SPOOFING ATTACK DETECTION IN THE PHYSICAL LAYER WITH COMMUTATIVE NEURAL NETWORKS

Daniel Romero[1]*, Peter Gerstoft*[2]*, Hadi Givehchian*[2]*, Dinesh Bharadia*[2]

[1]Dept. of ICT, University of Agder, Norway. Email: daniel.romero@uia.no
[2]Electrical and Computer Engineering, University of California, San Diego, USA

ABSTRACT

In a spoofing attack, an attacker impersonates a legitimate user to access or tamper with data intended for or produced by the legitimate user. In wireless communication systems, these attacks may be detected by relying on features of the channel and transmitter radios. In this context, a popular approach is to exploit the dependence of the received signal strength (RSS) at multiple receivers or access points with respect to the spatial location of the transmitter. Existing schemes rely on long-term estimates, which makes it difficult to distinguish spoofing from movement of a legitimate user. This limitation is here addressed by means of a deep neural network that implicitly learns the distribution of pairs of short-term RSS vector estimates. The adopted network architecture imposes the invariance to permutations of the input (commutativity) that the decision problem exhibits. The merits of the proposed algorithm are corroborated on a data set that we collected.

***Index Terms*—** Spoofing attack, wireless networks, deep learning.

1. INTRODUCTION

The pervasive presence of wireless communications in virtually all human activities has spurred the proliferation of techniques that seek to illicitly obtain private data, compromise the uptime of remote services, or impersonate other users [1–4]. Spoofing attacks, which pursue the latter goal, are especially problematic since they endow the attacker with the capacity to access and modify data intended for or produced by a legitimate user. Detecting this kind of attacks is therefore of paramount importance to guarantee data security.

To hinder the ability of an attacker to impersonate a legitimate user, cryptographic techniques are employed in various communication layers, from the medium access control (MAC) layer to the application layer. However, these security barriers are readily bypassed by an attacker with the user credentials. For this reason, many techniques to detect spoofing attacks in the physical layer have been developed, typically by exploiting physical layer characteristics that are specific to the transmitter, such as carrier frequency offset (CFO), or features of the channel, such as received signal strength (RSS) or Angle of Arrival (AoA). These spoofing detection techniques generally match observed features to previously collected ones for a given legitimate transmitter.

More specifically, imperfections such as CFO, I/Q offset, and I/Q imbalance, which uniquely characterize the analog hardware of transmitters, have been used to verify user identity in [5–8]. Unfortunately, these techniques need knowledge of the communication protocol of the transmitter and may fail under changes in the environment; e.g. in temperature [6]. These limitations have been partially alleviated in [9–11], which employ AoA and TDoA features, and in [12], where a neural network is trained on signal-to-noise ratio (SNR) traces obtained in the sector level sweep process in mm-Wave 60 GHz IEEE 802.11ad Networks. However, they still require synchronization and/or knowledge of the communication protocol.

In comparison, RSS-based techniques do not need to know protocols or decode signals, which greatly enhances their generality and applicability for spoofing detection [13–17]. The most common technique in this context is to apply K-means to RSS measurements collected by multiple receivers [13, 18, 19]. The idea is to leverage the dependence of RSS signatures on the location of the transmitter (and potential attacker) to declare an attack iff transmissions with the same user ID (as provided by higher communication layers) are determined to be originated at different locations. Unfortunately, this means that attacks are declared *even when a legitimate user transmits from different locations because it moves*. Thus, to satisfactorily tell spoofing from motion, one needs to employ a *higher-level algorithm* that utilizes the decisions of the aforementioned *low-level* RSS-based detectors for multiple pairs of transmissions. To see the intuition, suppose for example that one transmission is received from point A, then B, then C, and then D, where all these spatial locations are declared to be different by the *low-level* algorithm. This suggests that the user is moving. In turn, if the low-level algorithm determines that A=C and B=D, then it is most likely that the legitimate user is at A and an

This research has been funded in part by the Research Council of Norway under IKTPLUSS grant 311994 and Intelligence Advanced Research Projects Activity (IARPA), via 2021-2106240007.

Data and code to reproduce the experiments are available at `https://github.com/fachu000/spoofing`

attacker at B or vice versa. Unfortunately, such (low-level) algorithms necessitate accurate RSS measurements, which in turn require long averaging time intervals. This limits their time resolution and, therefore, their usage by higher-level algorithms to distinguish spoofing from movement.

