Spoofing Detection in the Physical Layer with Graph Neural Networks

Tien Ngoc Ha
Department of ICT, *University of Agder*
Grimstad, Norway
tien.n.ha@uia.no

Daniel Romero
Department of ICT, *University of Agder*
Grimstad, Norway
daniel.romero@uia.no

***Abstract*—In a spoofing attack, a malicious actor impersonates a legitimate user to access or manipulate data without authorization. The vulnerability of cryptographic security mechanisms to compromised user credentials motivates spoofing attack detection in the physical layer, which traditionally relied on channel features, such as the received signal strength (RSS) measured by spatially distributed receivers or access points. However, existing methods cannot effectively cope with the dynamic nature of channels, which change over time as a result of user mobility and other factors. To address this limitation, this work builds upon the intuition that the temporal pattern of changes in RSS features can be used to detect the presence of concurrent transmissions from multiple (possibly changing) locations, which in turn indicates the existence of an attack. Since a localization-based approach would require costly data collection and would suffer from low spatial resolution due to multipath, the proposed algorithm employs a deep neural network to construct a graph embedding of a sequence of RSS features that reflects changes in the propagation conditions. A graph neural network then classifies these embeddings to detect spoofing attacks. The effectiveness and robustness of the proposed scheme are corroborated by experiments with real-data.**

***Index Terms*—Graph neural networks, spoofing attack, physical layer security, deep learning, cybersecurity, wireless networks.**

I. Introduction

The prevalence of wireless communications has engendered a panoply of security threats, including the unauthorized interception of private data, disruptions to remote services, and user impersonation. Among these pernicious threats, spoofing attacks pose a particularly troublesome challenge since, in these attacks, malevolent actors intercept and manipulate data originally intended for legitimate users [1]–[4]. The detection and mitigation of these attacks are pivotal for data security. Although cryptographic techniques have traditionally been employed across various communication layers to fortify security, the potential access of attackers to the credentials of legitimate users introduces a serious vulnerability. Consequently, the research community has increasingly focused on detecting spoofing attacks in the physical layer.

For instance, [5]–[8] leverage transmitter hardware imperfections, such as carrier frequency offset (CFO), in-phase and quadrature (I/Q) offset, and I/Q imbalance, to verify user identity. Regrettably, these methodologies necessitate knowledge of the communication protocol and may prove ineffective in the face of environmental changes, such as fluctuations in temperature [6]. These constraints are somehow mitigated in [9]–[11], which rely on angle of arrival (AoA) and time difference of arrival (TDoA) features, and in [12], where a neural network is trained using signal-to-noise ratio (SNR) traces. Nonetheless, these approaches still demand synchronization and/or knowledge of the communication protocol. In contrast, techniques reliant on received signal strength (RSS) measurements do not require knowledge of the communication protocol or signal decoding, thus significantly augmenting their generality and applicability for detecting spoofing attempts [13]–[17]. The predominant approach in this context involves applying clustering primitives to RSS measurements collected by multiple receivers, such as the access points of a WiFi network [13], [18], [19]. By exploiting the dependence of RSS signatures on the transmitter locations, an attack is detected if transmissions with the same user identifier are found to originate at different locations. Consequently, this approach results in false alarms when the channel conditions change, as for example when a legitimate user moves.

To remedy this limitation, the key realization in this work is that it is possible to tell spoofing from motion and other effects by analyzing the temporal changes in RSS features. To illustrate this idea, consider a network that sequentially receives frames from locations denoted as A, B, C, and D. If all these locations are distinct, it is natural to ascribe these variations to the movement of the legitimate user. In contrast, if the received transmissions alternate between points A and B in a pattern such as A, B, A, B, A, B, etc., it is more likely that one user is transmitting from location A and another from location B, which indicates the presence of an attack. To the best of our knowledge, the work at hand is the first to exploit this kind of information.

To this end, this paper introduces a spoofing attack detection scheme where a graph embedding is constructed to capture the pattern of changes in RSS features over a sequence of frames. Then, a *graph neural network* (GNN) classifies such graph embeddings as either corresponding to an attack or to legitimate user activity, which may include user movement. The graph is constructed by utilizing a *position-change de-*

This research has been funded in part by the Research Council of Norway under IKTPLUSS grant 311994.

The code and pre-trained models are available at https://github.com/uiano/gnn_spoofing_detection.

tector (PCD) that determines whether a given pair of frames was transmitted from different locations. Since changes in the RSS measurements corresponding to different frames may be caused either by the movement of the transmitter or by the variability due to the finite number of samples used in the computation of these measurements, the PCD is designed as a deep neural network that detects position changes by implicitly learning the distribution of RSS estimates from signal samples. The proposed scheme can be readily deployed due to the simplicity of the procedure for collecting the required data set. Specifically, RSS features must be collected at different locations but those locations need not be recorded.

The rest of the paper is structured as follows. Sec. II formulates the problem. Sec. III presents the proposed spoofing detection scheme. Sec. IV presents an extensive performance evaluation using real data. Finally, Sec. V concludes the paper.

