Machine learning approaches for position and user authentication in wireless systems

Abstract

Physical layer authentication has been proposed as an effective alternative scheme to traditional upper layer-based techniques, such as cryptography, in order to provide a secure authentication scheme. In this paper, we exploit the area specific attenuation values to perform location-based authentication, i.e. granting access to the network only to users located in a pre-defined area. We first exploit the Neyman-Pearson (N-P) optimal criterion to formulate the problem as hypothesis testing. Secondly, we implement the authentication system by exploiting to Machine Learning (ML) techinques, namely the Neural Networks (NNs) and the Support Vector Machine (SVM). We then show how the output of the NN trained with two different loss functions, namely the Mean Squared Error (MSE) and the Cross Entropy (CE), are related to N-P optimal hypothesis testing. The same result has been derived for SVM. By introducing a realistic channel model we then show that the proposed NN implementation of the authentication system is convenient respect to N-P. The problem of network planning is hence addressed and lastly we propose a possible atteck strategy.

Index Terms

Physical layer security, authentication, neural network, auto-encoder, support vector machine

I. Introduction

Traditional authentication systems are based on upper-layer techniques such as cryptography. However, as the network latency requirement becomes stricter, the timely sharing of security keys in large and dense networks may not be supported by this type of networks. This is further complicated by the computational cost of key generation and detection. On the other hand, as the computational power of devices grows, the time spent in cracking a digital security key may be remarkably shortened [1]. Physical layer techniques have been proposed as an alternative to digital key generation. These techniques exploit properties of the communication channel such as the received signal strength (RSS) to distinguish radio transmitters. In this paper, we exploit area specific attenuation values to distinguish between a legitimate and non-legitimate area, implementing in such a way a location-based authentication system.

Traditional detection systems are based on Neyman-Pearson (N-P) lemma which rejects a hypothesis in favor of the other based on the comparison of the likelihood ratio with a suitably chosen threshold value. An example application of this criterion is [2], where the N-P framework has been exploited to implement an authentication system suited for multiple wiretap channels with correlated fading. A generalization of the N-P framework is presented in [3], where physical layer authentication is performed over a Rayleigh fading channels and detection is implemented via generalized likelihood ratio test. A further example is [4], where Machine Learning (ML) algorithms have been exploited in order to find the best threshold needed for hypothesis testing.

However, the knowledge of the channel statistics as well as its model may not be available. In this context, the computation of the likelihood functions needed for hypothesis testing can be estimated by the available data but may not be accurate. ML techniques have been introduced as an alternative method to perform user authentication without requiring prior knowledge of the channel statistics. This issue has been tackled by [5], where no assumption is made on the channel model and logistic regression has been proposed as alternative to hypothesis testing. In [6], a feed-forward neural network receives as input the Euclidean distance between successive channels and the Pearson coefficient as feature space to perform authentication. Literature overview on SVM to be added ($https://www.researchgate.net/publication/279246263_Robust_indoor_localization_and_tracking_using_GSM_f$; $https://ieeexplore.ieee.org/abstract/document/7037452/$).

In this paper, we compare the performance of the authentication system based on N-P optimality criterion, henceforth referred to as N-P detector, with ML based approaches. In particular, we investigate the performance of the ML-based authentication systems implemented via Neural Network (NN) and via Support Vector Machine (SVM). We consider two different NN architectures: the multi-layer perceptron and the auto-encoder. For the multi-layer perceptron, we investigate two loss functions: the Cross Entropy (CE) and the Mean Squared Error (MSE). We show that both loss functions guarantee the same performance of the optimal N-P detector in terms of true positive and false negative rate. Further details about the architectures and the training objective function will be discussed in section IV.

In section V we consider the problem of performing hypothesis testing when samples from one of the two classes are available and no information on the other class can be obtained. The auto-encoder NN is proposed as solution for this problem.

In section VI, we introduce the SVM and we show how it can be exploited to implement an

authentication system. Furthermore, we show that the output of such a system is related to the N-P lemma.

In sections IX and X we confirm via numerical results the theoretical parts of the previous section and investigate the performance of the different implementations of the authentication system.

In section VII, we consider the problem of network planning, i.e., find the optimal positions of the base stations (BSs) needed to implement the authentication system. We show that the NN can be exploited in the optimization procedure.

The contribution of this paper can be summarized as follows:

- We propose an implementation of a physical layer-based location authentication system which exploits ML techniques to perform hypothesis testing;
- We show that the performance obtained with this ML techniques are optimal in a N-P sense;
- We compare different ML techniques and show how these can be exploited to implement different levels of the authentication system, from the planning of the network to authentication and attacks.

The following notation will be used throughout the paper: bold letters $\boldsymbol{x}$ refer to vectors, whereas capital bold letters $\boldsymbol{H}$ refer to matrices, $\mathbb{E}[]$ denotes the expected value, e_i denotes the all zero vector except for component i which is equal to 1 and $(\cdot)^T$ denotes the transpose.

II. System Model

Consider a network with N_{bs} BSs, where BS n, $n = 1, ..., N_{\text{bs}}$, is located in position $\boldsymbol{x}_{\text{bs}}^{(n)} = (X_{\text{bs}}^{(n)}, Y_{\text{bs}}^{(n)})$, where entries represent respectively the x and y coordinates.

Consider a user equipment (UE) transmitting with power P_{tx}. The received power at the n^{th} BS is

$$P_{\text{rc}}^{n}[dB] = P_{\text{tx}}[dB] - a^{(n)}[dB], \tag{1}$$

where $a^{(n)}$ is the attenuation incurred by the transmitted signal to BS n. Assuming that the transmitting power of each UE is known, the BS can compute the attenuation value $a^{(n)}$. As signals transmitted from different locations incur to different attenuation we exploit such values to implement the authentication system.

We propose two models for the attenuation coefficient $a^{(n)}$. Consider a UE located in position $\boldsymbol{x}_{\text{ue}} = (X_u, Y_u)$. Let us define the distance between the UE and the BS as

$$L(\boldsymbol{x}_{\text{ue}}, \boldsymbol{x}_{\text{bs}}^{(n)}) = \sqrt{(X_{\text{bs}}^{(n)} - X_u)^2 + (Y_{\text{bs}}^{(n)} - Y_u)^2}. \tag{2}$$

When considering line of sight (LOS) links the path loss is modelled as

$$P_{\mathrm{LOS}}^{(n)}[dB] = 20\log_{10}\left(\frac{f 4\pi L(\boldsymbol{x}_{\mathrm{ue}}, \boldsymbol{x}_{\mathrm{bs}}^{(n)})}{c}\right), \tag{3}$$

where f is the carrier frequency and c is the speed of light. We refer to the *LOS scenario* the case where $a^{(n)} = P_{\mathrm{LOS}}$.

When considering non-LOS links path loss is defined as

$$P_{\mathrm{non-LOS}}^{(n)}[dB] = 40\log 10\left(\frac{L(\boldsymbol{x}_{\mathrm{ue}}, \boldsymbol{x}_{\mathrm{bs}}^{(n)})}{10^3}\right) + 21\log 10\left(\frac{f}{10^6}\right) + 80; \tag{4}$$

The shadowing term $s \sim \mathcal{N}(0, \sigma_s^2)$ is modelled as a zero mean σ_s^2 variance Gaussian random variable. We will refer to the *full scenario* when considering both LOS non-LOS links, shadowing effects and fading. In the full scenario the attenuation toward the n^{th} BS, due to the shadowing effects, is modelled as normal random variable with zero mean and variance

$$\sigma_f^2(n) = P_l^{(n)} n - 4,34s, \tag{5}$$

and hence the attenuation coefficient due to fading is modelled as

$$a^{(n)}[dB] = \mathcal{N}(0, \sigma_f^{-2}(n)). \tag{6}$$

Let us denote as $\mathcal{A}_0$ the legitimate area, i.e., where UEs are allowed to access the network and as $\mathcal{A}_1$ the area where UEs are not allowed to access the network. Assume that $\mathcal{A}_0$ is a sub-region of $\mathcal{A}_1$. Let us denote as $|\mathcal{A}_0|$ the area in m^2 of the legitimate region and as $|\mathcal{A}_1|$ the area in m^2 of the non-legitimate region. Since $\mathcal{A}_0 \subset \mathcal{A}_1$ the size of the overall area coincides with $|\mathcal{A}_1|$.

