Detection of Malicious Primary User Emulation Based on a Support Vector Machine for a Mobile Cognitive Radio Network Using Software-Defined Radio

Ernesto Cadena Muñoz 1,* , Luis Fernando Pedraza Martínez 2 and Jorge Eduardo Ortíz Triviño 1

1 Systems and Industrial Department, Universidad Nacional de Colombia, Bogotá 111321, Colombia; jeortitz@unal.edu.co
2 Technological Faculty, Universidad Distrital Francisco José de Caldas, Bogotá 111931, Colombia; lfpedrazam@udistrital.edu.co
* Correspondence: ecadenam@unal.edu.co

Received: 26 June 2020; Accepted: 14 July 2020; Published: 10 August 2020

Abstract: Mobile cognitive radio networks provide a new platform to implement and adapt wireless cellular communications, increasing the use of the electromagnetic spectrum by using it when the primary user is not using it and providing cellular service to secondary users. In these networks, there exist vulnerabilities that can be exploited, such as the malicious primary user emulation (PUE), which tries to imitate the primary user signal to make the cognitive network release the used channel, causing a denial of service to secondary users. We propose a support vector machine (SVM) technique, which classifies if the received signal is a primary user or a malicious primary user emulation signal by using the signal-to-noise ratio (SNR) and Rényi entropy of the energy signal as an input to the SVM. This model improves the detection of the malicious attacker presence in low SNR without the need for a threshold calculation, which can lead to false detection results, especially in orthogonal frequency division multiplexing (OFDM) where the threshold is more difficult to estimate because the signal limit values are very close in low SNR. It is implemented on a software-defined radio (SDR) testbed to emulate the environment of mobile system modulations, such as Gaussian minimum shift keying (GMSK) and OFDM. The SVM made a previous learning process to allow the SVM system to recognize the signal behavior of a primary user in modulations such as GMSK and OFDM and the SNR value, and then the received test signal is analyzed in real-time to decide if a malicious PUE is present. The results show that our solution increases the detection probability compared to traditional techniques such as energy or cyclostationary detection in low SNR values, and it detects malicious PUE signal in MCRN.

Keywords: cognitive radio; primary user emulation; security systems; software-defined radio; support vector machine

1. Introduction

The constant evolution of services and applications, the Internet of things, and the need for more bandwidth in wireless networks lead us to develop, implement and improve technologies such as mobile cognitive radio networks (MCRN), which helps to manage the spectrum scarcity problem [1]. The general concept of cognitive radio is an intelligent communication system that adapts in real-time to the radio environment; it is flexible and makes a better use of frequency resources. For this task, it continuously senses the radio frequency environment to find spectral holes where the primary user (PU) is not transmitting, and it involves the identification of PU activity in the spectrum and frequency hopping in case PU signal detected [2].
The first step in MCRN is to detect the PU presence; we can find several spectrum-sensing techniques such as energy detection, location, feature detection, matched filter detection, covariance-based detection, and cooperative spectrum sensing, among others. The main objective is to detect if there is a PU signal at a specific frequency, and the system must do this in real-time so that the algorithms are not complex and have a good time response [1].

Due to this requirement, a specific attack on the MCRN appears. This is called the primary user emulation (PUE), because it senses the medium all the time, and a secondary user (SU) can adapt its signal to mimic the PU signal, making the MCRN or other SUs release the frequency and hop into another one or cancel the transmission [3]. The objective of the attacker in a malicious attack is to obtain control of the channel while avoiding any user of it—it does not transmit information or communicate by itself, but it is hard to distinguish between a PU or a PUE [4].

The problem of finding this difference is at its worst when analyzing the real wireless environment where PU, SU, and PUE are moving, because traditional detection systems such as energy, location or others are based on the static position attacker, that is, when they move, the systems lead to a false state of alarm [5].

We propose to use a support vector machine (SVM) [6] to learn the behavior of cellular modulation techniques such as Gaussian minimum shift keying (GMSK) or orthogonal frequency division multiplexing (OFDM), and with this information and the received data from a spectrum analyzer, recognize a malicious PUE attack on the network. An energy detector is used to release the channel very quickly and to not interfere with a possible PU that is in the process of communicating. Then, the Rényi entropy of this signal is calculated, and with the signal-to-noise ratio (SNR) value, the SVM learns the PU signal behavior. With these signals learned, the hyperplane is calculated and the system is ready to decide the PUE presence in real-time with the implementation of the algorithm on a software-defined radio (SDR) [7].