II. Problem Formulation

Let $\mathcal{X} \subset \mathbb{R}^3$ comprise the coordinates of all points in the spatial region of interest, where both legitimate users and attackers are located. A transmitter at $\boldsymbol{x} \in \mathcal{X}$, which can be the legitimate user or an attacker, sends a signal $s(t)$, where t denotes time. This signal, modeled as an unknown wide-sense stationary stochastic process, is received by N receivers, such as the access points or base stations of a wireless network. Let $h_n(\boldsymbol{x}, t)$ denote the impulse response of the channel between the transmitter and the nth receiver, which is assumed to be time-invariant over the duration of a frame. The received signal at the nth receiver is given by

$$r_n(\boldsymbol{x}, t) = h_n(\boldsymbol{x}, t) * s(t) + z_n(t), \tag{1}$$

where $*$ denotes convolution and $z_n(t)$ is additive white Gaussian noise (AWGN) with variance σ^2 and independent of $s(t)$. Thus, one can define the *received signal strength* (RSS) $f_n(\boldsymbol{x}) := \mathbb{E}|r_n(\boldsymbol{x}, t)|^2$, where $\mathbb{E}$ denotes expectation.

To estimate the RSS of a frame received by the nth receiver, consider a set of K samples $\mathcal{K}_n := \{r_n(\boldsymbol{x}, t+kT)\}_{k=1}^{K}$, where T is the sampling period and t is the time when the frame begins. The RSS can be estimated as

$$\hat{f}_n(\boldsymbol{x}) := \frac{1}{K} \sum_{k=1}^{K} |r_n(\boldsymbol{x}, t + kT)|^2. \tag{2}$$

If $r_n(\boldsymbol{x}, kT)$ is ergodic for each $\boldsymbol{x}$, it follows that $\hat{f}_n(\boldsymbol{x})$ converges to $f_n(\boldsymbol{x})$ as $K \to \infty$.

For notational convenience, the RSS values estimated by all receivers are collected into a *feature vector*, generically represented by $\hat{\boldsymbol{f}}(\boldsymbol{x}) := [\hat{f}_1(\boldsymbol{x}), \ldots, \hat{f}_N(\boldsymbol{x})]^\top$. Note that since K is finite, measuring the RSS I times for location $\boldsymbol{x}$ yields I different estimates of $\boldsymbol{f}(\boldsymbol{x}) := [f_1(\boldsymbol{x}), \ldots, f_N(\boldsymbol{x})]^\top$. These estimates will be denoted as $\hat{\boldsymbol{f}}^{(i)}(\boldsymbol{x}), i = 1, \ldots, I$.

To introduce the notation for frame sequences, let $\boldsymbol{x}[j]$ denote the location of the user that transmits the jth frame at the moment of transmitting that frame and let $\hat{\boldsymbol{f}}[j] := \hat{\boldsymbol{f}}(\boldsymbol{x}[j])$. The feature vectors corresponding to a sequence of J frames are collected into matrix $\hat{\boldsymbol{F}} := [\hat{\boldsymbol{f}}[1], \ldots, \hat{\boldsymbol{f}}[J]]$. Out of these

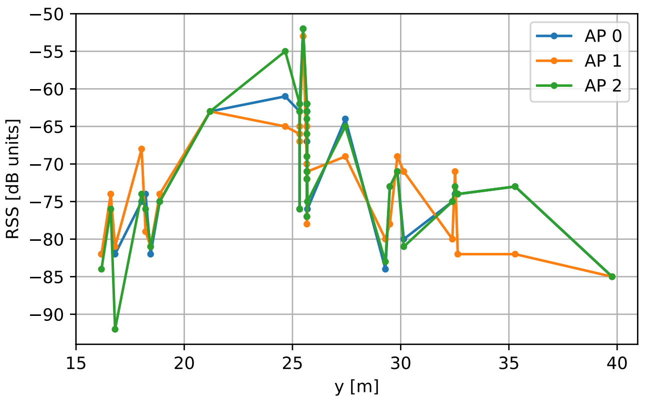

Fig. 1: RSS measurements along a line vs. their y-coordinate. It is observed that small variations in the y-coordinate often result in larger RSS changes than large variations in the y-coordinate. For example, the difference between $y = 24$ and $y = 29$ is around 24 dB, whereas the difference between $y = 16$ and $y = 40$ is less than 5 dB. This is caused mainly by multipath and suggests that accurately estimating the position from RSS measurements is not generally possible.

J frames, J_L belong to the legitimate user and J_A to the attacker, where $J = J_L + J_A$. The set of indices of the frames belonging to the legitimate user is represented by $\mathcal{J}_{\text{L}} \subset \{1, \ldots, J_L\}$ whereas the set of indices of the frames belonging to the attacker is represented by $\mathcal{J}_{\text{A}} \subset \{1, \ldots, J_A\}$.

Given $\hat{\boldsymbol{F}}$, the problem is to decide between the following hypotheses:

$$\begin{cases} \mathcal{H}_0 : \mathcal{J}_{\text{A}} = \emptyset \\ \mathcal{H}_1 : \mathcal{J}_{\text{A}} \neq \emptyset. \end{cases} \tag{3}$$

To this end, a dataset comprising the feature vectors $\mathcal{D} := \{\hat{\boldsymbol{f}}^{(i)}(\boldsymbol{x}_m), m = 1, \ldots, M, i = 1, \ldots, I\}$ is given, where M is the number of distinct measurement locations, i.e. $\boldsymbol{x}_m \neq \boldsymbol{x}_{m'}$ $\forall m \neq m'$.

