A PSD-based fingerprinting approach to detect IoT device spoofing

Florent Galtier, Romain Cayre, Guillaume Auriol, Mohamed Kaâniche, Vincent Nicomette

A PSD-based fingerprinting approach to detect IoT device spoofing

Florent Galtier*, Romain Cayre*‡, Guillaume Auriol*†, Mohamed Kaâniche*, Vincent Nicomette*†
*CNRS, LAAS, 7 avenue du colonel Roche, F-31400
†Univ de Toulouse, INSA, LAAS, F-31400
‡APSYS.Lab, APSYS
Email: *firstname.lastname@laas.fr ‡firstname.lastname@airbus.com

***Abstract*—Spoofing attacks are generally difficult to detect and can have potentially harmful consequences on computer networks and applications. Wireless IoT networks, in the context of smart buildings or smart factories, are particularly vulnerable to these attacks. In this paper, we present a new physical device fingerprinting approach aiming at detecting spoofing attacks in wireless IoT environments. The proposed approach is based on the analysis of some properties of the physical signals emitted by connected devices, using their Power Spectral Density (PSD) to extract a frequency profile of their communications. This approach does not require any expensive equipment, is easy to deploy, and is resilient to non predictable phenomena in transmissions. The detection of spoofing attacks consists in comparing the fingerprint of a transmitting device with previously stored fingerprints of legitimate devices, by measuring the similarity of the corresponding PSDs and applying a community detection algorithm. The efficiency of this approach has been successfully tested using various experimental setups with connected devices supporting different wireless protocols (BLE, Zigbee). We also discuss the practical applicability of our approach, e.g. in an industrial environment by analysing its scalability and proposing solutions to tune and optimize its deployment at a large scale.**

***Index Terms*—Smart environments, IoT, fingerprinting, SDR, PSD**

I. Introduction

The rapid and massive expansion of IoT devices in recent years has led to the development of a large number of heterogeneous wireless communication protocols designed to adapt to the constraints of these new systems (e.g. low power consumption, ...), while security is generally not considered as a primary concern. Indeed, many security vulnerabilities affecting wireless IoT protocols have been published in recent years [1] [2]. Several studies have also shown that multiple wireless-based attacks targeting IoT devices can be easily performed, using low cost hardware and open source software.

Many of these attack strategies rely on *spoofing* operations, allowing an attacker to easily impersonate a legitimate node. Multiple examples can be highlighted: Man-in-the-Middle, forged traffic injection, deauthentication, etc. These attacks allow an attacker to compromise the availability, confidentiality or integrity of a communication by reproducing data identifying a legitimate device. However, many of these attacks are out of the scope of traditional network security monitoring tools, as it is quite easy to run spoofing attacks that remain undetected at link-layer and above OSI layers.

One of the most relevant defensive strategies to detect this type of active attacks is fingerprinting. Indeed, this technique makes it possible to identify an unknown device based on some physical characteristics, even if it attempts to impersonate a legitimate one. However, existing fingerprinting techniques are generally not easy to deploy in practice. Indeed, some of them require the use of costly *Digital Signal Processing* hardware, whereas in the context of IoT there is a strong requirement for cost reduction. Moreover, most of these techniques have been validated only by simulation, and not experimentally under real-world conditions which might significantly differ from simulations.

In this paper, we present a novel method for fingerprinting IoT devices that aims to address these limitations. Unlike existing solutions which generally focus on the analysis of the transient phase during which a device starts communicating, which requires the use of high precision equipment, our method is based on the analysis of the entire physical signals emitted by the devices without relying on expensive hardware, and is validated and assessed experimentally in a variety of experimental conditions.

The proposed approach is designed to provide a complementary protection against spoofing attacks in Smart buildings, factories or homes. It focuses on analysing the Power Spectral Density (PSD) of physical signals to identify legitimate objects and use the fingerprints of these objects to detect potential intruders in the wireless environment.These fingerprints allow each device to be uniquely identified because they reflect the imperfections of hardware emitter components which are specific to each device. PSD has been chosen instead of the most commonly used Fast Fourier Transform (FFT), particularly for its resilience to phase offsets. The effectiveness of this technique is based on the difficulty for an attacker to clone the imperfections of the hardware that produce the spectrum variations. Indeed, to our knowledge, cloning such a fingerprint would require the use of a very expensive transceiver capable of operating at very high sampling rates, while being able to compensate for the impact of its own imperfections on the received and sent signals, which implies building a complex model of the interferences it produces.

The contributions of this paper are threefold:

- We present a novel low-cost fingerprinting method based on the PSD analysis and classification of IoT devices

entire signals, unlike most of existing solutions that focus on the transient phase of the emitter, and thus require high quality receivers. Our approach is generic and protocol-independent, supporting the main modulation schemes traditionally used in smart environments.
- We tested experimentally our approach to validate its practical feasibility, showing promising results in terms of detection efficiency of illegitimate devices and scalability.
- We make the data collected during our experiments and our results publicly available to facilitate the reproducibility of our work and enable the development of other fingerprinting solutions adapted to our context (see [3]). We are not aware of similar data available to the scientific community.

The paper is organized as follows. In Section II, we discuss related work focused on wireless devices identification solutions. Then, we give an overview of our approach in Section III. The different steps of our approach (the construction of the fingerprints and the detection process) are detailed in Section IV. In Section V, the results of various experiments carried out in real-word conditions are presented to assess the relevance of our approach. The performance and scalability of the approach are also discussed. The generalization of our approach to address dynamic environments with mobile objects and real-time execution is discussed in Section VI. Finally, Section VII concludes the paper and outlines future work.

II. Related work

This section describes some related works focusing on fingerprinting methods for wireless devices.

A common approach is based on the analysis of the transient phase during which the transmitter starts to communicate. Indeed, this transient phase exhibits certain distinctive characteristics that are directly influenced by the components involved and the manufacturing process of the transmitter. Some examples are proposed in [4]–[6]. Another relevant work by Boris Danev's [7], [8] focusing on RFID devices fingerprinting, relies on an analysis of the transient phase and of the response time together with the identification of the modulation used by the device. These solutions that focus on the analysis of the transient transmission phase are quite demanding in terms of reception quality, and require the use of expensive receivers to collect a maximum amount of information over a short period at the beginning of the transmission of a frame, which is difficult in practice. Our objective being to propose a low-cost approach, the solution explored in this paper is based on the monitoring and acquisition of the entire signal, which requires less precision of the receiver to get the same amount of data from the emitter. We also want to be able to recognise devices without having to identify the specific modulation used as proposed by Danev.