This paper is organized as follows: Section 2 presents the previous work; Section 3 introduce SVM concepts and the proposal for malicious PUE detection; the SDR testbed is presented in Section 4; Section 5 presents the results. The discussion is conducted in Section 6, and Section 7 presents the conclusions.

2. Previous Work

In MCRN, resources such as frequency band, modulation type, distance, power, noise, and interference have an impact on the overall system. We try to balance or maximize the use of these resources according to the environment, the technology, and the user services needed. A modern machine learning (ML) technique can be useful for providing a mechanism to maximize the use of the available resources. In our case, the learning capability enables the MCRN to learn and adapt the frequency management according to the wireless environment in real-time [1].

Cognitive radio (CR) is based on intelligence [8], and the learning process is very important to obtain the cognitive part. CRs must learn from the environment and decide their actions. There has been a growing interest in applying ML algorithms to the cognitive radio network (CRN). ML has been proposed and discussed in topics such as complex computation and the error propagation of training algorithm, and the majority of machine learning researchers work on techniques such as linear regression, logistic regression and support vector machines to decide if there is PU presence. ML can be reduced to four learning strategies: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [9].

Supervised ML applications such as support vector machines (SVM) result in better PU or PUE detection performance in an MCRN for an expected probability of false alarm when compared with traditional methods. We have to design the training set size, which is the input data in the learning process. Depending on this design, computation time and memory will be required to solve the decision problem. Once this learning process is done, the system can distinguish between signals, and only data points which lie at the margin of the hyperplane are qualified to be categorized as support vectors [2].
Individual spectrum sensing has the hidden primary user (PU) problem, but first we have to optimize the individual sensing process, because the cooperative scheme needs a common channel to transmit, and it requires more resources of the MCRN. It is expected to achieve a higher spectrum-sensing reliability when compared with individual spectrum sensing. All secondary users in the MCRN share their sensing information with the fusion center for a final decision with a hard or soft fusion algorithm and an OR/AND rule to make the final decision [10]. Machine learning techniques are also used for energy detection in cooperative spectrum sensing for CRN, and are even comparable with unsupervised techniques such as K-means [11].

According to [11], the supervised SVM classifier with the linear kernel performs well in terms of the detection probability. The computational complexity of the SVM-linear classifier is compensated by its higher detection capability and comparatively lower training and classification delay.

The detection systems can integrate an artificial neuron network (ANN) into a cognitive engine to learn from environment measurements and analyze how the status of the network affects its performance on different channels. The ANN and the feed-backward ANN belong to supervised learning; the training dataset must be labeled by a human, which is not easy in a real RF environment. The environment adaptation of unsupervised learning algorithms is better than supervised learning algorithms for CRN [9].

Another application is CRN based on genetic algorithms, where these algorithms are used to control channel selection schemes (GA-CSS) for CRs by considering the random arrival of data rate and bit error rate (BER) in order to select the optimal control channel. Besides, an improved genetic algorithm-based spectrum assignment model helps to achieve higher system utility with more cognitive users under the same population size. Learning algorithm-based decision-making is another major task that has been widely investigated in with regard to CR applications such as pattern classification [12].

Linear prediction (LP) algorithms are considered as an important member of statistics and mathematics and, in this case, the spectrum prediction and future values are predicted as a linear function of previous samples. LPs are used in digital signal processing (DSP) due to their remarkable simplicity. They have been implemented for the prediction of the spectrum in the time domain. Some common models for linear prediction models include the auto-regressive (AR) model, the mobile average (MA) model, the model of auto-regressive moving average (ARMA) and the auto-regressive integrated moving average model (ARIMA) [13,14].

The Bayesian interference (BIF) is another approach that been a classic prediction method in CRN. It has been implemented as a part of the Markov system. The scheme can be defined as a derived probability distribution of the system. It is calculated as the posterior probability distribution according to the Bayes rule, from the prior probability distribution and the likelihood function. The problem is to specify a prior probability distribution because it has a significant influence on the complexity of the subsequent computation [13].

A machine learning framework is proposed for primary user emulation attack detection in cognitive radio networks in [15]. It is based on classification models that exploit features extracted using the proposed pattern-described link signature method (PDLS) from the channel impulse response (CIR) in orthogonal frequency division multiplexing (OFDM)-based transceivers [15].

The particle swarm optimization (PSO) algorithm has been applied to the hyper-parameter optimization of the SVM to establish a prediction model for the heat exchanger. A support vector regression (SVR) model is also used to achieve the online prediction of thermal resistance denoising for low-temperature superheaters, showing a prediction accuracy above 98% without adding expensive devices to the system by predicting the results with the SVR with 80% of the samples used for training and 20% for testing [16].