This observation calls for RSS-based approaches capable of effectively solving the aforementioned low-level decision problem with RSS estimates that average just a small number of received samples. To understand the challenge, recall that RSS is generally estimated by averaging the squared magnitude of samples of the received signal. For a large number of samples, these estimates converge to the actual RSS. Unlike existing works, which generally assume converged RSS estimates, a method with high temporal resolution must rely on short-term averages, which randomly fluctuate around the true RSS. Developing such a method is the main contribution of the present paper. To this end, the distribution of such noisy RSS estimates is implicitly learned in a data-driven fashion using a deep neural network (DNN). The adopted architecture abides by the commutative nature of the decision problem. The data collection process just involves recording a small number of samples corresponding to a few tens of different transmitter locations. The high accuracy of the proposed algorithm is evaluated on a data set of real measurements that we collected as part of this research work.

2. MODEL AND PROBLEM FORMULATION

Let $\mathcal{X} \subset \mathbb{R}^3$ index the spatial region of interest, where all transmitters, both legitimate users and attackers, are located. A transmitter at a fixed position $\boldsymbol{x} \in \mathcal{X}$ transmits a signal whose complex baseband equivalent version is $s(t)$, where t denotes time. This signal, modeled as an unknown wide-sense stationary stochastic process, is received by N static receivers, which may be, e.g., access points or base stations. After downconversion, the signal at the n-th receiver reads as

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

where $*$ denotes convolution, $h_n(\boldsymbol{x}, t)$ is the unknown deterministic impulse response of the bandpass equivalent channel between location $\boldsymbol{x}$ and the n-th receiver, and $z_n(t)$ is noise. As usual, $z_n(t)$ is assumed wide-sense stationary and uncorrelated with $s(t)$, which implies that the mean-square magnitude of $r_n(\boldsymbol{x}, t)$ is independent on t. Thus, one can define the RSS of signal plus noise as

$$f_n(\boldsymbol{x}) := 10 \log_{10} \mathbb{E}|r_n(\boldsymbol{x}, t)|^2, \tag{2}$$

where $\mathbb{E}$ denotes expectation. To estimate $f_n(\boldsymbol{x})$, the n-th receiver averages the square magnitude of K samples of $r_n(\boldsymbol{x}, t)$, that is:

$$\hat{f}_n(\boldsymbol{x}) := 10 \log_{10} \sum_{k=0}^{K-1} |r_n(\boldsymbol{x}, kT)|^2, \tag{3}$$

where T is the sampling interval. If $r_n(\boldsymbol{x}, kT)$ is ergodic for each $\boldsymbol{x}$, as typically assumed, it follows that $\hat{f}_n(\boldsymbol{x})$ converges to $f_n(\boldsymbol{x})$ when $K \to \infty$.

A fusion center or central controller forms the RSS vector estimate $\hat{\boldsymbol{f}}(\boldsymbol{x}) := [\hat{f}_0(\boldsymbol{x}), \ldots, \hat{f}_{N-1}(\boldsymbol{x})]^\top$ by gathering these N RSS estimates. It is assumed that the position $\boldsymbol{x}$ of the transmitter does not change significantly during the process of acquiring and collecting these estimates, which is a mild assumption if K is small, as considered here.

With this notation, one can readily formulate the problem as follows. Suppose that an RSS vector estimate is obtained for two transmissions and let $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ denote the (potentially equal) locations where the transmissions have been originated. Given $\hat{\boldsymbol{f}}(\boldsymbol{x}_1)$ and $\hat{\boldsymbol{f}}(\boldsymbol{x}_2)$, the problem is to decide between the following hypotheses:

$$\begin{cases} \mathcal{H}_0 : \ \boldsymbol{x}_1 = \boldsymbol{x}_2 \\ \mathcal{H}_1 : \ \boldsymbol{x}_1 \neq \boldsymbol{x}_2. \end{cases} \tag{4}$$

To assist in this task, a data set of feature vectors $\{\hat{\boldsymbol{f}}^{(i)}(\boldsymbol{x}_m), m = 0 \ldots M-1, \ i = 0, \ldots, I-1\}$ is given, where $\boldsymbol{x}_m \neq \boldsymbol{x}_{m'}$ for all $m \neq m'$ and $\{\hat{\boldsymbol{f}}^{(i)}(\boldsymbol{x}_m)\}_{i=0}^{I-1}$ denote I estimates of $\boldsymbol{f}(\boldsymbol{x}_m) := [f_0(\boldsymbol{x}_m), \ldots, f_{N-1}(\boldsymbol{x}_m)]^\top$.