III. Location authentication

Consider a location authentication system based on physical layer information. In such a system, the legitimate user, traditionally referred to as Alice, sends a message to a central system, traditionally referred to as Bob, that is in charge of authenticating it. The process is divided in two phases: location association and location verification. In location association phase, Alice sends a message x to Bob through a secure channel h. Bob associate the channel h to Alice and what Bob expects is that each successive message coming from Alice will go through a channel that is very similar to h. During the location verification phase, Bob receives a message from an unknown user and estimates a channel $\hat{h}$: if $\hat{h}$ is very similar to h, then Bob can conclude that

the message comes from Alice. Otherwise, nothing can be said on the identity of the transmitter and the message is considered non-secure.

We exploit this two-step authentication procedure in order to grant access to the network only to users located in area A_0. Let us define the attenuation vector $\boldsymbol{a} = [a^{(1)}, ..., a^{(N_{\rm bs})}]$, whose $n^{\rm th}$ entry is the attenuation value measured at the $n^{\rm th}$ BS for the current message. During the location association phase, we collect attenuation vectors from both area A_0 and area A_1 in order to create the sets $\mathcal{S}_0$ and $\mathcal{S}_1$ of the attenuation vectors of the two areas. The idea is that different areas are associated to different attenuation vectors and, if $N_{\rm bs}$ is sufficiently large, we can discriminate area based on attenuation vectors. During the location verification phase, the system compares the current attenuation vector, i.e., the one associated to the current message, with those associated to area A_0 and if a match is detected the user is granted access to the network.

This problem has a straightforward formulation as an hypothesis testing problem. Let us define the two hypothesis: a) hypothesis $\mathcal{H}_0$, i.e., the user is transmitting from area $\mathcal{A}_0$; b) hypothesis $\mathcal{H}_1$, i.e., the user is transmitting from area $\mathcal{A}_1$.

Given the attenuation vector $\boldsymbol{a}$ we want to determine which of the two hypothesis is more likely to be true in order to implement the aforementioned authentication system. We exploit the Neyman-Pearson lemma to test the hypothesis. Let us define the log likelihood-ratio (LLR)

$$\mathcal{L} = \log\left(\frac{p(\boldsymbol{a}|\mathcal{H}_0)}{p(\boldsymbol{a}|\mathcal{H}_1)}\right), \tag{7}$$

where $p(\boldsymbol{a}|\mathcal{H}_.)$ is the probability of measuring the attenuation vector $\boldsymbol{a}$ given that hypothesis $\mathcal{H}_.$ is verified. According to the N-P lemma, the most powerful test is the one that compares the value $\mathcal{L}$ with a threshold value Λ in order to choose the more likely hypothesis. Let us define two error probabilities: $P_{\rm FA}$ as the false alarm probability, i.e. the probability $P(\hat{\mathcal{H}} = \mathcal{H}_1|\mathcal{H}_0)$ that a legitimate user is classified as non-legitimate; $P_{\rm MD}$ as the mis-detection probability, i.e., the probability $P(\hat{\mathcal{H}} = \mathcal{H}_0|\mathcal{H}_1)$ that a non-legitimate user is classified as legitimate. The N-P lemma ensures that, given a false alarm probability value the minimum mis-detection probability is obtained by choosing

$$\begin{cases} \mathcal{H}_1 & \text{if} \quad \mathcal{L} \leq \Lambda \\ \mathcal{H}_0 & \text{if} \quad \mathcal{L} > \Lambda. \end{cases} \tag{8}$$

This method ensures that the performance of the authentication system are optimal. However the statistics needed to compute the LLR (7) are not always available. A numerical approximation

can be obtained in certain cases, but we will show that this requires a large data-set to get a good approximation. In the following sections we introduce different ML techniques that will be exploited to implement the authentication system without requiring a-priori knowledge of the channel statistics.

IV. Neural network overview

A NN is a function of the type $f : \mathbb{R}^N \to \mathbb{R}^O$ which maps a set of N real values into O real values. A NN is identified by three types of layer: the input layer, the hidden layers, and the output layer. This architecture is also known as multy-layer perceptron (MLP)

Let us consider a NN with $L-1$ layers and denote as $N^{(\ell-1)}$ is the size of the $(\ell-1)^{\text{th}}$ layer. The output of the n^{th} neuron of the ℓ^{th} layer is

$$y_n^{(\ell)} = \sigma\left(\boldsymbol{w}_n^{(\ell-1)}\boldsymbol{y}^{(\ell-1)} + b_n^{(\ell)}\right), \tag{9}$$

i.e., a mapping via an activation function σ of the weighted linear combination with weights $\boldsymbol{w}_n^{(\ell-1)} \in \mathbb{R}^{1\times N^{(\ell-1)}}$ of the outputs $\boldsymbol{y}^{(\ell-1)} \in \mathbb{R}^{N^{(\ell-1)}\times 1}$ of the previous layer plus a bias $b_n^{(\ell)} \in \mathbb{R}^{N^{(\ell-1)}\times 1}$. The types of layer are characterized by the weights and the activation functions to be used. In the input layer each neuron corresponds to a single value of the input vector, hence , by defining as the layer 0 the input layer, we have $\boldsymbol{w}_n^{(0)} = e_n$. The activation function of the input layer is the identity function.

The weights of each layer are updated by means of a supervised learning algorithm which feed to the network a set of input values and the corresponding labels, i.e. known output values, such that the input values are mapped to the corresponding output labels. The activation function of the hidden layers is the sigmoid function

$$\sigma(x) = \frac{1}{1-e^{-x}}, \tag{10}$$

whereas the activation function of the output layer is a $\tanh^{-1}$ function

$$\tanh^{-1}(x) = \frac{1}{2}\left(\frac{1+x}{1-x}\right). \tag{11}$$

Consider a network with N_{BS} BSs. Since the information available at BS n is the attenuation value a_n we use the attenuation vector $\boldsymbol{a}$ as input of the MLP. The input layer will be hence composed by N_{BS} neurons. The objective is to train a MLP with a single output neuron whose value is used for hypothesis testing.

In the next sections we investigate the effects of training the MLP with two different loss functions: the MSE and the CE.

A. MSE training

Consider a set of training attenuation vectors $\boldsymbol{a}^{(i)}$, $i = 1, ..., S$ with target output $\theta^{(i)}$, $i = 1, ..., S$ where $\theta^{(i)}$ is equal to 1 if $\boldsymbol{a}^{(i)}$ has been generated form a user located in area $\mathcal{A}_0$ whereas $\theta^{(i)} = -1$ if $\boldsymbol{a}^{(i)}$ has been generated form a user in $\mathcal{A}_1$. Let us define as $\boldsymbol{\theta}_{\mathrm{MSE}}$ the vector of target values, i.e., $\boldsymbol{\theta}_{\mathrm{MSE}} = [\theta^{(1)}, ..., \theta^{(S)}]$.