Another approach, used in PARADIS [9] for IEEE 802.11 devices, consists in capturing the frames and their demodulated version, in order to create an ”ideal” version of the modulated signal and then compare it with the received signal to capture specific artifacts that are inherent to the transmitter or the channel. However, this approach is protocol-dependent and is difficult to generalise to other modulation schemes and protocols. Based on our experience, capturing and processing signals in real-time is challenging due to potential desynchronisation of the received signal and the regenerated one.

Other state of the art solutions use Physical Unclonable Functions (PUF) to support device authentication and identification. They actually implement ”challenge-response” mechanisms related to the physical characteristics of the devices, and cover much more than simple signal processing, as they are based on any uniquely identifiable mechanism that can be linked to any physical object. Some works are related to Radio-Frequency based PUF for IoT security, such as the RF-PUF [10], an approach based on amplitude, phase and DC offsets relatively to an ”ideal signal” in protocols using a 16-Quadrature Amplitude Modulation (16-QAM), e.g., such as for IEEE 802.11, and on classification with a neural network to identify devices. These solutions are usually not generic and costly and their efficiency is generally assessed by simulation.

Some interesting works focus on the characterization of the device behaviour when faced with specific errors or alterations of the packets. For instance, in [11], the authors study the reception rate of different devices when packet headers are modified, to uniquely identify devices based on their tolerance to those modified packets. Some physical fingerprinting methods have been also proposed for specific applications or devices, for example to distinguish two brands of mobile phones [12], or in [13]. These methods focus on specific protocols and on the differences they can detect among a given set of devices. Our approach, however, aims to address all possible protocols.

In the general context of device fingerprinting, several studies focused on upper-layers fingerprinting, analysing features such as counters, headers content and software or hardware specific parameters. For example, several fingerprinting and anti-fingerprinting methods have been investigated in the context of web-browsers in order to identify a user without the use of cookies [14]–[16].

Our goal is to propose a new device fingerprinting approach that can efficiently mitigate spoofing attacks that are not detected at the link-layer, while being inexpensive (e.g., not requiring high-quality receivers) and easy to deploy. Our approach based on the PSD of the physical signal, sampling at a low rate the entire signal, is designed to fulfil this objective.

III. Approach overview

This section outlines the principles of our approach and our assumptions about the targeted context and threat model.

A. Context

Our approach is intended for use in a smart environment, in which the *legitimate connected devices* to be monitored are deployed at specific locations of the environment , using heterogeneous wireless communication protocols, such as BLE, Zigbee, WiFi, ... The generalization of our approach to IoT environments including mobile objects is discussed in Section

VI. Typical use cases include smart buildings equipped with many sensors aiming at optimizing the energy consumption of the building, a smart factory in which some legitimate connected devices are used to perform some specific tasks, or smart homes in which different sensors of a physical intrusion detection system are deployed at different locations. The proposed approach aims at detecting an attacker entering the smart environment, carrying some connected devices implementing various wireless protocols that they use to impersonate a legitimate device. We assume that the attacker is able to run spoofing attacks targeting any layer above the link layer of the OSI model. For that purpose, they may use a programmable dongle supporting the targeted protocol, or an SDR (Software Defined Radio) to perform replay attacks or to re-create a physical-layer signal corresponding to the link-layer data to be transmitted. However, we assume that the attacker is not able to exactly replicate the physical imperfections of the legitimate transmitter. Indeed, for replay attacks, a high quality SDR would be required and, for a pure spoofing attack, the attacker would also need high expertise in signal processing to be able to replicate such imperfections while sending arbitrary data.

B. Architecture of the approach

Our approach is designed to create physical fingerprints of the legitimate devices in the environment, and then to detect potential intrusions by comparing with these fingerprints captured signals that are identified at the link-layer as being transmitted by one of the registered devices. It can be decomposed into three main steps:

1) *Fingerprint creation:* acquisition of Physical Protocol Data Unit (PDU) signals from each legitimate device, and computation of the PSDs associated to these different signals. Each set of PSDs from a device constitutes its fingerprint. The fingerprints of all legitimate devices are then saved in a fingerprint database.
2) *Cluster computation and Device fingerprints similarity analysis:* measurement of the similarities between all the device fingerprints, then clustering of the different PSDs, using a community detection algorithm. The similarity matrix and the community for each PSD are then saved in an additional database, called *device communities* database, for further analyses.
3) *Intrusion detection:* The PSD of each incoming signal whose address corresponds to a legitimate device is compared to the fingerprints previously registered in the database in Step 1, by computing its similarity with each fingerprint in order to estimate whether or not it belongs to a known community.

This approach is illustrated in Figures 1 and 2. Fingerprint creation is first performed off-line, for each legitimate device of the smart environment. These fingerprints along with the device similarity matrix and the identified communities, are saved respectively in the fingerprint and the device communities databases. This fingerprint creation is performed inside the smart environment, including all the legitimate devices at their dedicated location, by means of Software Defined Radio (SDR) devices, purposely installed in the environment at strategic locations. The intrusion detection is then performed on-line, by capturing the signals emitted by the different connected devices of the smart environment (including possible malicious devices carried by attackers entering the environment and impersonating a legitimate device at link layer[1]) and comparing these signals to the previously saved fingerprint. This comparison algorithm takes as inputs the similarity matrix and the communities previously saved in the device communities database. Note that it is easy to integrate new legitimate objects into the smart environment. This simply consists in executing the two first steps described above in order to create the fingerprint of this new object, and updating the similarity matrix and the clusters.

