Comparison of Machine Learning Approaches Based on Multiple Channel Attributes for Authentication andSpoofing Detection at the Physical Layer

Andrea Stomaci, Dania Marabissi*, and Lorenzo Mucchi

Department of Information Engineering, University of Florence, Italy.
Email: andrea.stomaci@unifi.it (A.S.); dania.marabissi@unifi.it (D.M.); lorenzo.mucchi@unifi.it (L.M.)
*Corresponding author

***Abstract*—The aim of this study is to assess the effectiveness of Physical Layer Authentication (PLA) in securing IoT nodes. Specifically, we present a PLA framework based on wireless fingerprinting, where the legitimated node is distinguished from potential attackers by exploiting the unique wireless channel features. To achieve this objective, we employ various machine learning approaches for anomaly detection, making use of a wide range of channel attributes in time-varying conditions. In particular, four different Machine Learning (ML) strategies in their one class version have been considered and compared: decision-tree, kernel-based, clustering and nearest neighbors. Our study highlights advantages and disadvantages of each method, considering parameters optimization, training requirements and time complexity. Results show that the use of multiple-attribute allows to achieve accurate detection performance. In particular, our results reveal that the kernel-based solution is the one that achieves best results in terms of accuracy, but the nearest neighbor's solution has very similar performance with a significant advantage in terms of complexity and no need for training, making it more suitable for time-varying contexts, and a promising choice for securing IoT nodes through PLA based on wireless fingerprinting. The other two alternatives have somewhat lower performance but low complexity. This research contributes valuable insights into enhancing IoT security through PLA techniques.**

***Keywords*—physical layer security, machine learning, authentication, spoofing detection**

I. Introduction

Internet of Things (IoT) is an essential element to develop smart environments and to provide emerging services and applications. The IoT paradigm involves a massive number of smart objects interacting with each other and with Internet. In many IoT applications, such as industrial, e-health, or autonomous vehicles, some constraints such as low-latency, accuracy, energy consumption as well as security are of paramount importance. Particularly, the presence of rogue de-vices claiming a false identity can expose IoT system to many types of security attacks that can have severe consequences, especially in a wireless environment. One of the first lines of defense is represented by the identification of legitimate nodes (authentication) and the detection of rogue devices (spoofing detection). Traditionally, this is done by means of higher-layers procedures that, however, may not be suitable for emerging IoT paradigms, in particular, when low-resourced devices are involved. As a consequence, different kinds of authentication and spoofing detection approaches have recently acquired a relevant importance in IoT systems, to provide an additional level of security protection. In particular, Physical Layer Authentication (PLA) methods, exploiting specific characteristics of the device and/or of the wireless channel have gained more and more attention by the research community [1]. We focus here on a PLA approach, called Wireless Fingerprinting (WF) [2], that aims to distinguish a legitimate node from a malicious one by exploiting the uniqueness of the wireless channel experienced by each node. We analyze and compare different decision criteria based on Machine Learning (ML). In particular, we consider approaches based on Nearest Neighbor (NN), Support Vector Machine (SVM), Isolation Forest (iForest) and k-means algorithms in their One-Class (OC) version.

Recently, many research efforts have been devoted to de- sign and investigate PLA and PHY-layer spoofing detection methods based on ML [3]. In comparison with traditional approaches, ML-based ones usually do not require setting of a threshold to achieve the detection in the hypothesis test and are not model-based. The right value of the threshold, as well as accurate models, are difficult to be obtained in time-varying contexts, thus strongly affecting the detection performance in terms of accuracy and speed. Moreover, ML approaches allow to consider multiple attributes, thus increasing the detection accuracy. Due to estimation errors and time-variations of parameters, single-attribute algorithms often are not able to differentiate the transmitters all the time. In addition, it is harder for a malicious node to estimate and imitate multiple attributes than a single one. Different PLA methods based on ML have been investigated under different conditions and environments. In general, non-parametric learning methods that do not require any a priori knowledge, but rely only on the available data, such as Nearest Neighbur (NN), kernel-based and decision tree,

Manuscript received August 8, 2023; revised August 30, 2023; accepted October 12, 2023; published February 26, 2024.