III. Spoofing Detection from RSS Features

Evidently, if the time between consecutive frames in $\hat{\boldsymbol{F}}$ is too long, then the vectors $\hat{\boldsymbol{f}}[j]$ may originate at highly distant locations due to user movement, even in the absence of attacks. As a result, $\hat{\boldsymbol{f}}[j]$ may be highly different from $\hat{\boldsymbol{f}}[j-1]$ and $\hat{\boldsymbol{f}}[j+1]$. Since this would also be the case in the presence of an attack, solving problem (3) becomes challenging as the distributions of $\hat{\boldsymbol{F}}$ under both hypotheses are highly similar. Therefore, it becomes imperative to introduce the following assumption:

Assumption 1: The frame rate is high relative to the speed of the users.

In other words, if $\hat{\boldsymbol{f}}[j]$ and $\hat{\boldsymbol{f}}[j+1]$ correspond to the same user, they will be reasonably similar.

In view of Assumption 1, one could consider a strategy to tackle problem (3) where the locations $\boldsymbol{x}[j]$ are first estimated based on $\hat{\boldsymbol{f}}[j]$, $j = 1, \ldots, J$, and an attack is declared if multiple transmissions are concurrently received from distant (possibly moving) users. However, this approach is not viable first because the data given in the problem formulation of Sec. II does not allow a reasonably accurate localization of the transmitters. Indeed, if the locations $\boldsymbol{x}[j]$ associated with the vectors in $\mathcal{D}$ were given, one could attempt to estimate the locations associated with the frames in $\hat{\boldsymbol{F}}$, for instance via fingerprinting-based localization [20], [21]. However, the error of such approaches is typically in the order of 10 m in indoor environments (see e.g. [20]), which would hinder detecting attacks where the attacker is relatively near the legitimate user. Fig. 1 illustrates why this is the case. Besides, collecting a data set where the positions of the measurement locations need to be recorded is highly costly since it would generally require the deployment of an auxiliary localization system, the use a mobile robot, or to manually measure the spatial coordinates of all measurement locations.

For this reason, the proposed scheme does not attempt to estimate the transmitter locations. Instead, it exploits the pattern of dissimilarities between the feature vectors $\hat{\boldsymbol{f}}[j]$. This is accomplished in two steps: First, each pair of vectors $(\hat{\boldsymbol{f}}[j], \hat{\boldsymbol{f}}[j'])$ is compared as described in Sec. III-A. Given these comparisons, a decision is made on the presence of an attack based on a graph embedding, as described in Sec. III-B.

A. Position-change Detection

This section presents a PCD, which is a detector that determines whether two given frames where transmitted from the same location. Specifically, given two feature vectors $\hat{\boldsymbol{f}}[j]$ and $\hat{\boldsymbol{f}}[j']$ respectively corresponding to (possibly equal) locations $\boldsymbol{x}[j]$ and $\boldsymbol{x}[j']$, the goal is to distinguish between the following hypotheses:

$$\begin{cases} \mathcal{H}_0^{\text{PCD}} : \boldsymbol{x}[j] = \boldsymbol{x}[j'] \\ \mathcal{H}_1^{\text{PCD}} : \boldsymbol{x}[j] \neq \boldsymbol{x}[j']. \end{cases} \tag{4}$$

A PCD is a function that maps a pair of feature vectors to a hypothesis, i.e., $d^{\text{PCD}} : \mathbb{R}^N \times \mathbb{R}^N \to \{\mathcal{H}_0^{\text{PCD}}, \mathcal{H}_1^{\text{PCD}}\}$.

To properly address (4), it is useful to consider the components behind the dissimilarity between $\hat{\boldsymbol{f}}[j]$ and $\hat{\boldsymbol{f}}[j']$. First, two feature vectors $\hat{\boldsymbol{f}}[j]$ and $\hat{\boldsymbol{f}}[j']$ are naturally different because K is finite, even when $\boldsymbol{x}[j] = \boldsymbol{x}[j']$. Second, $\hat{\boldsymbol{f}}[j]$ and $\hat{\boldsymbol{f}}[j']$ will be different when $\boldsymbol{x}[j] \neq \boldsymbol{x}[j']$ because of the different propagation phenomena undergone by the signals propagating from either location to the receivers. This includes effects such as path loss, shadowing, and fading. The latter is caused by multipath and dominates in indoor environments; see for example Fig. 1. The complexity of these phenomena calls for a PCD that learns to solve (4) in a data-driven fashion. To this end, in this work, d^{PCD} is implemented using a DNN, as described next.

1) Architecture: Following standard practice, the detector is designed to decide $\mathcal{H}_1^{\text{PCD}}$ when a detection statistic $T^{\text{PCD}} : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ exceeds a predefined threshold, and $\mathcal{H}_0^{\text{PCD}}$ otherwise. In this way, the problem of designing d^{PCD} becomes that of designing a function T^{PCD}.