3. DNN-BASED SPOOFING DETECTOR

This section describes the architecture and training process of the proposed detector.

3.1. Data Set

As indicated in Sec. 2, the given data set comprises vectors of the form $\hat{\boldsymbol{f}}^{(i)}(\boldsymbol{x}_m)$. To train a DNN-based detector, a data set of pairs of such vectors needs to be constructed with both pairs that correspond to the same transmitter location and with pairs that correspond to different transmitter locations. If one wishes to minimize the probability of error, as described later, it is desirable that the resulting data set contains the same number P of pairs from each of these categories. The procedure is formally described next.

First, for $p = 0, \ldots, P-1$, draw m_p uniformly at random from $\{0, \ldots, M-1\}$ and draw i_p and i'_p uniformly at random without replacement (i.e. no replacement *for each* p) from $\{0, \ldots, I-1\}$. Then, form

$$\mathcal{D}_\mathrm{s} := \{(\hat{\boldsymbol{f}}^{(i_p)}(\boldsymbol{x}_{m_p}), \hat{\boldsymbol{f}}^{(i'_p)}(\boldsymbol{x}_{m_p})), \\ p = 0, \ldots, P-1\} \subset \mathbb{R}^N \times \mathbb{R}^N.$$

Similarly, for $p = 0, \ldots, P-1$, draw m_p and m'_p uniformly at random without replacement (again for each p) from $\{0, \ldots, M-1\}$ and draw i_p and i'_p uniformly at random without replacement (i.e. no replacement *for each* p) from

$\{0, \ldots, I-1\}$. Then, form

$$\mathcal{D}_{\rm d} := \{(\hat{\boldsymbol{f}}^{(i_p)}(\boldsymbol{x}_{m_p}), \hat{\boldsymbol{f}}^{(i'_p)}(\boldsymbol{x}_{m'_p})), \\ p = 0, \ldots, P-1\} \subset \mathbb{R}^N \times \mathbb{R}^N.$$

3.2. Architecture

A detector (or binary classifier) is a function $d : \mathbb{R}^N \times \mathbb{R}^N \to \{\mathcal{H}_0, \mathcal{H}_1\}$ that returns a hypothesis for each input $(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')$. In the signal processing terminology, such a detector is constructed by setting $d(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') = \mathcal{H}_1$ iff $G(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')$ exceeds a certain threshold, where $G : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is termed *detection statistic*. The probability of error is minimized when

$$G(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') = \log \frac{\mathbb{P}[\mathcal{H}_1 | \hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}']}{\mathbb{P}[\mathcal{H}_0 | \hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}']} = \log \frac{\mathbb{P}[\mathcal{H}_1 | \hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}']}{1 - \mathbb{P}[\mathcal{H}_1 | \hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}']}, \quad (5)$$

and the threshold is 0 [20, Sec. 3.7]. Satisfying (5) is not generally possible using a finite data set, but a function $G(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')$ can be learned to approximately satisfy (5), as described in Sec. 3.3. The threshold to be used with the learned function will still be 0.

Observe that, if one implements G directly as a DNN $G_{\boldsymbol{w}}$, where $\boldsymbol{w}$ is the vector of parameters such as weights and offsets, it will generally happen that $G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') \neq G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}', \hat{\boldsymbol{f}})$, which is inconsistent with the symmetry of (4). Thus, the proposed approach is instead to use a DNN to implement an auxiliary function $\tilde{G}_{\boldsymbol{w}}$ and then set

$$G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') = \frac{\tilde{G}_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') + \tilde{G}_{\boldsymbol{w}}(\hat{\boldsymbol{f}}', \hat{\boldsymbol{f}})}{2}. \quad (6)$$

It can be easily seen that $G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') = G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}', \hat{\boldsymbol{f}})$ and that the 2 in the denominator (6) can be absorbed into $\tilde{G}_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')$.