are more suitable for IoT applications. Indeed, a priori knowledge of attributes requires static properties that cannot be applicable for many scenarios, such as 5G and beyond networks; in addition, the knowledge of the attacker is generally not realistic. In [4], the authentication is based on channel impulse response features extracted using statistical analysis and coefficients correlation. Attributes are then selected and classified by means of an SVM algorithm. The system is designed for a MIMO system exploiting spatial information in a static environment. Time-varying conditions are instead considered in [5] where a kernel machine-based scheme is proposed. Here, the dimensionality of multiple attributes is reduced by means of the kernel function. In [6] SVM and k-means clustering algorithms are applied and compared in an amplify and forward Unmanned Aerial Vehicle (UAV) context. The extracted attributes are the mean and the variance of the received samples and artificial data based on the statistical knowledge of channel state information are used to train the ML algorithms. Different ML algorithms are used in [7] to test the accuracy of body sensors authentication using Received Signal Strength Indicator (RSSI) and Frame Loss Rate as attributes. The simulated scenario is very limited, and no time variability is considered. In [8], authentication strategies, based on different ML algorithms and on statistical criteria based on a hypothesis testing approach, are evaluated, and compared in a time-varying environment. In particular, SVM and NN algorithms are considered. ML algorithms classified data exploiting the real and the imaginary parts of the channel coefficients measured on multiple carriers of an Orthogonal Frequency Division Multiplexing (OFDM) system. More recently, in [9], different ML methods are simulated over the entire estimated channel matrix to produce binary authentication (legitimate/malicious). In [10], carrier frequency offset and RSSI are used as parameters to check the performance of ML methods, concluding that considering two parameters is better than only one. An overview of existing studies in PLA is reported in [11]. The paper, anyway, focuses on ML and DL methods, but it does not discuss the specific physical parameters to extract from the received signal. As shown in previous papers, there is not a single ML method more suitable for all scenarios and datasets, that should be checked with several ML algorithms. For this reason, we compare here different ML algorithms, considering their OC version because we assume no knowledge about the malicious node, hence, only legitimate nodes' samples are available during training. Assuming no-knowledge about the attacker is more realistic, moreover, in [8], it has been shown that the use of SVM and NN algorithms in their binary version does not bring any advantage over their OC version.

This paper provides an exhaustive comparison of anomaly detection methods based on known ML approaches, that is not present in the literature to the best of our knowledge. Detection accuracy, time complexity, optimal parameters and training requirements are evaluated. We compare four different strategies: kernel-based, clustering, nearest neighbors and decision-tree, while generally, only some of them are considered and compared. Moreover, we consider a time- varying scenario and we use a large set of attributes. Even if multiple-attribute algorithms are considered an effective mean to improve the authentication accuracy, in the literature they are usually limited to multiple observations of the same attribute, or to the extraction of multiple features of the same attribute [4, 6, 8], rather than to effective multiple-attribute as in [5]. Among others, we exploit both the delay and the Angle of Arrival (AoA). Particularly, AoA is exploited in [12] to validate the claimed GPS location information in a vehicle- to-roadside communication using a two-side hypothesis testing problem and in [13] to create a unique signature that is used in a challenge-response protocol. All these aspects have been rarely considered in previous papers, and not jointly.

The paper is organized as follows. In Section II the system model is described while the proposed authentication/spoofing detection framework is introduced in Section III. Different solutions are described and compared. The numerical results are provided in Section IV. Finally, some conclusions are drowned in Section V.

II. System Model

We focus on a wireless IoT system where resource-constrained nodes are connected to a sink node that collects, elaborates and distributes information thanks to a Wireless Sensor Network (WSN). IoT nodes simply collect information and communicate with the sink node using radio transceiver with a single antenna. The sink node is a more powerful node performing more complex operations and is equipped with a multiple antenna system for transmitting/receiving data. Consequently, the sink node is able to extract spatial information from received signals. IoT nodes must be able to authenticate in order to ascertain their legitimacy as a communication source and to avoid the presence of malicious nodes that attempt to subtract information from or inject false information into the network, by posing as legitimate node. The Access Point (AP) of the WSN acts as sink node and has in charge the node authentication of the devices communicating with it. Hence, the AP (Named Bob) receives a signal transmitted from a legitimate IoT device (named Alice) and has to confirm its identity. In the area is present a rogue device (Eve) that tries to impersonate Alice. Bob wants to identify communications form Alice and to detect potential anomalies due to Eve attacks exploiting the WF.

A. Channel Model

The communication channels between Alice and Bob, and Eve and Bob are corrupted by Additive Withe Gaussian Noise (AWGN), pathloss and time-varying fading. We assume here a multipath fading channel defined for IEEE 802.11ac WSN [14] whose statistical distributions have been extracted from actual channel measurements. It is suitable for indoor communications operating at 5 GHz with frequency bands up to 160 MHz. We refer to the Model-D scenario in [15], defined for large open indoor environments with mobility in the range 0-5 km/h. More in detail, we consider static

transmitting/receiving nodes but the propagation channel is time-varying due to the mobility of the scatters in the environment. A single input multiple out- put propagation channel is considered, with one transmitting antenna and Q receiving antennas. The channel received by each antenna is then modelled as a Tapped Delay Line (TDL) with *L* paths. The channel matrix can be written as

$$\boldsymbol{H}(t) = \sum_{l=1}^{L} \boldsymbol{H}_l(t)\delta(t-\tau_l)(\mathrm{t})$$

where $\boldsymbol{H}_l(t)$ is the *l*-th path channel vector of Q elements, τ_l is the delay of the *l*-th path and $\delta()$ is the unit Dirac function (i.e., $\delta(t) = 1\ if\ t = 0, \delta(t) = 0$ otherwise). The model assumes a LoS propagation, hence the first path is Rice-distributed, while other paths are Rayleigh distributed. As a consequence, the channel matrix can be divided in two matrices: one fixed $\boldsymbol{H}_l^F(t)$ representing the LoS (non-variable) part, and one $\boldsymbol{H}_l^V(t)$ representing the NLoS (variable) part

