Security of short and long range wireless networks based on physical layer monitoring mechanisms

Florent Galtier

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue le *17/02/2023* **par :**

Florent Galtier

Sécurité des réseaux sans-fil courte et longue portée basée sur des mécanismes de monitoring de la couche physique

---

JURY

Jean-François LALANDE	Professeur des universités	Rapporteur
Samia BOUZEFRANE	Professeur des universités	Rapporteur, Présidente
Mohamed KAÂNICHE	Directeur de recherche	Directeur de thèse
Guillaume AURIOL	Maître de conférences	Co-directeur de thèse
Vincent NICOMETTE	Professeur des universités	Examinateur
Florent BRUGUIER	Maître de conférences	Examinateur
Jose LOPES ESTEVES	Ingénieur de recherche	Examinateur
Gregory BLANC	Maître de conférences	Examinateur

École doctorale et spécialité :
EDSYS : Informatique 4200018
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Mohamed KAÂNICHE et *Guillaume AURIOL*
Rapporteurs :
Jean-François LALANDE et *Samia BOUZEFRANE*

Acknowledgments

I would first like to thank the consecutive directors of the LAAS-CNRS, Liviu Nicu and Mohamed Kaâniche, for welcoming me during this thesis, and Helene Waeselynck, current head of the TSF team (Tolérance aux fautes et Sûreté de Fonctionnement informatique) in which I worked during those three years.

I would then like to express my gratitude towards my different supervisors, Mohamed Kaâniche, Guillaume Auriol and Vincent Nicomette, for their constant support, their help during the redaction of articles or of this manuscript, and during the various experiments that were needed. They brought a lot to me during these years, be it from a technical or human perspective.

I also want to thank the different examinators for this manuscript and for my defence, Samia Bouzefrane, Jean-François Lalande, Florent Bruguier, José Lopes Esteves and Gregory Blanc, for accepting to review my works.

I would also like to thank particularly Pascal Acco for his insights on signal processing, that helped me a lot from a theoretical point of view in my different works.

I don't think I would have been able to make this without Romain Cayre and Jonathan Roux, who preceeded me in the same team on the topic of the Internet of Things, and especially Romain, with who I had the privilege to work during most of my thesis, and whose motivation and ingenuity greatly fueled my own will to continue working in this field.

Finally, I would like to offer my thanks to all the people that I met in this team during those three years, and especially the other thesis students and interns, for all the good times we had, that helped me stay motivated until the end.

Contents

List of Figures

List of Tables

Introduction

The massive addition of connectivity to multiple devices across houses, buildings and even companies, changes durably the structure of network environments, forming an interconnected network of various devices, called the Internet of Things. Be it for data collection, remote control, coordination, or even to add intelligence to traditional tools, communication capabilities are progressively embedded in all items of everyday life. Nowadays, this phenomenon ranges from connected fridges or lightbulbs for home automation, to vehicular networks or health-related devices, now able to communicate, often wirelessly, with each other. They form an unusual type of environment, where everything becomes interconnected with various, heterogeneous communication protocols, and their mobility also adds dynamicity to the environments in which they come and go.

Unfortunately, these devices, especially those for the public, are often developed without consideration for their security, their manufacturers focusing more on the addition of new functionalities. This makes wireless networks composed of multiple of those items all the more difficult to analyse, monitor and protect. Moreover, their mobility and connectivity can make them prime targets for attackers, as they can be compromised in a low-security environment, for example at home, from an infected personal computer, to then attack other, more secure environments in which they are brought. The wireless nature of these networks also increases the risks, as the communications become accessible to all in transmission range. As it increases the attack surface of all networks in which they interconnect, it then becomes a primordial issue to find new ways to secure connected devices, especially considering some of their most critical use cases, such as connected healthcare devices. This is especially true as those devices can also connect to more traditional networks, thus bringing a possible aditional entry point into otherwise well-protected networks.

Some traditional security measures, such as authentication mechanisms or cryptography, could be applied to such networks, especially when they do not depend on the communication medium. For example, applying encryption to communications is important in such networks, due to the ease of implementing passive eavesdropping attacks. However, some other traditional approaches are incompatible with the particular nature of wireless networks, and with the fact that the devices often only embark small microcontrollers having low computational power. For example, network-based firewalls are hard to adapt to these networks, due to the difficulty of blocking transmissions, and host-based firewalls would need to only perform light packet analysis. Furthermore, the heterogeneity of the protocols they use can also complicate the implementation of security measures. First, different protocols may use different frequency ranges, sometimes far from each other (as an example, Bluetooth uses the 2.4-2.5GHz band, while some small transceivers use 868MHz, or even 433MHz bands). Because of this, monitoring various protocols may require a high number of listening equipments, to cover the entirety of the channels they use.

Then, some of those protocols are proprietary, their specifications not being fully available, making them harder to audit.

As traditional measures can be insufficient, new measures, adapted to the IoT, were created. However, most of those solutions are limited to a single protocol's characteristics, thus complicating the securing of whole, heterogeneous networks. Furthermore, limiting the protection, and especially the monitoring of an environment to measures for the protocols legitimately introduced in it overlook several risks, such as sensitive data exfiltration via some other protocols. Moreover, those methods are hard, if not impossible to design for proprietary protocols without public specifications. Only a few methods are generic, and they often remain costly, and need to be deployed on several frequency bands simultaneously. Indeed, it is currently impossible to monitor precisely all frequency bands with only one equipment, because of the high sampling rate required to cover all protocols. Some methods are able to watch wider bands by only registering reception power at a frequency at a given time, sweeping over the monitored band, but then loose all information on the data that was sent.

Numerous defense methods focus on the binary data that is transmitted. However, due to the simplicity of most IoT devices, attacks on the data, especially impersonation attacks, are easy to perform, and hard to detect. Even worse, this overlooks completely all threats related to the wireless medium.

Starting from these observations, this PhD was focused on the impact of the wireless communication medium itself on security. We explored the risks, often overlooked, of isolation breach between protocols in such environments. Then, we designed new defensive approaches to counter them, and improve monitoring capabilities by being able to analyse communications without making heavy assumptions on their content.

This thesis is organized as follows: in the first chapter, we briefly present its context, defining the wireless networks on which we worked, and the related security implications. We also present in broad terms the offensive and defensive state of the art in wireless networks.

Then, after introducing some important signal processing concepts in chapter 2, required to understand the implications of using a wireless medium, we explain in chapter 3 how some devices can be used to communicate with protocols that they were not designed to emit or receive. We present a practical example, realized in common in Romain Cayre, called Wazabee[Cayre 2021b]. This attack allows us to either receive or emit valid Zigbee frames while using Bluetooth Low Energy transceivers, now present in most computers or mobile phones. Then, we present the theory that would allow performing similar attacks with other kinds of protocols.

In chapter 4, we present our first defensive contribution, consisting in a low-cost fingerprinting approach for wireless emitters[Galtier 2020]. This approach is able, by receiving enough data with less precision than more costly approaches of the state of the art, to differentiate devices by detecting imperfections in the frequency profile of their emissions. This method aims in particular at countering spoofing attacks, where the attacker takes the place and identity of a legitimate device, that

are otherwise highly difficult to detect.

We then present our second defensive contribution in chapter 5, an approach to automatically analyse wideband receptions in order to detect, identify and analyse all wireless emissions it could contain. This method first identifies the modulation in use and its parameters, then analyses the demodulated binary stream to infer information about the packet structure. The method presented makes a minimal number of assumptions on the emissions, and is easily improvable with new modulations thanks to a modular structure.

To conclude, after synthesizing the contributions presented in this manuscript, we present the possible extensions and improvements that could be brought to them.

Chapter 1

Context and objectives of the thesis

Contents

In this chapter, we introduce the general context of this PhD, which mainly focuses on the security of the wireless communication protocols of the Internet of Things and especially the security of the lower layers of these protocols. We first start by describing the evolution of the development of wireless networks in IoT environments, then we focus on the security challenges raised by this evolution. After that, we define the security terminology that is used all along this document before presenting the existing attacks, defences and security challenges related to the Physical Layer aspect of such communications. We finally motivate the research work proposed in the thesis and its main contributions.

1.1 General context

1.1.1 Evolution of network communications: the Internet of Things

In recent years, the development of easy-to-use and easy-to-deploy wireless protocols has lead to a new phenomenon. Many devices, such as those that were already connected with each others or with computers by wired links, but also devices that were originally small appliances, manual tools or even objects that didn't embed any electronics, began to embed wirelessly communicating chips, and to interconnect with each other. By analogy with the original meaning of the word *Internet*, which originally referred to a network interconnecting machines, this phenomenon became known as the *Internet of Things*.

Nowadays, this idea of embedding communication capabilities into all objects has lead to interconnecting devices with radically different applications, and criticalities: from connected televisions, or lightbulbs, to alarms or tools, for which security is a major concern.

Additionally, this allowed for new types of environments, fully integrating such connected devices. First, they allow for better home automation, with all kinds of devices and appliances connected to a smartphone or computer, leading to so-called "smart homes". For example, it is now possible to install connected window shutters or garage doors, that can be opened or closed remotely. Similarly, some alarm systems can now also be remotely controlled using connected devices. More recently, we witnessed the development of voice assistants, also allowing to control various devices in the house. Moreover, many such devices appear everyday, with various and sometimes heteroclit usages, spreading in now smart homes: connected fridges, connected scales, connected flower pots... These devices and their interconnection form complex communication environments, and the forecasts suggest that the situation will continue to intensify in the next years, with more than 40 billion devices expected by 2025 [European-Comission 2022].

Then, by applying this automation at a larger scale to buildings to manage their energy consumption or production, it leads to "smart buildings". Those environments include a large amount of heterogeneous devices, for multiple purposes, from entertainment to physical security, interconnected through users' phones, computers or gateways. From a security point of view, these environments allow for easier attacks and harder monitoring, due to the quantity and heterogeneity of the emitters they contain, but the criticality of such attacks is limited.

Industry is also using such interconnected devices, for remotely-controlled tools, maintenance or monitoring, for example in a production line. This kind of environment, so-called "smart factory", is often more controlled, and contains known protocols in smaller quantities, at least in a usual operating situation. This generally makes them easier to monitor, but they are sensitive to people entering the environment with a device they own, introducing potential new, unmonitored and vulnerable protocols.