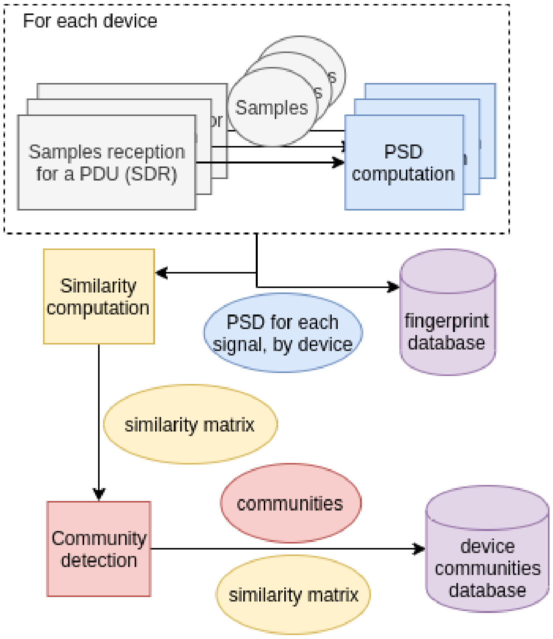

Fig. 1: Fingerprint creation and cluster computation

IV. Detailed description of the approach

In this section, we detail the different steps of our approach and their integration. Subsection IV-A explains the motivations for choosing the PSD for the creation of fingerprints, and how these fingerprints are computed. Subsection IV-B describes the algorithms used to measure the similarity between PSDs and the methodology to compute PSD clusters. Finally, subsection IV-C describes how the intrusion detection is performed for each incoming signal, based on these clusters and the PSDs stored in the database.

A. Fingerprint creation

1) Signal acquisition: To capture the signals transmitted by connected devices, a receiver that can support all different types of modulations is required, while being affordable and

[1] A device can be impersonated either by using its link-layer address if it exists, or by mimicking its behaviour.

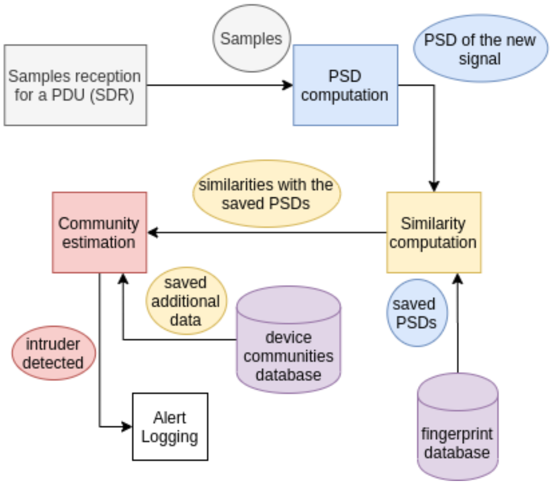

Fig. 2: Intrusion detection

easy to use with a standard computer. This is why, we have chosen to use Software Defined Radio (SDR) based devices such as the HackRF One [17] and the LimeSDR Mini [18].

These SDR receivers can be set to operate at given frequency and sampling rate, and transmit a stream of the captured signals to the computer dedicated to processing them.

The SDR receivers provide streams from which Physical PDUs must be extracted. For this purpose, we used rising and falling amplitude edges to detect PDUs, then tried to demodulate them according to the protocol studied. When a valid PDU is found, the signal corresponding to its entire transmission is then saved.

2) Power Spectral Density Analysis: In order to extract the frequency characteristics of a signal, a conventional approach would be to use a Discrete Fourier Transform (or DFT). Indeed, the DFT is a reversible transformation that converts a discrete signal in the time domain into an amplitude distribution in the frequency domain, limiting the loss of information. However, in our approach, we want to be able to deal with non perfect transmitters that possibly produce different phase offsets for each transmission. As the frequency is proportional to the derivative of the signal's phase, non predictable phase offsets lead to different spectral representations of a same device. As a consequence, the DFT may be problematic for the construction of our fingerprints.

A more relevant approach, commonly used in signal processing, is based on the Power Spectral Density, or PSD, of the signal. The PSD measures the power distribution of the frequencies of an entire signal, but with a loss of time information (unlike DFT, PSD is not a reversible transformation). It is calculated as follows:

$$PSD(s(t))(f) = DFT(s(t) \cdot s^{*}(-t))(f)$$

the ”·” operation being the convolution between signals and s^{*} being the conjugate form of the temporal signal s.

Among the interesting properties the PSD exhibits, the one we are interested in is its independence from phase offset. PSD, like DFT, has the property of isolating the spectral components of a signal. We assume that a specific object, in addition to the frequencies related to the payload sent, exhibits specific transmission profiles on different frequencies, which are highly dependent on its physical components and the quality of the manufacturing. As illustrated in Figure 3, we believe the PSD is a relevant candidate to efficiently isolate different emitters by their frequency usage profile[2]. In this figure, two different Bluetooth Low Energy (BLE) devices are analysed: a dongle and a lightbulb. The PSD curves correspond to three PDU transmissions from each device. It can be seen that different PSD profiles are associated to each BLE device.

We then decided to use PSD-based fingerprints of a device in our approach. To build these fingerprints, we record a set of physical signals from the device, each corresponding to the entire emission of a single PDU, then we compute the PSDs of those signals and store them as the fingerprint of the device.

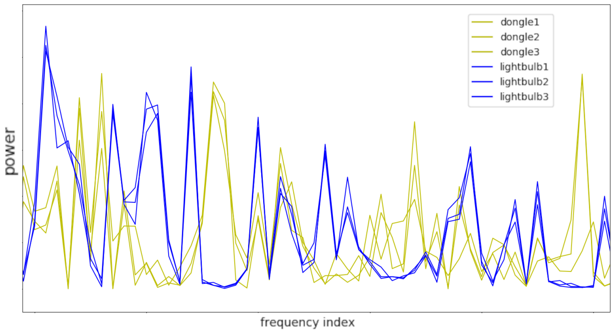

Fig. 3: PSD for two distinct BLE devices (3 PDUs each)

B. PSDs similarity analysis and clustering

1) Similarities computation: Our approach is based on the measurement of similarities between pairs of PSDs. For that purpose, we experimented several metrics:

- Metrics based on the occurrence of specific frequencies among top 10% frequencies with highest measured power: The similarity measure corresponds to the number of common frequencies in the top 10% of the PSD pair to be compared.
- Metrics based on top 10% highest power frequencies, and the difference between their ”rank”: we proceed the same way as for the previous metric, taking into account in addition the difference of ”rank” of the common frequencies in the sorted 10% of highest power, to compare their ”importance” in the signal.
- Metrics based on top 10% highest power frequencies, and the maximal ”rank” of each one: instead of comparing the ranks, we take into account the highest one in the pair to compare, to measure its maximal importance.
- Metrics based on the distance between the PSDs curves.