Function $\tilde{G}_{\boldsymbol{w}}$ will be implemented as a composition of more fundamental functions called layers. The following is a description of the layers used in our experiments, but other possibilities can be considered. The first layer $\tilde{G}_1(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')$ is non-trainable and complements the input with N linear features to facilitate learning:

$$\tilde{G}_1(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') = [\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}'] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}. \quad (7)$$

The subsequent 3 (hidden) layers are fully connected with leaky ReLU activations and 512 neurons [21]. Finally, the output layer is also fully connected and has a single neuron with a linear activation.

3.3. Training

From (5), it follows that

$$\mathbb{P}[\mathcal{H}_1 | \hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}'] = \frac{1}{1 + \exp[-G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')]} := \sigma(G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')), \quad (8)$$

where σ denotes the so-called sigmoid function. Given the way the data set was constructed, it follows that the log-likelihood of $\boldsymbol{w}$ is given by

$$\mathcal{L}(\boldsymbol{w}; \mathcal{D}_{\rm s}, \mathcal{D}_{\rm d}) = \sum_{(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') \in \mathcal{D}_{\rm s}} \log(1 - \sigma(G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')) \\ + \sum_{(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}') \in \mathcal{D}_{\rm d}} \log(\sigma(G_{\boldsymbol{w}}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}}')). \quad (9)$$

Following standard practice, a local minimum of the loss function $-\mathcal{L}(\boldsymbol{w}; \mathcal{D}_{\rm s}, \mathcal{D}_{\rm d})$ can be efficiently approximated using stochastic gradient descent [21].

To avoid overfitting, a common technique is to apply an early-stopping approach, where optimization is halted when the *validation accuracy* stops increasing. To this end, the given M transmitter locations are split into $M_{\rm tr}$ locations for training and $M_{\rm val} := M - M_{\rm tr}$ locations for validation. From the first, $2P_{\rm tr}$ pairs are constructed as explained in Sec. 3.1 and substituted in (9) to obtain the training loss. On the other hand, $2P_{\rm val}$ pairs are obtained from the $M_{\rm val}$ validation locations to obtain the validation accuracy, i.e., the number of validation pairs successfully classified.

It was observed that an ℓ_1 regularizer on the weights of the first trainable layer improves learning, likely because the variability across space of some features is too chaotic and, thus, not informative. Hence, promoting sparsity in this layer encourages the DNN to select the most informative features.

4. EXPERIMENTS

This section empirically validates the proposed algorithm using real data. A link to the code and data set is provided on the first page. Data collection took place in a room of the University of California, San Diego, as depicted in Fig. 1. A transmitter was sequentially placed at 52 locations. For the sake of reproducibility, the transmitted signal $s(t)$ is a 5 MHz sinusoid at a carrier frequency of 2.3 GHz. For every transmitter, four receivers with four antennas each record 4888 samples of the received signal. Thus, N can be set between 1 and 4·4=16.

The proposed DNN-based classifier (DNNC) is compared with three benchmarks. The first two, termed distance-based classifiers (DBCs), decide $\mathcal{H}_1$ iff $\|\hat{\boldsymbol{f}} - \hat{\boldsymbol{f}}'\|_q > \gamma$, where q is 2 for DBC(ℓ_2) and 1 for DBC(ℓ_1). The threshold γ is adjusted to maximize the accuracy over the training pairs. Along the lines of [13, 14], the third benchmark is a K-means classifier that clusters the training vectors $\hat{\boldsymbol{f}}^{(i)}(\boldsymbol{x}_m)$ into C clusters. Then it applies the same rule as DBC(ℓ_2) over the two feature vectors that respectively result from collecting the C Euclidean distances from the given $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{f}}'$ to all centroids.