Finally, a growing number of critical environments are also becoming increasingly connected. For example, transportation means, such as cars or planes, now have multiple short- or long-distance communication capabilities, to offer various services to the passengers. Another example is the medical field, where hospitals are also becoming more and more connected, for example to remotely monitor patients. Given the sensitivity of such environments and the possible consequences of a compromise, it becomes critical to ensure sufficient security on the connected devices and communication protocols they use.

To sum up, devices with interconnection capabilities, at various levels of criticality, are emerging and spreading in all kinds of environments, whether at home or in professional settings. They offer a new potential attack surface, especially when they are embedded in their most critical applications: in the health domain, for example with pacemakers, in the transportation domain, such as in smart cars

or planes, whether during their manufacturing process or their use. It is therefore necessary to study more in detail the different implications of their omnipresence on the security of the information systems they communicate with.

1.1.2 Wireless networking protocols

The protocols used by such devices are as numerous as their various uses, if not more. They range from short-range protocols for communication between small, local devices and a controller, for example with the well-known Bluetooth or Zigbee protocols, to long-range protocols for interconnecting devices through a Wide Area Network, for example with mobile telephony or the LoRa protocol. The specifications of these protocols are defined by different kinds of entities, such as a group of standard organizations for mobile telephony, a single dedicated one for Bluetooth, or individual companies, as is the case for the LoRa or Enhanced ShockBurst protocols. Some device manufacturers even create their own simple protocol on top of existing chips.

Moreover, the applicative protocols above can also completely differ from one object to another, depending on the level of expertise and involvement of the manufacturer. Indeed, a large number of connected devices currently sold on the market are developed by companies that were not originally involved in the software or networking industries, to add new functionalities to their devices, mainly as selling arguments. Therefore, a significant proportion of them are not aware of the security issues involved with embedding such features.

As a consequence, we regularly see the reappearance of old vulnerabilities that were discovered long before in more standard networks, along with the appearance of whole new vulnerabilities directly linked to the new nature of such devices. Even worse, the growing number of competing companies implementing connected functionalities pushes them to add even more features, as fast as possible, thus neglecting even more the security processes that should accompany the development of their devices.

Thus, due to the heterogeneity, number and lack of security awareness in the protocols that constitute this Internet of Things, it becomes all the more critical to investigate the potential attacks that could target it, and the defences that address them.

1.2 IoT communication security

In the following, we look at the different attacks and defence measures from the perspective of the Physical, Link, Network and Application Layers of the OSI model. Indeed, in the devices and protocols that we consider, when other layers are present, they are rarely specific to the IoT, and use implementations that are independent of it.

From the Application Layer perspective, as said in the previous section, the IoT lags significantly behind more traditional networks in terms of experience in

security. We can see manufacturers periodically re-introducing vulnerabilities such as in house made cryptographical suites.

Let us take two examples of such vulnerabilities, presented in [Newlin 2016]. The first one concerns the Logitech Unifying protocol. The Logitech Unifying protocol chooses to XOR a mask generated with state-of-the-art AES with key-presses, but 1) apart from the key, all information entered into the AES block was known because sent as cleartext in the packets, 2) it could be replayed, thus triggering decryption with the same output of said AES block, and finally 3) the key-releases were coded with a sequence of `\x00` bytes. Therefore, an attack can wait for a key-release event, and then inject encrypted key-presses by replaying the same values of the AES block input as in the key-release, XORing the key to encrypt with the key-release encrypted part, which, since the key-release is only 0s, was exactly the output of the AES block. The second example concerns simpler vulnerabilities, already seen before in more traditional networks, that re-appeared, such as with Microsoft wireless keyboard protocols [Schroeder 2010]. Originally, this protocol encrypted key-presses by XORing them with the address of the device, which was obviously sent as plaintext in the transmissions. This could allow an attacker listening to the communications to easily compute the mask from the known address, and decrypt all transmissions.

More generally, the Application Layer of IoT networks is often simple, without security measures such as encryption implemented, and greatly suffers from the lack of knowledge and awareness of the manufacturers in information security.

The Network Layer is not widely represented in the IoT world ; indeed, most protocols are limited to peer-to-peer communications. However, some exceptions are notable: first, protocols connecting with a core-network, for example linked to the more traditional "Internet", need to implement this layer. The most obvious example would be mobile telephony, but we could also cite the 6LoWPAN protocol, implementing IPv6 over IEEE 802.15.4 links, i.e. the same lower layers as Zigbee. In this category, the aforementioned core-network often benefits from pre-existing experience in security, and corresponds more to a traditional network than to the communications in the scope of this thesis. Then, other protocols can build up mesh networks, such as Zigbee or Bluetooth Mesh, based on Bluetooth Low Energy. However, the Network part of these protocols is, to our knowledge, quite light, and most of the security relies on the lower and upper layers. Indeed, the "mesh" aspect in these networks is often light for complexity reasons, and is closely intertwined with the lower-layer protocol. It should however be noted that, in the case of Bluetooth Mesh, encryption is always active, which is not the case in more regular Bluetooth Low Energy communications. To sum up, the security of the Network Layer in the IoT is either closely linked to the security of a more traditional "core-network", or heavily dependent on the lower layers.

Most of the specifications of the short-distance IoT protocols is focused on the two lower layers, namely the Physical and Link Layers. As such, they include a significant part of the innovative features and potential additional attack surface that comes with the IoT. As we will develop further in the following, the Link Layer

in particular can sometimes suffer from similar issues as the Application Layer, due to the lack of experience of people using the protocols. One glaring example is the fact that, despite the presence of cryptographical suites in the Bluetooth Low Energy specification, and its now wide use as it is integrated in most computers and phones nowadays, most applications of the protocol still do not activate the encryption on their communications, or only use a weak one[Ryan 2013, Zuo 2019]. Let us note that this kind of vulnerability is inherent to the implementations and integrations of the protocols, rarely to the protocols themselves when they are not developed specifically for the devices by their manufacturer. However, some protocols can be weak by themselves, at different levels. Some can be limited to the simplest functionalities needed for communication, and hence not include security measures, while some others are more mature from the security point of view, containing more complex and hidden vulnerabilities. For example, we encountered and experimented on small devices, such as connected outlets or doorbells, using protocols that only carried commands in cleartext, without any form of security.

Finally, the specificity of the wireless protocols is its communication medium: all communications are transported via electromagnetic waves, visible by anything present in their way. This obviously eases the insertion of a potential attacker in the communication, when they would have had to physically access cables to measure the signal passing inside with wired communications. Moreover, the interferences caused by such an attacker who would just listen to the communications can be highly difficult to measure, rendering passive attacks nearly, if not completely, impossible to detect. This new medium also allows access to an external attacker, by introducing a malicious emitter between the actors of a legitimate communication. Additionally, the wireless nature of the transmissions can also bring entirely new phenomena, due to the reflections of the waves on obstacles, interferences between electronic components and antennas, or also the directionality of some transmissions.

In this thesis, we focus more precisely on these two layers, which we refer to as the "low layers" in the following.

Before entering into details of the security of these low layers, we first define in the next subsection the security terminology that we adopt in this manuscript.

1.2.1 Security terminology

For the following, we need to define the security related vocabulary used in ourc context. To do so, we define its different concepts according to [Laprie 1995], then updated in [Avizienis 2004], first from the wider point of view of dependability, then focusing on security.

1.2.1.1 Dependability

According to the applications of the system, different facets of dependability may be highlighted. Thus, dependability can be broken down to different but comple-

mentary properties:

- availability: readiness for usage
- reliability: service continuity
- safety: non-occurrence of catastrophic consequences for the environment
- confidentiality: non-occurrence of unauthorized disclosure of information
- integrity: non-occurrence of inadequate information alterations
- maintainability: ability to conduct repairs and introduce evolutions
- security: the combination of confidentiality, integrity and availability

A failure of the system occurs when the delivered service deviates from implementing the system function, that is, from what the system is intended for. An error is that part of the system state which may lead to a failure: an error affecting the service is an indication of a failure occurring of which has occurred. The adjudged or hypothesized cause of an error is a fault.

Development of a dependable system requires the combined use of a set of methods that can be listed as follows:

- fault prevention: how to prevent the occurrence or introduction of faults
- fault tolerance: how to provide a service capable of implementing the system function despite faults
- fault removal: how to reduce the presence (number, severity) of faults
- fault forecasting: how to estimate the presence, creation and consequences of faults

1.2.1.2 Security

Security has for goal to protect a system against faults defined as intentionally harmful. In the following, we will apply the dependability attributes defined previously to the context of security.

The attributes of dependability can be specialized to security as follows:

- Availability: prevent unwanted information withholding
- Confidentiality: prevent unwanted information disclosure
- Integrity: prevent unwanted information modifications

The MAFTIA project [Powell 2001] addressed specifically the case of faults due to human interaction for the security. In the following, we will also mainly focus on this class of faults, and more precisely on the case of intrusions. Intrusions can

be defined as an external and harmful fault, resulting from an attack managing to exploit a vulnerability. Those vulnerabilities can be intentional, and thus participate in the intrusion process, or accidental. The vulnerabilities, be them intentional or not, contribute to the potential weakness of the system, and are thus important to monitor to prevent malicious interactions.

Moreover, there can also be unintentional "attacks", external faults due to a wrong usage of the system by a user. However, those faults, that can be critical for the system, will fall outside the scope of this thesis, because often depending on the user's knowledge and formation.

The methods to make a system dependable can also be specialized for security as follows:

- fault prevention: fault prevention can be split in preventing attacks, vulnerabilities or intrusions as a whole. Preventing attacks means dissuading malicious users to attack the system, for example by law. Preventing vulnerabilities amounts to an introduction of security checks during the design and development processes, for example via formal methods or user security awareness. Preventing intrusions can be done through authentication methods, authorization, firewalls or by preventing attacks and vulnerabilities.

- fault tolerance: in our case, we focus primarily on intrusion tolerance. It consists in building a system able to detect an intrusion, repair itself and reconfigure while keeping the service available and/or protecting the data integrity during an attack. We could also mention redundancy or diversification methods, which can ensure a certain level of security even when part of the components are compromised.