Let us define as $\tilde{\boldsymbol{\theta}}_{\mathrm{MSE}}$ the vector of the output values of the MLP, whose i^{th} component is the output of the MLP obtained with the i^{th} training vector. The MLP training is performed via gradient descent minimizing the MSE, defined as

$$MSE = \mathbb{E}[||\tilde{\boldsymbol{\theta}}_{\mathrm{MSE}} - \boldsymbol{\theta}_{\mathrm{MSE}}||^2]. \tag{12}$$

Since the output of the neural network $\theta^{(i)}$ is a continuous value in $-1, 1$, in order to perform classification, a suitable threshold value λ must be chosen, such that the input vector $\boldsymbol{a}^{(i)}$ is classified as

$$\boldsymbol{a}^{(i)} \in \begin{cases} \mathcal{A}_0 \,\text{if}\, \theta^{(i)} > \lambda \\ \mathcal{A}_1 \,\text{if}\, \theta^{(i)} \leq \lambda, \end{cases} \tag{13}$$

where $\boldsymbol{a}^{(i)} \in \mathcal{A}_{\cdot}$ denotes that the attenuation vector $\boldsymbol{a}^{(i)}$ has been generated from a UE located in area $\mathcal{A}_{\cdot}$.

The test to be performed is hence similar to the one performed by the N-P detector. In the next section we show the relation that exists between MSE training and the optimal hypothesis testing.

B. MSE training connection with N-P lemma

In the previous section we showed that a NN implements a function $f(\boldsymbol{a}, \boldsymbol{W})$, where $\boldsymbol{a}$ is the vector of input parameters and $\boldsymbol{W}$ is the matrix whose $\ell\mathrm{th}$ column is obtained by stacking the vector of weights $\boldsymbol{w}_n^{(\ell)}$ and bias $b_n^{(\ell)}$ of each neuron of the $\ell\mathrm{th}$ layer. In this section we are going to prove the following theorem:

Theorem 1. *Consider a NN with perfect training and a sufficient number of parameters such that training reaches a global minimum. Then the classificator obtained by training the NN via MSE is equivalent to the classificator obtained via N-P lemma.*

Proof. Consider the function $f(\boldsymbol{a}, \boldsymbol{W})$ implemented by the NN and the Bayes optimal discriminant function

$$g_0(\boldsymbol{a}) = \mathbb{P}(\mathcal{H}_0|\boldsymbol{a}) - \mathbb{P}(\mathcal{H}_1|\boldsymbol{a}). \tag{14}$$

The MSE obtained by a NN performing classification with target outputs $-1, 1$ for the two hypothesis can be written as [7]

$$E_t(\boldsymbol{W}) = \mathbb{E}\left\{(f(\boldsymbol{a}, \boldsymbol{W} - g_0(\boldsymbol{a}))^2\right\} + \left\{1 - \mathbb{E}[g_0(\boldsymbol{a})]^2\right\}. \tag{15}$$

The NN training problem is solved by minimizing $E_t(\boldsymbol{W})$ respect to the parameters $\boldsymbol{W}$ and we notice that the second term in (15) does not depend on $\boldsymbol{W}$. From [7], when the number of parameters goes to infinity, we obtain

$$f(\boldsymbol{a}, \boldsymbol{W}) \approx \mathbb{P}(\mathcal{H}_0|\boldsymbol{a}) - \mathbb{P}(\mathcal{H}_1|\boldsymbol{R}). \tag{16}$$

By recalling that $\mathbb{P}(x|y) = \mathbb{P}(y|x)\mathbb{P}(x)/\mathbb{P}(y)$ we can write

$$g_0(\boldsymbol{a}) = \frac{\mathbb{P}(\boldsymbol{a}|\mathcal{H}_0)\mathbb{P}(\mathcal{H}_0) - \mathbb{P}(\boldsymbol{a}|\mathcal{H}_1)\mathbb{P}(\mathcal{H}_1)}{\mathbb{P}(\boldsymbol{a})}, \tag{17}$$

which in turn can be written as

$$g_0(\boldsymbol{a}) = \frac{\mathbb{P}(\boldsymbol{a}|\mathcal{H}_0)\mathbb{P}(\mathcal{H}_0) - \mathbb{P}(\boldsymbol{a}|\mathcal{H}_1)\mathbb{P}(\mathcal{H}_1)}{\mathbb{P}(\boldsymbol{a}|\mathcal{H}_0)\mathbb{P}(\mathcal{H}_0) + \mathbb{P}(\boldsymbol{a}|\mathcal{H}_1)\mathbb{P}(\mathcal{H}_1)}. \tag{18}$$

By imposing a threshold λ on $g_0(\boldsymbol{a})$ and reorganizing we obtain

$$\frac{\mathbb{P}(\boldsymbol{a}|\mathcal{H}_0)}{\mathbb{P}(\boldsymbol{a}|\mathcal{H}_1)} > \frac{\mathbb{P}(\mathcal{H}_1)}{\mathbb{P}(\mathcal{H}_0)}\frac{1+\lambda}{1-\lambda} = \lambda^*, \tag{19}$$

which is equivalent to the N-P criterion. □

C. CE training

Consider the training attenuation vectors set $\boldsymbol{a}^{(i)}$, $i = 1, ..., S$ with target output vector $\boldsymbol{\theta}^{\mathrm{CE}}$, where the i^{th} component is equal to 1 if $\boldsymbol{a}^{(i)}$ has been generated form a user located in area $\mathcal{A}_0$ whereas $\theta_i = 0$ if $\boldsymbol{a}^{(i)}$ has been generated form a user in $\mathcal{A}_1$. Let us define as $\tilde{\boldsymbol{\theta}}^{\mathrm{CE}}$ the vector whose i^{th} component $\tilde{\theta}_i^{\mathrm{CE}}$ is the output of the NN obtained with the i^{th} training vector. The NN training is performed via gradient descent minimizing the CE defined as

$$CE = -\sum_{i=1}^{S}\left(\tilde{\theta}_i^{\mathrm{CE}} log\left(\theta_i^{\mathrm{CE}}\right) + \left(1 - \tilde{\theta}_i^{\mathrm{CE}}\right) log\left(1 - \theta_i^{\mathrm{CE}}\right)\right). \tag{20}$$

In this case the output of the NN is a continuous value between 0 and 1 and a suitable threshold value must be chosen in order to assign label 0 or 1 to the considered input vector.

Comparison with a threshold is the key step of the N-P lemma and we notice that this is also required when performing CE training. In the next section we show how the two methods are related.

D. CE training connection to N-P lemma

When training is performed with CE loss function the output of the NN represents the probability $\mathbb{P}(\mathcal{H}_0|\boldsymbol{a}^{(i)})$ of being in hypothesis $\mathcal{H}_0$ given that the attenuation vector is $\boldsymbol{a}^{(i)}$ [8], i.e.,

$$f(\boldsymbol{a}^{(i)}, \boldsymbol{W}) \approx \mathbb{P}(\mathcal{H}_0|\boldsymbol{a}^{(i)}), \tag{21}$$

where $\boldsymbol{W}$ is the matrix whose ℓth column is obtained by stacking the vector of weights $\boldsymbol{w}_n^{(\ell)}$ and bias $b_n^{(\ell)}$ of each neuron of the ℓth layer.

In this section we prove the following

Theorem 2. *Consider a NN with perfect training and a sufficient number of parameters such that the training reaches a global minimum. Then the classificator obtained by training the NN via CE is equivalent to the classificator obtained via N-P lemma.*

Proof. Since we are considering a two class classification problem the probability $\mathbb{P}(\mathcal{H}_1|\boldsymbol{a}^{(i)})$, i.e., the probability of being in hypothesis $\mathcal{H}_1$ given that the attenuation vector is $\boldsymbol{a}^{(i)}$, is obtained as

$$\mathbb{P}(\mathcal{H}_1|\boldsymbol{a}^{(i)}) = 1 - \mathbb{P}(\mathcal{H}_0|\boldsymbol{a}^{(i)}). \tag{22}$$

By imposing a threshold on the output of the NN we obtain

$$f(\boldsymbol{a}^{(i)}, \boldsymbol{W}) \approx \mathbb{P}(\mathcal{H}_0|\boldsymbol{a}^{(i)}) > \lambda, \tag{23}$$

which can be rewritten as

$$2\mathbb{P}(\mathcal{H}_0|\boldsymbol{a}^{(i)}) - 1 > \hat{\lambda} \tag{24}$$

$$\mathbb{P}(\mathcal{H}_0|\boldsymbol{a}^{(i)}) - (1 - \mathbb{P}(\mathcal{H}_0|\boldsymbol{a}^{(i)})) > \hat{\lambda} \tag{25}$$

$$\mathbb{P}(\mathcal{H}_0|\boldsymbol{a}^{(i)}) - \mathbb{P}(\mathcal{H}_1|\boldsymbol{a}^{(i)}) > \hat{\lambda}. \tag{26}$$

We hence obtained the same formulation of (16) and, by following the same steps of the proof of theorem 1 we see that imposing a threshold on the output of the CE-trained NN is equivalent of performing hypothesis testing with N-P lemma. □

V. REPLICATOR NEURAL NETWORK

In this section we consider the case where discrimination between the two areas $\mathcal{A}_0$ and $\mathcal{A}_1$ is performed based on data generated only from the authentic region, i.e., the available attenuation vectors $\boldsymbol{a}^{(i)}$ used for training are generated from UEs located in $\mathcal{A}_0$. This type of problem is usually referred to as *one-class classification*.