In general, machine learning techniques, if trained correctly, can be a good approach to minimize the delay of the detection and easily use complex models. They use features such as energy statistics, the probability vector of occupancy over time as an input to the model, and propose a classification problem to find the optimal decision if there is a PU, an attacker, or noise [17].
Contributions

In this paper, we describe the malicious PUE attack in the MCRN comparing the theory, simulations, and experiments in the SDR testbed to establish a system to detect this attack, which, to the best of our knowledge, is the first approach to detect this attack with this method. The selfish attack is described in [5]. In this paper, we focus on a malicious attack using ML. The main contributions of this paper are summarized as follows:

- Analyze and extract the PU signal behavior in systems with different modulations such as GMSK or OFDM.
- Establish the SNR and entropy as the two dimensions of the SVM.
- Improve energy detection by adding entropy and feature recognition.
- Define and implement the SVM with the learning behavior as an input generating the decision of the presence of a malicious PUE.
- Train and test the SVM with real values from the SDR.
- Predict the results of the malicious PUE signal.
- Model and implement the malicious PUE attack in the MCRN.
- Implement the MCRN testbed and experiment with the SVM learning stage, calculating the SNR and entropy based on the energy.

The rest of this paper is organized as follows: Section 3 introduces the SVM system model; the testbed for the experiments is described in Section 4; numerical results are presented in Section 5; in Section 6, a discussion takes place; finally, conclusions are drawn in Section 7.


In this section, we define the MCRN, users and interactions, variables, and scenarios for the attack. We propose the SVM for malicious PUE detection by using the energy detector and entropy detector.

In the primary network, operators have a BTS, an eNodeB depending on the technology; this element establishes a communication with the PU in some licensed frequencies by using some modulation methods such as GMSK or OFDM. The MCRN uses a spectrum analyzer to sense the medium and detect the spectral holes according to the selected frequency range. When it finds a hole, it starts the communication with the SU that needs some service. SUs connected to the MCRN utilize a cognitive base station (CBS) device to operate phone calls or messages (or anything else that is required) while continuously listening to the presence of the PU [18].

The PUE attack is given by an SU, which, in this case, emulates the PUE signal as a jammer for interfering with the transmission in the selected frequency range. Depending on the detection strategy of the MCRN, the attack will be identified as a licensed PU and has to change its frequency or deny the service to the SUs. One of the most used methods for detection is energy detection, as it senses the PU power in a defined frequency and if it is over a threshold, it is identified as a PU. This is where the PUE attacks by generating a signal in the frequency range. Detections methods vary in terms of energy, features, cyclostationary, and location detection, trying to separate and decide if what they encounter is a PU or a PUE signal [19].

The MCRN, users, and interactions are shown in Figure 1. The PUE attack can affect the MCRN and the primary network depending on the used configuration [20].

We analyzed the mobile network signals such as 3G or 4G systems, identifying the patterns in the energy detection in time and calculating the entropy of these values. Combining these with the SNR value and with the measured values, we obtain a database for the training system of the SVM.

The SVM algorithm generates the hyperplane based on the feature vectors maximizing the space between the limit values for each case in the presence or absence of a PU. With these values, the system can now decide if the experimental input is a PU or a malicious PUE attack [1].

The SVM is divided into two parts [21]: The first one is the SVM learning process where we extract parameters such as SNR, entropy, and modulation signal behavior of GMSK and OFDM modulations.
from the energy detector that makes an analog to digital conversion (A/D). The second is the fast Fourier transform (FFT), which calculates the square and makes an average of the energy signal.

**Figure 1.** Primary network (BTS), mobile cognitive radio network (MCRN) (cognitive base station—CBS), users and primary user emulation (PUE) attack.

These results are stored in the SVM and train the algorithm just one time. This is shown as the SVM learning process in Figure 2.

**Figure 2.** Proposed support vector machine (SVM) learning and classification process.

After the learning process, the algorithm calculates the hyperplane and vectors and is ready to classify the inputs. In the middle, part of the data is used to test the algorithm and measure the accuracy. The second part is to decide the presence of a malicious PUE. For this process, we made the energy detection and entropy calculation so that it is capable of identifying the signal. These values are the inputs for the SVM classification system and, according to the database and the algorithm of the SVM, it is decided if the presence is a PUE or not. This is shown in Figure 2 as the SVM classification process.