$$\boldsymbol{H}_l(t) = \sqrt{\gamma_l}\left(\sqrt{\frac{\zeta}{\zeta+1}}\boldsymbol{H}_j^F(t) + \sqrt{\frac{\zeta}{\zeta+1}}\boldsymbol{H}_j^V(t)\right) =$$

$$\sqrt{\gamma_l}\left(\sqrt{\frac{\zeta}{\zeta+1}}\begin{bmatrix} e^{j\phi_1(t)} \\ e^{j\phi_2(t)} \\ \vdots \\ e^{j\phi_Q(t)} \end{bmatrix} + \sqrt{\frac{\zeta}{\zeta+1}}\begin{bmatrix} X_1(t) \\ X_2(t) \\ \vdots \\ X_Q(t) \end{bmatrix}\right)$$

where

- $X_i(t)$ is the NLoS channel coefficient at the *i*-th receiving antenna. X_i coefficients are modelled as correlated complex Gaussian random variables with zero mean and unitary variance;
- $\phi_i(t)$ represents the phase difference between the transmitter and the *i*-th receiving antenna;
- is the Rice factor;
- γ_l is the mean power of the *l*-th path at the receiver.

For what concerns the pathloss, it is modelled with a free- space loss break-point model with two slopes [16]: LF S, the free-space pathloss with slope value 2 up to dBP, the break- point distance, and a slope value of 3.5 afterwards

$$L(d) = \begin{cases} L_{FS}(d), & for\ d \le d_{BP} \\ L_{FS}(d_{BP}) + 35\log_{10}\left(\frac{d}{d_{BP}}\right), & for\ d > d_{BP} \end{cases}$$

where *5<d<100* is the distance (in meters) between transmitter and receiver. In addition to the usually used channel coefficients, this paper proposes ML-based approaches that exploit also channel attributes that are rarely considered, the delay and particularly, the spatial information (i.e., AoA) of the received signals. In actual systems each node experiences different delay values that can also vary in time, and the AoA depends on the position of the node. Since the power delay profile in the IEEE model is fixed for any channel realization, we have integrated the IEEE model with the WINNER II [16] channel model. In [16] the *l*-th path of each channel realization is characterized by a random delay whose average value τ_l^{avg} is exponentially distributed with parameter λ. Moreover, for taking into account the time-variability, the delay of each path, τ_l, is uniformly distributed around the mean delay τ_l^{avg} with variance $\sigma_\tau^2 = 1/\lambda$. Values have been taken from B3 model in [16]. Similarly, the AoA of the signal's paths are randomly distributed around a mean value μ that is given by the geometrical angle connecting the IoT node with the AP. The AoA is a Gaussian-distributed random value, $N(\mu, \sigma_{AoA}^2)$, with variance σ_{AoA}^2 from the B3 model in [16].

III. Proposed Authentication/Spoofing Detection Framework

In this paper we propose the use of the WF (the full title) approach to authenticate legitimate nodes and detect rogue devices using different ML approaches. Let us assume that the IoT network is composed by a transmitting node (Alice) and one sink node (Bob) which performs node authentication. The authentication/detection framework proposed here works in two phases:

- *Phase I*: Bob identifies Alice by means a traditional authentication protocol and collects a set with size n of data (referred as training dataset in what follows) received from Alice in order to extract her WF and train the ML algorithm.
- *Phase II*: Bob receives a message without assurance that it comes from Alice, hence, he tries to verify its authenticity by extracting the WF attributes from the signal and it feeds them to the ML anomaly detection algorithm. A positive result of the ML algorithm means that a match of the WF extracted from the message against the WF acquired during the Phase I is found, so the sender of the message is considered legitimate. Conversely, a negative result implies a message rejection and consequent countermeasures, e.g., a new Phase I authentication of the sender. It is important to stress that phase II allows a continuous authentication of the IoT nodes, without any resource burden for the IoT nodes since all operations are performed by the sink node. So, this approach is suitable for resource-constrained IoT nodes.

The WF method operates by extracting multiple attributes from the channel between sender and receiver: (1) the Received Signal Strength (RSS), (2) the AoA of the main path, (3) the maximum path delay, and (4) the signal energy. In what follows the ML-based authentication/anomaly detection schemes that have been considered here are described. In particular, their *one class* version has been considered: they distinguish only one class (Alice) and everything else is considered an anomaly (Eve). These schemes do not require any knowledge about Eve, they can operate in absence of negative class training samples (i.e., without collecting samples from Eve). In general, these kinds of schemes operate by defining a

decision test around the positive class (Alice), for separating and identifying new fingerprints as legitimate or not. The classification is based on two parameters: the *distance* between the test element (i.e., the element to be classified) and the training dataset characterizing the legitimate node (i.e., data used for training during Phase I), and a *threshold*. A sample is positively classified if the distance is lower than the threshold, otherwise it is classified as negative (i.e., an anomaly). Different algorithms define these parameters differently. The performance of the algorithms has been evaluated in terms of

- *True negative rate* is the ratio between the number of anomalous samples correctly detected and the total number of anomalous samples. It represents the probability of correctly detecting an anomaly (Eve);
- *False negative rate* is the ratio between the number of legitimate samples mistaken for anomalies and the total number of legitimate samples. It represents the probability of mistaking an authorized node (and blocking it), i.e., Alice is identified as a malicious node;
- *Balanced Accuracy BA* is the average between the true positive rate (i.e., the probability that a legitimate sample is correctly identified) and the true negative rate.