In principle, this function could be directly implemented as a DNN. However, such a simple approach would result in a non-commutative T^{PCD}, that is, $T^{\text{PCD}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')$ will generally differ from $T^{\text{PCD}}(\hat{\boldsymbol{f}}', \hat{\boldsymbol{f}})$, which is clearly undesirable. To remedy this issue, a symmetrization technique will be adopted. Specifically, T^{PCD} will be implemented based on an auxiliary function $\tilde{T}^{\text{PCD}}$ by setting $T^{\text{PCD}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') := (\tilde{T}^{\text{PCD}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') + \tilde{T}^{\text{PCD}}(\hat{\boldsymbol{f}}', \hat{\boldsymbol{f}}))/2$. Observe that this implies that $T^{\text{PCD}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') = T^{\text{PCD}}(\hat{\boldsymbol{f}}', \hat{\boldsymbol{f}})$ regardless of $\tilde{T}^{\text{PCD}}$. Thereby, $\tilde{T}^{\text{PCD}}$ can be safely implemented as a DNN.

The architecture of the subnetwork $\tilde{T}^{\text{PCD}}$ is detailed next. Since $\tilde{T}^{\text{PCD}}$ is concerned with dissimilarities, it is natural to include an initial non-trainable layer that yields $10\log_{10}[\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}', \hat{\boldsymbol{f}} - \hat{\boldsymbol{f}}']$ when the input to the network is $[\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}']$. This facilitates learning from moderate-sized datasets. This layer is followed by three hidden fully-connected layers with 512 neurons and leaky ReLU activations [22]. The output layer contains a single neuron with a linear activation.

2) Data set: To train T^{PCD}, a dataset comprising pairs of vectors from $\mathcal{D}$ is constructed. The feature vectors in P of these pairs correspond to the same transmitter location. In the remaining P pairs, they correspond to different transmitter locations.

Specifically, the pairs of the first kind are generated for $p = 1, \ldots, P$ by first drawing m_p uniformly at random from the set $\{1, \ldots, M\}$. Then, i_p and i_p' are drawn uniformly at random without replacement from $\{1, \ldots, I\}$. This process results in the set $\mathcal{D}_{\text{s}} := \{(\hat{\boldsymbol{f}}^{(i_p)}(\boldsymbol{x}_{m_p}), \hat{\boldsymbol{f}}^{(i_p')}(\boldsymbol{x}_{m_p})),\ p = 1, \ldots, P\} \subset \mathbb{R}^N \times \mathbb{R}^N$. To generate the pairs of the second kind, draw m_p and m_p' uniformly at random without replacement from the set $\{1, \ldots, M\}$ for $p = 1, \ldots, P$. Drawing i_p and i_p' as before yields $\mathcal{D}_{\text{d}} := \{(\hat{\boldsymbol{f}}^{(i_p)}(\boldsymbol{x}_{m_p}), \hat{\boldsymbol{f}}^{(i_p')}(\boldsymbol{x}_{m_{p'}})),\ p = 1, \ldots, P\} \subset \mathbb{R}^N \times \mathbb{R}^N$. The DNN can then be trained on the dataset $\mathcal{D}_{\text{s}} \cup \mathcal{D}_{\text{d}}$.

Recall from Sec. II that obtaining $\mathcal{D}$ involves collecting the I RSS estimates $\hat{\boldsymbol{f}}^{(i)}(\boldsymbol{x}_m)$, $i = 1, \ldots, I$, for each of the M locations $\boldsymbol{x}_m$, $m = 1, \ldots, M$. A simpler approach may be to collect a single estimate with a large K so that it approximately equals $\boldsymbol{f}(\boldsymbol{x}_m)$ and then generate the $\hat{\boldsymbol{f}}^{(i)}(\boldsymbol{x}_m)$ synthetically. The procedure is described next for the case where $r_n(\boldsymbol{x}, t)$ is approximately Gaussian distributed, which would be the case e.g. if $s(t)$ is an orthogonal frequency division multiplexing (OFDM) signal; see e.g. [23].

To this end, express $r_n(\boldsymbol{x}, t)$ as $r_n(\boldsymbol{x}, t) = f_n(\boldsymbol{x})\epsilon_n(\boldsymbol{x}, t)$, where $f_n(\boldsymbol{x})$ is the true RSS and $\epsilon_n(\boldsymbol{x}, t)$ is a circularly symmetric zero-mean Gaussian random variable with unit variance uncorrelated over t. Then, (2) becomes

$$\hat{f}_n(\boldsymbol{x}, t) = \frac{|f_n(\boldsymbol{x})|^2}{K} \sum_{k=1}^{K} |\epsilon_n(\boldsymbol{x}, t + kT)|^2 \tag{5a}$$

$$= \frac{|f_n(\boldsymbol{x})|^2}{2K} \sum_{k=1}^{K} \left(\left[\sqrt{2}\text{Re}\{\epsilon_n(\boldsymbol{x}, t + kT)\}\right]^2 + \left[\sqrt{2}\text{Im}\{\epsilon_n(\boldsymbol{x}, t + kT)\}\right]^2 \right). \tag{5b}$$

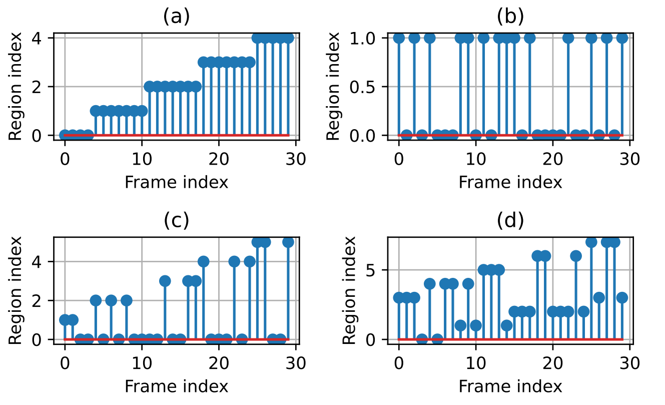

Fig. 2: Examples of region sequences without an attack ((a)) and with an attack ((b)-(d)).

