# Using One Class SVM to Counter Intelligent Attacks against an SPRT Defense Mechanism

Juan Parras*, Santiago Zazo

*Information Processing and Telecommunications Center*
*Universidad Politécnica de Madrid*
*ETSI Telecomunicación, Av. Complutense 30, 28040 Madrid (Spain)*

**Abstract**

A widely used defense mechanism in current wireless sensor networks is based on using a Sequential Probability Ratio Test (SPRT). This test does not require a fixed sample size, and this reduces the number of communications required and the battery consumption, both of which are of capital interest in such networks. However, SPRT assumes that the distributions under test do not change, and this assumption needs not hold for current attackers.

We divide this work in two parts, First, we develop an optimal attack strategy against a Bernoulli SPRT mechanism which is used in many defense mechanisms in current literature. The control law for such an attacker turns out to be easy to implement and very effective, thus posing a significant threat for the defense mechanisms that use such SPRT. Second, we make use of One Class Supporting Vector Machines to obtain a modified SPRT test that is able to detect, not only such an attacker, but potentially any other attack mechanism that has not a similar spectrum to the expected signal from normal sensors. Our work is validated via simulations, showing that the attacker we propose is a real threat to SPRT mechanisms, but also, that our proposed defense mechanism can efficiently cope with such an attacker.

*Keywords:*
SPRT, Optimal Attack, One Class Supporting Vector Machine

*Corresponding author

*Email addresses:* `j.parras@upm.es` (Juan Parras ), `santiago.zazo@upm.es` (Santiago Zazo)

## 1. Introduction

The detection problem tries to characterize the statistical behavior of an observed signal. This problem is usually posed as a hypothesis test (HT) and has been widely studied in the signal theory field [1]. One of many possible taxonomies to classify HT is based on the sample size: whether needed number of samples to make a decision is fixed in advance or not. The former case is the most usual: the well-known Neyman-Pearson HT belongs to this kind [2], [1], among many others, such as Rao or Wald tests [3]. The latter case is known as sequential hypothesis test, and traces back to the work of Wald on the Sequential Probability Ratio Test (SPRT) [4] [5]. In an SPRT, a sample from the signal of interest is collected at each time step $n$ and used to update a statistic. The updated statistic may be used either to make a decision if a certain threshold is reached or to collect another sample.

The use of SPRT is, thus, very attractive in many scenarios, as in Wireless Sensor Networks (WSN), in which the ability to make a decision requiring fewer communications among sensors means a lower battery and bandwidth consumption. SPRT also allows making a decision as soon as possible and adapts easily to working with online data. However, SPRT assumes that the underlying distribution of the signal under test does not change with time. Since SPRT is widely used to detect malicious behavior, this means that the malicious behavior is assumed to be static. In this work, we show that this is a dangerous assumption that can be used to exploit such systems, and also, we propose a defense mechanism that can be used to detect attacks which need not be static with time.

### 1.1. Prior work

As mentioned before, the properties of SPRT make it ideal for WSN applications, thus, it is no surprise that many WSN mechanisms make use of it. It has been used for cooperative spectrum sensing, in which several sensors send to a central entity their local spectrum sensing report and SPRT mechanisms are used to do the information fusion [6], [7], [8]. In [9], [10] and [11], SPRT is used for detecting sensors that have been compromised and replicated. In [12] and [13], SPRT is used to detect a selective forwarding attack, in which a compromised sensor drops packets. SPRT can also be used for DDoS attack detection [14] and spam detection [15]. Thus, SPRT finds many applications currently in WSN, specially when trying to detect a malicious behavior.

As we have mentioned, SPRT is valid only if the malicious behavior is static, and hence, it would be possible to exploit an SPRT based defense mechanism by means of a dynamic behavior attack. However, in many works this limitation of SPRT is not taken into account. For instance, in [9], [7], [11], [12] or [14], SPRT is used in environments in which the attackers may use a dynamic attack strategy and hence, compromise the defense mechanism.

To the best of our knowledge, the only study on such dynamic attacks on WSN is [16], where a cooperative spectrum sensing defense mechanism is exploited using reinforcement learning tools. While that work is focused on a concrete defense mechanism, ours focuses on the SPRT method, which is part of many defense mechanisms. If such attackers became frequent, the defense mechanisms would have to evolve in order to defend against such threats. In current literature, there are several tools designed for dealing with changes in the statistical behavior of a signal, such as quickest detection tools [17] or, for discrete time signals, repeated hypothesis tests [18].

In this work, we make use of a well-known tool in the field of novelty detection as the One Class Supporting Vector Machine (OCSVM)[19]. This method allows detecting signals whose features differ from the ones with which the OCSVM was trained [20]. In a misbehavior setting as ours, standard SVM needs to have access both to examples of normal and malicious behavior, as in [21], and hence, it becomes specialized in detecting a single attack type. However, the main advantage of OCSVM is that they need only have access to normal behavior examples in order to be trained, which in our security setting means that they could potentially detect any type of misbehavior.

## *1.2. Contributions*

The main contributions of our work are:

1. We obtain the optimal behavior that an attacker should follow to exploit a Bernoulli SPRT. We choose the Bernoulli distribution because it appears in many WSN defense mechanisms, such as [15], [9], [7], [11], [12] or [14].
2. We develop OCSVM-SPRT: a modification on the SPRT mechanism that makes use of an OCSVM that is able to deal with the novel optimal attacker we propose. Since an OCSVM is used, OCSVM-SPRT may potentially deal with any attacker whose spectral features do not match the ones of normal sensors.

The rest of the paper goes as follows: in Section 2, we describe the SPRT defense mechanism that we use throughout our work. Then, in Section 3, we describe the attacker we propose. Section 4 obtains the optimal control law that such an attacker should use against the SPRT mechanism described in Section 2. After, Section 5 introduces and explains OCSVM-SPRT. Section 6 studies the performance of OCSVM as a function of its parameters, and then, Section 7 presents a study case in which we analyze the performance of OCSVM-SPRT in a cooperative spectrum sensing setup. Finally, Section 8 presents the advantages and disadvantages of our approach, as well as draws some conclusions.

## 2. SPRT detection mechanism

We describe an SPRT based detection mechanism which is similar to the one present in many current literature approaches, such as [15], [9], [7], [11], [12], [14] or [10]. Even though concrete details differ between these works, the main lines of the detection mechanisms are similar to the mechanism that we introduce in this section. Also, note that though we only provide results for this model, we strongly believe that they could be extended to different signal characterizations (such as the one in [6]).

### *2.1. The detection problem*

We assume a discrete time signal $x_n$, where $n = 0, 1, 2, ...$ is the time index. The detection problem consists in collecting enough information in order to decide between two hypotheses $H_0$ and $H_1$:

$$\begin{cases} H_0 : x_n \sim Q_0, & n = 0, 1, 2, ..., \\ H_1 : x_n \sim Q_1, & n = 0, 1, 2, ..., \end{cases} \tag{1}$$

where $Q_0$ and $Q_1$ are statistical distributions that characterize the behavior of the signal $x_n$ under normal and malicious behavior respectively. In our case, we assume that each $x_n$ follows a Bernoulli distribution of parameters $\theta_0$ under $H_0$ and $\theta_1$ for $H_1$. The concrete malicious behavior varies depending on the setting, for instance:

- In [10], the authors describe an attack in which sensors could have been compromised. A compromised sensor would send a piece of information $x$ which can be accurate ($x = 0$) or inaccurate ($x = 1$). They try to detect as soon as possible inaccurate sensors, and note that attacking sensors try to provide as much inaccurate information to the network as possible.

- In [11], the authors also try to make a difference between legitimate and illegitimate sensors in a network. In order to do so, they define a binary variable $x$ that combines information of distance and signal power. Under this setting, an attacker tries not to be discovered while trying to compromise other sensors, which means that they will often cause $x = 1$.

- In [12], the authors try to detect a selective forwarding attack, in which $x = 0$ denotes a successful forwarding and $x = 1$ a packet drop. Note that an attacker tries to drop as many packets as possible.

- In [15], the authors propose a spammer detection algorithm, in which $x$ denotes whether the emails sent by a user are spam ($x = 1$) or not ($x = 0$). A user which surpasses a certain threshold is blocked, hence, a spammer will try to send as much spam as possible while also trying not to be detected.