The first type of metrics aims at isolating the frequencies that are actually specific to the devices, and compare their

[2] We discuss the experimental efficiency of our approach in Section V.

importance in the different signals. However, considering only the frequencies with the highest power may lead to ignoring potential weaker parts of the signal's spectrum, that are also characteristic of the device. Similarly, isolating only the weakest parts would ignore relevant frequencies. Therefore, we also experimentally explored the possibility to only use the "ranks" over the entire frequency spectrum. However, with the whole spectrum or only a part of it, the results were at best equivalent to those obtained with distance-based metrics, and at worst equivalent to a random identification in the case of only analysing the presence of these specific frequencies among the most powerful ones. As a consequence, and also because of its simplicity, we opted for distance-based metrics.

We tested several distances, such as the euclidean distance between PSDs, average and max by-frequency L1 and L2 distances, also called the Manhattan distance and the Euclidean distance. We obtained the best results with the max by-frequency L2 distances between PSDs as the distance between two PSDs defined as follows:

$$D(PSD1, PSD2) = \max((PSD1(f) - PSD2(f))^2) \quad (1)$$

This distance was thus adopted in our approach. The next step consists in defining a similarity measure to estimate the proximity between a given PSD and other PSDs. Since our PSDs are normalised, the distance D is always between 0 and 1. Hence, we decided to simply take $1 - D$ as our similarity measure. However, in order to have a clear separation between similar and dissimilar PSDs, we decided to amplify the differences by lowering the values close to 0, leading to the following similarity measure:

$$S(PSD1, PSD2) = [1 - D(PSD1, PSD2)]^{amp} \quad (2)$$

Parameter amp is evaluated empirically for each protocol.

2) Community detection: The next step is then to create "clusters" of PSDs corresponding to the physical PDUs from a given source. To visualize the similarity between PSDs, a graph is generated in which each node is associated to one individual signal's PSD, and the edges between the nodes are labeled with a weight corresponding to the similarity measure between the PSDs. A community detection algorithm is then applied in order to group into clusters similar PSDs that are likely to correspond to communications from the same device.

Several community detection algorithms are proposed in the literature. Random walk algorithms, such as walktrap [19], randomly crawl a graph to compute, for each pair of nodes, estimates of the probabilities to move from one to the other in a given number of steps. Then, based on these probabilities and nodes degrees it estimates the likelihood for each pair of nodes to be in a same community. Communities that are close according to a distance based on these probabilities and degrees are then grouped iteratively, starting with each node in an independent community. Another approach is based on k-means and k-medioids [20] algorithms, which take as a parameter the number of clusters to build, start by randomly adding nodes to clusters, then iteratively compute the center of each cluster (either a barycenter for k-means, or the node considered as the most central for k-medioids) and add again nodes to the cluster of the nearest center. Modularity-based algorithms, such as Girvan-Newman [21] or the fast-greedy [22] (which is faster in the case of sparse graphs), add nodes to individual clusters, then merge iteratively the two clusters that maximise the modularity of the whole graph (the modularity measures the quality of the partition based on the number of inter-community and intra-community edges), forming a dendrogram with the successive merges.

In our approach we chose the fast-greedy algorithm to build "communities" of PSDs, each corresponding to a given object, for several reasons:

- Weights can be assigned to edges.
- It is quite fast. A graph with n nodes, m edges and a dendrogram describing the community structure of depth d, results in a $O(md \log_2 n)$ time complexity . Indeed, in our case, since we have a near-complete graph $m = O(n^2)$, and the dendrogram depth is in $O(\log_2 n)$, the fast-greedy algorithm has a complexity in $O(n^2 \log_2^2 n)$.
- It is deterministic, unlike random walk algorithms.
- In the implementation we used (see Section V), it can estimate the number of clusters to create, unlike k-means or k-medioids for which this number must be provided by the end user. This would be difficult in our approach since, even if the number of legitimate devices is known, the presence of outliers could lead to the creation of additional clusters, whose number is unknown.

3) Cleaning and saving the data: The final step is to identify potential outliers, to avoid taking them into account in the cluster databases, and to find out whether an incoming signal being analyzed corresponds or not to one of the known devices. We define a PSD as *correctly identified* by the community detection algorithm if it is included in a cluster containing a majority of the PSDs from the same device.

Three types of outliers can be distinguished:

- *Unidentified signals:* signals forming small external communities outside the "main" communities containing a majority of signals from a device.
- *Identified distant signals:* signals included in the right community, but located far from its other members.
- *Wrongly identified signals:* signals included in a wrong community.

We first address the issue of unidentified signals, forming small external communities, by removing the clusters with too few members which are likely to correspond to outliers. For each experiment, we empirically defined a threshold below which the cluster is considered too small, relatively to the number of signals per device in the experiment.

Identified distant outliers which correspond to PSDs correctly identified in a cluster but far from the other members of the cluster are also removed. These are problematic because a signal from another source, with interferences from the environment, may be close to those outliers, and hence could be recognised as member of the same cluster. To identify these outliers, we measure, for each PSD, its average similarity with the other PSDs of the same cluster. Then, we calculate, for each one of them, the average similarity between a given PSD and the other members of this cluster:

$$\overline{S}(PSD, cluster) = \frac{1}{|cluster|} \sum_{PSD_i \in cluster} S(PSD, PSD_i) \quad (3)$$

Assuming that the signals follow a normal distribution around the average, and that most of the signals are legitimate, an assumption consistent with our experimental results, we measure the standard deviation of these average similarities $\sigma_{\overline{S}}$ in a given cluster, and remove the PSDs whose average similarity with the cluster is too far from $\overline{\overline{S}}$, the average of the different $\overline{S}$ for this cluster defined as follows:

$$\overline{\overline{S}}(cluster) = \frac{1}{|cluster|} \sum_{PSD_i \in cluster} \overline{S}(PSD_i, cluster) \quad (4)$$

We chose to remove all PSDs that have less than 99.7% chances to be inside the cluster according to the hypothesis of a normal distribution, which means the PSDs whose associated $\overline{S}$ is below a threshold defined as follows:

$$threshold = \overline{\overline{S}} - 3 * \sigma_{\overline{S}} \quad (5)$$

These $\overline{S}$ values, along with the average and standard deviation for each cluster, are saved to be used subsequently for intrusion detection, as explained in IV-C.

