsack problem. *Comput. Electr. Eng.* **2018**, *67*, 454–468.
47. Li, W.; Wang, G.G. Elephant herding optimization using dynamic topology and biogeography-based optimization based on learning for numerical optimization. *Eng. Comput.* **2021**, 1–29, <https://doi.org/10.1007/s00366-021-01293-y>.
48. Wiering, M.; Van Otterlo, M. Reinforcement learning. *Adapt. Learn. Optim.* **2012**, *12*, 51.
49. Oliehoek, F.A.; Spaan, M.T.; Vlassis, N. Optimal and approximate Q-value functions for decentralized POMDPs. *J. Artif. Intell. Res.* **2008**, *32*, 289–353.
50. Bernstein, D.S.; Givan, R.; Immerman, N.; Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* **2002**, *27*, 819–840.
51. Dibangoye, J.S.; Amato, C.; Buffet, O.; Charpillet, F. Optimally solving Dec-POMDPs as continuous-state MDPs. *J. Artif. Intell. Res.* **2016**, *55*, 443–497.
52. Smola, A.; Gretton, A.; Song, L.; Schölkopf, B. A Hilbert space embedding for distributions. In Proceedings of the International Conference on Algorithmic Learning Theory, Sendai, Japan, 1–4 October 2007; pp. 13–31.
53. Zhang, L.; Ding, G.; Wu, Q.; Zou, Y.; Han, Z.; Wang, J. Byzantine attack and defense in cognitive radio networks: A survey. *IEEE Commun. Surv. Tutorials* **2015**, *17*, 1342–1363.
54. Urkowitz, H. Energy detection of unknown deterministic signals. *Proc. IEEE* **1967**, *55*, 523–531.
55. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Computer Society, **2016**. pp. 1–3534.
56. Demirkol, I.; Ersoy, C.; Alagoz, F. MAC protocols for wireless sensor networks: A survey. *IEEE Commun. Mag.* **2006**, *44*, 115–121.
57. Yadav, R.; Varma, S.; Malaviya, N. A survey of MAC protocols for wireless sensor networks. *UbiCC J.* **2009**, *4*, 827–833.
58. Parras, J.; Zazo, S. Wireless Networks under a Backoff Attack: A Game Theoretical Perspective. *Sensors* **2018**, *18*, 404.
59. Anderson, T.W. On the distribution of the two-sample Cramer-von Mises criterion. *Ann. Math. Stat.* **1962**, *33*(3), 1148–1159.
60. Bianchi, G. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE J. Sel. Areas Commun.* **2000**, *18*, 535–547.
61. Parras, J.; Zazo, S. Using one class SVM to counter intelligent attacks against an SPRT defense mechanism. *Ad Hoc Netw.* **2019**, *94*, 101946.
62. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep reinforcement learning that matters. In Proceedings of the AAAI Conference on Artificial Intelligence, 2018; New Orleans, LA, USA, February 2–7, 2018; pp. 3207–3214.
63. Zhu, F.; Seo, S.W. Enhanced robust cooperative spectrum sensing in cognitive radio. *J. Commun. Netw.* **2009**, *11*, 122–133.
64. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August, 2017; pp. 1126–1135.

-
65. Jamal, M.A.; Qi, G.J. Task agnostic meta-learning for few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June, 2019; pp. 11719–11727.
 66. Payal, A.; Rai, C.S.; Reddy, B.R. Analysis of some feedforward artificial neural network training algorithms for developing localization framework in wireless sensor networks. *Wirel. Pers. Commun.* **2015**, *82*, 2519–2536.
 67. Hernandez-Leal, P.; Kaisers, M.; Baarslag, T.; de Cote, E.M. A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. *arXiv* **2017**, arXiv:1707.09183.
 68. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839.