- fault removal: in the case of intrusions, only vulnerability elimination remains pertinent. Indeed, one cannot completely prevent the presence of attacks, and if a vulnerability exists, an intrusion could always exploit it. However, completely eliminating all possible faults from a system remains a difficult, if not impossible task depending on its complexity. Thus, the designers can try to reduce the risks by applying, for example, verification, diagnosis or correction to their system. Verification refers to checking if the system satisfies desired properties, diagnosis searches for the faults preventing from satisfying them, and correction then applies the necessary measures to satisfy the missing properties.

- fault forecasting: finally, fault forecasting contains a set of methods to identify vulnerabilities, attacks, potential intrusions on a system and measure the impact of errors on the security attributes of the system, while reporting those identifications.

In order to design methods for more dependable systems, we need to understand two elements for the system to protect:

- The attack model: defining the potential threats, to identify the system's vulnerabilities in order to propose appropriate solutions to prevent, tolerate, remove or forecast malicious activities

- The attack surface: defining the attack vectors, i.e. the external access points that could contain vulnerabilities potentially exploitable by an attacker to perform an intrusion

1.2.2 Low layers security

1.2.2.1 Link Layer security

As explained above, the two layers that bring the most novelty and potential for attacks and defence measures, compared to more traditional wired networks in the IoT world and in wireless communications in general, are the Physical and Link Layers of the OSI model.

Concerning the Link Layer only, there are several classes of vulnerabilities in wireless communications. A first but important one, from the confidentiality point of view, is the fact that many communications using short-distance wireless protocols do not use encryption. This may be to simplify the development of small connected objects, or because of a lack of knowledge about the use or impact of encryption on communications, with some manufacturers not understanding the risk of eavesdropping, and others believing that encryption will decrease the speed and/or availability of the link. However, as the wireless communication medium is easily accessible for anyone with a compatible receiver, this lack of data protection allows full readability of everything sent by the device everywhere around it. Despite its criticality, especially in a smart factory context, this vulnerability is unfortunately widely spread, with some protocols leaving manufacturers free to use encryption or not, and others not implementing Link Layer encryption, leaving at least the metadata open for all to see. Let us note, however, that more mature protocols such as those used for mobile telephony have been using strong encryption since many years already.

Similarly, some communications, especially smaller-scale ones, do not implement authentication mechanisms. This is often the case for simple communicating devices, but it is also possible, even in Bluetooth Low Energy, to only use the Link Layer address of a device as an identification feature, or even accept connections from any device. This leads to other vulnerabilities where any attacker, in the direct vicinity of an object implementing such protocol, could connect to a device as would a legitimate device, leading for example to a potential malicious takeover of the device. The example of BLE is even more significant because most devices on which one can connect broadcast beacon-like packets, and, often not being able to accept more than one connection, completely stop emitting them when another device connects to them, leading to a disappearance of the object and potential Denial of Service just by an attacker establishing connections.

To allow authentication, several methods exist. First, it is possible to use pre-shared keys, for example with WiFi networks with a password. They can also be embedded into the hardware, as with mobile telephony where a key specific to a user is embedded in the SIM card, reducing the risk of a potential disclosure of the key. The vulnerabilities of these methods are the same as for more traditional networks, an attacker gaining access to a terminal being potentially able to extract the pre-shared key from it, with varying levels of difficulty depending on how the key is stored. Then, some protocols implement pairing procedures, allowing two devices to exchange keys for later authentication and link encryption. A usual example of this is the pairing between Bluetooth devices, where each device can ask for a code displayed on the other one to check the user establishing the pairing has access to both. However, in Bluetooth Low Energy, there also exist a pairing method called "Just-Works". When, usually, the two devices would exchange temporary keys that allow them to generate long term keys for later communications, "Just-Works" use a hard-coded value of 0 instead. Sadly, as shown by Zuo et al. in [Zuo 2019], where they built a tool to crawl free IoT apps on the Google Play Store, 61.3% of the 18 000 applications they tested were using this method, showing its spread in the IoT industry. Even if such weak pairing methods could allow protecting the communications from a basic passive eavesdropping, it offers no security overall.

These kinds of vulnerabilities also make it easier to deploy Man-in-The-Middle attacks, in which an attacker stands between two communicating devices, and communicates with each of them by impersonating the other, allowing the attacker to either transparently forward packets, drop some or even modify some data. Furthermore, data integrity in the IoT protocols is often only based on the use of Cyclic Redundancy Checks (CRC), the algorithm being open in most cases, thus not providing any protection against message tampering. Their combination can lead to vulnerabilities such as BTLEJuice[Cauquil 2016], where the attacker, having only two BLE-compatible emitters, can 1) connect to some device connectable with one of them, which then stops sending its beacon-like packets, 2) expose the second one with the same address as the device to which the first one is connected, and start sending the same beacons, while 3) identifying the services offered by the device, to finally 4) wait for a connection by another device, showing it a clone of the first one, to run a full Man-in-The-Middle.

1.2.2.2 Physical Layer security

The Physical Layer can also present vulnerabilities, inherent to the wireless nature of the communications. Firstly, as said before, the communication medium is available to everyone within transmission range, allowing for easier eavesdropping. This characteristic also allows an attacker to directly interfere with the transmitted signal, the best known example being jamming[Xu 2005]. This technique consists in sending another signal during a legitimate transmission to prevent the legitimate recipient from receiving the correct data. It ranges from the simpler continuous jamming[Shintani 2020], where an attacker completely floods the channels used

by the targeted protocol with a continuous and high-strength signal, to reactive jamming[Nguyen 2014, Wilhelm 2011], where the jammer remains silent until it recognizes a specific pattern, triggering the interfering signal. This type of attack is easy to detect when used naively, by monitoring the received power on the channel and suspiciously high bit-error-rates. Some works also make use of the difference in error-rates between the jammed sections and the rest of the packet to detect reactive jamming [Spuhler 2014]. However, they are hard to prevent, as the detection occurs after the attack was already carried. In addition to detecting and stopping the source of the attack, the ways to resist jamming would be either to change channel, for example with channel hopping protocols like Bluetooth or protocols that jump on loss of connection like Logitech Unifying, or to increase transmission power. However, this last measure can be energy-consuming and results in a tug-of-war with the attacker to know who transmits the loudest. It would also be possible to correct the errors by isolating the jamming signal and subtracting it or by implementing error-correction codes, though these measures are dependent on the predictability of the signal or the amount of errors it generates, respectively.

A more complex variant of this attack, called overshadowing [Wilhelm 2012], consists in replacing the original signal with one chosen by the attacker. This method could also be detected by monitoring the received power in search for an abnormal increase, but if the attack is skilfully carried on, the amount of incorrectly received packets will not increase.

Another threat, this time not directly over existing legitimate communications, is the use of the wireless medium to exfiltrate data [Okhravi 2010]. This type of attack, called a covert channel, requires monitoring the communication medium to detect suspicious communications. However, this task is not trivial, especially when the attacker uses frequencies that are not used by common protocols, or at least the ones used in the considered environment.

1.2.2.3 Combined lower layers

Finally, it is also important to consider the combination of the two layers, or the "low layers" as a whole, from a security point of view. Indeed, some threats or defences exploit jointly relevant features of the physical communication medium and the Link Layer characteristics of the protocol in use. A glaring example of this is the recent vulnerability Injectable[Cayre 2021a]: in Bluetooth Low Energy, a connection is divided into *connection events*, where one of the devices must transmit at a precise timing. However, to cope with the potential inaccuracy of the clocks of the two communicating devices, an error margin was introduced around this timing, called "window widening". We then used this margin to transmit before the legitimate device while impersonating it. This attack can successfully inject any kind of packets into an existing communication, if we either sent our entire packet before the legitimate one, or started sending before it, and the interference did not introduce errors into our transmission.

1.3 Contributions of the thesis

In this thesis, we mainly focus on the security of the two lower layers of wireless communications in their entirety. We first study them from an offensive point of view, by exposing threats of pivoting attacks based on the similarity between low layers, that are often disregarded. Then, we focus more on the defensive aspects related to these layers, by proposing novel methods to monitor and analyse the wireless communication medium from a low-layer perspective. Three main contributions can be highlighted.

First, we explore a new attack model that comes from a class of methods known in the signal processing community as Cross Technology Communications. This class covers ways to communicate with a receiver designed for a protocol A, using an emitter designed to transmit according to a different protocol B. As we will show, joint research in both this domain and the security of wireless communication has been minimal, and until recently no real methods that do not require cooperation from the different parties, and thus appliable in an offensive context, have been published. In chapter 3, we present joint work with Romain Cayre that led to the design of Wazabee, a pivotal attack based on Bluetooth Low Energy chips to attack Zigbee networks, also presented in his PhD thesis[Cayre 2022]. Furthermore, we present some thoughts and possible future works on the possibility of implementing similar attacks in other protocol pairs, to emphasize the need to consider such threats in the attack surface of a wireless network.

Then, in chapter 4, we propose a novel method for identifying communicating emitters based on the fingerprinting of Physical Layer emissions of each device. This method, which is designed to be less expensive than most from the state-of-the-art ones, allows for a better monitoring of a wireless network against potential identity theft, which is an omnipresent threat in the IoT and wireless communications in general.

Finally, in chapter 5, and to complement the previous method, we introduce a new approach to automatically identify emissions in a wideband reception, and recover the modulation used with its parameters, along with a maximum amount of parameters on the Link Layer protocol above it. This allows for a full monitoring of a wide frequency band, and to detect potential anomalies, such as covert channels or pivotal attacks. Furthermore, this approach also helps to audit unknown wireless networks, automatically retrieving most of the information needed by an expert to properly analyse transmissions without having to manually analyze the physical emissions and to implement protocol-specific approaches.

Chapter 2

Signal processing prerequisites

Contents

In this chapter, we introduce some background information on the basics of signal processing that are necessary for a good understanding of this manuscript. We first explain the general principles behind digital modulations, and their most usual variants. Then, we detail the complex representation of a signal, essential to better understand the usual representation of numerous modulations. Finally, we briefly explain the issue of spectral efficiency, and the usual ways to address it in signal processing, as well as the methods to demodulate the signal at the right moment, based on the signal's eye-diagram, which we will use later in the manuscript.