It follows that $2K\hat{f}_n(\boldsymbol{x},t)/|f_n(\boldsymbol{x})|^2$ is a χ^2 random variable with $2K$ degrees of freedom. Thus, I estimates $\hat{f}_n^{(i)}(\boldsymbol{x},t)$, $i = 1,\ldots,I$, can be obtained by generating I realizations of such a χ^2 random variable.

3) Training: The DNN is trained using a binary cross-entropy loss function. Within the dataset, a subset comprising M_{val} validation points is reserved for validation, while the remainder $M_{\text{tr}} = M - M_{\text{val}}$ are used for training.

B. Graph Neural Network based Spoofing Detection

The decisions of the PCD for all pairs of frames will be used next to detect spoofing attacks. To intuitively understand why this is possible, recall from Sec. III-A that the PCD decides $\mathcal{H}_1^{\text{PCD}}$ when the dissimilarity between the given feature vectors $\boldsymbol{f}[j]$ and $\boldsymbol{f}[j']$ owes to the difference between the propagation phenomena experienced at $\boldsymbol{x}[j]$ and $\boldsymbol{x}[j']$. In other words, if $\boldsymbol{x}[j]$ and $\boldsymbol{x}[j']$ are so close that both points see similar propagation conditions to all receivers, the PCD decides $\mathcal{H}_0^{\text{PCD}}$. For didactical purposes, it is useful to split the space into propagation *regions* and assume that $d^{\text{PCD}}(\boldsymbol{f}[j],\boldsymbol{f}[j']) = \mathcal{H}_0^{\text{PCD}}$ if $\boldsymbol{x}[j]$ and $\boldsymbol{x}[j']$ belong to the same region and $d^{\text{PCD}}(\boldsymbol{f}[j],\boldsymbol{f}[j']) = \mathcal{H}_1^{\text{PCD}}$ otherwise. Using the decisions of the PCD, one can therefore assign each frame in the given sequence to a region.

This assignment is illustrated in Fig. 2, which sheds light into why it is possible to solve (3) using the decisions of the PCD. Fig. 2a shows an example where all frames are generated by a single moving user. Due to Assumption 1, groups of consecutive frames are declared by the PCD to belong to the same region. In turn, Fig. 2b depicts the case where two transmissions are concurrently taking place from different regions, which indicates the presence of an attack. Finally, Fig. 2c and Fig. 2d respectively correspond to the case where one or both of the concurrently transmitting users move.

It is important to note that the above considerations are provided to develop intuition, but in practice do not hold exactly. In particular, the decisions of the PCD will not generally be transitive, that is, it may hold that $d^{\text{PCD}}(\boldsymbol{f}[j],\boldsymbol{f}[j']) = \mathcal{H}_0^{\text{PCD}}$ and $d^{\text{PCD}}(\boldsymbol{f}[j'],\boldsymbol{f}[j'']) = \mathcal{H}_0^{\text{PCD}}$ but $d^{\text{PCD}}(\boldsymbol{f}[j],\boldsymbol{f}[j'']) = \mathcal{H}_1^{\text{PCD}}$. However, this may approximately hold. For this reason, it is useful to construct a graph $\mathcal{G}$ where the j-th node corresponds to $\boldsymbol{f}[j]$ and there is an edge between nodes j and j' if $d^{\text{PCD}}(\boldsymbol{f}[j],\boldsymbol{f}[j']) = \mathcal{H}_0^{\text{PCD}}$. Clearly, since d^{PCD} is commutative (cf. Sec. III-A1), this graph is undirected.

Clearly, if the space could be split into propagation regions, as discussed earlier, then this graph could be partitioned into one component per region and the presence of an attack would be characterized by an alternating pattern between components as in Fig. 2b-2d. However, since this is not exactly the case, it makes sense to train a GNN to detect attacks based on $\mathcal{G}$ in a data-driven fashion.

1) Architecture: A GNN [24] exploits the relation between node features and the graph topology by performing a sequence of message-passing steps or *layers*, where the features associated with a node at layer l depend on the features of that node and the neighboring nodes at layer $l-1$. Specifically, if $\boldsymbol{\phi}_\nu^{(l)}$ represents the features of node ν at layer l, then

$$\boldsymbol{\phi}_\nu^{(l)} = G_1^{(l)}\left(\boldsymbol{\phi}_\nu^{(l-1)}, \bigoplus_{\nu' \in \mathcal{N}_\nu} G_2^{(l)}(\boldsymbol{\phi}_\nu^{(l-1)}, \boldsymbol{\phi}_{\nu'}^{(l-1)})\right), \tag{6}$$

where $G_1^{(l)}$ and $G_2^{(l)}$ are conventional DNNs, $\mathcal{N}_\nu$ contains the set of neighbors of node ν, and $\bigoplus$ is an aggregation operator such as a summation or maximum operator. The output of the GNN can be computed by another aggregation operator applied to the concatenation of the vectors $\boldsymbol{\phi}_\nu^{(L)}$ for all ν, where L is the number of layers.