In all these cases, the condition $\theta_0 < \theta_1$ is satisfied. Thus, $x_n \in \{0, 1\}$, and $P(x_n = 1) = \theta$ and $P(x_n = 0) = 1 - \theta$. In other words, the malicious behavior causes $x = 1$ to happen as often as possible.

In an HT, there are two different errors: the type I error or false alarm probability is the probability $H_0$ is rejected, provided that $H_0$ is actually true. The type II error is the probability of accepting $H_0$, provided that $H_1$ is actually true. We denote by $\alpha$ to the type I error probability and $\beta$ to the type II error probability. The values of $\alpha$ and $\beta$ determine the stopping rule that is used in the test. There is always a tradeoff between having a low false alarm probability and a high power test, which is defined as $1 - \beta$ and is the probability of correctly rejecting the null hypothesis.

## *2.2. SPRT*

The SPRT for our signal model presents the following test statistic for the sample $n$:

$$LR_n = \frac{\theta_1^{s_n}(1-\theta_1)^{n+1-s_n}}{\theta_0^{s_n}(1-\theta_0)^{n+1-s_n}} \tag{2}$$

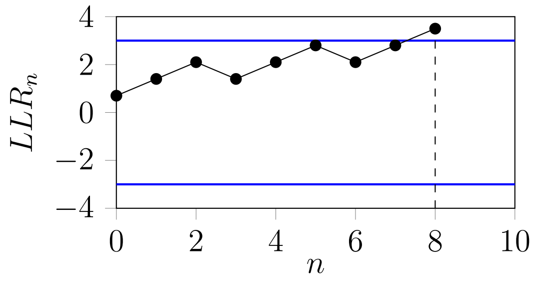


Figure 1: Illustration of an SPRT. The upper blue line is $h$, the lower blue line is $l$. The black continuous line is the $LLR_n$, the test statistic of the SPRT. The dashed line indicates $N-1$, the time in which a decision is made by the SPRT. In this case, since $LLR_n \geq h$, $H_0$ the test decision is to reject $H_0$. Note that in samples $n \leq 7$, SPRT does not have enough information to make a decision and hence, another sample is collected.

where $s_n = \sum_{i=0}^{n} x_i$, $s_n \in [0, n+1]$ and $n \in \{0, 1, ...\}$. It is usual working with the log-likelihood ratio: the SPRT from (2) becomes:

$$LLR_n = s_n \log\left(\frac{\theta_1}{\theta_0}\right) + (n+1-s_n)\log\left(\frac{1-\theta_1}{1-\theta_0}\right) \tag{3}$$

and the decision rules of test (3) can be approximated [5] as:

$$\begin{cases} \text{Reject } H_0 & \text{if} \quad LLR_n \geq h \\ \text{Accept } H_0 & \text{if} \quad LLR_n \leq l \\ \text{Take another sample} & \text{if} \quad \text{otherwise} \end{cases} \tag{4}$$

where $h$ and $l$ are defined as:

$$h = \log\left(\frac{1-\beta}{\alpha}\right), \quad l = \log\left(\frac{\beta}{1-\alpha}\right) \tag{5}$$

Note that (4) means that the SPRT procedure gathers new samples until a certain threshold in the statistic $LLR_n$ is surpassed. An illustration is found in Figure 1: note that $LLR_n$ produces a random walk: the SPRT is finished when it surpasses a certain threshold. Also, note that we can rewrite (3) as:

$$LLR_n = s_n \log\left(\frac{\theta_1(1-\theta_0))}{\theta_0(1-\theta_1)}\right) + (n+1)\log\left(\frac{1-\theta_1}{1-\theta_0}\right) \tag{6}$$

and then define:

$$A = \log\left(\frac{\theta_1(1-\theta_0))}{\theta_0(1-\theta_1)}\right), \quad B = \log\left(\frac{1-\theta_1}{1-\theta_0}\right) \tag{7}$$

we can rewrite (6) as:

$$LLR_n = As_n + B(n+1) \tag{8}$$

Also, a sequential formulation for (3) can be obtained by noting that:

$$LLR_n = \begin{cases} LLR_{n-1} + A + B & \text{if} \quad x_n = 1 \\ LLR_{n-1} + B & \text{if} \quad x_n = 0 \end{cases} \tag{9}$$

or even more compactly as:

$$LLR_n = LLR_{n-1} + B + Ax_n, \quad LLR_0 = B \tag{10}$$

Note that (10) allows an easy and sequential implementation of the SPRT test that we have defined, which updates the $LLR_n$ statistic using only simple operations. In case that $x_n = 1$ is received, the $LLR_n$ adds up $A + B$, and in case that $x_n = 0$, only $B$ is added.

## 3. Attacker model

In order to model our attacker, we assume that there is a malicious agent that can modify the signal $x_n$, either directly or indirectly. For our purposes in this work, we assume that the agent is able to directly modify $x_n$, and hence, treat $x$ as the stream of actions of the agent. In our model, we have assumed that a malicious behavior means that $x_n = 1$ as often as possible. Thus, we assume that the agent receives an instantaneous reward of $+1$ each time that $x_n = 1$. This reward scheme causes that the agent tries to increase the mean value of the signal $x$ and thus, $\theta > \theta_0$. This is consistent with using $H_1 : \theta_1 > \theta_0$.

The agent tries to maximize its total cumulative reward $R$, defined as:

$$R(x_n) = \sum_{n=0}^{\infty} \delta^n x_n \tag{11}$$

where $\delta \in (0,1)$ is a discount factor that gives more weight to the rewards obtained in closer time steps than in the future. The use of $\delta$ is very sensible in volatile environments such as wireless networks: an attacker does not know how long it will be able to attack and thus, it cannot be infinitely patient for attacking.

Also, $\delta$ allows that the total reward remains finite: note that the minimum value for $R$ using (11) is $R = 0$ for $x_n = 0, \quad \forall n$, and the maximum value is $R = (1-\delta)^{-1}$, for $x_n = 1, \quad \forall n$, where we used:

$$\sum_{k=a}^{b} \delta^k = \frac{\delta^a - \delta^{b+1}}{1-\delta}, \quad \delta \neq 1 \tag{12}$$

## 4. Optimal camouflage algorithms

Now, we proceed to show the optimal control that the agent should use in order to maximize its total cumulative reward when facing an SPRT detection mechanism. We denote by $N$ the number of timesteps required by the SPRT to make a decision. In practical implementations, $N$ is usually bounded to avoid system lockouts, although this truncation is suboptimal [22]. That is, in a truncated SPRT, there is a maximum number of samples that the test will gather before making a decision: $N-1$. A predefined decision is fixed beforehand in case that sample $N-1$ is reached without having made a decision and $l < LLR_{N-1} < h$. We consider that if the truncated SPRT test reaches sample $N-1$ without making a decision, $H_0$ is rejected.

The problem that the agent must solve is the following one, where we use (12) and consider $N-1$ as the time in which the SPRT makes a decision:

$$\begin{aligned}
\max_{x_n} \quad & \sum_{n=0}^{\infty} \delta^n x_n = \sum_{n=0}^{N-1} \delta^n x_n + \frac{\delta^N}{1-\delta} \\
\text{s.t.} \quad & x_n \in \{0,1\}, \quad s_n = \sum_{n=0}^{n} x_n \\
& LLR_n < h, \quad \forall\, n \leq N-1 \\
& LLR_{N-1} \leq l < h
\end{aligned} \tag{13}$$

Note that in (13):

- The function that the agent needs to maximize is split in two terms. The first term refers to the timesteps in which the SPRT detection mechanism is active, in which the agent needs to find an optimal control law for $x_n$ such that it is not discovered. The second term includes the timesteps after a decision is made: at these timesteps the SPRT mechanism is not active, and hence, the agent can always use $x_n = 1$.

- The constrain $LLR_n < h$ allows the agent not to be discovered by the SPRT mechanism. It forces SPRT to never reject $H_0$.
- The constrain $LLR_{N-1}$ simply indicates that, at timestep $N-1$, in which a decision is made, $H_0$ is accepted (i.e., the attacker has not been discovered yet).
- An assumption that we make is that $l < LLR_0 < h$, that is, that the $LLR$ initial value does not allow SPRT to make a decision. This condition is usually satisfied by normal SPRT parameters and in our case, using (7) and (8) turns out to be:

$$\frac{\beta}{1-\alpha} < \frac{1-\theta_1}{1-\theta_0} < \frac{1-\beta}{\alpha} \tag{14}$$

  Note that this means also that $l < h$ in the last constrain.