After the removal of these outliers, we obtain a list of "clean" clusters and the corresponding signals. Then, the following information is needed for intrusion detection:

- The PSDs of the signals used for this step.
- The similarity matrix between the different PSDs.
- The cluster associated with each one of them (outliers are labeled as in an "outlier cluster" numbered -1).
- The average and standard deviation of the average similarity between a PSD and the rest of the cluster for each cluster.

C. Detection

The detection phase consists in analysing the signals captured in the operational smart environment in order to estimate whether they come from a known legitimate device or from an unknown one. A new incoming physical PDU is hence analysed as follows:

- The signal corresponding to this PDU is isolated according to the approach described in Section IV-A1.
- The PSD of the signal, noted P_s is compared to the other ones (by computing the similarity measure defined by equation 2).
- The average similarity for each PSD in each cluster is computed (except cluster -1) as defined in equation 3.
- This average similarity $\overline{S}(P_s, cluster)$ is compared to the average $\overline{\overline{S}}$ (as defined in equation 4) and standard deviation $\sigma_{\overline{S}}$ of $\overline{S}$ previously saved for each cluster, identifying possible clusters for the objects based on the proximity of its average similarity to the reference one, using the threshold defined in equation 5. If the PSD fits with more than one cluster, the most relevant is selected according to the similarity to the reference average similarity and the standard deviation of those similarities. Otherwise we consider the signal as an anomaly (that may correspond to an attack).

Note that we deliberately did not consider the approach that consists in re-computing the clustering algorithm to obtain the cluster in which the new signal's PSD would be located. Indeed, this algorithm would take too much time whereas our detection approach must be performed in real-time.

V. Experiments

This section presents several experiments we carried out in order to assess the relevance of our approach. We first describe the experimental setup in Section V-A. Section V-B is dedicated to the presentation of small scale experiments, 1) using two devices sending the same data each at two different positions to evaluate the impact of position on device recognition, and 2) using three different devices at static positions, still sending the same data. Section V-C is dedicated to the presentation of higher scale experiments on sets of 10 different devices, to evaluate our performances with more emitters. Finally, in Section V-D, we describe some experiments carried out on sets of around 20 identical devices. A summary of the different parameters used in all the experiments, which we further explain through the following subsections, can be found in Table I. The section ends with a discussion about the scalability and the performances of our approach. Note that the signals we captured, along with the similarity matrices and the results from our clustering, can be found in [3].

A. Experimental setup

Our objective was to design a tool that would be easily accessible to researchers, requiring more affordable hardware than the approaches based on high-precision captures. Accordingly, we selected two different SDRs during our experiments: the HackRF One, cheaper and faster to set up, for prototyping, and the Lime SDR Mini, a little more expensive but offering better capture precision and stability, for the final results. Our implementation uses Python3 and the igraph [23] package for graph creation and management as well as for the fast-greedy community detection algorithm. For all experiments, the SDR device (HackRF One for testing, then LimeSDR for the final results) was plugged into a laptop located at a specific position.

For our experiments, we used very common connected devices, that anyone can easily buy, including connected lightbulbs, motion detectors, mobile phones or USB dongles. The selected devices cover the three main modulation types:

- *Amplitude modulation :* used for example by remote controls communicating with a modulation called Pulse Width Modulation (PWM), transmitting during predefined time slots, the duration of the transmission within the slot encoding the different bits.
- *Phase modulation :* used by objects communicating with Zigbee protocol, which is based on IEEE 802.15.4 using

TABLE I: Parameters of the different experiments

experiment	number of devices	sets per device	set size for fingerprint creation	set size for testing	number of signals for fingerprint creation	total number of signals for testing
B-1	2	2	133	67	26600	13400
B-2	3	1	67	33	20100	9900
D	10	1	67	33	67000	33000
E-Zigbee	20	1	67	33	134000	66000
E-BLE	18	1	67	33	120600	59400

a Gaussian Minimum Shift Keying (GMSK, a phase modulation where the sign of the phase variation is used to encode the bits).

- *Frequency modulation :* used by objects implementing BLE and Enhanced ShockBurst protocols, both using a Gaussian Frequency Shift Keying (GFSK).

For our experiments, all incoming signals are demodulated to ensure that they are generated from an identified device, before processing them by our approach. Acquisition, demodulation and signal processing were all implemented in Python3 to ease prototyping. An alternative solution would be to use a compiled language to improve real-time performances.

B. First experiment

For the first experiment, we considered a reduced set of BLE devices to assess, at a small scale, the efficiency of our approach in creating distinct fingerprints for different devices. For this experiment, we considered two different setups. For each setup, the experiment consisted first in running the fingerprints creation and PSD clustering steps of our approach and then in evaluating the efficiency of our detection algorithm in the presence of illegitimate devices. In the first setup (B-1 in I), a connected power outlet and a BLE embedded chip were used to transmit the same data. Two different locations are also considered to capture the PDUs sent by the two sources.

As illustrated in Figure 4, the PDUs from the outlet and those from the dongle are perfectly separated into two distinct clusters, independent of the location of the signal acquisition device, without errors and without any outlier found during the cluster creation step. The nodes represent the PSDs and the edges represent the similarities between PSDs, weighted by our similarity measure. The graph is visualised using a positioning algorithm known as the *spring layout*, which positions nodes on a graph by grouping ”close” nodes relatively to the weight of edges linking them, representing their similarities. This graph is only used for this visualisation purpose, and is not used in the fingerprinting or intrusion detection algorithms.