OC-k(j)NN is an authentication/anomaly detection algorithm derived from the k-NN classification algorithm [17] that selects the class of a sample to be tested as the most frequent among its k nearest neighbours. OC-*k(j)*NN algorithm is adapted to a single class problem: first the k nearest neighbours, $\{y_1, \cdots, y_k\}$, of the test element x, are found in the training dataset, then the j nearest neighbors, $\{z_{i1}, \cdots, z_{ij}\}$, for each of the first k neighbors (i.e., $i = 1, \cdots, k$) are found in the same dataset . The average Euclidean distances, $\bar{D}_{xy}$ between *x* and its *k* nearest neighbors, and $\bar{D}_{yz}$ between those *k* neighbors and their own *j* closest neighbors are calculated. The element to be test, *x* is considered an anomaly if $\frac{\bar{D}_{xy}}{\bar{D}_{yz}} > 1$. Indeed, if *x* is an anomaly, it likely lays in a region where the samples are less clustered, so the distance of *x* form its closest neighbours (belonging to the training dataset) is on average greater than the distance of between those neighbours and their own closest neighbours (also belonging to the training dataset). In terms of complexity, there is no need of a training phase for OC-*k(j)*NN. During the test phase, assuming a number of extracted features *p*, and a training dataset size *n*, the algorithm calculates *k* times the distance of an element to be tested to every point in the training dataset extracting every time that at minimum distance ($O(nkp)$). Then for the *k* selected neighbours it calculates *j* times distances to other points in the dataset extracting every time the element at minimum distance ($O(kjnp)$). Then the algorithm calculates the ratio between the average distances. Neglecting the complexity for calculating the distances the complexity is $O(kjnp)$.

OC-SVM is based on SVM algorithms that use a non-linear function (kernel) to map input data into a space withhigher dimensions named the feature space, and that find decision boundaries to separate classes. OC-SVM has only one class, the boundary is decided using the available training dataset, and any new data that lies outside that boundary is classified as an anomaly. We consider the solution that uses a hyperplane (a plane in m-dimensions) for the decision boundary [18]. During the training phase, elements of the dataset are projected in the feature space using a Gaussian kernel and then they are separated from the origin using a hyperplane minimizing the distance of the hyperplane from the origin. A parameter $\nu \in [0, 1]$ is used as upper bound for the fraction of elements of the kernel transformed training dataset that lies outside the hyperplane, so that a low value of ν means that a few outliers are allowed, and the hyperplane is closer to the origin. Moreover, ν represents the lower bound for the number of support vectors that are critical elements of the dataset that define the decision boundary and are used to calculate the distances. SVM is a convex quadratic programming problem with linear constraints. Training complexity of non-linear SVM is generally between $O(n^2)$ and $O(n^3)$ depending on the implementation. The test complexity depends on the used kernel function and the number of support vectors, s, since the kernel function must be computed for each support vector. Hence, the complexity is $O(spf)$ where f is the complexity of the kernel function.

iForest is an anomaly detection algorithm belonging to decision tree algorithms [19], it is based on a ensemble random binary trees, called isolation trees (*iTree*s). During the training, *T* different iTrees are built by splitting the dataset into sub-sets until each partition has only one element or a multiple of that same element. When the size of training dataset, n, is big a sub-set of the whole dataset is used with dimension $\psi = min(n, 256)$. iTrees are created by successively splitting the resulting sub-set at each step, randomly selecting an attribute and a value in its range so that the split generates two complementary sub-sets. The path length, $h(x)$, is defined as the number of nodes traversed through the iTree to reach the leaf containing *x*. The anomaly score is then calculated as $s(x, n) = 2^{-\frac{E[h(x)]}{C(n)}}$ using the path length averaged on all iTrees, $E[h(x)]$, normalized to the average path length $c(n)$ of an unsuccessful search in a binary search tree built over a training dataset of n elements [19]. It is expected that features of an anomalous element differ significantly from training dataset elements, in particular an anomaly should have a path length shorter than the average. The anomaly score is compared with a threshold $\rho \in [0, 1]$: values over the threshold are classified as anomalies. During the training stage, *T* iTrees are built by recursively splitting the dataset of size ψ. The complexity of the training is $O(T\psi log\psi)$. The anomaly detection complexity for a single element is $O(Tlog\psi)$.