In order to solve this problem we exploit a NN that implements a *replicator neural network (RNN)*, i.e., a network that is trained to: a) convert high-dimensional inputs to low-dimensional codes in the hidden layer; b) reconstruct the high-dimensional input at the output layer from the low-dimensional codes of the hidden layer [9].

Consider a set $\mathcal{U}$ and a set of vectors $\mathcal{V} \in \mathcal{U}$. If this vectors are used for training the auto-encoder, each new input vector $\boldsymbol{v} \in \mathcal{U}$ is mapped at the output with a small reconstruction error, whereas each input vector $\boldsymbol{v} \notin \mathcal{U}$ is mapped at the output with a large reconstruction error.

The capacity of the auto-encoder to replicate only certain values at the output is due to the fact that the hidden layer is able the extract the features of the set $\mathcal{U}$ used for training. This is particularly true when the size of the hidden layer is lower than the size of the input layer, i.e. when $M < N$ [10].

Consider an RNN with three layers, i.e., a size N input layer, a size M hidden layer and a size N output layer. The function mapping the input layer to the hidden layer is of the form (9), whereas the non-linearity is removed when mapping the hidden layer to the output layer. This is due to the fact that the network must reconstruct real values and hence the output shall not be limited in a range of values.

The n^{th} output $y_n^{(o)}$ of the output layer is obtained from

$$y_n^{(o)} = \boldsymbol{w}_n^{(o)} \boldsymbol{y}^{(h)} + b_n^{(h)} \tag{27}$$

where $\boldsymbol{w}_n^{(o)}$ is the $1 \times M$ vector of weights assigned to the n^{th} output, $\boldsymbol{y}^{(h)}$ is the $M \times 1$ vector of the outputs of the hidden layer and $b_n^{(h)}$ is the n^{th} bias of the hidden layer.

We here exploit the RNN to implement the authentication system described in section III.

Consider the input attenuation vector $\boldsymbol{a}$ and its replicated value $\hat{\boldsymbol{a}}$ at the output of the auto-encoder after training. Let us define the reconstruction error

$$\epsilon = \frac{1}{N} \sum_{i=1}^{N} |a_i - \hat{a}_i|^2, \tag{28}$$

where a_i denotes the i^{th} component of the vector. The authentication system performs a test comparing the reconstruction error ϵ with a threshold value γ and chooses

$$\boldsymbol{a} \in \begin{cases} \mathcal{A}_0 \,\text{if}\, \epsilon < \gamma \\ \mathcal{A}_1 \,\text{if}\, \epsilon \geq \gamma. \end{cases} \tag{29}$$

VI. Support Vector Machine

A SVM [8] is a supervised learning model that can be used for classification and regression. We focus here on binary classification, i.e., given the input attenuation vector $\mathbf{a}^{(i)} \in \mathbb{R}^N$ the SVM returns $y_i = 1$ if $\mathbf{a}^{(i)}$ has been generated form a user located in area $\mathcal{A}_0$ whereas $y_i = -1$ if $\mathbf{a}^{(i)}$ has been generated form a user in $\mathcal{A}_1$.

The classification problem is modelled as a function $y : \mathbb{R}^N \to \mathbb{R}$ defined by

$$y(\mathbf{a}^{(i)}) = \mathbf{w}^T \phi(\mathbf{a}^{(i)}) + b, \tag{30}$$

where $\phi : \mathbb{R}^N \to \mathbb{R}^K$ is a feature-space transformation function, $\mathbf{w} \in \mathbb{R}^K$ is the weights vector and b is a bias parameter. The decision function is

$$y_i = \text{sign}(y(\mathbf{a}^{(i)})). \tag{31}$$

Geometrically, the SVM defines a hyperplane in $\mathbb{R}^K$ defined by $y(\mathbf{a}^i) = 0$ which separates the features of the two classes.

A. Training and Relation with N-P Lemma

Given a training set $\{\mathbf{a^{(i)}}\}_{i=1}^S$, the SVM searches for the hyperplane maximizing the margin, i.e., the perpendicular distance between the decision boundary and the closest data point. Moreover we want to minimize the mis-classification errors.

The relation with N-P lemma comes from the least squares SVM (LS-SVM), an extension of the SVM first introduced in [11]. In [12] it is shown that SVM and LS-SVM are equivalent under mild conditions. The LS-SVM defines the optimization problem as

$$\min_{\mathbf{w},e} \quad f_l(\mathbf{w}, e) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\frac{1}{2}\sum_{i=1}^{S} e_i^2 \tag{32a}$$

$$\text{subject to}\, y_i[\mathbf{w}^T\phi(\mathbf{a}^{(i)}) + b] = 1 - e_i \quad i = 1, \ldots, S. \tag{32b}$$

From the constraints in (32) and the fact that $y_i = \pm 1$ we have

$$e_i^2 = (1 - y_i y(\mathbf{a}^{(i)}))^2 = (y_i - y(\mathbf{a}^{(i)}))^2, \tag{33}$$

that is the squared error between the soft output of the LS-SVM $y(\mathbf{a}^i)$ and the correct training label y_i. We now prove the equivalence between the LS-SVM and N-P classificators.

Let us first consider the following lemma that establishes the convergence of the learning phase of SVM, as the training sample set becomes large.

Lemma 1. *For training samples $\boldsymbol{a}^{(i)}$ from a finite alphabet $\mathcal{A}$, taken with a given static probability distribution, for large number of training samples, i.e., as $S \to \infty$, the vector $\boldsymbol{w}$ of the LS-SVM converges in probability to a vector of finite norm $||\mathbf{w}||_2 = \mathbf{w}^T\mathbf{w}$.*

Proof. Given a finite alphabet $\mathcal{A} = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_M\}$ of M elements for $\boldsymbol{a}^{(i)}$, we indicate with $p_{\boldsymbol{a}^{(i)},y_i}(\boldsymbol{\alpha}_j, y)$, with $y \in \{-1, 1\}$, the joint probability of input vector $\boldsymbol{a}^{(i)}$ and corresponding output y_i, $i = 1, \ldots, S$.