For the problem at hand, a test statistic is obtained with a GNN and then compared to a threshold to decide between $\mathcal{H}_0$ and $\mathcal{H}_1$. In the adopted architecture, $L = 3$ layers and functions $G_1^{(l)}$ and $G_2^{(l)}$ are implemented as single-layer fully-connected DNNs with 64 output neurons and ReLU activations. The operator $\bigoplus$ is a summation whereas the output of the GNN is obtained by averaging the features of all nodes at the last layer and applying a trainable affine transformation. Since the order of the nodes is relevant (cf. Fig. 2), the features in the first layer are set so that $\boldsymbol{\phi}_\nu^{(1)}$ equals the index of node ν.

2) Dataset: To train the GNN, realizations of $\mathcal{G}$ must be generated under both $\mathcal{H}_0$ and $\mathcal{H}_1$. This involves generating frame sequences $\hat{\boldsymbol{F}}$ under both hypotheses. Under $\mathcal{H}_0$, the trajectory $\boldsymbol{x}(t)$ of the (single) user is generated as follows. First, obtain the time duration of the frame sequence, given by JR_{J}, where R_{J} is the number of frames per second. The length $\Delta\boldsymbol{x}$ of the trajectory is therefore $\Delta\boldsymbol{x} = JR_{\text{J}}v$, where v is the speed of the user. Then, randomly draw a straight line segment of length $\Delta\boldsymbol{x}$ in $\mathcal{R}$. The trajectory is therefore $\boldsymbol{x}(t) = \tilde{\boldsymbol{x}} + \boldsymbol{d}vt$, where $\tilde{\boldsymbol{x}}$ is the starting point and $\boldsymbol{d}$ is the unit direction vector of the line. For each j, $\hat{\boldsymbol{f}}[j]$ is obtained by randomly selecting one of the vectors in $\mathcal{D}$ that correspond to the location that lies closest to $\boldsymbol{x}(j/R_{\text{J}})$.

Under $\mathcal{H}_1$, the frame sequences of both users are generated following the above procedure. Then, the frame sequence $\hat{\boldsymbol{f}}_1[0],\ldots,\hat{\boldsymbol{f}}_1[J-1]$ of user-1 is merged with the frame

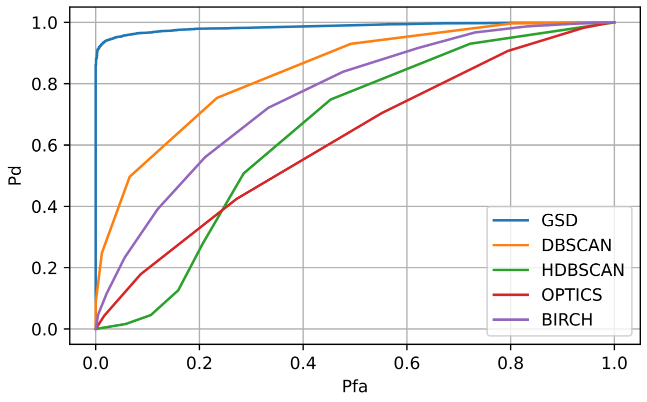

Fig. 3: ROC curves of the proposed algorithm and the benchmarks (10 frames/s, $J = 30$, $K = 150$, $N = 5$).

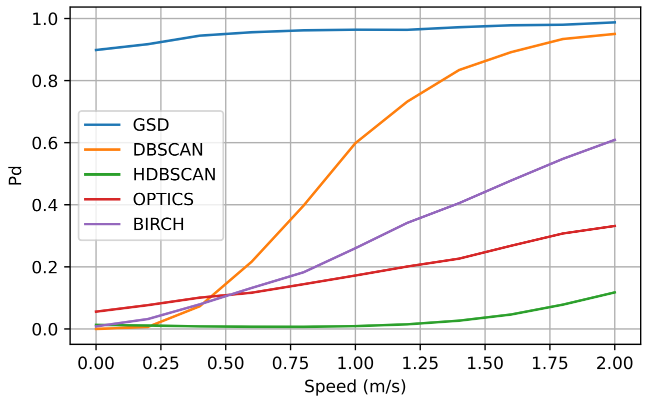

Fig. 4: Pd vs speed of the proposed algorithm and the benchmarks for a fixed probability of false alarm $P_{\rm FA}$ (10 frames/s, $J = 30$, $K = 150$, $N = 5$, $P_{\rm FA} = 0.1$).

sequence $\hat{\boldsymbol{f}}_2[0], \ldots, \hat{\boldsymbol{f}}_2[J-1]$ of user-2 into a sequence $\hat{\boldsymbol{f}}[0], \ldots, \hat{\boldsymbol{f}}[J-1]$ where either $\hat{\boldsymbol{f}}[j] = \hat{\boldsymbol{f}}_1[j]$ or $\hat{\boldsymbol{f}}[j] = \hat{\boldsymbol{f}}_2[j]$, both with probability 1/2 and independently along j.

3) Training: The GNN model is trained using a binary cross-entropy loss function.