In the second setup (B-2 in I), we run a similar experiment with three different transceiver: the same BLE-connected power outlet and two different BLE transceivers from different manufacturers (CSR, which is a BLE dongle, and BLE-chip, which is a Raspberry Pi’s Broadcom embedded chip). The results of the clustering algorithm are displayed in Figure 5, again showing a clear separation between the different devices.

In order to assess the detection efficiency of illegitimate devices, we collected 200 signals from each device, and adopted a 100-cross validation approach. The devices are first separated into two sets, one playing the role of the legitimate device, the other the role of the attacker (in the second setup of the experiment, we considered one device as the attacker and two as legitimate).

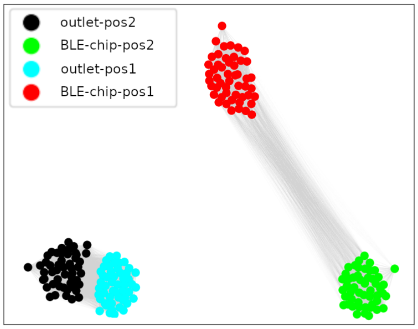

Fig. 4: First experiment results Visualisation - Setup B-1

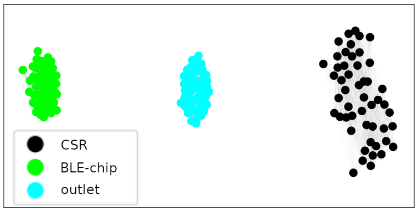

Fig. 5: First experiment results Visualisation - Setup B-2

At each iteration, the data set of each legitimate device is split in two parts: two thirds of the data set are used to compute the fingerprints, similarities and clusters, and the last third is used to assess the detection efficiency (to evaluate the false positive rate). Similarly, one third of the data set associated to an attacker device is used to run the detection algorithm and check whether the corresponding PSDs are correctly included in new clusters, different from those associated to the legitimate devices, or are considered as outliers. We chose to use only a third of the attacker’s PDUs for each test to reduce the difference between legitimate and non legitimate packet numbers. In this experiment, we used 100 signals in each set (each position for each device), ran 100 iterations of this process, and decided to consider a cluster to be too small if it contains less than 49 members. We used the same values for those parameters in each of the following experiments.

The results obtained after running those 100 iterations are presented in Table II. The metrics used are defined as follows:

- *Accuracy*: assesses the success rate of our algorithm, calculated with the formula $\frac{TP+TN}{Total}$.

- *Precision*: is related to the probability of false alarm, given by $\frac{TP}{TP+FP}$.
- *Recall*: is related to the non-detection probability, calculated with the formula $\frac{TP}{TP+FN}$.

With TP/FP being the true/false positive rates, and TN/FN being the true/false negative rates.

TABLE II: First experiments - results

Metric	B-1	B-2
Accuracy	91.73%	100%
Precision	83.46%	100%
Recall	100%	100%
TP	33000	33000
TN	27541	66000
FP	5459	0
FN	0	0

From this first experiment, we conclude that our approach is able to distinguish different objects on small sets. It can also be seen from the first setup of the experiment that the position has a significant effect on the frequency profile of the devices, but no significant impact on the efficiency of our approach to separate different emitters. The sensitivity of the frequency profile to the position can be explained by the impact of multipath-delay in wireless communications, especially indoors, that generates Inter-Symbol Interferences (ISI) in the signals due to reflections on different surfaces, and hence depends on the positions of the emitter and receiver and the surfaces present around them.

C. Generalisation - second experiment

In this experiment, we validated the approach at a larger scale, using ten different BLE devices: 1) a Bluetooth USB dongle from Cambridge Silicon Radio, 2) an iPhone, 3) a Samsung smartphone, 4) a raspberry pi 3B (using its Broadcom BCM43438 WiFi/BLE chip), 5) a WiFi/Bluetooth embedded chip (Qualcomm Atheros QCA6174), 6) an electrical outlet with a Texas Instruments (TI) BLE transceiver, 7) 2 different connected lightbulb models, also using TI BLE transceiver, 8) a thermometer using a TI BLE transceiver, and 9) a Bluetooth-connected key ring (using a BK3231 chip).

The results of the clustering algorithm are displayed on Figure 6. Accuracy, Precision and Recall results generated from 100 cross-validation assessments, with the same proportions as the previous experiment, and selecting each time randomly half of the emitters as intruders, are presented in Table III.

TABLE III: Different devices, BLE - results

Metric	BLE different devices
Accuracy	91.50%
Precision	85.81%
Recall	98.01%

The visualisation of the PSDs and their similarities shows the presence of several outliers, that form small clusters of one or two PSDs, but despite their presence, the different devices are well isolated, their PSDs forming separate groups.

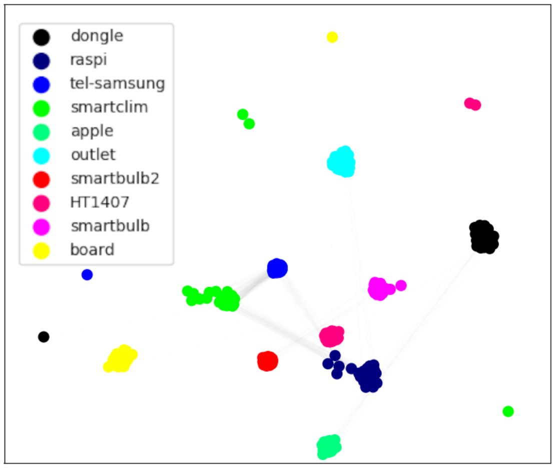

Fig. 6: Visualisation of the data from the different BLE devices

The results show that our approach is able to efficiently detect the ”attackers” and recognise the legitimate registered devices, though with a higher false positives rate compared to the smaller-scale previous experiment. This is reflected by a lower precision (which stems from the presence of outliers, or from the relative proximity between some devices). Nevertheless, the overall efficiency of our approach to detect an intrusion inside a smart environment containing diverse legitimate connected devices, is high a recall rate over 98%.

D. Identical devices

In the last experiment, we tested our approach on sets of identical BLE and Zigbee devices: 18 NRF52840 chips implementing BLE protocol and 20 XBee chips implementing ZigBee protocol. The devices emit the same data from similar positions. The results for the Zigbee and BLE cross-validations can be found in Table IV, and the visualisation of the PSDs similarities for Zigbee devices is presented in Figure 7. Moreover, we can conclude that the LimeSDR gives indeed better results than the HackRF in our Zigbee experiment, while they remain comparable for BLE.