OC-kmeans is a modified version of the *k*-means clustering algorithm [20] that aims at dividing a given dataset into k clusters where each element is closer to the

center of its cluster than to the center of other clusters. This is achieved through an iterative technique whereby the clustering operation is performed several times, using the resulting centres from each previous iteration as a starting point for the next one. We use here a modified version of the *k*-means algorithm to achieve anomaly detection [21]. This is done by dividing elements of the dataset in two clusters (i.e., *k = 2*). Then the average distance $\overline{D}$ between the two clusters' centers is calculated and used as threshold for the following decision test. During testing operations, received samples and training dataset are clustered in two clusters and the resulting distance V between the two clusters is compared with $\alpha\overline{D}$, where α is a scaling factor. If $V \leqslant \alpha\overline{D}$ a positive result is assumed, i.e., the received samples belong to Alice, otherwise an anomaly is detected. *k*-means shows complexity $O(npk)$ for each iteration: for each element of the dataset the distance from the k centroids is calculated using a vector of dimension p. In the anomaly detection implementation described before, both training and test phases are based on a dataset clustering with $k = 2$, then distances among clusters are evaluated and compared. Hence, the complexity for both phases is $O(2npI)$ where I is the number of iterations of the algorithm.

IV. Numerical Results

Performance of the previously described ML-based anomaly detection methods are presented and compared. Numerical results have been derived by means of simulations. To have results not depending on a single specific dataset (i.e., a specific position distribution of nodes in the area), results from multiple datasets have been averaged. For each dataset, Eve and Alice are randomly placed with a uniform distribution in a square area A = *20 × 20 m*, with the sink node in the centre. It is assumed that Alice's signal is received with a Signal-to-Noise Ratio (SNR) of 10 dB, while the SNR of Eve is consequently calculated considering its position in the area. The channel model and the probability distribution of the channel attributes have been described in Sec. II, taking into account also their time-variability due to the scatterers' movement up to 5 km/h. The carrier frequency is 5.25 GHz with a bandwidth of 80 MHz. Firs of all, we have evaluated the impact of different parameters' settings on detection performance of the algorithms for selecting the optimum ones. Moreover, the impact of the training dataset size is evaluated. The OC-*k(j)*NN algorithm depends on two parameters *k* and *j*. Fig.1 (a) shows the balanced accuracy (BA) vs the parameter k for different values of *j*. Training dataset size is fixed at *n = 200* samples. The algorithm behaves better when there is a higher unbalance between the two parameters with $k < j$. Indeed, performance increases as $\frac{k}{j}$ decreases up to certain point, then benefits tend to disappear. In particular, values of $k = [1, 5]$ are those that provide the best results with $j = [10, 20]$. With equal k/j, using a lower value of k gives almost the same performance but with lower complexity (for example the point $k/j = 5/10$ is similar to the value for $k/j = 10/20$ but the first one has lower complexity.) These results are confirmed by Table I where the BA is presented for different values of the training dataset size, *n*, for values of k and j in their best ranges ($k = 1, 5$ and $j = 10, 20$). We can see that better performance is achieved with $k/j = 1/20$ and a training dataset with limited size ($n \in [50 - 100]$), evidenced in the gray table part. OC-SVM requires to set the parameter ν that represents an upper bound for the fraction of outliers of the training dataset and a lower bound for the number of supporting vectors. Fig. 1(b) shows BA vs *ν* for different values of the training dataset size, n. We can see that as n increases, BA performance significantly increases (up to *n = 500*) and the impact of ν decreases. Conversely, for small n values it is preferable to work with higher values of ν, since a higher number of support vectors improves the detection accuracy. However, the value of ν should be limited in order to limit the computational complexity. Previous considerations can be drawn also from Table II where BA is reported for different values of n and a selected range of values of $\nu \in [0.05, 0.5]$.

TABLE I. OC-K(J)NN: BA VS TRAINING DATASET SIZE *N*

	1st # neighbours			
	k=1		*k=5*	
	2nd # neighbours			
Training dataset (n)	*j=10*	*j=20*	*j=10*	*j=20*
30	0.7931	0.9286	0.9829	0.5665
50	0.9772	0.9903	0.8910	0.9610
100	0.9749	0.9911	0.8783	0.9534
200	0.9736	0.9897	0.8766	0.9466
300	0.9746	0.9898	0.8778	0.9442

TABLE II. OC-SVM: BA VS TRAINING DATASET SIZE N

Training dataset (n)	**Upper bound elements outside the hyperplane *ν*)**					
	0.05	*0.10*	*0.20*	*0.30*	*0.40*	*0.50*
100	0.9046	0.9475	0.9852	0.9880	0.9889	0.9892
300	0.9411	0.9840	0.9892	0.9902	0.9903	0.9904
400	0.9918	0.9930	0.9932	0.9932	0.9928	0.9925
500	0.9928	0.9939	0.9937	0.9936	0.9933	0.9927