Additionally, we also introduce a now widespread communication scheme, used notably in WiFi or 5G networks: the Orthogonal Frequency Division Multiplexing, or OFDM. This particular scheme embeds more information directly in the frequential domain, which has consequences on the methods to analyse it, and limits the use of traditional signal processing methods to treat it. Let us note that the main contributions proposed in this manuscript are not designed to fit the specific constraints of OFDM, notably because the connected objects we considered in this PhD do not support this modulation scheme. However, we chose to describe the OFDM scheme in this chapter because, as we discuss in the following, our different contributions can still be adapted to support, to a certain extent, OFDM-based protocols.

2.1 Digital modulations

A signal is often represented as a function of time, as in the example of figure 2.1. This type of signal is in general of the form:

$$s(t) = A * \cos(2 * \pi * f * t + \varphi) \tag{2.1}$$

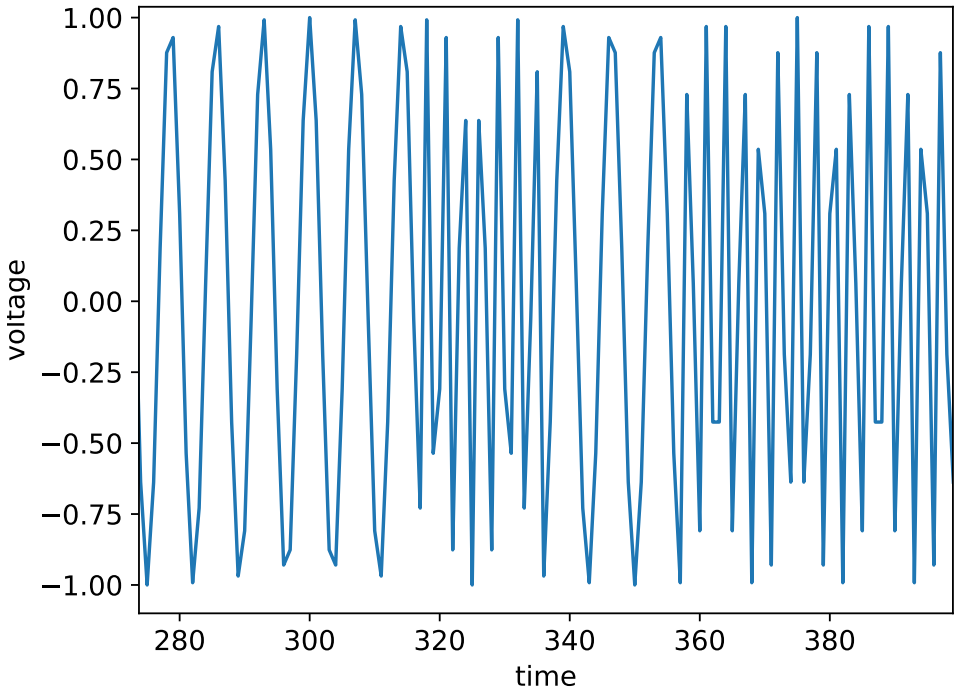


Figure 2.1: Temporal representation of a signal (frequency modulation)
[2]

with $s(t)$ the signal at time t, A its amplitude, f its frequency and φ a phase offset, which can be assimilated to a time delay.

The signal can then be modulated by coding information on one of its characteristics, for example by varying its amplitude depending on the data to send. Throughout this manuscript, we focus exclusively on digital modulations, meaning that binary information is modulated over the signal. These are most of the time only based on three main modulation schemes:

- **Amplitude Shift Keying, or ASK :** the information is coded over the signal's amplitude, hence resulting in the following formula :
  $s_{ASK}(t) = m(t) * \cos(2 * \pi * f * t + \varphi)$, with $m(t)$ the *modulant*, i.e. signal containing the information to modulate. An example of such modulation is shown in figure 2.4.

- **Frequency Shift Keying, of FSK :** the information is coded over shifts in the signal's frequency : $s_{FSK}(t) = A * \cos(2 * \pi * f(t) * t + \varphi)$. The example of figure 2.1 is a frequency modulation, the frequency increasing when modulating a "1" and decreasing when modulating a "0".

- **Phase Shift Keying, or PSK :** the information is coded over shifts in the

phase offset : $s_{PSK}(t) = A * \cos\big(2 * \pi * f * t + \varphi(t)\big)$. An example of such modulation is shown in figure 2.2.

A digital modulation can be built on one of those schemes, such as with the 2FSK, modulating a "0" with a frequency and a "1" with another, use more possible symbols to code more bits, such as with the Quadrature Phase Shift Keying, or QPSK, using four different phase offsets to code either "00", "01", "10" or "11", or even combine several of them, such as with the Quadrature Amplitude Modulations family, or QAM, combining amplitude and phase modulations.

2.2 Complex representation and constellations

A sinusoidal signal as described in equation 2.1 can also be re-written using the angle sum identity $\cos(a + b) = cos(a)cos(b) - sin(a)sin(b)$ as follows:

$$s(t) = A * \cos(2 * \pi * f * t)\cos(\varphi) - A * \sin(2 * \pi * f * t)\sin(\varphi) \tag{2.2}$$

Or, by using the fact that $\sin(x) = \cos(x + \frac{\pi}{2})$:

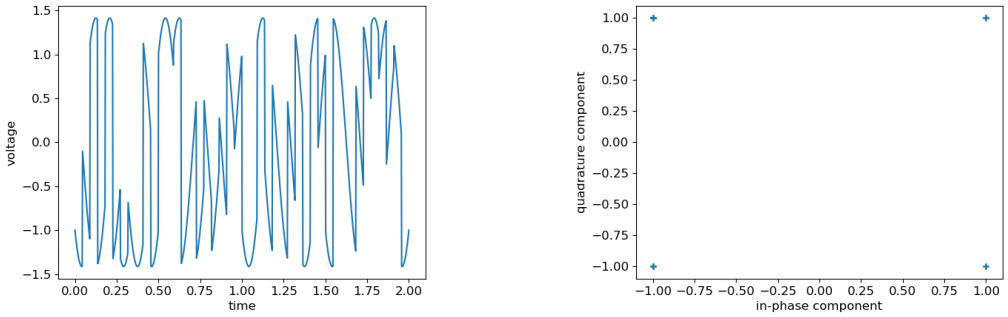
$$s(t) = A * \cos(2 * \pi * f * t)\cos(\varphi) - A * \cos(2 * \pi * f * t + \frac{\pi}{2})\sin(\varphi) \tag{2.3}$$

In this equation, the $A * \cos(\varphi)$ component is called *in phase*, as it is carried by the original sine wave, and the $A * \sin(\varphi)$ component is called *quadrature*, as it is carried by a sine wave with a phase offset of $\frac{\pi}{2}$, hence orthogonal to the first one.

These two orthogonal carriers can, for example, be used to transmit two symbols at the same time. To represent signal samples, we then plot each one of them on a two-dimensional plane, using by convention the *in phase* component, also called I, as X coordinate, and the *quadrature* component, also called Q, as Y coordinate. This is also called the *complex representation* of the signal, as it can also be seen as the representation of $I + i * Q$ in the complex plane.

Using this representation is often easier to visualize phase modulations, the phase being represented as the complex numbers' arguments, as show in figure 2.2, where we represented a QPSK-modulated signal first the same way as before, and then the corresponding points in the complex domain. The complex representation allows to see in a single picture all symbols, here the four different phases used by a QPSK modulation. This figure showing an ideal case, what is shown on figure 2.2b is the theoretical set of symbols for the modulation, also called the modulation's *constellation*. On the temporal view, however, the phase shifts are visible, but hard to decipher. In general, it is easier to identify a frequency modulation on the temporal representation, a phase modulation on the complex representation, an amplitude modulation being identifiable on both. However, the complex representation as shown in figure 2.2b masks the temporal aspect, not allowing to see the sequence of symbols. It is then also possible to represent the signal in three

dimensions, one being the time and the two others representing the in-phase and quadrature components.



(a) Temporal representation (b) Complex representation

Figure 2.2: Representations of a QPSK modulated signal (ideal)

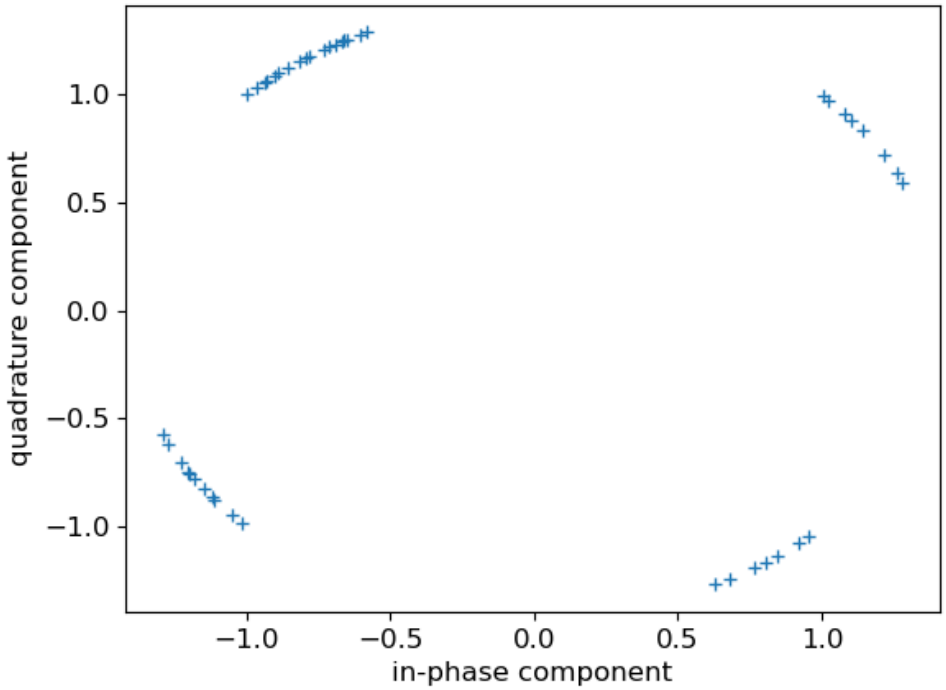


Figure 2.3: Complex representation of a QPSK modulated signal (error on the frequency)

Another interest of this representation is that is allows to see the sine waves as a rotation on the complex plane. Moreover, a frequency shift Δf appears as a multiplication of $I + iQ$ by $\cos(2 * \pi * \Delta f * t) + i * \sin(2 * \pi * \Delta f * t)$, which can

also be written $e^{2i*\pi*\Delta f*t}$, which results in the addition of a rotation of frequency Δf in this representation of the signal.