In other words, the agent must be able to attack and camouflage while the SPRT mechanism is running. When the SPRT mechanism is not running (i.e., $n \geq N$), the agent can attack without needing to camouflage. Note that in truncated SPRT, $N$ is fixed, while in SPRT, $N$ is not fixed and hence, the actions of the agent will determine the final time $N$, which means that $N$ is a value to optimize.

### *4.1. Optimal control for truncated SPRT*

First, we proceed to obtain the optimal control for the problem (13), assuming that there is a truncated SPRT, which means that $N-1$ is fixed. First, note that the agent prefers using $x_n = 1$ as often as possible because that way, its reward is maximized. Also, note that since rewards are discounted, if the agent has to choose between using $x = 1$ at timestep $n$ or at timestep $m > n$, the agent always prefer the former because it provides a larger reward due to the discount factor: $\delta^n \cdot 1 > \delta^m \cdot 1$ for $\delta \in (0, 1)$. Intuitively, this means that the agent will try to use $x = 1$ wherever possible.

However, the agent may not always use $x_n = 1$ for all $n$. Using (10), the agent can predict its $LLR_n$ value depending on its action. Since we considered that $\theta_1 > \theta_0$, then we obtain from (7) that $B < 0$ and $A + B \geq 0$. This means that:

- If the agent uses $x_n = 0$, $LLR_n = LLR_{n-1} + B < LLR_{n-1}$. In other words, the $LLR_n$ value decreases by using $x_n = 0$.

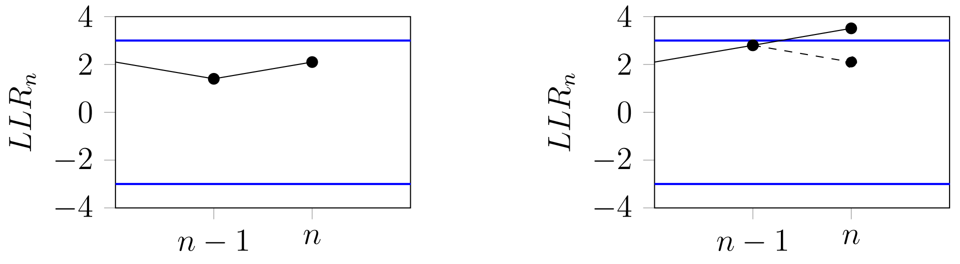


Figure 2: Illustration of the constrain that $LLR_n < h$ in problem (13), where $h$ is the upper blue line and the black lines represent $LLR_n$. In both plots, we show what would happen if the agent used $x_n = 1$. In the left plot, $LLR_n = LLR_{n-1} + A + B < h$ and hence, the agent could play $x_n = 1$. However, in the right plot, $LLR_n = LLR_{n-1} + A + B > h$ (solid line) and if the agent played $x_n = 1$, $H_0$ would be rejected and the agent would be discovered. Instead, the agent should use $X_n = 0$, which would decrease the $LLR_n$ value (dashed line).

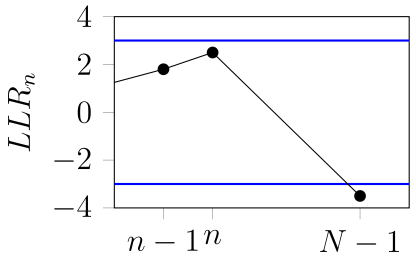
- If the agent uses $x_n = 1$, $LLR_n = LLR_{n-1} + B + A \geq LLR_{n-1}$. In other words, the $LLR_n$ value is non-decreasing by using $x_n = 1$.

However, there are two constrains in problem (13) that may prevent the agent from using $x_n = 1$. The first is that $LLR_n < h$. As we just noted, by using $x_n = 1$ the agent may increase its $LLR_n$ and hence, it may, eventually, violate that constrain. In order to avoid that, the agent can play $x_n = 1$ if $LLR_{n-1} + A + B < h$. This is illustrated in Figure 2.

Another constrain in problem (13) that may prevent the agent from using $x_n = 1$ is that $LLR_{N-1} \leq l$. We assume that $l < h$, which is satisfied if $\alpha + \beta < 1$, which is our case. Note that the agent can only decrease $LLR_n$ by using $x_n = 0$, thus, in order to satisfy $LLR_{N-1} \leq l$, the agent will have to play $x = 0$ sometimes. As we noted before, the agent prefers using $x = 1$ as often as possible, and hence, it will delay using $x = 0$ to satisfy $LLR_{N-1} < l$ as many timesteps as possible. Namely, if the agent is at timestep $n$, he could play $x_n = 1$ and then, play $x = 1$ at timesteps $[n+1, N-1]$ and satisfy $LLR_{N-1} \leq l$ if $LLR_{n-1} + A + B(N-n) \leq l$. This is illustrated in Figure 3, and intuitively tries to delay using $x = 0$ as many timesteps as possible in order to satisfy the constrain on $l$.

Thus, the optimal control for the agent is:

$$\begin{cases} x_n = 1 & \text{if} \quad LLR_{n-1} + A + B < h \text{ and } LLR_{n-1} + A + B(N-n) \leq l \\ x_n = 0 & \text{if} \quad \text{otherwise} \end{cases} \tag{15}$$



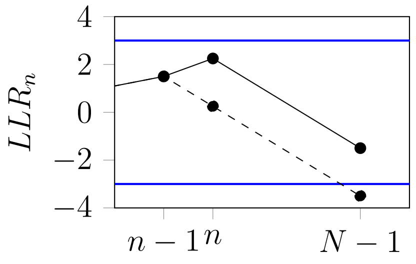


Figure 3: Illustration of the constrain that $LLR_{N-1} \leq l$ in problem (13), where $l$ is the lower blue line and the black lines represent $LLR_n$. In both plots, the solid black lines indicate the evolution of the $LLR_n$ if the agent used $x_n = 1$ and then $x = 0$ for $n \in [n+1, N-1]$. In the left plot case, the agent satisfies that $LLR_{N-1} \leq l$, thus, it can use $x_n = 1$. However, in the right plot, the agent does not satisfy $LLR_{N-1} \leq l$ if $x_n = 1$, and hence, the agent would have to use $x_n = 0$ to satisfy the constrain (dashed line).

### *4.2. Optimal control for non-truncated SPRT*

In case that the SPRT is not truncated, note that the actions of the agent determine $N-1$, the time in which the SPRT makes a decision. In the previous Section, we showed that for fixed $N-1$, the agent would need to use $x = 0$ at several timesteps. As we noted, the agent would rather use $x = 1$, and this would imply that $N-1 \to \infty$. In other words, the agent would cause the SPRT to never reach a decision, and its optimal control would be (see (15)):

$$\begin{cases} x_n = 1 & \text{if} \quad LLR_{n-1} + A + B < h \\ x_n = 0 & \text{if} \quad \text{otherwise} \end{cases} \tag{16}$$

### *4.3. Optimal control for the attacker*

Now, we proceed to obtain Lemma 1, which sums up the optimal control obtained in the previous two Sections.

**Lemma 1.** *Consider the following discrete time control problem, in which the controller chooses $x_n$ and may choose $N-1$:*

$$\max_{x_n} \sum_{n=0}^{N-1} \delta^n x_n + \frac{\delta^N}{1-\delta}$$

$$\begin{aligned} s.t. \qquad & x_n \in \{0,1\}, \quad s_n = \sum_{n=0}^{n} x_n \\ & LLR_n < h, \quad n \leq N-1 \\ & LLR_{N-1} \leq l < h \end{aligned}$$

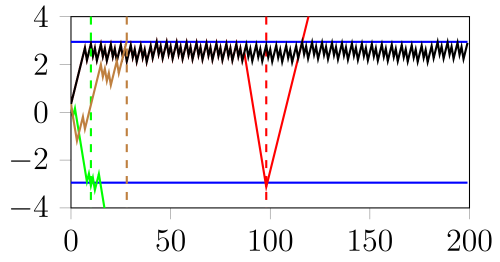


Figure 4: Example of control under several situations. For all cases, $\theta_0 = 0.5$ and $\theta_1 = 0.7$. The blue lines are the $LLR_n$ thresholds from (4), for $\alpha = \beta = 0.05$. Green line is the case in which there is no attack (i.e., $x \sim Bernoulli(\theta_0)$). Brown line is the case in which there is a simple attack (i.e., $x \sim Bernoulli(\theta_1)$). Red line is the case in which the attacker follows the control law from Lemma 1 when the SPRT test finishes after 100 samples. Black line is the case in which the attacker follows the control law from Lemma 1 when the SPRT does not have a predefined finishing time. The dashed vertical lines indicate when each test ends. While SPRT is able to detect the simple attack, is unable to detect the control law we describe in Lemma 1, independently on whether the SPRT test is truncated or not.