The SVM is divided into two parts [21]: The first one is the SVM learning process where we extract parameters such as SNR, entropy, and modulation signal behavior of GMSK and OFDM modulations from the energy detector that makes an analog to digital conversion (A/D). The second is the fast Fourier transform (FFT), which calculates the square and makes an average of the energy signal.

The SVM algorithm generates the hyperplane based on the feature vectors maximizing the space between the limit values for each case in the presence or absence of a PU. With these values, the system can now decide if the experimental input is a PU or a malicious PUE attack [1].
hyperplane allowing for the maximum margin is called an optimal separating hyperplane [22] as is shown in Figure 3.

![Figure 3. Support vector machine (SVM) operation with two dimensions [22].](image)

In Figure 3, assuming \( H \) as the optimal hyperplane, \( H_1 \) and \( H_2 \) are the two support planes chosen by the support vectors parallel to \( H \). The margin is defined as the distance between the two support planes. To make the classification, we have to separate the two classes of data by maximizing the margin so that the hyperplane divides the two classes of data [23]. The hyperplane can be defined as Equation (1).

\[
H : w \times x + b = 0 \\
H_1 : w \times x + b = -1 \\
H_2 : w \times x + b = 1
\]  
where \( x \) represents the argument, \( w \) is the weight and \( b \) is a constant value [23]. Based on \( w \), we need to minimize the vector norm of \( w \) to maximize the category margin, and hence the objective function is Equation (2) [24].

\[
\text{min} \left( \frac{1}{2}||w||^2 \right) \\
\text{s.t.} \quad y_i(w \times x_i + b) \geq 1 \\
i = 1..n
\]  

A Lagrange dual method is used to obtain the optimal solution. After introducing the Lagrange multiplier, the problem changes to Equation (3) [23].

\[
\text{min} \left( \frac{1}{2}||w||^2 \right) - \sum_{i=1}^{n} a_i (y_i(w \times x_i + b) - 1) \\
\text{s.t.} \quad a_i \geq 0
\]  

The solution can be simplified to obtain the final classification decision function Equation (4) [23].

\[
f(x) = \text{sgn}((w \times x_i) + b) \\
f(x) = \text{sgn}\left(\sum_{i=1}^{n} a_i x_i x + b\right)
\]  
where \( x_i \) are the support vectors, \( a \) is the Lagrange multiple vectors and \( \text{sgn}(x) \) is the symbolic function [23].
Once the general functionality of the SVM has been explained, we proceed to describe the structure of the SVM for the detection model, which is shown in the following pseudocode:

1. GNURadio starts measuring the energy signal at the desired frequency and estimates the SNR value by using the RTLSDR device using the equations described in Section 3.2.
2. The entropy of the energy is calculated with a modified GNURadio main python code using the equations described in Section 3.3.
3. Using the previously installed scikit software, the kernel SVM code is invoked with the fit function and the supported vectors, and the margin and the hyperplane are calculated.
4. The results of the learning process are stored in the scikit program.
5. The second part (classification process) begins with measuring the averaged energy signal and SNR values in real-time by using the RTLSDR device.
6. The entropy of the averaged energy signal is calculated.
7. The prediction code is invoked with entropy and SNR value as an input.
8. The classification is made with the results of the prediction code by giving a binary result if there is a PU (1) or a malicious PUE (0) present.
9. According to these results, the SDR-USRP-2922, configured as a CBS for the MCRN, starts the transmission to the SU.

### 3.1. PUE Definition

The mathematical definition of PUE considers \( n(t) \) as the noise signal, \( h(t) \) as the impulse response of the system, \( s(t) \) as the received signal from a PU, \( s'(t) \) as the mimic signal from the PUEA and \( y(t) \) as the received signal \([25]\), as shown in Equation (5).

\[
y(t) = \begin{cases} 
n(t) & \text{SU} \\
h(t) \times s(t) + n(t) & \text{PU/PUE} 
\end{cases}
\]  

(5)

For malicious PUE, the PU signal behavior helps us to recognize if it is a PU or a PUE, but the whole system, including energy, entropy, and SVM, allows the detection of malicious PUE.