By the Glivenko–Cantelli theorem we have that with probability 1 as $S \to \infty$ there are $Sp_{\boldsymbol{a}^{(i)},y_i}(\boldsymbol{\alpha}_j, y)$ training vectors $\boldsymbol{\alpha}_j$ with associated true value y in any training sequence. All these training points will have the same value e_i, from (32b), that will appear $Sp_{\boldsymbol{a}^{(i)},y_i}(\boldsymbol{\alpha}_j, y)$ times in the sum $\sum_{i=1}^{S} e_i^2$. Note that in the training ensemble there could be two equal instances $\mathbf{a}^{(m)} = \mathbf{a}^{(n)} = \alpha_j$, but with different labels $y_m \neq y_n$. Therefore, for $\mathbf{a}^i = \alpha_\mathbf{j}$ we can have two possible values for e_i, depending on y_i, and we denote them with $e_{j,1}$ and $e_{j,-1}$. This translates in only $2M$ *distinct* constraints of the type (32b). Asymptotically, for $S \to \infty$, problem (32) becomes

$$\min_{\mathbf{w},e} \quad f_l(\mathbf{w}, e) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + CS\frac{1}{2}\sum_{j=1}^{M}[p_{\boldsymbol{a}^{(i)},y_i}(\boldsymbol{\alpha}_j, 1)e_{j,1}^2 + p_{\boldsymbol{a}^{(i)},y_i}(\boldsymbol{\alpha}_j, -1)e_{j,-1}^2] \tag{34a}$$

$$\text{subject to } [\mathbf{w}^T\phi(\boldsymbol{\alpha}_j) + b] = 1 - e_{j,1} \quad j = 1, \ldots, M. \tag{34b}$$

$$-[\mathbf{w}^T\phi(\boldsymbol{\alpha}_j) + b] = 1 - e_{j,-1} \quad j = 1, \ldots, M. \tag{34c}$$

which solution provides the convergence value (in probability) of vector $\boldsymbol{w}$. Note that while the original problem (32) as $S \to \infty$ has infinite constraints, it can be solved by the problem (34) that includes a finite number $2M$ of constraints.

We now solve (34) with the dual formulation following the same steps used for the LS-SVM in [11] with minor modifications. In particular, when building the Lagrangian we need to introduce two sets of multipliers: v_k for (34b) and u_k for (34c), $k = 1, \ldots, M$. Then, by deriving with respect to $\mathbf{w}$ and setting to 0, we obtain

$$\mathbf{w}\mathbf{w}^T = \sum_{k=1}^{M}\sum_{h=1}^{M} k(\boldsymbol{\alpha}_k, \boldsymbol{\alpha}_h)(v_k v_h + u_k u_h - 2v_k u_h), \tag{35}$$

where we used the fact that the kernel function is symmetric w.r.t. its inputs. We conclude that $\mathbf{w}$ has a finite norm since the rhs of (35) is a finite sum.

□

Theorem 3. *Consider a LS-SVM with perfect training, i.e., the training reaches a global minimum of $f_l(\mathbf{w}, e)$ given an infinite number of training points $\boldsymbol{a}^{(i)}$. Then the classificator obtained by training the LS-SVM and by thresholding the soft output* (30) *is equivalent, in the MSE sense, to the classificator obtained via N-P lemma.*

Proof. Let f_l^* be the solution to the optimization problem (32). Since we're assuming perfect training, then $f_l^* < +\infty$ and the optimization variables $\mathbf{w}$ will converge as S grows. We can then write

$$\lim_{S\to+\infty} \mathbf{w}^T\mathbf{w} < +\infty. \tag{36}$$

Consider now

$$\lim_{S\to+\infty} \frac{1}{S} f_l(\mathbf{w}, e) = \frac{C}{2} \lim_{S\to+\infty} \frac{1}{S} \sum_{i=1}^{S} e_i^2 = \frac{C}{2} \mathrm{E}_t(\mathbf{w}, b), \tag{37}$$

where $\mathrm{E}_t(\mathbf{w}, b) = \mathbb{E}\left[\left(y_i - y(\mathbf{a}^{(i)})\right)^2\right]$ is the expected value carried out w.r.t. the training points $\mathbf{a}^{(i)}$. The first equality comes from (36), while the last equality comes from the strong law of large numbers. In the limit, the optimization problem (32) is equivalent to

$$\min_{\mathbf{w},b} \quad \mathrm{E}_t(\mathbf{w}, b), \tag{38}$$

where we dropped the constraints in (32) by substitution using (33). The optimization problem is the same as in the NN case and from [7] we have that the couple $(\mathbf{w}^*, b^*)$ minimizing (38) and parametrizing (30) is such that

$$y(\mathbf{a}^i; \mathbf{w}^*, b^*) \approx \mathbb{P}(\mathcal{H}_0|\mathbf{a}^{(i)}) - \mathbb{P}(\mathcal{H}_1|\mathbf{a}^{(i)}). \tag{39}$$

The proof now is the same as in Theorem 1. □

Note that in order to build the receiver operating characteristic (ROC) it is required to change the decision function (31) with the thresholded version

$$y_i = \begin{cases} +1 & y(\mathbf{a}^{(i)}) \geq \gamma^* \\ -1 & y(\mathbf{a}^{(i)}) < \gamma^*. \end{cases} \tag{40}$$

VII. BSs' OPTIMAL POSITION

As attenuation maps are different at each BS and depend on the surrounding environment in terms of shadowing effects the performance of the authentication system depends on the number of BSs and their position. In this section we derive an approach to optimally locate BSs so that the authentication system attains the best performance.

Consider a sufficiently large training set which is representative of the attenuation vectors measured from both $\mathcal{A}_0$ and $\mathcal{A}_1$. An MLP trained with such a set is not likely to find a large number of outliers when tested. In this sense we can say that if a MLP attains better performance that another one in the training phase, then it attains better performance also in the testing phase. We hence test different positions for N_{BS} BSs and, for each one, we train a MLP with a sufficient number of training vectors and compute the minimum value obtained by the gradient descent algorithm used for training.

Fig. 1 shows the performance obtained with 8 random positions for $N_{\mathrm{BS}} = 5$ BSs. For all curves is reported the minimum MSE obtained during training and the corresponding area under the curve (AUC) after testing. We see that MSE values can be divided in ranges, each one corresponding to different values of AUC. In particular we notice that lower ranges of MSE correspond to higher values of AUC. For MLPs attaining MSE values within the same range we must instead also perform testing in order to choose the configuration with better classification performance. Hence, in order to find the optimal configuration we implement an iterative algorithm which, for each configuration, build a training set and trains a MLP and updates the position of the BSs according to the obtained MSE value. When the difference between the current MSE value and that obtained at the previous iteration is smaller than a threshold value λ_{diff} then a testing set is built and the relative AUC values are computed. The configuration with higher AUC is then selected as the optimal one.

Notice however that the problem is non-convex as the map realization depends on the BSs position and on a random shadowing and fading realization. Therefore we exploit the particle swarm optimization (PSO) [13].

PSO is an iterative optimization algorithm based on social behavior of animals (e.g. birds flocking and fish schools). Consider a particle as a set of positions for the BSs and consider a total number of P particles. Each one is a possible candidate solution of the optimization problem. Each particle is described by its position $\boldsymbol{x}_p$, which is a N_{BS} dimensional vector containing the

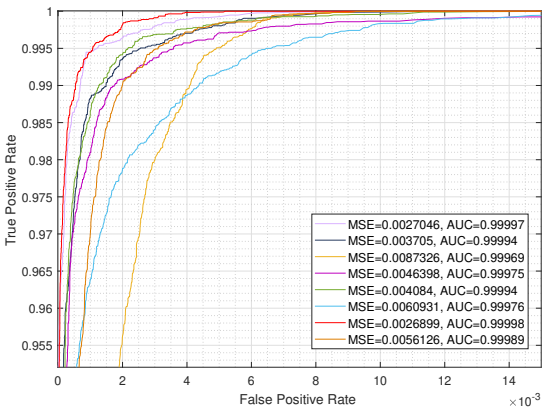

Fig. 1. ROC curves for different BS positions. Lower MSE are associated to higher AUC

positions of the BSs and representing a possible solution, and its velocity $\boldsymbol{v}_p$. Starting from a random initialization of all the particles at each iteration both the positions $\boldsymbol{x}_p$ and the velocities $\boldsymbol{v}_p$ are updated. Two optimal values are defined in each iteration: the global optimal value found so far by the entire population and a local optimal value for each particle, i.e., the optimal value found by the individual p up to the current iteration. We define as $\boldsymbol{o}_g$ the position of the the global optimal values and as $\boldsymbol{o}_p$ the position of the optimal value found by particle p at the current iteration.