IV. Performance Evaluation

This section assesses the performance of the proposed scheme using the dataset from [20], which contains RSS measurements of 992 WiFi access points at 4846 locations across 4 floors. To ensure a sufficient spatial density, only the measurements collected at $M = 648$ locations on the first floor are used. 20% of them are reserved for testing. Since each measurement location lies out of the range of most of the access points, the $N = 5$ access points that are measured at the greatest number of the selected locations are considered. The procedure described in Sec. III-A2 is then used to generate $I = 1000$ feature vectors for each location. A link to the code is provided on the first page.

The proposed algorithm, referred to as *GNN-based Spoofing Detection* (GSD), is compared against four benchmarks which, along the lines of [13], [18], [19], rely on clustering the feature vectors. This is intuitive as the frames transmitted from the same location will tend to be clustered together. The number of clusters is then used as a test statistic and the threshold is obtained to attain a target probability of false alarm ($P_{\rm FA}$). The considered clustering algorithms include *density based spatial clustering of applications with noise* (DBSCAN) [25], *hierarchical DBSCAN* (HDBSCAN) [26], *ordering points to identify the clustering structure* (OPTICS) [27], and *balanced iterative reducing and clustering using hierarchies* (BIRCH) [28].

Fig. 3 depicts the receiver operating characteristic (ROC) curves [29, Ch. 3] of GSD and the benchmarks. It is seen that GSD results in a significantly higher probability of detection ($P_{\rm D}$) for each $P_{\rm FA}$.

Fig. 4 analyzes the influence of the speed of the users in the $P_{\rm D}$ for a given $P_{\rm FA} = 0.1$ for the compared algorithms.

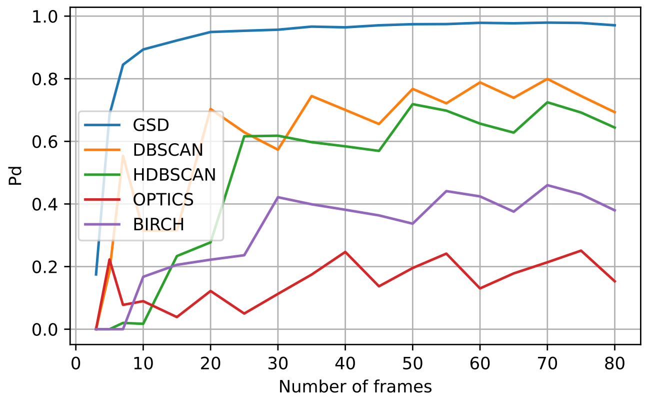

Fig. 5: Pd vs number of frames of the proposed algorithm and the benchmarks for a fixed probability of false alarm $P_{\rm FA}$ (10 frames/s, $K = 150$, $N = 5$, $P_{\rm FA} = 0.1$).

Interestingly, speed seems to positively impact the $P_{\rm D}$ of all algorithms, especially those based on clustering. This is because the the number of clusters per user increases with the speed and, therefore, the test statistic will tend to be more different between hypotheses.

Fig. 5 investigates the impact of J on the detection performance. As expected, $P_{\rm D}$ tends to increase with J. However, a wiggling effect is observed for the benchmarks. This does not vanish even if the number of Monte Carlo iterations is increased. The cause is the discrete nature of the test statistic of the benchmarks.

Finally, Fig. 6 analyzes how $P_{\rm D}$ evolves as a function of K. It is remarkable that GSD attains a very large $P_{\rm D}$ even for a small K, which suggests that the PCD successfully learned to distinguish the two sources of variability in the feature vectors described in Sec. III-A.

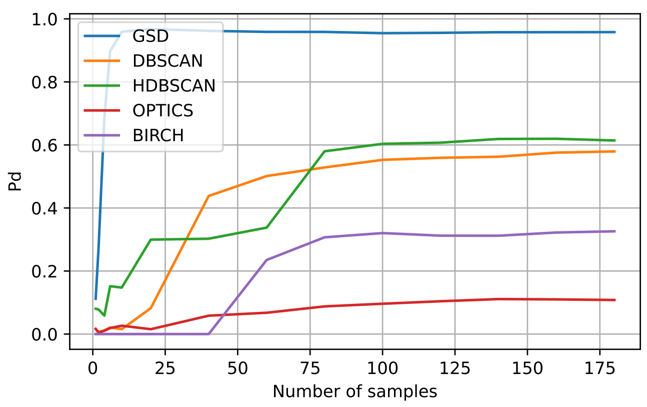

Fig. 6: P_{D} vs number of frames of the proposed algorithm and the benchmarks for a fixed probability of false alarm P_{FA} (10 frames/s, $J = 30$, $N = 5$, $P_{\text{FA}} = 0.1$).

V. Conclusion

This work considered the problem of detecting spoofing attacks in the physical layer, which is motivated by the vulnerability of cryptographic techniques when the credentials of the legitimate user are compromised. Unfortunately, prior schemes based on RSS measurements raise false alarms in the presence of channel changes or user movement. To remedy this limitation, this work introduced a deep learning detector robust to these effects. Since localization-based approaches would suffer from a low spatial resolution due to multipath effects, a position change detector based on a deep neural network is used to build a graph embedding of the RSS features. The temporal pattern of changes in the propagation conditions captured by this embedding is then learned by a GNN, which then decides on the presence of a spoofing attack. Empirical evaluation with real-world data showcases the effectiveness of this scheme as well as its robustness to the mobility of the user and attacker.