### 3.2. Energy Calculation

An additive white Gaussian noise (AWGN) channel is assumed, and we use a time-based energy detection method as in \([26]\). \( N \) is the total number of samples of the energy and \( Y(n) \) is added during one \( S_i \) detection interval, as shown in Equation (6).

\[
Z(Y_n) = \frac{1}{N} \sum_{n=1}^{N} |Y(n)|^2
\]  

(6)

The energy detector compares \( Z \) to a threshold \( \lambda \) and makes a decision regarding the presence or absence of signals. The probability of detection and the probability of false alarm for energy detection is defined in Equation (7) \([26]\).

\[
P_d = p(Z \geq \lambda|H1), \\
P_{fa} = p(Z \geq \lambda|H0).
\]  

(7)

\( Z \) is a Gaussian random variable with a mean of zero and variance of \( \sigma m^2 \), and \( \gamma \) is the signal-to-noise-ratio (SNR). The probability of false alarm (\( P_{fa} \)) and the probability of detection (\( P_d \)) are defined in Equation (8) \([26]\).

\[
P_d = Q\left(\frac{\lambda-N-N\times \gamma}{\sqrt{2\times N \times 4\times N\times \gamma}}\right) \\
P_{fa} = Q\left(\frac{\lambda-N}{\sqrt{2\times N}}\right)
\]  

(8)
In our proposed solution, we do not define a hypothesis or a threshold; we take the energy samples and filter, average, and use them as an input for the entropy calculation.

3.3. Entropy Calculation

The selected entropy detector is in the frequency domain as it is not sensitive to noise. This system is considered to improve the detection of PU presence in the network and operates with low SNR. It is considered a system without a priori knowledge of the signal. A binary hypothesis test is used to find out the presence of the PU/PUE signal, where \( H_0 \) is the noise in the absence of a signal and \( H_1 \) indicates the presence of a signal, as described below [27].

\[
y(n) = \begin{cases} 
  u(n) & H_0 \\
  s(n) + u(n) & H_1 
\end{cases}
\]  

(9)

A discrete Fourier transform (DFT) is applied to (9) and (10) is obtained [28].

\[
\overline{Y}(k) = \begin{cases} 
  \overline{U}(k) & H_0 \\
  \overline{S}(k) + \overline{U}(k) & H_1 
\end{cases}
\]  

(10)

where \( \overline{Y}(k) \), \( \overline{S}(k) \) and \( \overline{U}(k) \) are the complex spectrum of the received signal, PU signal, and noise, respectively. The DFT size \( K = N \) [28]. We use the Rényi entropy described below in Equation (11) [29].

\[
H_\alpha(Y) = \frac{1}{1-\alpha} \times \log \left( \sum_{i=1}^{n} p_i^\alpha \right)
\]  

(11)

Given \( Y \) discrete events, \( p_i \) is the probability of occurrences in the \( i \)th bin, \( \alpha \) is the Rényi entropy order, where \( \alpha \geq 0 \) and \( \alpha \neq 1 \). Using the histogram method to estimate the received signal probability of detection, for a given number of bins \( L \), Rényi entropy is estimated by Equation (12) for a statistic test [29,30].

\[
T(Y) = \frac{1}{1-\alpha} \times \log \left( \sum_{i=1}^{l} k_i^\alpha \right)
\]  

(12)

In the entropy detector, we compare the test statistics with a threshold for calculating the probability of detection. We will use the entropy value as an input to the SVM directly without making any decision regarding the detection by itself, as it requires the SVM classification results.

4. SDR Testbed for Experiments

For the experiments, we use an RTLSDR 2832 device for spectrum analysis connected to a PC by the USB port, and for the MCRN, an NI USRP-2922 is used with GNURadio [31] with a basic CR protocol implemented. This device works in a range of 400 MHz to 4.4 GHz. To emulate the PUE, we use another USRP which transmits a malicious signal in the selected frequency. We use the uplink (UL) signal to measure the energy. The CBS detects the PU/PUE signal and, according to these values, it assigns a frequency and sends the SU signal. The PUE detects PU signals and sends malicious PUE signals to confuse the CBS detection system which recognizes it as a PU, as can be seen in Figure 4. In the learning process, we use the primary network of an operator to measure the signals in GMSK and OFDM in a real phone call. We estimate the energy signal for each case. This information is the input for the SVM system with the SNR signal.

For experiments, the PUE is configured to transmit a signal in the absence of a PU signal to avoid interfering with the operator’s network. It has previous knowledge of the PU frequencies, signals, and characteristics to detect it, but the PUE can interfere with this network in a hostile configuration, subsequently affecting PUs and SUs. A python program is implemented to receive the energy data
from RTLSDR and the SNR and calculate the entropy. With these data, the program extract the behavior in the learning process and calculates the support vectors and the hyperplane.

![Diagram of software-defined radio (SDR) testbed for experiments.](image)

**Figure 4.** Software-defined radio (SDR) testbed for experiments.

In the detection process, the program receives these data, and with the previous knowledge of signals, the SVM decides if there is a malicious PUE present in the system using scikit [32]. This can be seen in the flowchart in Figure 5.