The position and velocity of the particles are updated at iteration t as

$$\boldsymbol{v}_p(t) = w\boldsymbol{v}_p(t-1) + \phi_1(t)(\boldsymbol{o}_p(t-1) - \boldsymbol{x}_p(t-1)) + \phi_2(t)(\boldsymbol{o}_g(t-1) - \boldsymbol{x}_p(t-1)), \tag{41}$$

$$\boldsymbol{x}_p(t) = \boldsymbol{x}_p(t-1) + \boldsymbol{v}_p(t), \tag{42}$$

where w is the inertia coefficient and ϕ_1 and ϕ_2 are random variables distributed respectively in $[0, c_1]$ and $[0, c_2]$, where c_1 and c_2 are defined as acceleration constants. The values of the inertia coefficient and of the acceleration constants are the parameters of the PSO problem. Typical values for this parameters are $w = 0.7298$, $c_1 = c_2 = 1.4961$ [13].

The algorithm steps for BSs positioning are reported in Algorithm 1. We initialize P particles with random positions for each of the N_{BS} in each particle. For each particle p we then build and train a NN and compute the achieved MSE value $\text{MSE}_p^{(0)}$. PSO is then exploited to iteratively update the position of the particles. Notice that in order to find the best local and global optimal positions MSE values at the current and previous iterations are compared. If this values are in

Algorithm 1: BSs positioning algorithm

Data: number of particles P, N_{BS}, λ_{diff}
Result: optimal position
Initialization: select random positions for the components of each particle;
build training set for each particle;
build and train a NN for each particle and obtain the training performance $\mathrm{MSE}_{\mathrm{p}}^{(0)}$, $p = 1, ..., N_p$;
$it = 0$;
repeat
  $it = it + 1$;
  for $p = 1, ..., P$ **do**
    update velocity and position vector of particle p via (41) and (42);
    build a training set;
    build and test a NN and compute $\mathrm{MSE}_{\mathrm{p}}^{(\mathrm{it})}$;
    if $|\mathrm{MSE}_{\mathrm{p}}^{(\mathrm{it}-1)} - \mathrm{MSE}_{\mathrm{p}}^{(\mathrm{it})}| < \lambda_{\mathrm{diff}}$ **then**
      compute $\mathrm{AUC}_{\mathrm{p}}^{(\mathrm{it}-1)}$ and $\mathrm{AUC}_{\mathrm{p}}^{(\mathrm{it})}$ for both NNs;
      update PSO vectors based on AUC_p;
    else
      update PSO vectors based on MSE_p;
    end
  end
until *convergence of particles positions*;

the same range, i.e., if $|\mathrm{MSE}_{\mathrm{p}}^{(\mathrm{it}-1)} - \mathrm{MSE}_{\mathrm{p}}^{(\mathrm{it})}| < \lambda_{\mathrm{diff}}$ then the NNs are also tested and decision is taken based on the AUC values, otherwise decision is taken based on MSE values.

The position of the particle that achieves the global minimum MSE or maximum AUC at the end of the optimization problem is the best position for the BSs. Notice that, as the optimization problem is non-convex, solving PSO is similar to a multi-start optimization considering P different starting points, which is a standard method used to avoid local minimums. Hence as the number P increases the probability of finding a local solution is reduced.

Notice that the BSs positioning problem could be solved without implementing a NN by mean of computation of the LLRs for the different positions. However we showed in Section X-A that when considering multiple BSs and including shadowing effects the exact computation of the LLR curve requires more data than those needed for training a NN. Hence the proposed solution is effective in terms of both authentication performance and number of needed data.

VIII. Attack strategies

The proposed techniques can be exploited in order to provide not only an authentication system, but also for implementing an attack strategy. The idea is here to use attenuation values

measured from the non-authentic area and to train a ML architecture based on these values.

The first architecture we exploit is the RNN. The RNN is here trained in order to reproduce at the output attenuation vectors measured from the non-authentic area. We propose two attack strategies: a random generation attack and a manifold based attack.

A. Random generation attack

After that the RNN has been trained with attenuation vectors measured from the non-authentic area, only attenuation vectors with the same statistical structure will be reproduced at the output with a reconstruction error ϵ (28) lower than γ. The idea is hence to generate a random attenuation vector $\boldsymbol{a}_{\text{test}}$ and to feed it to the RNN: if the output has a reconstruction error lower than γ then $\boldsymbol{a}_{\text{test}}$ is considered as belonging to the non-authentic area, otherwise the it is considered as a possible candidate for a successful attack. This vector is hence tested and, if the attack is not successful, we label it as belonging to the non-authentic area.

As new randomly generated vectors are not successful or have a reconstruction error lower than γ we gain information about the structure of the non-authentic area. We hence update the RNN training in a mini-batch fashion when a sufficient number of new vectors (good results can be achieved with batch size of $m = 3$ or $m = 4$ new vectors per update [14]) is obtained. These vectors are stored in matrix $\boldsymbol{A}_{\text{test}}$ and when m vectors populate the matrix mini-batch gradient descent is executed on the RNN to update its parameters.

The algorithm steps are reported in Algorithm 2.

B. Manifold based attack

The second attack strategy exploits the encoding properties of the RNN. Consider an optimal-compression RNN [15] and let $\Phi : [0,1]^M \rightarrow \mathbb{D}$ be a smooth orientation-preserving diffeomorphism of the unit cube $[0,1]^M \subset \mathbb{R}^M$ onto the data manifold $\mathbb{D} \subset \mathbb{R}^N$. The natural coordinates of a point $\boldsymbol{x}$ on $\mathbb{D}$ are defined as the Cartesian coordinates of its pre-image $\Phi^{-1}(\boldsymbol{x})$ on the cube. From the theorem in [15] we know that a RNN configured to perform a reconstruction of a value on the data manifold $\mathbb{D}$ within a MSE ϵ outputs at the hidden layer neurons the natural coordinates of the given input.

This allows us to construct the manifold of the non-authentic area and hence to generate data that are certainly out of the training area, reducing hence the time needed to generate vectors that do not have the same characteristics of those used for training.

Algorithm 2: Random generation attack

Data: weight and bias vectors of the trained RNN
Result: authentic attenuation vector

repeat
  generate random attenuation vector $\boldsymbol{a}_{\rm test}$;
  compute reconstruction error ϵ of $\boldsymbol{a}_{\rm test}$ via (28);
  if $\epsilon < \gamma$ **then**
    $\boldsymbol{a}_{\rm test}$ belongs to non-authentic area;
    store $\boldsymbol{a}_{\rm test}$ in $\boldsymbol{A}_{\rm test}$;
  else
    test $\boldsymbol{a}_{\rm test}$ for an attack;
    if *attack is successful* **then**
      return $\boldsymbol{a}_{\rm test}$
    else
      store $\boldsymbol{a}_{\rm test}$ in $\boldsymbol{A}_{\rm test}$;
    end
  end
  if *size(*$\boldsymbol{A}_{\mathbf{test}}$*)* $= m$ **then**
    update RNN training via mini-batch gradient descent;
    delete vectors from $\boldsymbol{A}_{\rm test}$;
  end
until *authentic attenuation vector has been obtained*;

In order to build the manifold of the training set we compute the outputs of the hidden layer when the vectors of the training set are fed as input.

IX. Numerical results: LOS scenario

In order to confirm the theoretical results obtained so far and compare the different solutions we test the performance of the authentication systems based on two measures: the *true positive rate* and the *false positive rate*.