*The discount factor is $\delta \in (0, 1)$. The constrain has the form $LLR_n = As_n + B(n + 1)$ and satisfies:*

- *$B < 0$ and $A + B \geq 0$*
- *$h < LLR_0 < l$*

*In this problem, the optimal control for $n \in [0, N-1]$ depends on whether $N - 1$ is fixed or not:*

- *If $N - 1$ is fixed:*

$$\begin{cases} x_n = 1 & \textit{if} \quad LLR_{n-1} + A + B < h \textit{ and } LLR_{n-1} + A + B(N-n) \leq l \\ x_n = 0 & \textit{if} \quad \textit{otherwise} \end{cases}$$

- *If $N - 1$ is not fixed:*

$$\begin{cases} x_n = 1 & \textit{if} \quad LLR_{n-1} + A + B < h \\ x_n = 0 & \textit{if} \quad \textit{otherwise} \end{cases}$$

Let us visualize the result of the control law obtained in Lemma 1. Note that the basic idea is that the agent is able to get close to the SPRT threshold without surpassing it. An illustration can be found in Figure 4, in which we can observe how the control law proposed is able to effectively attack the SPRT mechanism while not being detected. More complete results are shown in Section 6, however, by now, note that the control law proposed in Lemma 1 can successfully overcome an SPRT based defense mechanism.

## 5. Defense mechanism

### *5.1. Parallel SPRT*

As we explained in the introduction, SPRT is not a good option when the attacker may change its behavior. A solution proposed to face this problem is given in [18], where several simultaneous SPRT tests are run in parallel, in order to detect changes in the signal $x$. This would imply that, for each new sample $x_n$ that arrives to the defense mechanism, a new SPRT test is initiated, and $n$ SPRT tests are updated. Note that this is a computationally demanding mechanism, since many SPRT tests must be run in parallel.

Also, note that an approach like this would not detect an attacker following the control law from Lemma 1. As we show in Figure 4, the attacker is as close as possible to the detection threshold without surpassing it. A second SPRT test would simply be a downshift in the $LLR_n$ curve, pushing it down and hence, making impossible that the second SPRT test detects the attacker. This reasoning extends to subsequent SPRT tests, which are unable to detect the attacker. Note, however, that this applies only to non-truncated SPRT: if truncation is applied, the agent could be detected after the last SPRT sample if the agent is unaware that there are several SPRT tests running. Another way in which an agent could exploit such mechanism would be simply initiating a control law for each sample $n$ and choosing the most restrictive one. Hence, a parallel SPRT is not only computationally expensive, but also is unable to detect adequately the attacker we have developed.

### *5.2. One-class SVM*

We propose using another well-known tool in the field of sequence classification in order to modify the SPRT defense mechanism: a One-Class Supporting Vector Machine (OCSVM) [19]. As described in [20], a OCSVM is an algorithm that maps an input vector $z$ according to whether $z$ belongs to a set $Z$ or not as follows:

$$f(z) = \begin{cases} +1 & \text{if} \quad z \in Z \\ -1 & \text{if} \quad z \notin Z \end{cases} \tag{17}$$

The algorithm takes a set of $z \in Z$ points, and then obtains $f$ by solving the following optimization problem:

$$\begin{aligned} \min_{w,\xi,\rho} \quad & \frac{1}{2}\|w\|^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho \\ & s.t.(w \cdot \Phi(z_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0 \end{aligned} \tag{18}$$

where $\Phi(z)$ is the feature map obtained by using a certain kernel, $i$ indexes the training inputs $z_i$, $l$ is the total number of training inputs, $\xi$ are the slack variables and $\nu \in (0,1)$ is a parameter that corresponds to the fraction of outliers in the input data set $Z$ [23]. Thus, note that a OCSVM needs only a training data set of valid data, in order to provide a decision function that later can be used for anomaly detection.

### *5.3. SPRT - OCSVM defense mechanism*

We make use of the capabilities of the OCSVM to propose a modified SPRT defense mechanism. First, we need to define which are the features $z$ that we want to use. Note that these features need to characterize a statistical signal, and a possible way of characterizing such signals is by using the power spectrum of the signal [24]. We use as feature the power spectrum of the $LLR_n$ signal, which is a random walk. We denote by $v$ a subsequence of $LLR_n$, and in order to avoid errors caused by the mean value of the signal we subtract the mean of $v$ (i.e., note that the $LLR_n$ signal does not have a constant mean). Mathematically, we propose using the following feature vector $z$:

$$z(v) = \left| FT\left( AC\left( v - \frac{\sum_{m=1}^{M} v_m}{M} \right)\right)\right| \tag{19}$$

where $FT$ denotes the Fourier Transform, $AC$ is the estimator of the aucotorrelation function, and $v$ is a vector formed by the $M$ most recent values of $LLR_n$. We use the full autocorrelation, hence, $z$ has a length $2M-1$. Hence, $z$ is the estimated Power Spectrum of $v$ with the mean subtracted. As we said, this choice of $z$ is sensible given the fact that $LLR_n$ is a random signal, and as we show in Section 6, the results provided by this characterization are quite good against the attack we propose. We note that the OCSVM training can be done offline by generating sequences of $x_n \sim Bernoulli(\theta_0)$, then obtaining the $LLR_n$ vector $v$ by using (3) and then obtaining $z$ by using (19). That way, we train the OCSVM to detect any sequence not generated

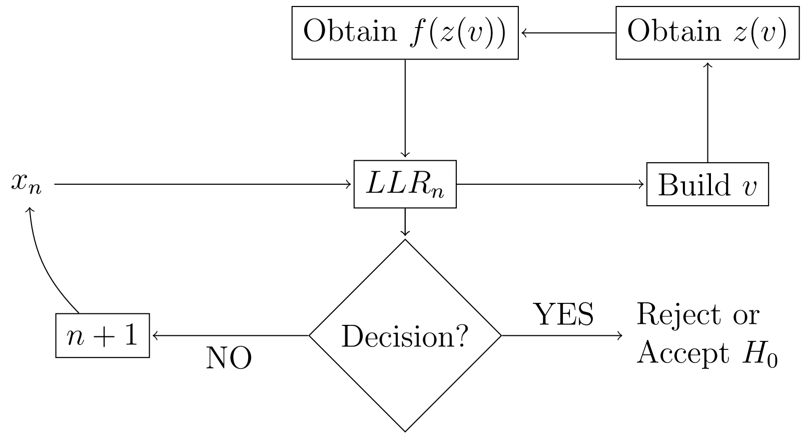


Figure 5: Flow diagram for the proposed OCSVM-SPRT defense mechanism, where the $LLR_n$ block implements (20).

by following a Bernoulli of parameter $\theta_0$. This means that the OCSVM will be able to detect, not only the attacker we propose, but any with a different spectral characterization from the signal that the OCSVM has been trained with.

Now, we need to decide how to include the additional information that the OCSVM provides. We propose using a modified SPRT with the following statistic $LLR'_n$:

$$LLR'_n = LLR'_{n-1} + B + Ax_n + \gamma(A+B)\left|\min\left(f\left(z\left(v\right)\right),0\right)\right|, \quad LLR'_0 = B \tag{20}$$

where $LLR'_{n-1}$ is the previous value of the test statistic, $B + Ax_n$ is the standard $LLR_n$ update for the SPRT as shown in (10), $\gamma$ is a small positive parameter that control how much we make use of the information given by the OCSVM, and $f$ is the OCSVM as defined in (17) using (19). Note that we include an additional term which depends on $A+B$, the increase on the $LLR_n$ value when $x_n = 1$ (10). We do so because the value of $A+B$ depends on $\theta_0$ and $\theta_1$ (7), and thus, we achieve that the last term in (20) depends on the concrete test parameters by being relative to the increase when $x_n = 1$. A flowchart illustrating the whole process can be seen in Figure 5.