![SVM system flowchart.](image)

**Figure 5.** SVM system flowchart.
The device performance parameters of center frequency, bandwidth, instantaneous dynamic range (IDR), noise floor, noise figure and sensitivity under the evaluated experiment can be seen in Table 1 according to [33].

Table 1. Device Parameters.

<table>
<thead>
<tr>
<th>Device</th>
<th>Center Frequency (MHz)</th>
<th>Bandwidth (MHz)</th>
<th>Noise Floor (dBm)</th>
<th>Noise Figure (dB)</th>
<th>Sensitivity (dBm)</th>
<th>IDR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRP2922</td>
<td>850</td>
<td>20</td>
<td>−113</td>
<td>5</td>
<td>−105</td>
<td>50</td>
</tr>
<tr>
<td>RTLSDR</td>
<td>850</td>
<td>3.2</td>
<td>−102</td>
<td>3.5</td>
<td>−97.5</td>
<td>20</td>
</tr>
</tbody>
</table>

5. Results

In this section, the results of the test scenarios are presented and analyzed.

5.1. GMSK Experiments

In the first part of the process (learning), we start the systems and read the data. For experiments, we use both GMSK and OFDM UL signals in a phone call as the input to the system. For some SNR values, we acquire the time signal and calculate the energy and entropy to identify the signal in a specific frequency. The frequency depends on the technology and operator, and we use a real operator and phone call to acquire these signals in GSM and 4G-LTE cellular systems.

We calculate the energy of the signal using the RTLSDR, and these data are transmitted to the PC where there are on average 100 samples. The results for the GMSK signal can be seen in Figure 6, where a periodic pattern can be seen in time. These results show that the energy of the modulated signal is not constant in time. If we take the energy in a time that is low (under −90 dBm), the results will have a wrong interpretation, thus, we calculate the entropy of the averaged energy signal according to [34].

![Figure 6. Average energy pattern for the Gaussian minimum shift keying (GMSK) signal.](image)

The next step is to calculate the entropy of the received energy signal. There are two stages: in the first one, there is only a noise signal; the second one is where a PUE attacks. These signals are processed and the entropy of the averaged energy received is calculated, as can be seen in Figure 7. The idea to use entropy to classify the signals is explained in this figure. The first 100 samples are the calculation of entropy in the absence of a malicious PUE signal, while the rest of the samples are taken when a modulated PU signal is present in the frequency. This is how the system detects the PUE attack.
The next step is to calculate the entropy of the received energy signal. There are two stages: in the first one, there is only a noise signal; the second one is where a PUE attacks. These signals are processed and the entropy of the averaged energy received is calculated, as can be seen in Figure 7. The idea to use entropy to classify the signals is explained in this figure. The first 100 samples are the calculation of entropy in the absence of a malicious PUE signal, while the rest of the samples are taken when a modulated PU signal is present in the frequency. This is how the system detects the PUE attack.

Based on this information, we observe that entropy values are linearly separable and we increase the probability of detection by implementing an SVM in two stages, namely, the learning part and the testing part. We proceed to take 350 samples for each SNR value from −25 dB to 0 dB and apply an SVM in the learning stage process to define the hyperplane and predict the values. The idea is to develop an SVM using two parameters, SNR and entropy, which are experimental values in an indoor laboratory. The parameters for this stage are shown in Table 2.

Table 2. Learning process parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Samples</td>
<td>10,000 in total: 8750 samples for learning; 1250 for testing</td>
</tr>
<tr>
<td>Averaged Values</td>
<td>350 samples for each signal-to-noise ratio (SNR) value</td>
</tr>
<tr>
<td>Noise Signal</td>
<td>Additive white Gaussian noise AWGN</td>
</tr>
<tr>
<td>Service</td>
<td>Phone Call</td>
</tr>
<tr>
<td>Frequency</td>
<td>831.8 MHz</td>
</tr>
<tr>
<td>Time of measurement</td>
<td>1 s for each SNR value</td>
</tr>
<tr>
<td>Confidence Level</td>
<td>95%</td>
</tr>
<tr>
<td>Margin of error</td>
<td>5%</td>
</tr>
</tbody>
</table>

After the learning process, we test the SVM algorithm with 1250 samples. The results of this achieve 99% effectiveness for the testing data according to the scikit report.