TABLE IV: Same manufacturer, same model - results

Metric	Zigbee LimeSDR Mini	Zigbee Hackrf	BLE LimeSDR Mini	BLE Hackrf
Accuracy	93.74%	81.09%	94.12%	95.81%
Precision	96.86%	85.85%	95.87%	94.89%
Recall	87.62%	66.20%	92.67%	97.16%

Even though in both cases, our visualisation highlighted a potential collision between two emitters (as shown in Figure 7 between xbee9 and xbee20 for Zigbee devices), the overall performance of our algorithm in separating the devices and detecting potential intruders, as reflected by accuracy, precision and recall measures, remains high.

Moreover, it is important to underline that the attacker model used in this experiment is quite pessimistic. Indeed, we consider that the attacker is able to know or guess precisely the

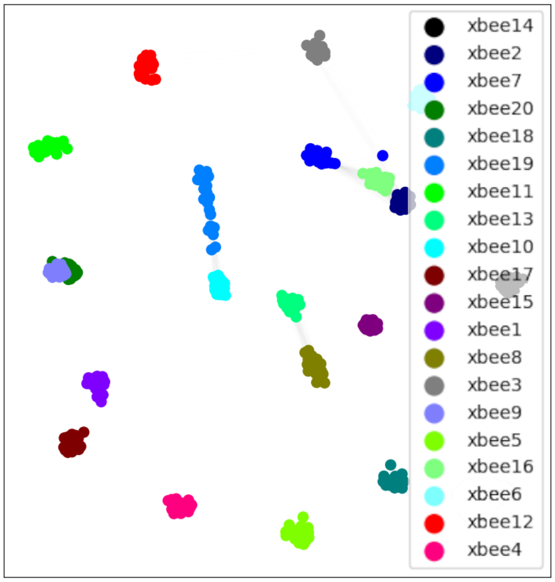

Fig. 7: Visualisation of the data from identical ZigBee devices

model of all legitimate devices, and is able to get a high number of identical copies of each specific model, and to use them in approximately the same location as the legitimate ones. This makes this kind of situation, and hence these collisions quite unrealistic in usual smart environments. Moreover, even in this pessimistic situation, our approach still exhibits fairly good detection results, which, we believe reinforces its relevance.

E. Performances and Scalability

In this section we address the scalability of our approach and estimate the associated performance overhead. Two main parameters are relevant for such analysis: the number of devices N to be fingerprinted, and the number of PDUs or signals M to be collected per device to create the associated fingerprints. The processing time is directly proportional to the number of similarity computations needed to run the community detection algorithm and create the fingerprints. This number is equal to $\binom{M*N}{2} = \frac{(M*N)*(M*N-1)}{2} = \frac{(M*N)^2-M*N}{2}$. Additionally, when a new device is included in the environment and needs to be fingerprinted, considering there were already $(N-1)$ fingerprints, the number of similarity computations is given by $\binom{M}{2} + (N-1)*M^2 = \frac{(M-1)*M}{2} + (N-1)*M^2 = \frac{(2*N-1)*M^2-M}{2}$. Figure 8 plots (on a log-scale) the evolution of the number of similarity computations with the number of devices, the number of PDUs per device varying between 10 to 100. Black curves correspond to the case where the fingerprints are computed for the number of devices indicated on the x-axis, and blue curves correspond to the case where a new device is added incrementally. It can be seen that for a given number of devices, increasing the number of PDUs leads to a dramatic increase of the number of similarities computed, and hence the processing time (the impact is quadratic). Accordingly, an optimal tradeoffs needs to be achieved between the number of devices and the number of PDUs.

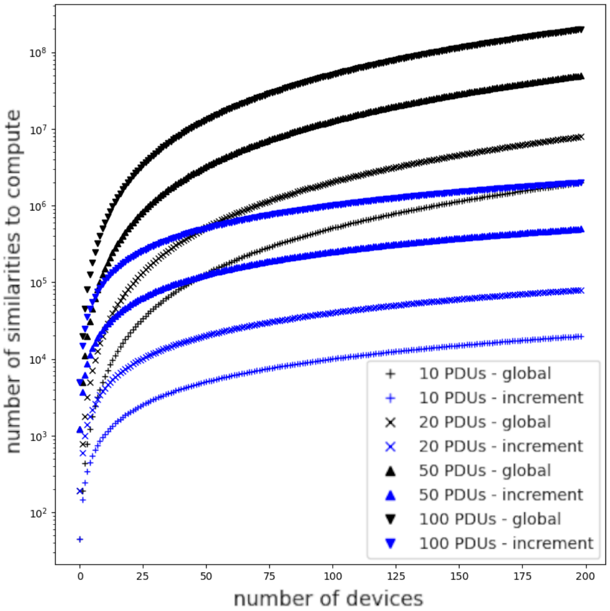

Fig. 8: Number of similarity computations estimation

The selection of the optimal number of PDUs should be based on the detection efficiency of the proposed approach. Table V presents the results of our intrusion detection algorithm in the case of the experiment with BLE devices presented in Section V-D, considering different numbers of PDUs per device, varying from 10, 20, 50 and 100, respectively.

TABLE V: Same manufacturer, same model - BLE results for different numbers of PDUs per device

Metric	100 PDUs	50 PDUs	20 PDUs	10 PDUs
Accuracy	94.8%	95.4%	94.5%	95.5%
Precision	93.2%	95.5%	96.5%	95.0%
Recall	97.0%	95.3%	92.9%	96.9%

The results of this experiment remain excellent with lower numbers of PDUs. The results clearly depend on the experimental environment and more experiments should performed in different environments to analyse the impact of reducing the number of PDUs on the detection effectiveness. We may observe a higher variability in more noisy environments.

Additionally, Table VI presents the average times, over 100 samples, to compute the fingerprints measured in our different experiments with different numbers of PDUs. These times remain significantly low even with 100 PDUs per device. Those times were obtained with a laptop equipped with an i7-7700HQ (3.8GHz) and a 8GB RAM.