References

[1] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2015.

[2] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms," in *ACM on Asia conf. computer commun. security*, 2016, pp. 413–424.

[3] J. Martin, D. Alpuche, K. Bodeman, L. Brown, E. Fenske, L. Foppe, T. Mayberry, E. C. Rye, B. Sipes, and S. Teplov, "Handoff all your privacy: A review of apple's bluetooth low energy continuity protocol," *arXiv preprint arXiv:1904.10600*, 2019.

[4] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *ACM Conf. Comput. commun. security*, 2011, pp. 75–86.

[5] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Int. Conf. Mobile Comput. Netw.*, 2008, pp. 116–127.

[6] H. Givehchian, N. Bhaskar, E. R. Herrera, H. R. L. Soto, C. Dameff, D. Bharadia, and A. Schulman, "Evaluating physical-layer BLE location tracking attacks on mobile devices," in *IEEE Symp. Security Privacy*. IEEE, 2022, pp. 1690–1704.

[7] P. Liu, P. Yang, W.-Z. Song, Y. Yan, and X.-Y. Li, "Real-time identification of rogue WiFi connections using environment-independent physical features," in *IEEE INFOCOM 2019-IEEE Conf. Computer Commun.* IEEE, 2019, pp. 190–198.

[8] T. D. Vo-Huu and G. Noubir, "Fingerprinting Wi-Fi devices using software defined radios," in *ACM Conf. Security & Privacy Wireless Mobile Netw.*, 2016, pp. 3–14.

[9] J. Xiong and K. Jamieson, "Secureangle: Improving wireless security using angle-of-arrival information," in *ACM SIGCOMM Workshop Hot Topics Netw.*, 2010, pp. 1–6.

[10] J. Xiong and K. Jamieson, "Securearray: Improving WiFi security with fine-grained physical-layer information," in *Annual Int. Conf. Mobile comput. & netw.*, 2013, pp. 441–452.

[11] X. Shi, B. D. O. Anderson, G. Mao, Z. Yang, J. Chen, and Z. Lin, "Robust localization using time difference of arrivals," *IEEE Signal Process. letters*, vol. 23, no. 10, pp. 1320–1324, 2016.

[12] N. Wang, L. Jiao, P. Wang, W. Li, and K. Zeng, "Machine learning-based spoofing attack detection in mmwave 60GHz IEEE 802.11 ad networks," in *IEEE Conf. Computer Commun.* IEEE, 2020, pp. 2579–2588.

[13] Y. Chen, W. Trappe, and R. P. Martin, "Detecting and localizing wireless spoofing attacks," in *Annual IEEE Commun. Society Conf. sensor, mesh ad hoc commun. netw.* IEEE, 2007, pp. 193–202.

[14] J. Yang, Y. Chen, W. Trappe, and J. Cheng, "Detection and localization of multiple spoofing attackers in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 44–58, 2012.

[15] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "PHY-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 10037–10047, 2016.

[16] K. Zeng, K. Govindan, D. Wu, and P. Mohapatra, "Identity-based attack detection in mobile wireless networks," in *IEEE INFOCOM*. IEEE, 2011, pp. 1880–1888.

[17] B. Alotaibi and K. Elleithy, "A new MAC address spoofing detection technique based on random forests," *Sensors*, vol. 16, no. 3, pp. 281, 2016.

[18] M. T. Hoang, Y. Zhu, B. Yuen, T. Reese, X. Dong, T. Lu, R. Westendorp, and M. Xie, "A soft range limited k-nearest neighbors algorithm for indoor localization enhancement," *IEEE Sensors J.*, vol. 18, no. 24, pp. 10208–10216, 2018.

[19] A. Sobehy, E. Renault, and P. Mühlethaler, "CSI-MIMO: K-nearest neighbor applied to indoor localization," in *IEEE Int. Conf. Commun.* IEEE, 2020, pp. 1–6.

[20] E. S. Lohan, J. Torres-Sospedra, H. Leppäkoski, P. Richter, Z. Peng, and J. Huerta, "Wi-Fi crowdsourced fingerprinting dataset for indoor positioning," *Data*, vol. 2, no. 4, 2017.

[21] P. Barsocchi, A. Crivello, D. La Rosa, and F. Palumbo, "A multisource and multivariate dataset for indoor localization methods based on WLAN and geo-magnetic field fingerprinting," in *2016 Int. Conf. Indoor Position. Indoor Navig. (IPIN)*, 2016, pp. 1–8.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT press, 2016.

[23] D. Romero and G. Leus, "Wideband spectrum sensing from compressed measurements using spectral prior information," *IEEE Trans. Signal Process.*, vol. 61, no. 24, pp. 6232–6246, Dec. 2013.

[24] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, 2009.

[25] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Knowledge Discovery and Data Mining*, 1996.

[26] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia Conf. Knowl. Discov. Data Min.* Springer, 2013, pp. 160–172.

[27] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, 1999.

[28] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, 1996.

[29] S. M. Kay, *Fundamentals of Statistical Signal Processing, Vol. II: Detection Theory*, Prentice-Hall, 1998.