Let us define as *true positive* the number of UEs located in area $\mathcal{A}_0$ which are considered in area $\mathcal{A}_0$ by the authentication system, whereas we define as *false positive* the number of UEs located in area $\mathcal{A}_1$ which are considered in area $\mathcal{A}_0$ by the authentication system. Furthermore let us define as *true negative* the number of UEs located in area $\mathcal{A}_1$ which are considered in area $\mathcal{A}_1$ by the authentication system, whereas we define as *false negative* the number of UEs located in area $\mathcal{A}_0$ which are considered in area $\mathcal{A}_1$ by the authentication system. We hence define the true positive rate as

$$\text{true positive rate} = \frac{\text{true positive}}{\text{true positive + false negative}}; \tag{43}$$

and the false positive rate as

$$\text{false positive rate} = \frac{\text{false positive}}{\text{false positive + true negative}}, \tag{44}$$

Consider the LOS scenario. Let us define the overall network area as a circle $\mathcal{C}$ with radius R_{out} and consider a single BS located at the center of $\mathcal{C}$. Consider the legitimate area $\mathcal{A}_0$ as a rectangle of height H and length L and with nearest point to the center of $\mathcal{C}$ at a distance $R_{\min}$. The non-legitimate area is $\mathcal{A}_1 = \mathcal{C} \setminus \mathcal{A}_0$.

Since we stated that in the LOS scenario the attenuation incurred by a UE only depends on its relative distance to the BS we can here compute a closed form solution for the LLR in (7) and hence compare the machine learning-based solutions to the N-P classification.

Consider a UE u transmitting a message to the BS located at a distance R_0 from the BS. The probability othat u is located at a distance $R \leq R_0$ in $\mathcal{A}_0$ is

$$\mathbb{P}(R \leq R_0 | \mathcal{A}_0) = \frac{1}{|\mathcal{A}_0|} \int_{R_{\min}}^{R_0} R a(R) dR, \tag{45}$$

where $a(R)$ denotes the angle of the circular sector located at distance R intersecting area $\mathcal{A}_0$.

By taking the derivative of (45) respect to R_0 we obtain the probability distribution function (PDF) of u transmitting from a distance R_0 given that it is located in $\mathcal{A}_0$ as

$$p(R_0 | \mathcal{A}_0) = \frac{1}{|\mathcal{A}_0|} R_0 a(R_0). \tag{46}$$

Following the same reasoning and considering that the length of the arc of circle with radius R_0 located in $\mathcal{A}_1$ is $2\pi - a(R_0)$, we obtain the PDF of u being at a distance R_0 given that it is located in $\mathcal{A}_1$ as

$$p(R_0 | \mathcal{A}_1) = \frac{1}{|\mathcal{A}_1|} R_0 \left(2\pi - a(R_0)\right), \tag{47}$$

from which we obtain the closed form solution for (7)

$$\mathcal{L} = \log\left(\frac{|\mathcal{A}_1| a(R_0)}{|\mathcal{A}_0| \left(2\pi - a(R_0)\right)}\right), \tag{48}$$

The attenuation value measured at the BS for a user is given by (3), where we consider a unitary transmitting power for each user and a carrier frequency of 2.12 GHz.

Fig. 2 show the true positive rate vs. false positive rate obtained by the authentication system with 10^5 training points and 10^5 testing points. In particular we here compare the performance of the N-P detector with the CE-trained and with the MSE-trained MLPs. Furthermore we show the performance of the MLPs when the hidden layer is composed by $2, 4$ and 8 neurons.

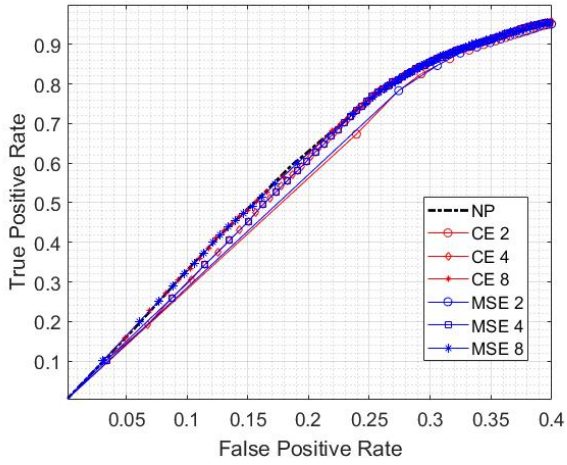

Fig. 2. True positive rate vs. false positive rate obtained in the LOS scenario with a single BS. Comparison between NP detector, MLP with MSE training and MLP with CE training with different number of neurons in the hidden layer.

We first notice that the CE-trained and MSE-trained MLPs obtain the same performance when considering the same number of neurons in the hidden layer. This confirms that the two training methods are performing hypothesis testing in an equivalent manner. Furthermore we notice that as the number of neurons at the hidden layer grows the performance of the MLP classificators approach those obtained by the N-P classificator, up to the point where they are the same with 8 neurons. This shows that both CE-trained and MSE-trained MLPs are performing the optimal N-P test and that convergence is obtained with a small number of neurons.

X. Numerical results: non-LOS scenario

A. *Shadowing effects*

In this section we consider the non-LOS scenario and we show that the ML-based solutions are convenient over the N-P-based one.

Fig. 3 shows a realization of the attenuation map. A spatial grid has been created in order to take into account the shadowing's spatial correlation over different locations in space. The network includes a single BS located at the map center and a squared authentic area $\mathcal{A}_0$, delimited in figure by the red line. Furthermore we consider two LOS paths, one parallel to the x axis and one parallel to the y axis and intersecting at the map center.

Consider the LLR (7). In the non-LOS context the computation of the two area dependent probabilities has no closed-form solution. A numerical solution is obtained by sampling the

attenuation values over the spatial grid of positions. Consider an attenuation value $\hat{a}$: the probability of measuring $\hat{a}$ given that the UE is located in area $\mathcal{A}_0$ is given by the number of positions (x_u, y_u) inside $\mathcal{A}_0$ where the measured attenuation $a(x_u, y_u) = \hat{a}$ over the total number of positions (x_u, y_u) in the entire map with the same attenuation $\hat{a}$, i.e.

$$\mathbb{P}(\hat{a}|\mathcal{A}_0) \approx \frac{\text{number of positions } (x_u, y_u) \in \mathcal{A}_0 \text{ s.t. } a(x_u, y_u) = \hat{a}}{\text{total number of positions } (x_u, y_u) \text{ s.t. } a(x_u, y_u) = \hat{a}} \tag{49}$$

An approximation of equation (7 is hence obtained as

$$\mathcal{L} \approx \frac{\text{number of positions } (x_u, y_u) \in \mathcal{A}_0 \text{ s.t. } a(x_u, y_u) = \hat{a}}{\text{number of positions } (x_u, y_u) \in \mathcal{A}_1 \text{ s.t. } a(x_u, y_u) = \hat{a}} \tag{50}$$

The approximation gets closer to the real value as the number of grid points over the map increases, as an higher number of points means a better statistical characterization of the attenuation over the map area.

Fig. 4 shows the true positive rate vs. the false positive rate obtained for the attenuation map in Fig. 3. In particular we here compare the results obtained with approximation (50) with the results obtained with the MSE trained MLP with different number of neurons in the hidden layer. For the computation of (50) we built a grid with $4.46 \cdot 10^6$ space points, whereas for training and testing the MLP we used respectively 10^5 and 10^6 points. We notice that the MLP achieves better results than the N-P-based detector. This means that the considered number of grid points is not sufficient to approximate the optimal N-P solution. Furthermore we tested the MLP with $1, 3$ and 10 neurons at the hidden layer. We see that the performance with 1 and 3 neurons are the same, whereas with 10 neurons we achieve the N-P solution, based on the results of the previous section.

Comparing the number of grid points and the number of training points used for the MLP we can hence conclude that the MLP-based solution is advantageous over the N-P-based one, as it requires a smaller number of points to achieve the optimal solution. Furthermore this implies that the MLP-based authentication system can be implemented without a-priory knowledge of the PDF of the hypothesis to be tested.