Intuitively, the modified SPRT test in (20) works as follows. We first obtain the standard $LLR_n$ value, and then, we use a OCSVM to obtain a second opinion on whether the $LLR_n$ subsequence $v$ has been generated by a normal sensor or an attacker. If the OCSVM detects a sequence that

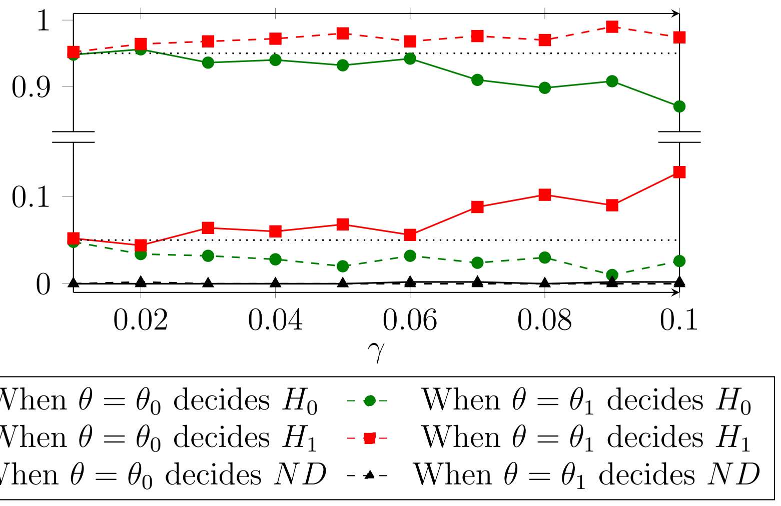


Figure 6: Example on the influence of $\gamma$ on the modified SPRT-OCSVM scheme proposed, for $\theta_0 = 0.5$ and $\theta_1 = 0.7$. The details on the OCSVM are in Section 6. For the SPRT, we consider $\alpha = \beta = 0.05$, without truncation and finishing the test after 200 SPRT iterations. The dotted lines represent the type I and II error of the SPRT without modification: note that our modified test performs gives an increasing performance under $H_1$ as $\gamma$ grows, while its performance under $H_0$ decreases as $\gamma$ grows. This is to be expected: the OCSVM modification helps to detect a deviation from $H_0$, however, under $H_0$ the OCSVM modification introduces an additional error since it increases the $LLR_n$ value.

differs from these it has been trained to detect, it returns $f = -1$. In that case, the $LLR_n$ is increased by a $\gamma(A + B)$ value, which is proportional to the increase in the $LLR_n$ signal when $x_n = 1$. If the OCSVM detects a normal sequence, then no modification is done. Note that the effect of the OCSVM is cumulative, and hence, our modified SPRT test detects faster an attacker if the OCSVM continuously says so. On the other hand, note that if the OCSVM states erroneously that a sequence comes from an attacker, the probability of type I error is increased compared to the standard SPRT test. This can be observed in Figure 6, in which we can observe how the error of the modified SPRT test increases with $\gamma$ under $H_0$, but decreases under $H_1$. In short, what we have done is obtaining a more precise test under $H_1$, which however, gives a higher error under $H_0$.

| | SPRT | SPRT-OCSVM |
|---|---|---|
| NA | 97/3/0 | $93,2/6.8/0$ |
| SA | 3.6/96.4/0 | 2/98/0 |
| OWT | 100/0/0 | 0/100/0 |
| ONT | 0/0/100 | 0/100/0 |

Table 1: Test results for $\theta_0 = 0.5$ and $\gamma = 0.05$, for all the tests simulated. Each table entry is the percentage of times that $H_0$ was decided / $H_0$ was rejected / no decision was taken. Observe how when facing the control law from Lemma 1, SPRT is totally unable to detect the attacker. However, the exact opposite happens with our proposed SPRT-OCSVM mechanism: it always detects such an attacker.

## 6. Empirical influence of the OCSVM-SPRT parameters

We proceed to study the influence of the parameters of OCSVM-SPRT using simulations. For all the SPRT tests, we choose $\alpha = \beta = 0.05$; note that these values satisfy the condition (14). We then take 10 values of $\theta_0$ equispaced in the range $\theta_0 \in [0.1, 0.7]$, and for each $\theta_0$ value, we define $\theta_1 = \theta_0 + 0.2$, that is, we use as $\theta_1$ another 10 equispaced values in the range $\theta_1 \in [0.3, 0.9]$.

For each $\theta_0$ value, we train an OCVSVM using 500 different $z$ vectors generated from a Bernoulli distribution with $\theta = \theta_0$, using $\nu = 0.1$ and a Gaussian Kernel; each $z$ vector has a length $M = 5$ and hence each $z$ has a length of 9 samples. We then obtain the validation error using another 500 $z$ vectors. We train 5 different OCSVM for each test and $\theta_0$ value, using the $OCSVM$ that provides the lower value for the sum of the training and validation error.

For each pair of $\theta_0$ and $\theta_1$ value, we average the results for 500 runs of each test, where all the tests are finished after 200 iterations - note that the test might not have decided by that time. We test for four different signal possibilities: (1) a situation of No Attack (NA), in which the signal $x \sim Bernoulli(\theta_0)$, (2) a Simple Attack (SA) situation, in which the signal $x \sim Bernoulli(\theta_1)$, (3) an attack situation in which the attacker uses the control law from Lemma 1 with $N-1 = 100$ (i.e., the test is truncated, Optimal With Truncation, OWT), and (4) an attack situation in which the attacker uses the same control law, without truncation (Optimal No Truncation, ONT). Each of these different situations are faced to an SPRT defense mechanism that uses (10) and also, to our modified SPRT-OCSVM scheme, using $\gamma = 0.05$,

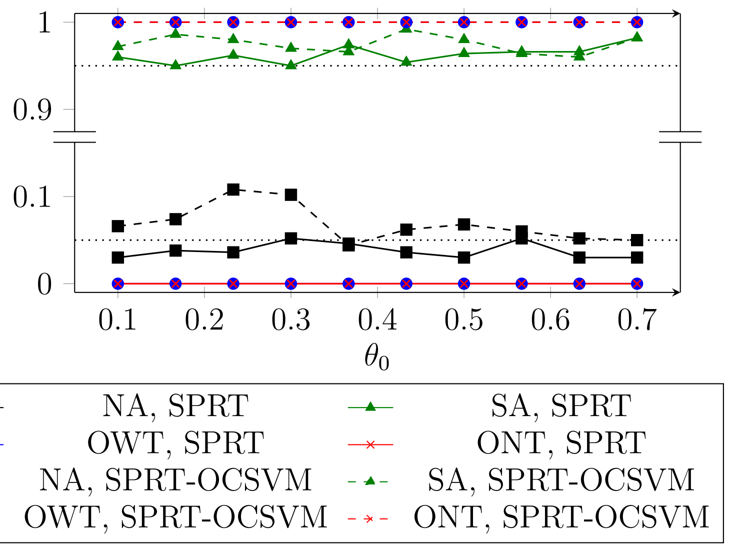


Figure 7: Proportion of $H_0$ rejections for the different schemes proposed as a function of $\theta_0$. The dotted lines correspond to the $\alpha$ and $1-\beta$ values of the tests. Note that under $H_0$ (i.e., NA), our proposed SPRT-OCSVM performs worse than SPRT, rejecting $H_0$ more often; and under $H_1$, SPRT-OCSVM works better than SPRT, as we advanced in Figure 6. However, note that the improvement in detecting an attacker following the control law from Lemma 1 is dramatic: while SPRT is never able to detect it, SPRT-OCSVM always detects the attacker.

because, as Figure 6 shows, it provides a nice tradeoff in the error under both $H_0$ and $H_1$.

The results can be observed in Table 1 and Figures 7, 8 and 9. First, in Table 1 we show the test results for $\theta_0 = 0.5$, and we note how the control law proposed in Lemma 1 allows that the attacker is never detected as such under an SPRT defense mechanism. Note that the attacker is able to either make that the SPRT test never reaches a decision if no truncation is done, or is able to be detected always as a normal agent if the SPRT is truncated. However, our proposed modification, SPRT-OCSVM, allows detecting such an attacker with high accuracy. As we advanced in the previous section, SPRT-OCSVM is able to perform better under attack by means of decreasing the test performance under $H_0$, i.e., when there is no attack. These results apply to all the tested values of $\theta_0$, as can be observed in Figure 7, where the proportion of times that each test rejects $H_0$ is represented as a function of $\theta_0$.