The last stage is to test the system with the modulated GMSK signal in the SDR. We capture the energy signal in real-time, calculate the entropy and average 100 values. These values are sent to the SVM which gives a binary result, one if a PUE is present and zero if a PU is present. A sample of the hyperplane calculation is shown in Figure 8, where the support vectors are the green circles, and the hyperplane and working zone are blue, showing the maximization of the margin. The yellow circles correspond to a PU signal and the black circles correspond to malicious PUE signal. We can see the support vectors and the hyperplane margin in the learning process. In the classifying process, when an entropy value is calculated, the software detects the malicious PUE signal according to the previously stored data from the learning process.

We take 10,000 samples for different SNRs and energy values with an objective $P_{fa}$ of 0.1 [28]. With these values, we measure the practical probability of detection as the times of effective detection over all samples. As an example, we compare the entropy detector found in [30], and the objective $P_D$ of 90% is obtained −2 dB below the simple entropy detector, while it is −9 dB below the entropy
detector used in [35]. Figure 9 shows this comparison, where an improvement in PD for low SNR values between −2 dB and −9 dB compared with entropy values from the literature can be seen. The detection system appears to work better than a simple entropy detector in low SNR values, and it detects malicious PUE properly.

Table 2. Learning process parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Samples</td>
<td>10,000 in total: 875 0 samples for learning; 1250 for testing</td>
</tr>
<tr>
<td>Averaged Values</td>
<td>350 samples for each signal-to-noise ratio (SNR) value</td>
</tr>
<tr>
<td>Noise Signal</td>
<td>Additive white Gaussian noise (AWGN)</td>
</tr>
<tr>
<td>Service Phone Call</td>
<td>Frequency 831.8 MHz</td>
</tr>
<tr>
<td>Time of measurement</td>
<td>1 s for each SNR value</td>
</tr>
<tr>
<td>Confidence Level</td>
<td>95%</td>
</tr>
<tr>
<td>Margin of error</td>
<td>5%</td>
</tr>
</tbody>
</table>

After the learning process, we test the SVM algorithm with 1250 samples. The results of this achieve 99% effectiveness for the testing data according to the scikit report.

The last stage is to test the system with the modulated GMSK signal in the SDR. We capture the energy signal in real-time, calculate the entropy and average 100 values. These values are sent to the SVM which gives a binary result, one if a PUE is present and zero if a PU is present. A sample of the hyperplane calculation is shown in Figure 8, where the support vectors are the green circles, and the hyperplane and working zone are blue, showing the maximization of the margin. The yellow circles correspond to a PU signal and the black circles correspond to malicious PUE signal. We can see the support vectors and the hyperplane margin in the learning process. In the classifying process, when an entropy value is calculated, the software detects the malicious PUE signal according to the previously stored data from the learning process.

We take 10,000 samples for different SNRs and energy values with an objective Pfa of 0.1 [28]. With these values, we measure the practical probability of detection as the times of effective detection over all samples. As an example, we compare the entropy detector found in [30], and the objective PD of 90% is obtained −2 dB below the simple entropy detector, while it is −9 dB below the entropy detector used in [35]. Figure 9 shows this comparison, where an improvement in PD for low SNR values between −2 dB and −9 dB compared with entropy values from the literature can be seen. The detection system appears to work better than a simple entropy detector in low SNR values, and it detects malicious PUE properly.

![SVM with Entropy](image1)

Figure 8. SVM with entropy with the GSMK signal.

![Probability of detection](image2)

Figure 9. Probability of detection with SVM vs. entropy for GMSK.

5.2. OFDM Experiments

We use an SDR to transmit and receive information with OFDM modulation similar to GMSK, and the learning process parameters are the same as those in Table 2.

The entropy calculated for the OFDM signal can be seen in Figure 10. The signal trend is similar to GMSK in terms of entropy. The first 50 samples are with the malicious PUE signal present, and the
rest are with the PU signal present. The threshold is very close and here the SVM helps us to classify values under these conditions where a simple threshold will lead us to an error.

![Entropy of OFDM received signal](image)

**Figure 10.** Entropy with and without of the orthogonal frequency division multiplexing (OFDM) signal.

The SVM machine is implemented with these entropy values and the hyperplane and support vector are calculated by using the learning process, as can be seen in Figure 11. Notably, the margin is smaller than the GMSK signal. If a simple energy detector or entropy detector is used based on a threshold, the results will increase false alarm results, but with the SVM, the results are better for low SNR values and it is able to detect malicious PUE.

![SVM with Entropy](image)

**Figure 11.** SVM with entropy with the OFDM signal.

We apply the same process of GMSK, that is, we take 10,000 samples for different SNR and energy values with an objective PFA of 0.1 [28]. With these values, we measure the practical probability of detection as the times of effective detection over all samples. We compare these practical results with the entropy detector found in [30], and the objective PD of 90% is obtained $-2$ dB below the simple...
entropy detector. Figure 12 shows this comparison, where an improvement in $P_D$ for low SNR values compared with entropy values from the literature can be seen, which shows that the malicious PUE can be detected with high performance and effectiveness.