Consider the network in Fig. 5. $N_{\text{bs}} = 5$ BSs gather attenuation values from the UEs. Each BS has its own attenuation map given by path loss, shadowing and fading realization. Each BS gathers the attenuation value of the transmitting UE and sends it to the central unit, which builds the attenuation vector $\boldsymbol{a}$ that is used for discriminating the location of the considered UE.

Fig. 6 compares the performance of the MLP-based solution with those of the RNN-based solution. Results have been obtained for the network configuration in Fig. 5. Both architectures

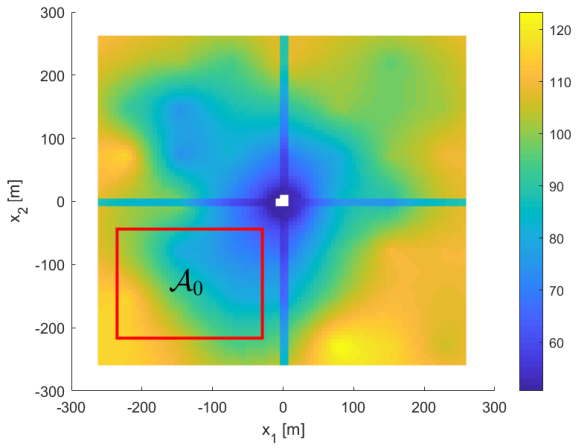

Fig. 3. Example of a realization of the attenuation map in the non-LOS scenario considering only the shadowing effects.

have been trained over a set of 10^5 attenuation vectors and tested over the same set of 10^5 attenuation vectors. Notice that the training set of the MLP comprises attenuation vectors measured in both $\mathcal{A}_0$ and $\mathcal{A}_1$. The MLP is trained with MSE loss function and we compare the results obtained with a number of neurons in the hidden layer from 1 to 5. The performance of the RNN-based solution are here presented for a number of neurons in the hidden layer from 1 to 5. Both architectures are composed by an input layer, a hidden layer and an output layer. Curves for the MLP are obtained by varying the threshold value λ, whereas for RNN are obtained by varying the threshold value γ. We notice that the performance of the RNN with £4£ neurons in the hidden layer are better compared to those obtained with the RNN with 4 and 5 neurons in the hidden layer. This is due to the fact that the features extracted by 3 neurons in the hidden layer better represent data in $\mathcal{A}_0$, whereas 4 and 5 neurons are over-representative. This confirms that the feature extraction process of the RNN is an efficient way to represent a set. We further notice that the performance of the RNN with 2 neurons in the hidden layer are worse than those obtained with the RNN with 3 neurons in the hidden layer. This is due to the fact that 2 neurons are not sufficient for the feature extraction process to provide a good characterization of area $\mathcal{A}_0$. We notice that the performance obtained with the MLP-solution are better than those obtained with the RNN.

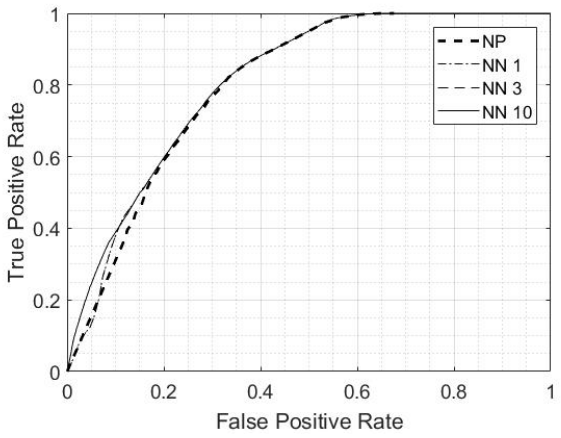

Fig. 4. True positive rate vs. false positive rate for the attenuation map in Fig. 3. Comparison between the N-P-based and the MLP-based detectors

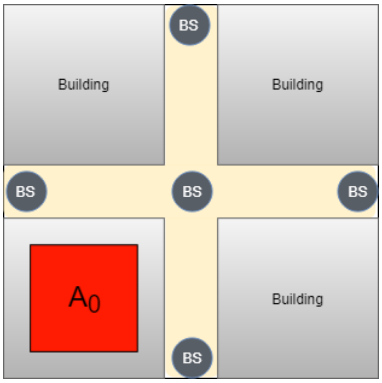

Fig. 5. Scenario with multiple BSs: each one gathers the attenuation value of the transmitting user and passes it to the central unit that builds the attenuation vector $\boldsymbol{a}$ used for discriminating the location of the user.

B. Fading effects

Let us now include the fading effects. The attenuation in point (x_u, y_u) is a Gaussian random variable with zero mean and variance given by the inverse of the value of the map in (x_u, y_u).

In order to compute the LLR in (7) we should here consider a N_{bs}-dimensional multivariate Gaussian distribution. However, as this equation has no closed-form solution and we showed in X-A that the MLP-based solution achieves the optimal results of the N-P detector with fewer points we implement the MLP-based solution. A larger number of BSs means a larger feature space to be fed to the MLP. We hence expect that the performance of the system increase with the number of BSs.

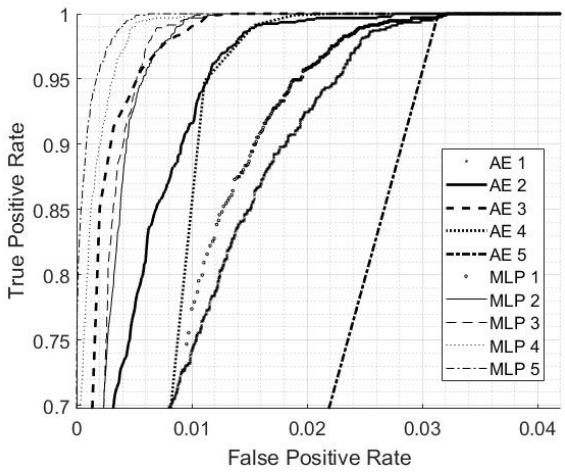

Fig. 6. True positive rate vs. false positive rate for the network configuration in Fig. 5. Comparison between the MLP-based and the RNN-based detectors with different number of neurons in the hidden layer.

Different fading realizations lead to different statistical characterization of the attenuation values measured in both areas. Therefore training the MLP over a single fading realization could lead to a non-optimal result in the successive realization. We hence propose to train and test the MLP over N_{avg} fading realizations for each BS.

Fig. 7 shows the true positive rate vs. the false positive rate in non-LOS scenario with fading effects. Different number of fading realizations for training a MLP with MSE loss function has been considered. We notice that as the number N_{avg} of fading realizations used for training increases the performance of the network get closer to those obtained without considering the fading effects. Furthermore we notice that, as expected, with a larger number of BSs the performance of the system increase, as can be seen comparing the curve in Fig. 4 and the 'No fading' curve in Fig. 7.

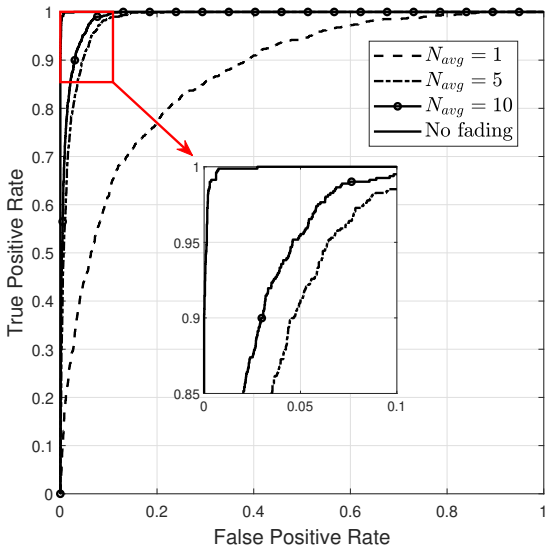

Fig. 7. True positive rate vs. false positive rate in non-LOS scenario with fading effects. Comparison between training with different number of fading realizations.