Proof. As shown in equation 2.3, the I and Q components are defined as the factors of the cosine and sine functions in the signal decomposition, usually $A*\cos(\varphi)$ and $A*\sin(\varphi)$.

If we add the frequency shift, the signal becomes:

$$s(t) = A*\cos\Big(2*\pi*f*t+(2*\pi*\Delta f*t+\varphi)\Big)$$

Hence the following new decomposition in in-phase and quadrature components:

$$s(t) = A*\cos(2*\pi*f*t)\cos(2*\pi*\Delta f*t+\varphi) - A*\cos(2*\pi*f*t+\frac{\pi}{2})\sin(2*\pi*\Delta f*t+\varphi)$$

By using the identities $\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$ and $\sin(a+b) = \sin(a)\cos(b) + \cos(a)\sin(b)$, and replacing $A*\cos(\varphi)$ and $A*\sin(\varphi)$ respectively by I and Q for an easier reading, we can then develop the in-phase and quadrature components of the new signal as follows:

$$\begin{cases} A*\cos(2*\pi*\Delta f*t+\varphi) = I*\cos(2*\pi*\Delta f*t) - Q*\sin(2*\pi*\Delta f*t) \\ A*\sin(2*\pi*\Delta f*t+\varphi) = I*\sin(2*\pi*\Delta f*t) + Q*\cos(2*\pi*\Delta f*t) \end{cases}$$

The complex representation of such a signal then becomes:

$$I*\cos(2*\pi*\Delta f*t) - Q*\sin(2*\pi*\Delta f*t) + i*\Big(I*\sin(2*\pi*\Delta f*t) + Q*\cos(2*\pi*\Delta f*t)\Big)$$

Which can be simplified as $\Big(I+iQ\Big)*\Big(\cos(2*\pi*\Delta f*t)+i\sin(2*\pi*\Delta f*t)\Big)$, or $\boxed{(I+iQ)*e^{2i*\pi*\Delta f*t}}$.

□

For example, for the QPSK in figure 2.2b, where we simulated an emission and reception at 3Hz, adding 0.03Hz on the emission side results in figure 2.3, where this rotation is clearly visible, and might hinder correct demodulation after some time.

2.3 Fourier Transform

One of the most useful tools in signal processing, along with other fields, is the *Fourier Transform*. This operation allows to represent functions of time or space from a frequential perspective. More precisely, it uses a decomposition of the signal in a sum of sinusoids at different frequencies, giving a new signal represented as a function of the frequeny. At each given frequency, the value of the Fourier transform is a complex number with a magnitude equal to the amplitude of the corresponding sine wave, and with an argument equal to its phase offset. Put more simply, if

$TF(s)(f)$ if the value of the Fourier Transform of signal s at frequency f, s is the sum, for each frequency f, of the terms $TF(s)(f) * \exp^{2i*\pi*f*t}$.

To be more precise, the Fourier Transform is defined as the following infinite integral:

$$TF(s)(f) = \int_{t=-\infty}^{+\infty} s(t) \exp^{-2i*\pi*f*t} \, dt \tag{2.4}$$

In computer science and digital signal processing, we work with discrete signals, meaning we have data points separated by sampling periods. In this case, we use the *Discrete Fourier Transform*, or DFT:

$$TF(s)(f) = \sum_{k} s(k) \exp^{-2i*\pi*f*t(k)} \tag{2.5}$$

With $t(k)$ the timestamp of the k^{th} sample, and $s(k)$ its value.

As we only have a finite number of points, we compute an approximation of the Fourier Transform on those points. In this case, we obtain the same number of points in the output as in the input, uniformly spread on the spectrum.

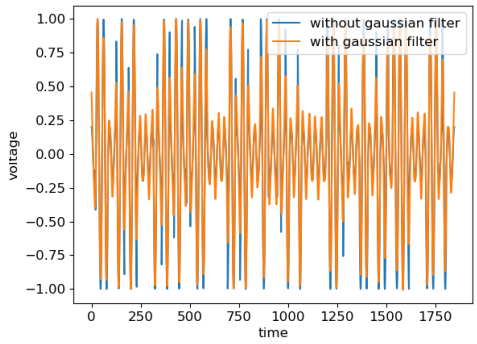
The spectrum covered by a Discrete Fourier Transform is defined by the sampling rate of the input: indeed, as stated in the *Nyquist-Shannon sampling theorem*, to correctly receive a signal that has a frequency f (relatively to the receiver's central frequency), one must use a sampling rate greater than $2 * f$. This also applies to frequencies before the central frequency, meaning that using a sampling rate of f_s while listening on a central frequency f_c, one can receive signals between $f_c - \frac{f_s}{2}$ and $f_c + \frac{f_s}{2}$. The DFT then outputs N frequencies, ranging from $f_c - \frac{f_s}{2}$ to $f_c + \frac{f_s}{2}$.

Finally, the Fast Fourier Transform, or FFT, is an optimized algorithm for computing the Discrete Fourier Transform: as the DFT requires ordinarily to sum N terms for N frequencies with an N-point signal, resulting in a quadratic complexity, the FFT is able to obtain the same result in $O(N \log(N))$. We will not go in more detail into this optimization, as various versions exist and have different properties on the input they accept.

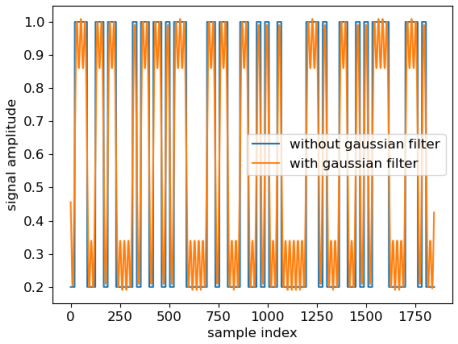
2.4 Pulse-shaping filters and moment of decision

An important concept in signal processing is the *spectral efficiency*, which is related to the width of the signal in the frequency domain. A less efficient, i.e. wider, signal limits the number of channels that can be used simultaneously because of the space it occupies, and requires that the equipment used by both the receiver and emitter support such wide bands.

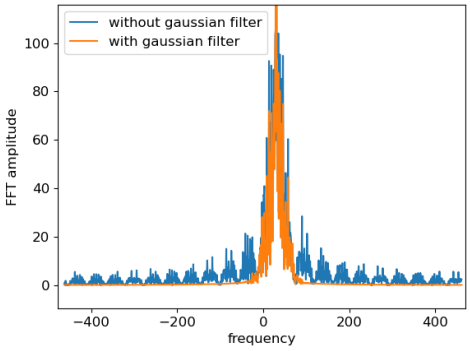
As we explained before, according to the *Nyquist-Shannon sampling theorem* one can only receive frequencies between $f_c - \frac{f_s}{2}$ and $f_c + \frac{f_s}{2}$. This means that a less spectrally efficient signal requires higher sampling rates from the equipment involved in the communications.



(a) Temporal representation



(b) Amplitude



(c) Fourier Transform

Figure 2.4: Comparison of two 2ASK signals with and without gaussian filter

This led to the creation of signals with shapes increasing their spectral efficiency. Indeed, using simple shapes for the signal generates abrupt slopes when transitioning from a symbol to the next one, introducing high-frequency components in the signal spectrum (one can imagine an abrupt transition from one value to another as a nearly-vertical slope, which would then have a nearly infinite derivation). One then needs shapes that can generate a continuous curve when symbols are put one behind the other. One of the most used shapes is the Gaussian curve, obtained by applying what is called a Gaussian filter to the original input to modulate, for example in the Gaussian Frequency Shift Keying, or GFSK, used in numerous Internet of Things protocols such as the well-known Bluetooth.

To illustrate, figure 2.4 shows the temporal representation of two signals, their instantaneous frequencies, i.e. derivation of the temporal signal, and spectrum without noise. Both signals modulate the same input, but the first one uses the frequency modulating a symbol for its entire duration, while the other uses a Gaussian-shaped input.

However, there is a drawback to using such pulse-shaping filters: during most of the symbol period, the signal lies between several possible values, increasing the

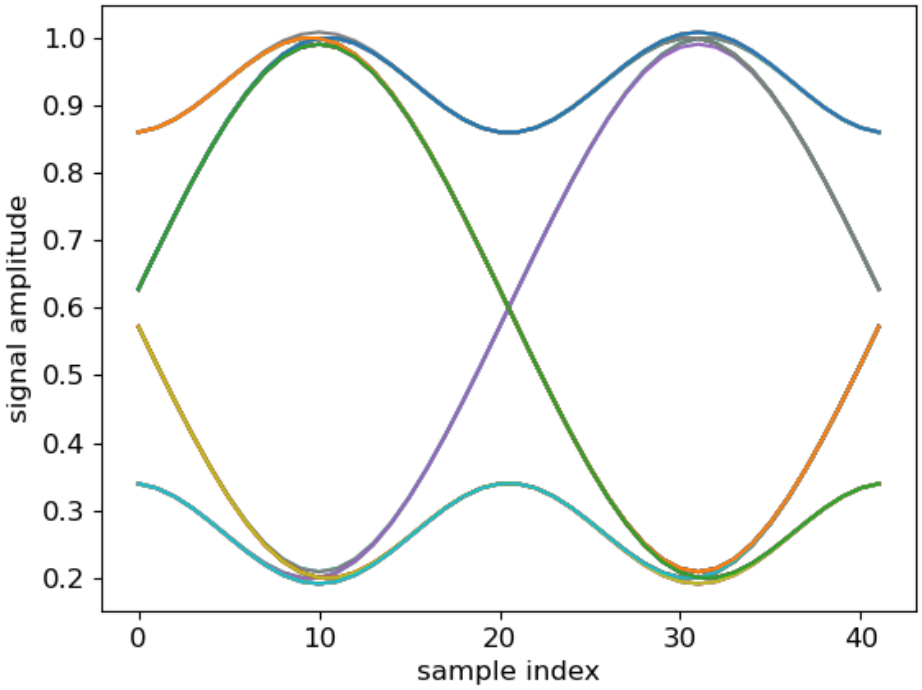


Figure 2.5: Eye diagram for an ASK with gaussian filter

risk of wrong demodulation especially in the presence of noise. We therefore need to choose the adequate sample in a given symbol period where the symbol currently sent is the clearest. To help us to do so, we can use another representation that allows us to see more clearly the different values the signal takes during each symbol period, called an eye-diagram. This representation is built by representing all pairs of symbols that were sent overlapping on a single time axis, as shown in figure 2.5 for the ASK with Gaussian filter of the previous example (here, as it is an amplitude modulation, the amplitude is represented).