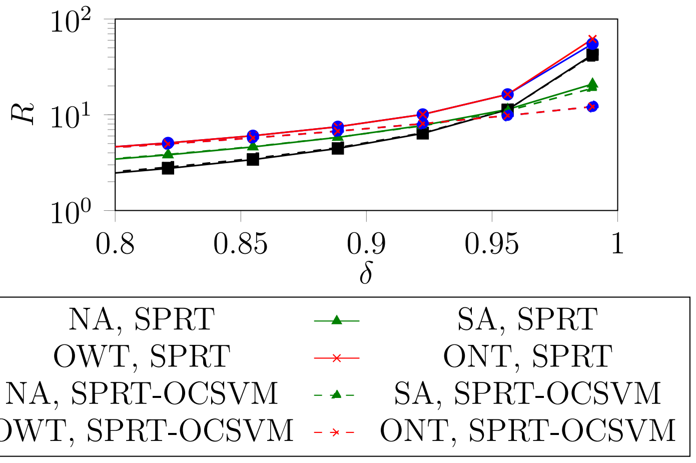


Figure 8: Detail on the total cumulative reward $R$ from (8) obtained for an attacker under the different schemes proposed. For low values of $\delta$, the use of SPRT or SPRT-OCSVM does not bring significant differences. However, as $\delta \to 1$, note how SPRT-OCSVM causes the attacker to obtain a lower reward than if he did not attack. While the attacker obtains an advantage in terms of $R$ against SPRT by using Lemma 1, this advantage vanishes when facing our proposed SPRT-OCSVM mechanism.

In terms of the total cumulative reward in (11), we can observe in Figure 8 that an attacker using the control law from Lemma 1 is able to obtain a better reward by following that control law if the defense mechanism is an SPRT. However, that control law is useless against our modified SRPT-OCSVM: note that as $\delta \to 1$ the agent receives no benefit in attacking, and it would receive a higher reward by behaving as a normal sensor. Note that this dependency on the value of $\delta$ comes from the fact that lower values of $\delta$ cause that the total reward strongly depends on the rewards at the first time steps. Since any detection method takes some time to make a decision, this problem cannot be easily solved. However, as $\delta$ approaches 1, the attacker puts a larger emphasis on future rewards and in this case, the ability to camouflage becomes crucial if the attacker wants to obtain a large reward.

Finally, in Figure 9 we can observe an example of the difference that our proposed approach has when compared to the standard SPRT. Note that if the OCSVM detects a signal that does not follow the expected spectral pattern, the modified $LLR'_n$ from (20) starts growing with respect to the SPRT $LLR_n$. Eventually, this means that the attacker is detected. Also, observe that the OCSVM brings a very small increase, which is controlled by the $\gamma$ parameter. As we noted before, a larger $\gamma$ brings a higher detection

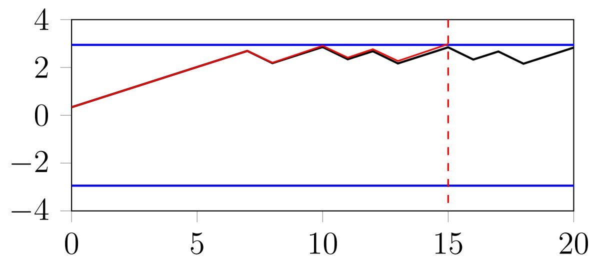


Figure 9: Example of detection for $\theta_0 = 0.5$ and $\theta_1 = 0.7$. The blue lines are the $LLR_n$ thresholds from (4). In both cases, we compare a realization of the control law from Lemma 1 without truncation, using SPRT (black) and SPRT-OCSVM (red). The dashed vertical line indicate when each test ends. Observe that, as in Figure 4, the SPRT is unable to detect the attack. However, SPRT-OCSVM is able to do so: when it believes that there is an attacker, it starts increasing slowly the $LLR_n$ value using (20). Note that this means that eventually, the attacker is detected.

under attack, because it increases the $LLR'_n$ faster, but that also means that the error increases under $H_0$.

## 7. Study case: Using OCSVM-SPRT in a Spectrum Sensing Data Falsification attack

In this section, we provide an example of how OCSVM-SPRT can be used as an additional security layer for existing defense mechanisms, using as study case a Spectrum Sensing Data Falsification (SSDF) attack. Consider a Wireless Sensor Network performing cooperative spectrum sensing: each sensor senses the spectrum locally and sends this data report to a centralized Fusion Center (FC), which uses a certain fusion rule to make a decision on whether the communications channel is busy or idle. This problem is of special interest in Cognitive Radio (CR) setups, in which each sensor would be a secondary node that would cooperate among them to transmit when there is no primary transmitting. These schemes are vulnerable to SSDF attacks, in which false reports are given by attacking sensors. A lot of effort has been addressed to design defense mechanisms against such attacks, as [7], [25], [26], [27], [28], [29], [30], [31], [32], [8] or [33].

### *7.0.1. Problem setup*

Let us assume a WSN with $M$ sensors, which wants to estimate the channel state in the slots $k = 1, 2, ..., K$. The actual state of the channel can be $y_k = 1$ if the primary is transmitting or $y_k = 0$ if the primary is idle. In each slot $k$, the FC asks several sensors $m$ for a report $u_{m,k}$, $m \in 1, 2, ..., M$, which may be $u_{m,k} = 0$ if sensor $m$ senses the channel idle or $u_{m,k} = 1$ if sensor $m$ detects a primary transmitting. When there are enough reports to make a decision, the FC uses a predefined fusion rule in order to obtain the channel state estimation $u_k$. Note that all the information that the FC receives from the sensors are the binary reports. The FC makes an error if $y_k \neq u_k$.

Each $u_{k,m}$ may differ from the actual $y_k$ due to errors in the sensing method or due to the presence of attacking sensors (ASs). We model the first case by assuming that each sensor has a probability of obtaining a wrong sensing result $P_e$, which is independent and equal for all sensors. Some possible attacks are always yes (AY), i.e., the AS always reports $u_{m,k} = 1$; always no (AN), i.e., the AS always reports $u_{m,k} = 0$ or always false (AF), i.e., the AS always reports the opposite of what it has sensed. Even though these are simple attacks, they often appear [7], [25], [8]. To these attack strategies, we add the control law from Lemma 1, which we name Intelligent Attack (IA).

There are several fusion rules that can be used by the FC to make a decision. A majority rule can be used, in which a maximum sample size $N_{mr}$ is fixed: the FC collects $N_{mr}$ reports and makes its decision by majority [7]. Note that the FC needs not having $N_{mr}$ reports to make a decision: whenever it has $(N_{mr} + 1)/2$ equal reports, the decision is made. Another popular fusion rule is based on SPRT in order to have a defense mechanism against SSDF attacks, such as SPRT and WSPRT [7], EWSPRT [25], RWSPRT [8] or S0/1 [33]. In our problem, we use the majority rule for its simplicity and EWSPRT (Enhanced Weighted SPRT), which makes use of a reputation scheme to give more weight to the reports of the sensors with good reputation and it also asks first sensors with higher reputations: this makes EWSPRT more advanced than SPRT and WSPRT [25]. We do not use RWSPRT and S1/0 because they require additional information from the transmissions to make a decision.

We use our OCSVM-SPRT algorithm to enhance the defense mechanism. For each sensor $m$, we run an instance of OCSVM-SPRT in order to detect sensors that are deviating from the expected behavior of a normal sensor. Thus, for each sensor $m$, we run an OCSVM-SPRT test where each $x_n$ is the $u_{k,m}$ for sensor $m$. Note that in this case, if the transmission probability of the primary is $P_{tr}$, the probability of receiving $u_{m,k} = 1$, which is used to

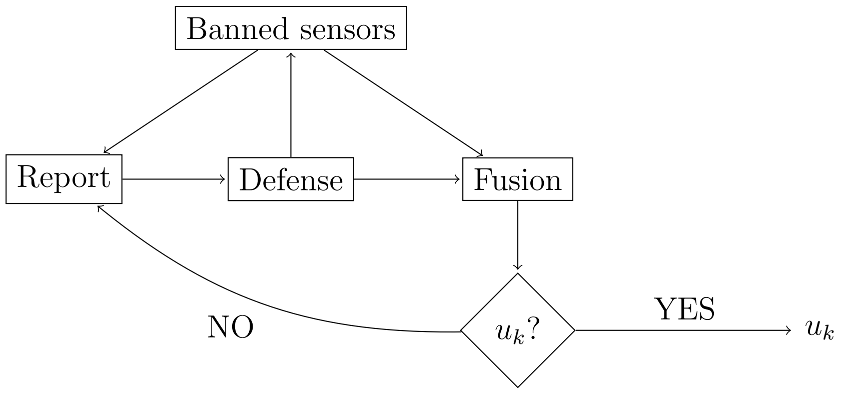


Figure 10: Flow diagram for each slot $k$ of the SSDF problem.