TABLE VI: Fingerprint creation measured times

#PDUs	B-1	B-2	D	E-BLE	E-Zigbee
100	10.09s	10.34s	1mn 40s	2mn 12.30s	3m 34.21s
50	1.85s	1.84s	8.88s	11.56s	21.00s
20	1.13s	1.13s	1.83s	1.84s	4.51s
10	1.04s	1.03s	1.21s	1.12s	2.41s

Based on the numbers presented in Table VI, the estimated time to create the fingerprints of 100 devices with 100 PDU per device is around 90 mn and it takes only a few minutes to create incrementally the fingerprint of an additional device. These times are significantly reduced to around 60 mn and 30 sec, respectively if we only consider 50 PDUS per device.

VI. Discussion

Our approach has been successfully tested in fairly static smart environments, including smart sensors that are not supposed to move much. This assumption corresponds to many real-life cases, such as smart buildings.

However, the proposed approach can be easily adapted to dynamic environments in which legitimate users can bring connected devices that have not been already fingerprinted. This requires that they register these devices before entering the smart environment, so that their fingerprint can be computed and saved in the corresponding database. The fingerprinting phase only requires a couple of minutes as shown in Table VI (note that our algorithms are currently implemented in Python, and can be further optimized), after which the device can be identified by our algorithm.

Another relevant problem is related to dynamic objects that can join, move around or leave the environment. This requires a regular update of the fingerprints using the incremental approach discussed in V-E. Note that dynamic fingerprints are also useful to mitigate the potential desynchronisation between the SDR receiver and the emitting devices.

Finally, it should be also noted that as the results of the fingerprinting are sensitive to multi-path delay, the fingerprints creation and the execution of the detection algorithm should be done in the same environment.

VII. Conclusion

In this paper, we investigated a novel approach for device identification and intruder detection based on the generation of device fingerprints. These fingerprints are elaborated from the PSDs of entire signals corresponding to PDUs from the device. The selection of the PSD is motivated by the need to capture the frequency characteristics of the emitter, without influence of the position or phase offset. This approach is inexpensive and is designed to detect potential link-layer spoofing attacks that would not be detected by upper-layer monitoring.

The various experiments that we have performed confirmed the high performance of our approach independently of the protocol used by the emitters. Our experimental results showed the high effectiveness of our approach in separating different objects, with a precision and a recall always higher than 85% when signal acquisition is carried out with the LimeSDR Mini, showing a probability of non detection and a probability of false alarm below 15%. These low probabilities ensure that, in an intrusion detection system, a high number of positives (packets coming from unidentified sources) observed over a short period of time for a stream of PDUs will indicate that the emitter is most likely an intruder.

For future work, we plan to carry out larger scale experiments to confirm these promising results with a higher number of sources in the environment. Moreover, we plan to improve our implementation to address the limits exposed in the previous section (by the use of a compiled language to optimize the performances and the prototyping of the dynamic signatures to mitigate desynchronisation problems).

Special thanks

We would like to thank the GEI department of INSA Toulouse for having provided the XBee modules for our experiments.

References

[1] D. Antonioli, N. O. Tippenhauer, and K. Rasmussen, "Bias: Bluetooth impersonation attacks," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2020.

[2] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 195–212.

[3] "Experiments datasets," https://gitlab.laas.fr/fgaltier/prdc-2020-psd-based-fingerprinting-datasets.

[4] J. Hall, M. Barbeau, and E. Kranakis, "Radio frequency fingerprinting for intrusion detection in wireless networks," *IEEE Trans. Dependable Secure Comput.*, 2005.

[5] S. Ur Rehman, K. Sowerby, and C. Coghill, "Rf fingerprint extraction from the energy envelope of an instantaneous transient signal," in *2012 Australian Communications Theory Workshop (AusCTW)*, 2012, pp. 90–95.

[6] M. Köse, S. Taşcioğlu, and Z. Telatar, "Rf fingerprinting of iot devices based on transient energy spectrum," *IEEE Access*, vol. 7, pp. 18 715–18 726, 2019.

[7] B. Danev, "Physical-layer identification of wireless devices," Ph.D. dissertation, ETHZ, 2011, pages 25-90.

[8] B. Danev, D. Zanetti, and S. Capkun, "On physical-layer identification of wireless devices," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–29, 2012.

[9] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Paradis : Physical 802 . 11 device identification with radiometric signatures," 2008.

[10] B. Chatterjee, D. Das, S. Maity, and S. Sen, "Rf-puf: Enhancing iot security through authentication of wireless nodes using in-situ machine learning," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 388–398, 2019.

[11] B. W. Ramsey, B. E. Mullins, M. A. Temple, and M. R. Grimaila, "Wireless intrusion detection and device fingerprinting through preamble manipulation," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 585–596, 2015.

[12] D. Nouichi, M. Abdelsalam, Q. Nasir, and S. Abbas, "Iot devices security using rf fingerprinting," in *2019 Advances in Science and Engineering Technology Int. Conferences (ASET)*, 2019, pp. 1–7.

[13] J. Hasse, T. Gloe, and M. Beck, "Forensic identification of gsm mobile phones," in *Proceedings of the first ACM workshop on Information hiding and multimedia security*, 2013, pp. 131–140.

[14] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: A survey," 05 2019.

[15] P. Eckersley, "How unique is your web browser?" pp. 1–18, 2010.

[16] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Nordic conference on secure it systems*. Springer, 2011, pp. 31–46.

[17] "Hackrf one official webpage," https://greatscottgadgets.com/hackrf/.

[18] "Limesdr mini official webpage," https://limemicro.com/products/boards/limesdr-mini/.

[19] P. Pons and M. Latapy, "Computing communities in large networks using random walks." *J. Graph Algorithms Appl.*, vol. 10, pp. 191–218, 01 2006.

[20] L. Kaufmann and P. Rousseeuw, "Clustering by means of medoids," *Data Analysis based on the L1-Norm and Related Methods*, pp. 405–416, 01 1987.

[21] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002. [Online]. Available: https://www.pnas.org/content/99/12/7821

[22] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 70, p. 066111, 01 2005.

[23] "igraph," https://igraph.org/.