We can see that best performance, highlighted in gray in the table, is achieved with a training dataset size around *n = 400* (higher values do not yield relevant benefits), also maintaining a low value of $\nu = 0.05$. A reduced value of $n = 300$ could be selected using a value of $\nu \geq 0.3$, that means the time complexity decreases during the training phase but increases in the running phase. iForest requires two parameters, the number of trees T and the threshold $\rho \in [0, 1]$. In Fig. 1c the BA vs ρ for different values of *T* and *n = 200* is shown. Curves show that the optimal threshold value is around $\rho = 0.6$ for all values of *T*. Moreover, there is not a significant benefit in increasing the number of trees over $T = 10$. Indeed, BS performance remains almost constant, but the time complexity increases. The impact of the training dataset size is shown in Table III where the BA for different values on n is reported varying parameters ρ and T in their optimal ranges. Again, we can see that the number of trees does not significantly affects the performance, while increasing *n* up to *n = 300* leads to an improvement, then the performance remains almost constant, thus a further increase would lead only a complexity increase. Best

values are achieved with $\rho = 0.6, n = 200 \div 300$ and T around *100*. OC-*k*means algorithm compares clusters' distances using a weight factor α. Fig. 1(d) shows the BA vs α for different values of n. Best accuracy is achieved with a limited training dataset size, indeed there is a benefit increasing n up to *20* but for higher values performance worsens. Also for what concerns the value of α, an increase is beneficial up to a certain value then performance decreases. This worsening is more evident when larger training dataset are considered. The best value of α varies with n, particularly, it increases as n decreases. This can be seen also from Table IV where BA for different values of n is reported for α in its best range $[1.1, 1.5]$. The grey part of the table highlights that the best accuracy is achieved with $n = 20$ and $\alpha = 1.4 \div 1.5$. The low value of n is beneficial also in terms of time complexity.

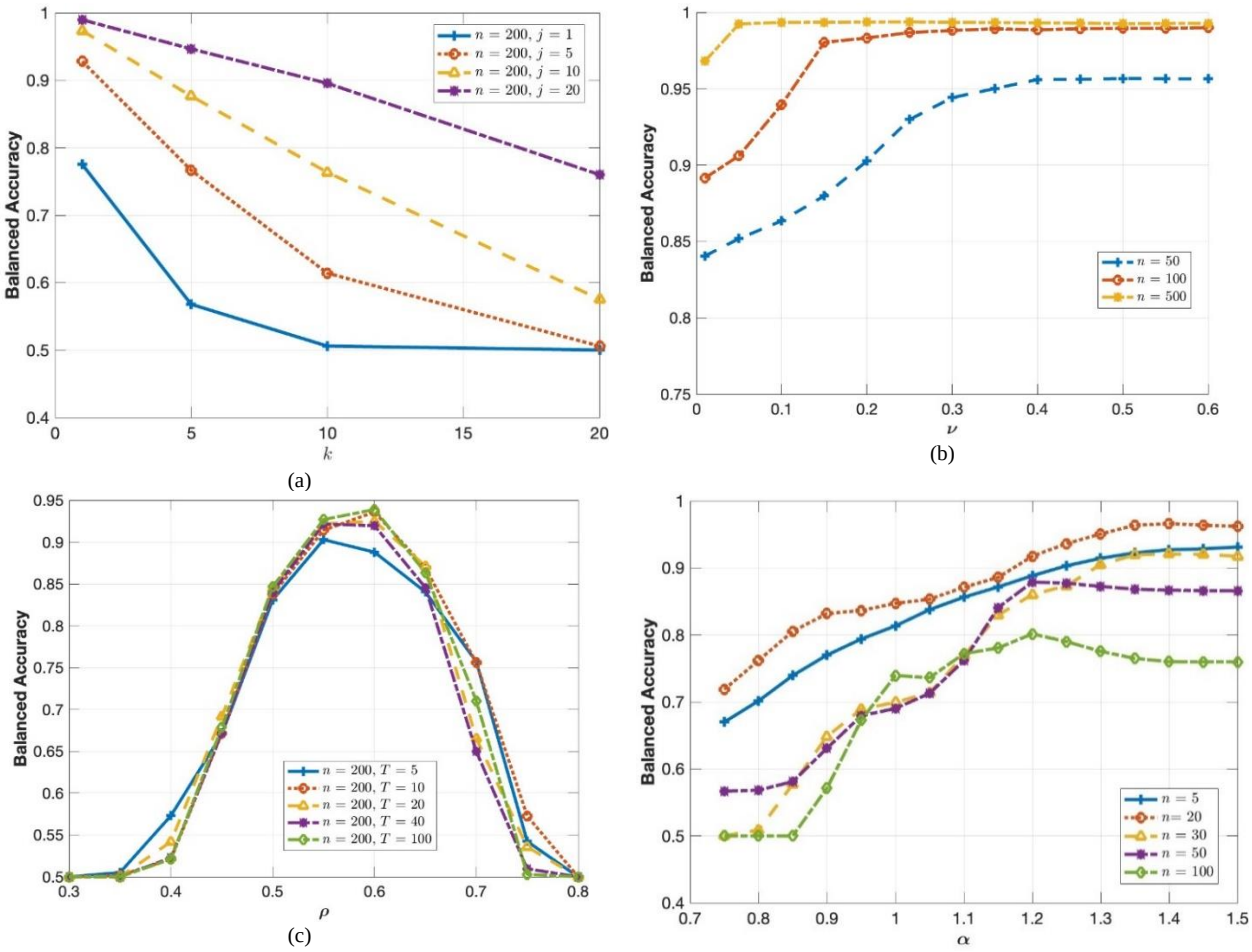

Fig. 1. Balanced Accuracy vs algorithms' parameters: (a) **OC-k(j)NN**. BA vs parameter k varying j, (b) **OC-SVM**. BA vs parameter ν varying n, (c) **iForest**. BA vs threshold ρ varying T, (d) **k-means**. BA vs distance weight α varying n