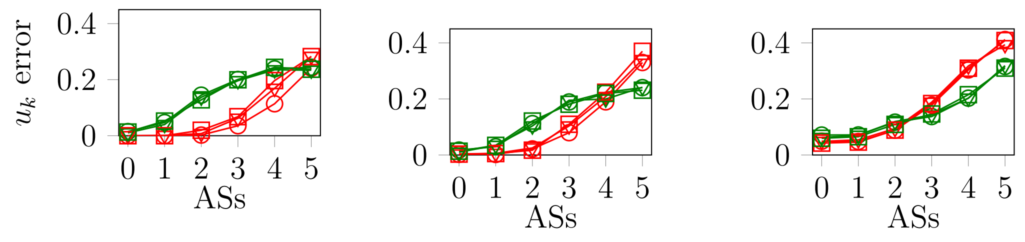
define the null hypothesis, is $\theta_0 = (1 - P_e) \cdot P_{tr} + P_e \cdot (1 - P_{tr})$. If sensor $m$ is detected as an attacker, the sensor is banned from the WSN and is not called again for reports.

The whole FC procedure can be seen in Figure 10. In each slot $k$, the FC asks for a report to a sensor $m$. This sensor $m$ is randomly selected among the sensors not banned if the majority rule is used, and under EWSPRT sensors are sorted by their reputations. Then, a defense mechanism is used to update the list of banned sensors, and if sensor $m$ has not been banned, its report is used to update the fusion rule. If more information is needed to make a decision, then the process starts over; otherwise, the decision $u_k$ is returned.

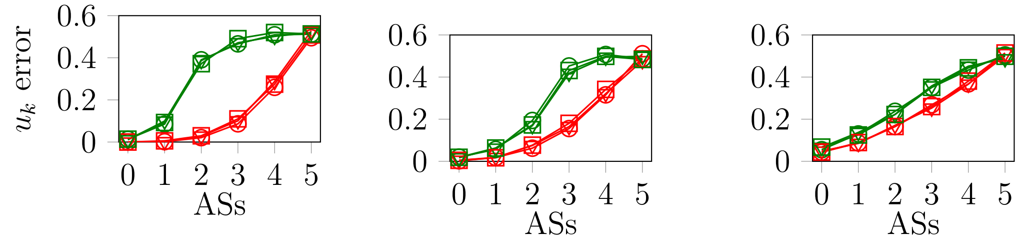
*7.0.2. Results*

We run the SSDF attack described in the previous section for a WSN with $M = 10$ sensors with $P_{tr} = 0.5$. We test for three possible defense mechanism: no defense mechanism, an SPRT and an OCSVM-SPRT mechanism. The two latter cases use the SPRT described in Section 2, with $\theta_0 = (1 - P_e) \cdot P_{tr} + P_e \cdot (1 - P_{tr})$, $\theta_1 = \theta_0 + 0.1$ and $\alpha = \beta = 0.05$. We use the same OCSVM that defined in Section 6 with $\gamma = 0.05$. Regarding the fusion rule parameters, we set $N_{mr} = 20$ for the majority rule, and for EWSPRT we use $\alpha = \beta = 0.05$ to define the sequential test thresholds. EWSPRT defines a maximum number of reports for system stability: after 20 reports, if no decision has been achieved, EWSPRT returns $u_k = 1$, which is a conservative decision that favors the primary [25].

We define a grid on the number of attackers and the sensing error com-



(a) FC average error for the Always No (AN) attack. From left to right: $P_e = \{0.1, 0.2, 0.3\}$. Lower is better.



(b) FC average error for the Always False (AF) attack. From left to right: $P_e = \{0.1, 0.2, 0.3\}$. Lower is better.

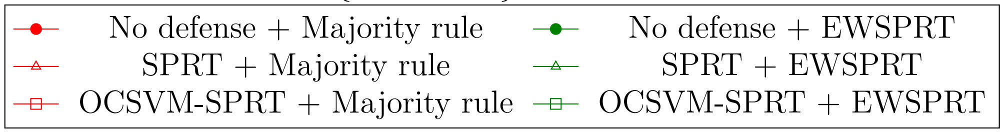
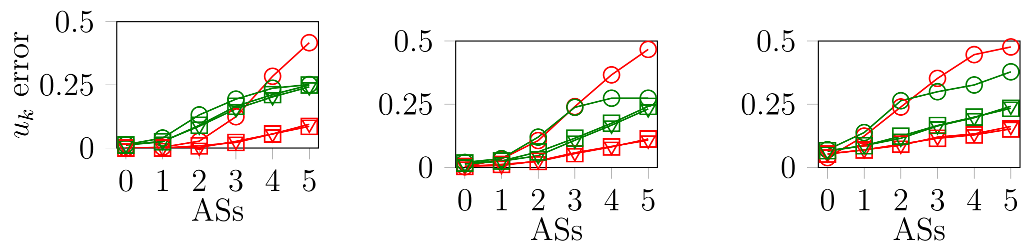
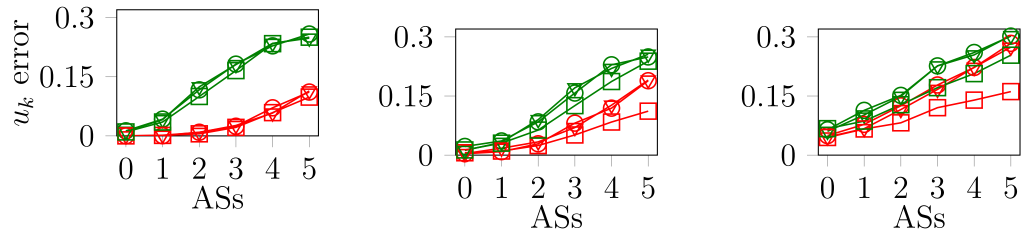


Figure 11: Results for the AN and AF attacks. Note that all attack strategies are successful, since the error increases with the number of ASs. In these two attacks, the choice of the defense mechanism does not make a significant different, but the fusion rule does: majority rule, in general, provides a lower error than EWSPRT.

bining $\{0, 1, 2, 3, 4, 5\}$ ASs and $P_e = \{0.1, 0.2, 0.3\}$. As attack strategies, we use AY, AN, AF and IA, and as defense mechanism, we use no defense mechanism, SPRT and OCSV-SPRT. For each combination of these parameters, the results are averaged over 50 realizations of each test, with $K = 50$. The results can be observed in Figures 11 and 12. All attacks are successful and increase their harm with the number of ASs: this is specially remarkable for the IA case, since the control law that the ASs follow was not derived to fool the fusion rules used. Note that using our proposed OCSVM-SPRT defense mechanism does not cause any negative impact in the decision error, except for AN attack with $P_e = 0.1$, and it does help against the IA, in which it provides the best results and helps to decrease the error for all the fusion rules tested.



(a) FC average error for the Always Yes (AY) attack. From left to right: $P_e = \{0.1, 0.2, 0.3\}$. Lower is better.



(b) FC average error for the Intelligent Attack (IA). From left to right: $P_e = \{0.1, 0.2, 0.3\}$. Lower is better.

| | | | |
|---|---|---|---|
| ● | No defense + Majority rule | ● | No defense + EWSPRT |
| △ | SPRT + Majority rule | △ | SPRT + EWSPRT |
| □ | OCSVM-SPRT + Majority rule | □ | OCSVM-SPRT + EWSPRT |

Figure 12: Results for the IA and AY attacks. Note that, again, all attack strategies are successful, since the error increases with the number of ASs. In these two attacks, the choice of the defense mechanism makes a difference: in the AY attack, when there are several ASs, having a defense mechanism decreases the error. In case of IA, note that as $P_e$ increases, OCSVM yields a lower error. Again, note that the majority rule consistently provides a lower error than EWSPRT.