This figure takes the shape of two "eyes", hence the diagram name, and shows all possible sequences of two symbols with the transitions between them. For example, a series of "1" bits results in the curve that stays around an amplitude of 1, or a "1" followed by a "0" passes through the first peak around 1, then by the center of the figure to go to the second lower peak around 0.2. On this diagram, the points where all curves focus on two positions show where the symbols are the clearest, with the lowest risk of confusion between them. In our example, the moment of decision is then around the tenth sample of each symbol period.

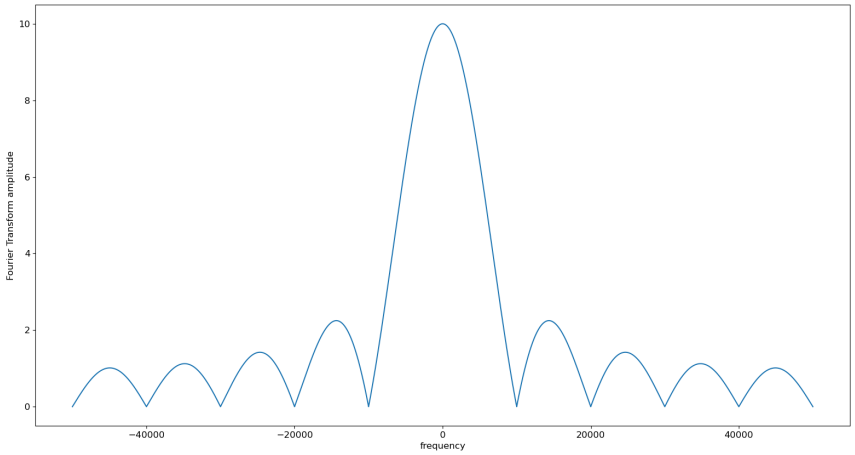


Figure 2.6: Fourier Transform of a door of period $\frac{1}{10000}$ seconds

2.5 The case of Orthogonal Frequency Division Multiplexing

Nowadays, many wireless protocols, especially those designed for long-distance communications, are abandoning the simple modulations described previously to favor a more complex, but more efficient modulation scheme : the Orthogonal Frequency Division Multiplexing, or OFDM. This scheme is used, among others, in WiFi (except its older versions), satellite communications, and even in mobile telephony with the downlink of 4G and 5G networks.

Its principle is to create series of fixed-length symbols, usually with a QAM, assigning each one of them to a specific frequency before sending its inverse Fourier Transform. By doing so, one then sends one symbol per period needed to send one of these blocks on a large number of channels.

To understand the main idea behind OFDM, an important point is the impact of the use of such blocks on the spectrum. The Fourier Transform is initially a decomposition of sinusoidal waves extending on an infinite time axis, so reducing a signal with some fixed frequential components to a given period of time T_0, as it is done in OFDM, has a direct impact on the spectrum of such block. As it is equivalent to multiplying the theoretical infinite signal by a function g with value 1 only during this period, its Fourier Transform is the result of the convolution between the theoretical Fourier Transform and the Fourier Transform of this function, which is known and is expressed by $TF(g)(f) = T_0 * \frac{\sin(\pi * f * T_0)}{\pi * f * T_0}$, represented in figure 2.6. This kind of function is called a sinc function. The interesting fact with this function is that it reaches its maximum value around frequency 0, and then periodically reaches 0 every $\frac{1}{T_0}$. This makes it possible to calculate which frequencies can be used by the channels, as each symbol is multiplied by a sinc : indeed, by choosing

to space them by a multiple of $\frac{1}{T_0}$ with T_0 the length of the block in time, each sinc reaches a maximum at the frequency where the symbol is, and does not interfere with the others.

To give a more concrete example, the 4G downlink uses blocks containing 2048 samples at 30.72MHz, which means it can use channels spaced by 15KHz, which would be inconceivable with other schemes.

To avoid interference between consecutive blocks, systems using OFDM choose between two different methods:

- **Guard periods:** this technique consists in introducing fixed-length periods of non-emissions between the blocks, to space them and reduce Inter-Symbol Interferences (ISI). However, this technique requires to be able to find precisely the period where the block was emitted in order to isolate it correctly and recover its Fourier Transform, and the symbols behind it

- **Cyclic prefixes:** this technique consists in copying a fixed length portion of the end of the block at its beginning. This way, the block plus prefix set contain the same frequential components as the initial block. Thus, by taking any slice of the size of the block inside this set, one should be able to retrieve correctly the symbols in the Fourier Transform. However, light ISI might happen in the beginning of the prefix and the end of the block, since there is no period where nothing is emitted

The last important point we would like to highlight about OFDM concerns a phenomenon known in signal processing as "channel selectivity": waves going from a point A to a point B interacts differently with the environment depending on their frequencies (due to refraction or reflection on obstacles for example); this leads to non-uniform distortion through the spectrum, from the receiver point of view. This phenomenon may alter the different symbols modulated with an OFDM scheme, depending on their positions in the spectrum. Thus, OFDM systems have introduced *reference symbols* in the emitted blocks: these symbols are fixed values sent on specific sub-channels, sometimes only at regular intervals but not all the time, and are defined in the protocol specifications. Then, the receiver uses the difference between the theoretical reference symbols and the signals it received in their place to estimate the distortion experienced by each of them. Then, by interpolating between the reference symbols, it is able to estimate the distortion introduced on the whole spectrum, in order to correct it.

As will be discussed later in this manuscript, these aspects, while beneficial to reliable data transmission using this scheme, raise issues for protocol-agnostic analysis of transmissions, as it would require prior knowledge of several parameters, notably the values and positions of the reference symbols in use.

As, from the defensive point of view, we focused on such approaches, we do not support this scheme natively in our monitoring and analysis approaches. Then, from an offensive point of view, it offers interesting perspectives, even if difficult

to exploit. Thus, we discuss possible adaptations and improvements to support OFDM-based protocols in the different chapters and the perspectives of this PhD.

2.6 Conclusion

In this chapter, we presented the necessary signal processing prerequisites for the rest of this manuscript. We first briefly presented the main principles of the most used digital modulations, namely frequency, amplitude and phase modulations. We then explained the notions of complex representation and modulation constellation, useful to understand the representations of various signals, in particular phase modulations. Subsequently, we introduced the spectral efficiency issue, how it is addressed by the use of pulse-shaping filters, and the crucial problem of the right moments to demodulate a signal that ensues, using a representation called an eye-diagram. Finally, we provided information about Orthogonal Frequency Division Multiplexing, a scheme now widely used in telecommunications, notably in WiFi or 5G networks.

In the following, we present the three main contributions of this PhD, respectively presented in Chapters 3, 4 and 5.

Chapter 3

Inter-protocolar pivotal attacks

Contents

In this chapter, we first present Wazabee[Cayre 2021b], a common work with Romain Cayre, which is a new approach that allows using full-Zigbee capabilities on Bluetooth devices, without any modifications on the devices or on the Zigbee protocol stack. Then, after having explained the theoretical basis behind this approach, we present a reflection on a possible offensive use of such pivots, and which protocols would be compatible in this way. We finally discuss the possible countermeasures against such attacks, before presenting our own defensive measures in the next chapters.

3.1 Introduction

The idea behind Wazabee came during the development of demodulators for both the GFSK modulation used by the Bluetooth Low Energy, and the O-QPSK used by several IEEE 802.15.4 based protocols, such as Zigbee or 6LoWPAN. Indeed, to simplify its development, we decided to leverage the equivalence between the O-QPSK and another modulation, called "Minimum Shift Keying" (MSK): the O-QPSK adds a delay of half a symbol period to the *quadrature* component of a

QPSK, thus only allowing transitions from adjacent symbols (for example, a "11" could only become a "01" or a "10", before becoming a "00"), and the MSK codes the information in shifts between adjacent symbols of a QPSK, the rotation direction giving the bit to demodulate. Thus, by knowing or estimating the first QPSK symbol of the O-QPSK, it is possible to demodulate it by first demodulating with an MSK demodulator, then re-tracing the transitions from the first state to retrieve the corresponding O-QPSK symbols.

Interestingly, according to diverse works in signal processing [Speth 2004], specific values of the distance between the two frequencies of a GFSK modulation give an equivalence between it and a Gaussian Minimum Shift Keying (GMSK), a version of the MSK where a gaussian filter was applied to the phase values to smoothen the transitions. As we will precise further in the following, since the value of this parameter for the Bluetooth Low Energy is close to the right one to get the equivalence, we coded the GFSK demodulator as a GMSK modulator. Furthermore, to simplify the development, we implemented a version ignoring the gaussian filter, by taking the most probable transition value in each symbol period.

Thus, when we started developing the Zigbee demodulator, we were surprised by the similarity of the two codes. As we will see later, this equivalence was already explored in the signal processing field of Cross Technology Communications (CTC), where several methods were proposed to communicate between Zigbee and Bluetooth Low Energy devices. However, these methods require some form of cooperation between the devices, introducing modifications to the protocols to allow them to connect, thus they could not be practically used in an offensive context.