The performance metric is the accuracy measured on a test data set constructed as described in Sec. 3.1 with the $52 - M$ locations that are not used for training. The accuracy of each algorithm is averaged using Monte Carlo simulation over the choice of the M training locations among all 52 available

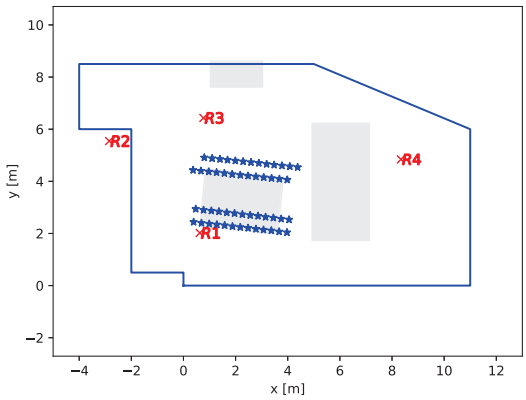

Fig. 1: Plan of the measurement room with outer walls (solid blue), and tables (grey). Locations of transmitters (blue star) and receivers (red cross) are indicated.

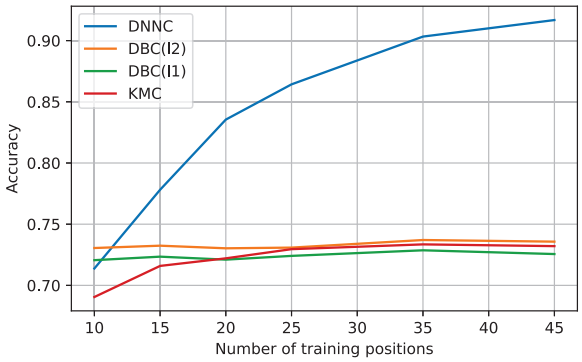

Fig. 2: Accuracy vs. number of training positions M for the proposed algorithm (DNNC) and the competing algorithms ($K = 16$, $N = 16$, $P_{\text{tr}} = 1250$, $P_{\text{val}} = 150$, $M_{\text{tr}} = 0.8M$, $C = 15$).

locations. The vector of parameters $\boldsymbol{w}$ of DNNC is randomly initialized at every Monte Carlo iteration.

The first experiment investigates the number M of training locations that need to be collected to attain a reasonable accuracy. Fig. 2 compares the accuracy of DNNC as a function of M with the benchmarks. It is observed that (i) the accuracy of DNNC is very high with just 45 locations, (ii) DNNC learns much more than the benchmarks with new data. The horizontal axis begins at $M = 10$ because DNNC needs to split the given M positions into both training and validation positions and, therefore, using $M < 10$ results in very noisy estimates of the training and validation losses; cf. [22, Ch. 2].

The second experiment assesses the influence of the number of features on the performance of DNNC. In this data set, the first 4 features correspond to the first receiver, the second 4 features to the second receiver, and so on. Fig. 3 shows

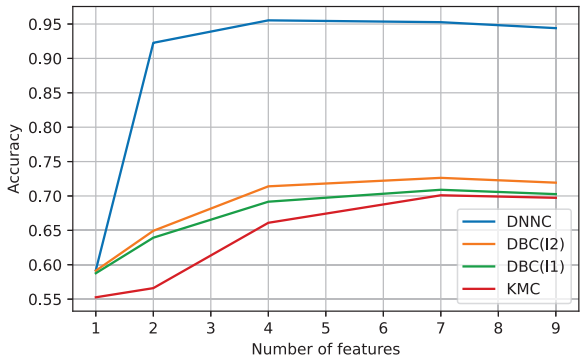

Fig. 3: Accuracy vs. number of features N for the proposed algorithm (DNNC) and the competing algorithms ($K = 16$, $P_{\text{tr}} = 1250$, $P_{\text{val}} = 150$, $M = 40$, $M_{\text{tr}} = 0.8M$, $C = 15$).

that two features already result in a very high accuracy for DNNC. Remarkably, this effect can be seen to be milder when the two selected features are obtained by different receivers, which suggests that there is more information in the joint distribution of the RSS estimates obtained by nearly-located antennas than in those obtained by distant antennas. One may ascribe this phenomenon to the fact that the latter are "less synchronized" than the former.

Finally, it is worth observed that in both experiments, the value of samples used by the RSS estimator in (3) is just $K = 16$, which demonstrates that DNNC meets the goal of providing a high temporal resolution. Further experiments omitted here due to lack of space show that the accuracy of the benchmarks decreases more abruptly than the one of DNNC if K is reduced below 16.