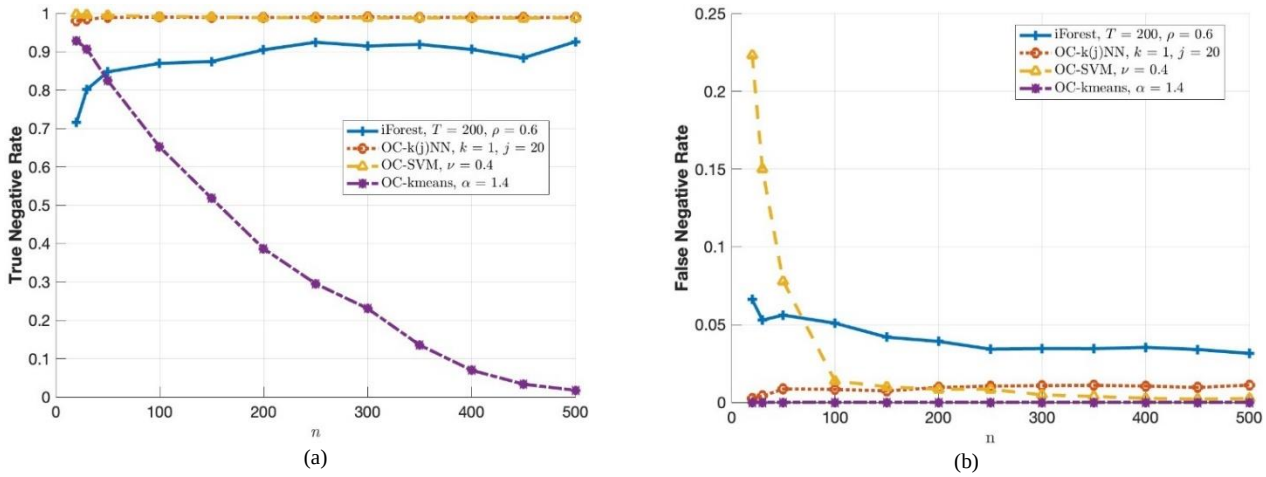

Fig. 2. Eve detection rate vs training dataset size: (a) True Negative Rate, (b) False Negative Rate

TABLE III. IFOREST: BA VS TRAINING DATASET SIZE N

	Threshold													
	$\rho = 0.55$							$\rho = 0.6$						
	# of Threes (T)													
Training dataset (n)	5	10	20	40	100	120	140	5	10	20	40	100	120	140
50	0.8413	0.8559	0.8610	0.8746	0.8816	0.8881	0.8854	0.8786	0.8863	0.8941	0.8828	0.8924	0.8878	0.8973
100	0.8384	0.8551	0.8992	0.9069	0.9075	0.9110	0.9095	0.8663	0.8867	0.9025	0.8978	0.9048	0.9097	0.9103
200	0.9028	0.9141	0.9219	0.9214	0.9269	0.9289	0.9241	0.8879	0.9359	0.9248	0.9196	0.9388	0.9198	0.9306
300	0.9021	0.9178	0.9255	0.9277	0.9289	0.9355	0.9304	0.9076	0.9159	0.9217	0.9333	0.9404	0.9414	0.9406
400	0.8993	0.9101	0.9273	0.9298	0.9344	0.9319	0.9326	0.9199	0.9309	0.9336	0.9290	0.9307	0.9405	0.9321

From previous results, we can see that OC-*k(j)*NN and OC-SVM are those achieving the best BA performance *(~ 99%)*, while OC-*k*means and iForest reach *96%* and *94%*, respectively. It is also interesting comparing the performance of different ML algorithms when their optimal parameters are used. Fig. 2 shows the true negative rate (*tnr*) and the false negative rate (*fnr*), that are the correct and false detection of Eve, for different algorithms. In terms of *fnr*, results confirm that OC-SVM and OC-*k(j)*NN achieve the highest values that are quite independent on the training dataset size. For what concerns the *fnr* instead, OC-*k(j)*NN achieves values almost constant with *n* but that do not go to zero, OC-SVM has values of *fnr* that depend on the training dataset size and go to zero around *n=400*. OC-*k*means presents good performance in terms of *fnr* while *tnr* strongly depends on *n* and reaches maximum values around *94%*. iForest presents worst performance, and even increasing n there is a floor.