However, the recently standardized Bluetooth 5 introduced new capabilities for Bluetooth Low Energy, among which the possibility to increase the sample rate, matching the one from Zigbee. Thus, we decided to make use of this new standard to build Wazabee, an attack allowing to communicate with Zigbee networks from Bluetooth Low Energy chips, without any cooperation from the attacked device.

3.2 Related work

In this section, we describe the different known possibilities to build a pivot inside a wireless network. To do so, we first study the case of devices already supporting several protocols, which can be attacked using one, then used with the other to access another part of the network, and then highlight the possibility of using single-protocol devices with low-layer equivalences to achieve a similar result.

3.2.1 Multiprotocol devices

A pivoting attack aims at taking advantage of the coexistence of multiple protocols in the same environment in order to compromise new objects. The most natural approach for this attack is to compromise an object supporting multiple radio communication protocols, allowing to perform the attack using the provided API. As an example, in [Bachy 2015], Bachy et al. compromise a smart-TV using *HbbTV*

communication protocol, then use it to reconfigure the firewall embedded in the ADSL box using LAN protocols (Ethernet or WiFi).

Several hardware devices allow such attacks to be carried out. For instance, *Software Defined Radio* devices are designed for a generic purpose, allowing communications through multiple protocols, regardless of their modulation and frequency bands. However, so far, these devices are only used for prototyping and experimentation purposes.

There are also chips that integrate different wireless devices. For example, *B-L475E-IOT01A* [IOT 2018], based on the *STM32L4* microcontroller intended for IoT devices, supports multiple wireless protocols (such as *Bluetooth*, *WiFi* or *NFC*). Similarly, the *CC2652R*[CC2 2019] from *Texas Instruments* is compliant to multiple radio technologies in the ISM band. The compromise of such a chip greatly facilitates the implementation of a pivoting attack targeting one of the wireless protocols supported by the chip. However, such chips are expensive, and their use is quite specific, which limits their deployment in IoT networks.

3.2.2 Single-protocol devices

Since most connected objects only embed one wireless device, the practical implementation of a pivoting attack is much more complex. We are not aware of existing research specifically addressing this issue from an offensive perspective. However, some contributions explored related topics. The most relevant contributions are related to *Cross-Technology Communications* (*CTC*) solutions, that are aimed at providing a communication system between two single-protocol devices supporting heterogeneous wireless communication protocols. However, to our knowledge these contributions did not investigate the use of this technology in security or in an offensive perspective. There are two main categories of *CTC*, named *Packet-level CTC* and *Physical Layer CTC*.

The *Packet-level CTC* approach relies on some information linked to the packets. As an example, K. Chebrolu et al. use packet duration in order to transmit data [Chebrolu 2009], while the *FreeBee* [Kim 2015] approach by S. Min Kim is based on the time interval between beacon frames. From an offensive perspective, these approaches could be interesting to exfiltrate some data, but they are not relevant for pivoting attacks. Other limitations, such as a low data throughput, are inherent to these approaches and hamper their practical use.

Physical Layer CTC approaches consist in emulating a technology using the signal generated by another one. As an example, Z. Li et al. simulate a *Zigbee* frame using a WiFi transceiver [Li 2017]. Similarly, W. Jiang et al. have presented an approach named *BlueBee* [Jiang 2017], allowing to simulate *Zigbee* frames using a *BLE* transceiver, and another approach called *XBee* [Jiang 2018], enabling to receive *Zigbee* frames from a *BLE* receiver. However, these solutions have major limitations that prevent their use in an offensive perspective. As an example, the selection of a *Zigbee* channel by *BlueBee* is based on the channel hopping algorithm of *BLE* connected mode, so it requires to establish a *BLE* connection with another

BLE device. Similarly, adding a specific identifier before the data included in the frame is needed in order to receive a *Zigbee* frame using *XBee*, so it requires the cooperation of the *Zigbee* transmitter. These constraints can be easily addressed if the use of *CTC* is legitimate and deliberate, however they prevent the use of these solutions in a context of attack and especially for pivoting attacks. Our approach overcomes these limits and provides a reliable two-way *CTC* that doesn't require the cooperation of other devices: as a consequence, it may be used in an offensive context.

The *Packet-in-Packet* strategy [Goodspeed 2011b], proposed by T. Goodspeed et al. consists in encapsulating a complete radio frame into an application-level *payload*: a misidentification of the beginning of the encapsulating frame by the receiver (e.g., due to interferences causing bitflips during the demodulation) can lead to the interpretation of the encapsulated frame. This strategy is particularly interesting for bypassing software checks performed at the protocol layer, and may thus allow attackers to access and control the lower layers of the radio device. The authors highlight a possible use of this attack to perform a pivoting attack, e.g., to inject radio traffic corresponding to a wireless protocol different from the protocol natively supported by the radio device, under certain specific conditions. However, this strategy can only be applied to a limited number of protocols, and can only be achieved if the modulations used have similar characteristics (frequency bands, bandwidth, etc). For instance, M. Millian and V. Yadav discuss the possibility of encapsulating *802.15.4* traffic into 802.11 frames [Millian 2015]. However, they stress the difficulty of such a strategy due to the differences between the two technologies.

T. Goodspeed has also discovered a vulnerability in the *nRF24L01+* chip, that facilitates sniffing and frame injection on a set of protocols (such as *Bluetooth Low Energy* or *Enhanced ShockBurst*) using *Gaussian Frequency Shift Keying* modulation. He was able to divert the use of a register dedicated to the address selection to select an arbitrary preamble [Goodspeed 2011a]. Exploiting this vulnerability allowed him to add a promiscuous mode for the *Enhanced ShockBurst*, which is not natively supported by the chip. However, it is also possible to divert the use of this register to detect specific preambles used by different wireless technologies, as long as similar modulations and bit rates are used. This vulnerability has been used by M. Newlin to develop a firmware aiming to add advanced sniffing capabilities for the *Enhanced ShockBurst* and *Mosart* protocols to the *nRF24* chip [Newlin 2016].

D. Cauquil has also disclosed a similar vulnerability in other *Nordic Semiconductors* chips [Cauquil 2017b], and has developed a similar tool for the *nRF51*. He was then able to implement communication primitives for a proprietary protocol not initially supported by the chip, allowing it to control a mini-drone [Cauquil 2017c]. An implementation of these primitives has been integrated into the *radiobit* [Cauquil 2017a] project.

These research works present some first techniques and experimental results that illustrate the practical feasibility of pivoting attacks targeting wireless protocols. However, these techniques have several limitations which strongly restrict their use: they require an active cooperation of other devices, or the modulation

of the native protocol and the pivoting protocol must be similar and sometimes depend on the use of specific chips (such as *Nordic SemiConductors nRF24* and *nRF51* chips). Our main contribution is to present a pivoting attack strategy that overcomes some of these constraints, allowing the implementation of communication primitives targeting a wireless technology using a modulation different from the one natively supported by the chips that doesn't require the cooperation of other nodes, and that could possibly be generalised to multiple hardware devices from different manufacturers.

3.3 Wazabee, diverting Bluetooth chips to attack ZigBee Networks

3.3.1 Principles of the attack

3.3.1.1 From GFSK to GMSK

Our approach is based on the similarities between the modulations used by Bluetooth Low Energy (BLE) and IEEE 802.15.4 protocols, such as Zigbee.

First, BLE uses a GFSK modulation, that consists in coding bits by using two symmetrical frequencies around the carrier, smoothing the transitions between two bits by using a gaussian filter, as we explained in chapter 2. In terms of rotation in the complex representation, it means the signal is either "rotating" clockwise or counterclockwise at the same speed, once the carrier frequency is removed from it.

Interestingly, in a simple 2FSK, by comparing the sampling frequency of the receiver and the rotation speeds, it is possible to estimate the position of the next sample depending on the coded bit. The sign of the angle between consecutive points could thus be used to demodulate the signal. For example, if the difference between one of the coding frequencies and the carrier is equal to a quarter of the sampling rate, it means that if the signal was "rotating" clockwise, the angle between two consecutive samples would be $-\frac{\pi}{2}$, and conversely if the signal was "rotating" counterclockwise, it would be $\frac{\pi}{2}$. In this specific case, the 2FSK becomes then equivalent to an other modulation at this sampling rate: the Minimum Shift Keying. As explained in the introduction of this chapter, the Minimum Shift Keying is a modulation where the bits are coded by either a rotation of $\frac{\pi}{2}$, or a rotation of $-\frac{\pi}{2}$ in the complex plane.

The ratio between the sample rate and the distance between coding frequencies in a 2FSK is a known quantity, sometimes called *modulation index*. Let us note, however, that the term "modulation index" is used in the different modulation classes, thus with different meanings, and can even have different definitions for a given modulation. Overall, the "modulation index" is a quantity used to compare different modulations by characterizing them, and in particular the distance between the symbols they use. In the following, we will use the most common definition of

the modulation index m for a 2FSK:

$$m = \frac{\Delta f}{f_s} \tag{3.1}$$

with Δf the difference between the two coding frequencies, and f_s the sampling frequency.

With this definition, as stated in the Bluetooth Core Specification v5.3[Blu 2021], the modulation index of the GFSK used in Bluetooth Low Energy is between 0.45 and 0.55. This means that the distance between the two coding frequencies is about half the sample rate, i.e., the distance between one of them and the carrier is approximately a quarter of the sample rate. Thus, looking at the consecutive peaks of the gaussian shapes used for the different symbols, the angle shift between two points is either $\frac{\pi}{2}$ or $-\frac{\pi}{2}$, depending on the modulated bit. In fact, according to the state of the art, this leads to a full equivalence between a GFSK with a modulation index $m = 0.5$ and a GMSK.