5. CONCLUSIONS AND DISCUSSION

This paper presented an algorithm that learns to distinguish whether two RSS feature vectors correspond to the same transmitter location or to a different transmitter location. The low number of samples required by this algorithm renders it suitable to construct a spoofing detector that can distinguish spoofing attacks from motion of a legitimate user; cf. Sec. 1. A DNN with an architecture that ensures symmetry in the decisions was designed and tested. Numerical experiments with real data corroborate the high accuracy of the proposed scheme. Future work will target the development of a high-level algorithm that relies on the decisions of the proposed scheme to attain robustness to user movement.

It is worth emphasizing that the collection of the data set entails small effort, since the locations of the training points $\boldsymbol{x}_m$ need not be measured. It suffices to ensure that $\boldsymbol{x}_m \neq \boldsymbol{x}_{m'}$ for all $m \neq m$. The measurements can be collected by a technician or recording signals in a time interval where no attacks are known (by some other means) to be taking place.

6. REFERENCES

[1] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis, "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2015.

[2] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens, "Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms," in *Proceedings of the 11th ACM on Asia conference on computer and communications security*, 2016, pp. 413–424.

[3] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik C Rye, Brandon Sipes, and Sam Teplov, "Handoff all your privacy: A review of apple's bluetooth low energy continuity protocol," *arXiv preprint arXiv:1904.10600*, 2019.

[4] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun, "On the requirements for successful gps spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 75–86.

[5] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, 2008, pp. 116–127.

[6] Hadi Givehchian, Nishant Bhaskar, Eliana Rodriguez Herrera, Héctor Rodrigo López Soto, Christian Dameff, Dinesh Bharadia, and Aaron Schulman, "Evaluating physical-layer ble location tracking attacks on mobile devices," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1690–1704.

[7] Pengfei Liu, Panlong Yang, Wen-Zhan Song, Yubo Yan, and Xiang-Yang Li, "Real-time identification of rogue wifi connections using environment-independent physical features," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 190–198.

[8] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir, "Fingerprinting wi-fi devices using software defined radios," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, pp. 3–14.

[9] Jie Xiong and Kyle Jamieson, "Secureangle: improving wireless security using angle-of-arrival information," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.

[10] Jie Xiong and Kyle Jamieson, "Securearray: Improving wifi security with fine-grained physical-layer information," in *Proceedings of the 19th annual international conference on Mobile computing & networking*, 2013, pp. 441–452.

[11] Xiufang Shi, Brian DO Anderson, Guoqiang Mao, Zaiyue Yang, Jiming Chen, and Zihuai Lin, "Robust localization using time difference of arrivals," *IEEE Signal processing letters*, vol. 23, no. 10, pp. 1320–1324, 2016.

[12] Ning Wang, Long Jiao, Pu Wang, Weiwei Li, and Kai Zeng, "Machine learning-based spoofing attack detection in mmwave 60ghz ieee 802.11 ad networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2579–2588.

[13] Yingying Chen, Wade Trappe, and Richard P Martin, "Detecting and localizing wireless spoofing attacks," in *2007 4th Annual IEEE Communications Society Conference on sensor, mesh and ad hoc communications and networks*. IEEE, 2007, pp. 193–202.

[14] Jie Yang, Yingying Chen, Wade Trappe, and Jerry Cheng, "Detection and localization of multiple spoofing attackers in wireless networks," *IEEE Transactions on Parallel and Distributed systems*, vol. 24, no. 1, pp. 44–58, 2012.

[15] Liang Xiao, Yan Li, Guoan Han, Guolong Liu, and Weihua Zhuang, "Phy-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10037–10047, 2016.

[16] Kai Zeng, Kannan Govindan, Daniel Wu, and Prasant Mohapatra, "Identity-based attack detection in mobile wireless networks," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 1880–1888.

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

[18] Minh Tu Hoang, Yizhou Zhu, Brosnan Yuen, Tyler Reese, Xiaodai Dong, Tao Lu, Robert Westendorp, and Michael Xie, "A soft range limited k-nearest neighbors algorithm for indoor localization enhancement," *IEEE Sensors Journal*, vol. 18, no. 24, pp. 10208–10216, 2018.

[19] Abdallah Sobehy, Eric Renault, and Paul Mühlethaler, "Csi-mimo: K-nearest neighbor applied to indoor localization," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

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

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

[22] V. Cherkassky and F. M. Mulier, *Learning from Data: Concepts, Theory, and Methods*, John Wiley & Sons, 2007.