TABLE IV. OC-KMEANS: BA VS TRAINING DATASET SIZE *N*

	Weight (α)				
Training dataset (n)	1.1	1.2	1.3	1.4	1.5
5	0.8641	0.8799	0.9102	0.9237	0.9272
20	0.8710	0.9173	0.9506	0.9661	0.9642
30	0.7687	0.8601	0.9050	0.9207	0.9173
50	0.7613	0.8786	0.8721	0.8666	0.8655
100	0.7718	0.8012	0.7753	0.7598	0.7595

V. CONCLUSIONS

This paper presented a ML-based PLA framework for identifying IoT nodes belonging to a WSN and detecting potential rogue devices trying to gain an unauthorized access. The identification is based on WF of the received signal, here characterized by various channel attributes. Different ML approaches have been evaluated and compared in terms of different metrics. OC-SVM and OC-*k(j)*NN resulted to be the algorithms providing best performance, even if OC-*k(j)*NN presents a higher *fn*r. On the other side OC-*k(j)*NN has the advantage of a reduced time complexity compared to OC-SVM, it does not require training, achieves good performance with a limited training dataset $n \in [50,100]$ and has linear complexity with *n*. On the contrary, OC-SVM has a training complexity that exponentially increases with *n* and for having low *fnr* the training dataset should be big. The last two algorithms have limited complexity but present a significant performance worsening, especially the iForest. Although this paper does not include experimental activities, it still provides original contributions, as reported before. Field tests are planned in our scheduled future works to validate the findings reported in this paper.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

AS, DM and LM conducted the research; AS analyzed the data; DM and LM wrote the paper; all authors had approved the final version.

FUNDING

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of Next Generation EU, partnership on "Telecommunications of the Future" (PE0000001 - program "RESTART").

REFERENCES

[1] J. Zhang, S. Rajendran, Z. Sun, R. Woods, and L. Hanzo, "Physical layer security for the internet of things: Authentication and key generation," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 92–98, 2019.

[2] L. Bai, L. Zhu, J. Liu, J. Choi, and W. Zhang, "Physical layer authentication in wireless communication networks: A survey," *J. Commun. Inf. Netw.*, vol. 5, no. 3, pp. 237–264, 2020.

[3] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Machine learning for the detection and identification of internet of things devices: A survey," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 298–320, 2022.

[4] J. Yoon, Y. Lee, and E. Hwang, "Machine learning-based physical layer authentication using neighborhood component analysis in mimo wireless communications," in *Proc. Int. Conf. Information and Commun. Tech. Convergence (ICTC)*, 2019, pp. 63–65.

[5] H. Fang, X. Wang, and L. Hanzo, "Learning-aided physical layer authentication as an intelligent process," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2260–2273, 2019.

[6] T. M. Hoang, N. M. Nguyen, and T. Q. Duong, "Detection of eavesdropping attack in uav-aided wireless systems: Unsupervised learning with one-class svm and k-means clustering," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 139–142, 2020.

[7] S. Kashani, F. Nait-Abdesselam, and A. Khokhar, "A channel-based authentication using machine learning for body sensor networks," *IEEE Globecom*, 2022, pp. 1103–1108.

[8] L. Senigagliesi, M. Baldi, and E. Gambi, "Comparison of statistical and machine learning techniques for physical layer authentication," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1506–1521, 2021.

[9] E. H. Enad and S. Younis, "Machine learning based decision stratigies for physical layer authentication in wireless systems," in *Proc. 2nd Annual Int Conf Information and Sciences (AiCIS)*, 2020, pp. 114–118.

[10] H. Fang, X. Wang, and S. Tomasin, "Machine learning for intelligent authentication in 5g and beyond wireless networks," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 55–61, 2019.

[11] L. Alhoraibi, D. Alghazzawi, R. Alhebshi, and O. B. J. Rabie, "Physical layer authentication in wireless networks-based machine learning approaches," *Sensors*, vol. 23, no. 4, p. 1814, Feb. 2023.
[12] A. Abdelaziz, R. Burton, F. Barickman, J. Martin, J. Weston, and C. E. Koksal, "Enhanced authentication based on angle of signal arrivals," *IEEE Trans. Vehic. Tech*., vol. 68, no. 5, pp. 4602–4614, 2019.
[13] J. Xiong and K. Jamieson, "Securearray: Improving wifi security with fine-grained physical-layer information," in *Proc. 19th Annual Int. Conf. Mobile Computing & Networking (MobiCom)*, 2013, pp. 441–452.
[14] J. Kermoal and et al., "A stochastic mimo radio channel model with experimental validation," *IEEE J. Sel. Areas Commun*., vol. 20, no. 6, pp. 1211–1226, 2002.
[15] V. Erceg *et al*., "Wireless lans indoor mimo wlantgn channel models," 2004.
[16] P. Kyosti, J. Meinila, L. Hentila, X. Zhao, T. Jamsa, C. Schneider,M. Narandzic, M. Milojevic, A. Hong, J. Ylitalo, V. M. Holappa, M. Alatossava, R. Bultitude, Y. Jong, and T. Rautiainen, "Winner II channel models," *IST-4-027756 WINNER II D1.1.2 V1.2*, vol. 2, 2008.
[17] S. S. Khan and A. Ahmad, "Relationship between variants of one-class nearest neighbors and creating their accurate ensembles," *IEEE Trans. Knowl. Data Eng*., vol. 30, no. 9, pp. 1796–1809, 2018.
[18] B. E. A. Scholkopf, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
[19] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.
[20] J. M. Queen *et al*., "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statistics and Probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.
[21] H. Liu, Y. Wang, J. Liu, J. Yang, and Y. Chen, "Practical user authentication leveraging channel state information (csi)," in *Proc. 9th ACM Symp. Information, Computer and Commun. Security*, 2014, pp. 389–400.