![Figure 12. Probability of detection with SVM vs. entropy for OFDM.](image)

### 6. Discussion

The results show that the proposed method increases the probability of detection, working significantly better in OFDM than energy or entropy-based detectors when compared with previous work in [30,35–37]. The simulations were carried out with the same parameters as the literature, and the experimental results show that for low SNR, the system detects the signal for above 90% for $-17.5$ dB in GMSK and $-11$ dB for OFDM. We apply the complete process to 10,000 samples with different SNR and energy values, and the results show that the malicious PUE can be detected with high performance and effectiveness in low SNR values. Table 3 includes a comparison for the detection accuracy with the measure of probability of detection vs. SNR values.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$-20$ dB</td>
<td>36%</td>
<td>4%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>$-18$ dB</td>
<td>82%</td>
<td>50%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>$-17.5$ dB</td>
<td>90%</td>
<td>85%</td>
<td>0%</td>
<td>0%</td>
<td>5%</td>
</tr>
<tr>
<td>$-15.5$ dB</td>
<td>92%</td>
<td>90%</td>
<td>0%</td>
<td>0%</td>
<td>22%</td>
</tr>
<tr>
<td>$-15$ dB</td>
<td>96%</td>
<td>94%</td>
<td>0%</td>
<td>0%</td>
<td>25%</td>
</tr>
<tr>
<td>$-10$ dB</td>
<td>99%</td>
<td>99%</td>
<td>83%</td>
<td>50%</td>
<td>60%</td>
</tr>
<tr>
<td>$-8.5$ dB</td>
<td>99%</td>
<td>99%</td>
<td>96%</td>
<td>90%</td>
<td>92%</td>
</tr>
</tbody>
</table>

The advantage of this proposal is that we do not need to calculate or measure the energy threshold or entropy threshold for the detection system. The learning process does not take too much time and it needs only 192 s to complete it one time. The computational complexity is intermediate in the learning process and low in the real-time decision, as the system averages the entropy and makes a decision based on the SVM hyperplane calculated in the learning process. If a new signal is required, we prepare the system for a new learning process and the algorithm process it automatically.

### 7. Conclusions

A malicious PUE detector was proposed based on an SVM with Rényi entropy analysis of energy received for an MCRN. The system was examined in an SDR scenario by using USRP NI-2922 configured with GMSK and OFDM modulations, and the expected results were simulated with Monte
Carlo experiments in AWGN channels. The results show that for the two modulations, the probability of detection increases when compared with other detection techniques, without the dependence of a threshold such as energy or entropy techniques by themselves and with better results in low SNR values.

The results show that the SVM outperforms the energy detector method found in the literature by about 12 dB and the entropy method by about 3 dB. The influence of noise decreases due to the entropy calculation, which allows the system to detect PUE with lower SNR values. The system works in the two modulations, but it shows a better performance in OFDM systems where the threshold is hard to find, and it is dynamic in other detection techniques.

The SDR experiments show that the SVM for detection can be implemented in MCRN, and the learning process does not take a significant amount of time for one completion. After that, the system can detect malicious PUE in real-time without an excessive complexity for calculations and with a high probability of detection under lower SNR values.

Regarding future research, we suggest exploring the cooperative scenario in MCRN, where all users are in movement in an indoor scenario and an outdoor scenario.

**Author Contributions:** Methodology, software, and hardware: E.C.M.; experiment design: E.C.M., L.F.P.M., and J.E.O.T.; performed the experiments: E.C.M., L.F.P.M.; results analysis, review and editing of the paper: E.C.M., L.F.P.M. and J.E.O.T. All authors participated in the discussion and proofread the work. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Colciencias, grant number Doctorado Nacional 757. The APC was funded by Universidad Distrital Francisco José de Caldas, CIDC.

**Acknowledgments:** We express our gratitude to Colciencias, Universidad Distrital Francisco José de Caldas, Centro de Investigaciones y Desarrollo Científico (CIDC) and Universidad Nacional de Colombia for the founding of this Project.

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

**References**


33. Cheema, A.A.; Salous, S. Spectrum Occupancy Measurements and Analysis in 2.4 GHz WLAN. *Electronics* 2019, 8, 1011. [CrossRef]


© 2020 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 (http://creativecommons.org/licenses/by/4.0/).