It may be surprising that EWSPRT, in general, provides a higher error than the simpler majority rule. This is due to the cases in which EWSPRT did not have enough information to make a decision after 20 reports and the test was truncated. This is consistent with [33], which shows that the number of samples required by EWSPRT significantly increases with the number of ASs. Even though EWSPRT was not able to make a decision within 20 reports in some cases, in average EWSPRT always required fewer reports to make a decision that the majority rule. In some cases, also, the OCSVM-SPRT defense mechanism reduced the average number of reports required to make a decision using both fusion rules. Hence, OCSVM-SPRT can be used as an additional security layer by keeping track of the behavior of each sensor, and banning those who deviate from the expected behavior: we compromise computational power for an additional security.

## 8. Conclusions

In this work, we show that there are several detection problems in WSN which are based on SPRT tools which are vulnerable to intelligent attackers, because SPRT assumes distributions that do not change with time. In the first part of our work, we show one attacker that takes advantage of this fact to develop an optimal control law that renders it undetectable for a Bernoulli SPRT.

Then, in a second part, we have proposed OCSVM-SPRT, a modified SPRT that allows detecting such intelligent attackers. The idea is that an OCSVM is trained with data generated according to the distribution of normal sensors, and it then is able to detect behaviors that do not match normal sensors. We propose using spectral features of the $LLR_n$ signal, which proves to be very effective against the attack we had previously developed in our simulations.

The approach we propose has several limitations. It requires a training stage, it requires evaluating an OCSVM in each time step and it gives worse results than standard SPRT under $H_0$. But it also provides significant advantages: the very definition of OCSVM means that our proposed SPRT-OCSVM test is able to detect, not only our attacker, but potentially any attacker whose spectral pattern differs from normal sensors. It also requires a minimal modification to the standard SPRT, and the tradeoff between detecting an attacker and worsening the results under $H_0$ can be controlled with a $\gamma$ parameter.

We conclude that SPRT is vulnerable to an intelligent attacker. The control law we propose in Lemma 1 is simple to implement and hence, is a real threat to current SPRT based defense mechanisms. We also note that the attackers have a wide set of possible attack strategies today [16], hence, the study on defense mechanisms able to counter these attackers is vital for the security of sensor networks.

## 9. Acknowledgements

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).

## References

[1] S. M. Kay, Fundamentals of Statistical Signal Processing: Estimation Theory, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] J. Neyman, E. S. Pearson, Ix. on the problem of the most efficient tests of statistical hypotheses, Phil. Trans. R. Soc. Lond. A 231 (694-706) (1933) 289–337.

[3] D. Ciuonzo, A. De Maio, P. S. Rossi, A systematic framework for composite hypothesis testing of independent bernoulli trials, IEEE Signal Processing Letters 22 (9) (2015) 1249–1253.

[4] A. Wald, Statistical decision functions which minimize the maximum risk, Annals of Mathematics (1945) 265–280.

[5] A. Wald, Sequential analysis, Courier Corporation, 1973.

[6] Y. Shei, Y. T. Su, A sequential test based cooperative spectrum sensing scheme for cognitive radios, in: Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on, IEEE, 2008, pp. 1–5.

[7] R. Chen, J.-M. Park, K. Bian, Robust distributed spectrum sensing in cognitive radio networks, in: The 27th Conference on Computer Communications, IEEE, 2008, pp. 1876–1884.

[8] J. Wu, T. Song, Y. Yu, C. Wang, J. Hu, Sequential cooperative spectrum sensing in the presence of dynamic byzantine attack for mobile networks, PloS one 13 (7) (2018) e0199546.

[9] J.-W. Ho, M. Wright, S. K. Das, Fast detection of mobile replica node attacks in wireless sensor networks using sequential hypothesis testing, IEEE transactions on mobile computing 10 (6) (2011) 767–782.

[10] S. K. Das, J.-W. Ho, A synopsis on node compromise detection in wireless sensor networks using sequential analysis (invited review article), Computer Communications 34 (17) (2011) 2003–2012.

[11] P. R. Vamsi, K. Kant, A lightweight sybil attack detection framework for wireless sensor networks, in: Contemporary computing (IC3), 2014 Seventh International conference on, IEEE, 2014, pp. 387–393.

[12] G. Li, X. Liu, C. Wang, A sequential mesh test based selective forwarding attack detection scheme in wireless sensor networks, in: Networking, Sensing and Control (ICNSC), 2010 International Conference on, IEEE, 2010, pp. 554–558.

[13] F. Gara, L. B. Saad, R. B. Ayed, An intrusion detection system for selective forwarding attack in ipv6-based mobile wsns, in: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE, 2017, pp. 276–281.

[14] P. Dong, X. Du, H. Zhang, T. Xu, A detection method for a novel ddos attack against sdn controllers by vast new low-traffic flows, in: Communications (ICC), 2016 IEEE International Conference on, IEEE, 2016, pp. 1–6.

[15] P. Bhadre, D. Gothawal, Detection and blocking of spammers using spot detection algorithm, in: Networks & Soft Computing (ICNSC), 2014 First International Conference on, IEEE, 2014, pp. 97–101.

[16] J. Parras, S. Zazo, Learning attack mechanisms in wireless sensor networks using markov decision processes, Expert Systems with Applications 122 (2019) 376–387.

[17] H. V. Poor, O. Hadjiliadis, Quickest detection, Vol. 40, Cambridge University Press Cambridge, 2009.

[18] M. Basseville, I. V. Nikiforov, et al., Detection of abrupt changes: theory and application, Vol. 104, Prentice Hall Englewood Cliffs, 1993.

[19] Z. Xing, J. Pei, E. Keogh, A brief survey on sequence classification, ACM Sigkdd Explorations Newsletter 12 (1) (2010) 40–48.

[20] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, Support vector method for novelty detection, in: Advances in neural information processing systems, 2000, pp. 582–588.

[21] Z. Hu, J. Zhang, X. A. Wang, Intrusion detection for wsn based on kernel fisher discriminant and svm, in: International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Springer, 2016, pp. 197–208.

[22] P. K. Varshney, Distributed detection and data fusion, Springer Science & Business Media, 2012.

[23] Y. Wang, J. Wong, A. Miner, Anomaly intrusion detection using one class svm, in: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004., IEEE, 2004, pp. 358–364.

[24] E. R. Dougherty, Random processes for image and signal processing, SPIE Optical Engineering Press, 1999.

[25] F. Zhu, S.-W. Seo, Enhanced robust cooperative spectrum sensing in cognitive radio, Journal of Communications and Networks 11 (2) (2009) 122–133.

[26] A. W. Min, K. G. Shin, X. Hu, Attack-tolerant distributed sensing for dynamic spectrum access networks, in: Network Protocols, 2009. ICNP 2009. 17th IEEE International Conference on, IEEE, 2009, pp. 294–303.

[27] N. Nguyen-Thanh, I. Koo, An enhanced cooperative spectrum sensing scheme based on evidence theory and reliability source evaluation in cognitive radio context, IEEE Communications Letters 13 (7) (2009).

[28] F. R. Yu, H. Tang, M. Huang, Z. Li, P. C. Mason, Defense against spectrum sensing data falsification attacks in mobile ad hoc networks with cognitive radios, in: Military Communications Conference, IEEE, 2009, pp. 1–7.

[29] E. Noon, H. Li, Defending against hit-and-run attackers in collaborative spectrum sensing of cognitive radio networks: A point system, in: Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st, IEEE, 2010, pp. 1–5.

[30] W. Wang, Y. Sun, H. Li, Z. Han, Cross-layer attack and defense in cognitive radio networks, in: Global Telecommunications Conference, IEEE, 2010, pp. 1–6.

[31] Q. Yan, M. Li, T. Jiang, W. Lou, Y. T. Hou, Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 900–908.

[32] F. Benedetto, A. Tedeschi, G. Giunta, P. Coronas, Performance improvements of reputation-based cooperative spectrum sensing, in: Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016 IEEE 27th Annual International Symposium on, IEEE, 2016, pp. 1–6.

[33] J. Wu, Y. Yu, T. Song, J. Hu, Sequential 0/1 for cooperative spectrum sensing in the presence of strategic byzantine attack, IEEE Wireless Communications Letters (2018).