3.3.1.2 From O-QPSK to MSK

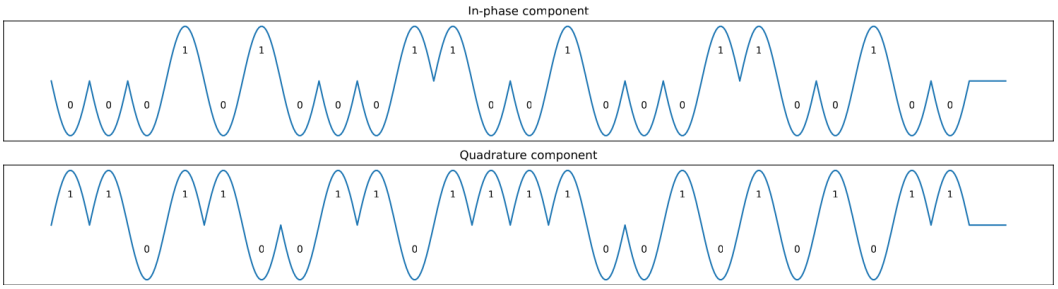


Figure 3.1: Representation of a QPSK with semi-sinusoidal pulse-shaping

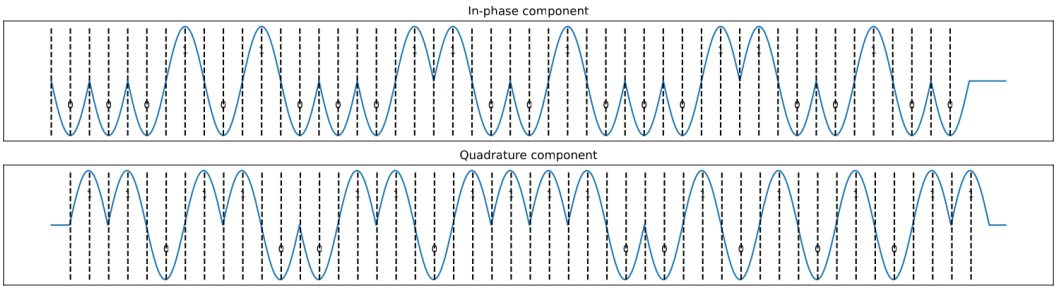


Figure 3.2: Representation of an O-QPSK with semi-sinusoidal pulse-shaping

Some IEEE 802.15.4 protocols, among which Zigbee and 6LoWPAN, use a same Physical Layer based on an Offset Quadrature Phase Shift Keying, or O-QPSK.

The principle behind this modulation is to avoid abrupt transitions between a phase and the symmetrical one across the origin of the complex plane, that can occur with a standard QPSK (for example when coding "11" then "00", the signal

jumps from a phase offset of $\frac{\pi}{4}$ to $\frac{-3*\pi}{4}$). To do so, the O-QPSK adds a time offset of half the symbol period to the quadrature component of the QPSK, as shown in figures 3.1 and 3.2. By doing so, it then forces to only change one symbol at a time.

Thus, it virtually only uses rotations of $\frac{\pi}{2}$ or $-\frac{\pi}{2}$ during half symbol-periods, resulting in a scheme similar to a Minimum Shift Keying. We can then, by estimating which of the four symbols was the first one, build an equivalence between series of these rotations and the sequence of states that was modulated in O-QPSK, making it possible to use an MSK demodulator to receive such IEEE 802.15.4 protocols. Moreover, as two communicating devices are not initially phase-synchronized, an MSK equivalent of an O-QPSK transmission, considering the first state to be "00" for example, is effectively received and correctly demodulated by an O-QPSK receiver, which "corrects" its phase synchronization to detect the right Zigbee preamble, independently of the phase from which the MSK started.

3.3.1.3 Low layers compatibility

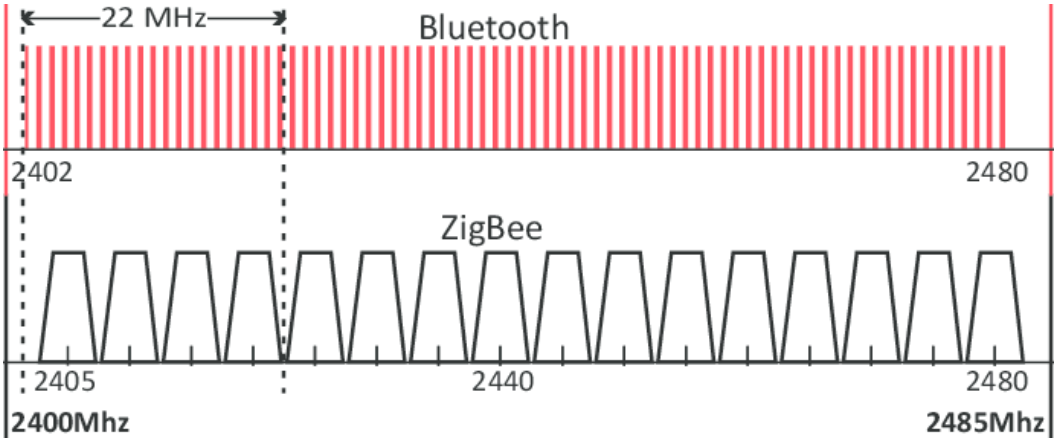


Figure 3.3: Overlapping of Bluetooth and Zigbee channels

Beyond the issue of the modulations, we needed to address three other issues: the correspondence of the data rates between the protocols, the channel coding of the two protocols, which introduces modifications to the data to send for link reliability, and finally the frequencies used by the two protocols.

The IEEE 802.15.4 O-QPSK Physical Layer defines a data throughput of 250 kbps for emissions in the 2450, 915, 780 and 2380 MHz bands. However, this rate defines the rate of "useful data", or data without channel coding. This Physical Layer also uses a technique of Direct Sequence Spread Spectrum, or DSSS, more precisely what they call a "16-ary quasi-orthogonal modulation technique", which transforms each of the 16 possible sequences of 4 bits in the data to send in 16 larger symbols of 32 bits, nearly orthogonal to each other. Thus, the data rate used on the wireless medium for these larger symbols is 2 Mbps.

The Bluetooth Low Energy protocol initially supported only a data rate of 1 Mbps with its LE 1M PHY; however, starting with the Bluetooth Core Specification

v5, it added the optional LE Coded and LE 2M PHY, respectively a coded 250 kbps link and a 2 Mbps alternative to the classic LE 1M.

Thus, LE 2M compatible BLE devices and Zigbee devices can communicate with the same bit rate over the physical medium. However, this takes the Zigbee's DSSS into account, which hinders the possibility to control what Zigbee devices send to communicate with a BLE receiver. As we will explain further in the following, we chose to focus on communicating as both a Zigbee receiver and emitter with BLE devices, and ignore the other direction because of this issue.

BLE devices also integrate a channel coding functionality called data whitening. This method consists in generating a pseudo-random binary mask for the data to send, known by the recipient of the communication, to XOR with it in order to improve the spread of "1" and "0" bits in the data that is then sent. We found two methods to bypass this measure, which we will explain further in the following: pre-generating the same mask to XOR the data before the whitening, thus reverting it, or, if possible, deactivate it with a low level control on the chip.

Finally, both Zigbee and BLE devices can communicate in the 2.4-2.5 GHz ISM band. As we are focusing on using BLE devices to communicate in a Zigbee network, we have to make them communicate on frequencies supported by Zigbee. A first and easy way to do so is to use channels that already overlap between Zigbee and BLE, shown on figure 3.3. Indeed, BLE uses 40 channels from 2402 MHz to 2480 MHz, 2MHz apart from each other, while, in the same band, the Zigbee channels are 5 MHz apart from each other, from 2405 MHz to 2480 MHz. This means the central frequencies $2410 + k * 10$ MHz, with k any integer from 0 to 7, are common to both. Another way is to use a common feature of Bluetooth 5 compliant BLE devices, which is to be able to select arbitrary frequencies in the 2.4-2.5 GHz band.

3.3.1.4 Re-implementing the Zigbee DSSS from a BLE device

All Zigbee devices, along with other IEEE 802.15.4 devices with similar lower layers, use the same DSSS technique, transforming 4-bits sequences into 32-bits sequences, called pseudo-random noise (PN) sequences, as shown in table 3.1. To communicate with such devices from GFSK-based protocols, we thus need, for each sequence of four bits of data, to first get the corresponding PN sequence, and then to compute the MSK equivalent of such sequence. The final equivalence, giving which bit sequence to send from an MSK device for each 4-bit sequence, can be found in table 3.2.

3.3.2 Experimentation

It is important to validate the *WazaBee* attack on chips from different manufacturers. We have chosen two different chips: nRF52832 designed by Nordic Semi-Conductors and CC1352-R1 designed by Texas instruments. In this section, we describe the proof of concept implementations on these chips, then we present the experiments conducted to evaluate our attack.

Block ($b_0b_1b_2b_3$)	PN Sequence (c_0c_1 ... $c_{30}c_{31}$)
0000	11011001 11000011 01010010 00101110
1000	11101101 10011100 00110101 00100010
0100	00101110 11011001 11000011 01010010
1100	00100010 11101101 10011100 00110101
0010	01010010 00101110 11011001 11000011
1010	00110101 00100010 11101101 10011100
0110	11000011 01010010 00101110 11011001
1110	10011100 00110101 00100010 11101101
0001	10001100 10010110 00000111 01111011
1001	10111000 11001001 01100000 01110111
0101	01111011 10001100 10010110 00000111
1101	01110111 10111000 11001001 01100000
0011	00000111 01111011 10001100 10010110
1011	01100000 01110111 10111000 11001001
0111	10010110 00000111 01111011 10001100
1111	11001001 01100000 01110111 10111000

Table 3.1: Block/PN sequence correspondence table

Block $b_0b_1b_2b_3$	PN Sequence - MSK encoding (m_0m_1 ... $m_{29}m_{30}$)
0000	11000000111011110101110011011 00
1000	10011100000011101111010111001 10
0100	01011001110000001110111101011 10
1100	01001101100111000000111011110 10
0010	11011100110110011100000011101 11
1010	01110101110011011001110000001 11
0110	11101111010111001101100111000 00
1110	00001110111101011100110110011 10
0001	00111111000100001010001100100 11
1001	01100011111100010000101000110 01
0101	10100110001111110001000010100 01
1101	10110010011000111111000100000 101
0011	00100011001001100011111110001 000
1011	10001010001100100110001111111 000
0111	00010000101000110010011000111 11
1111	11110001000010100011001001100 01

Table 3.2: Correspondence table of PN sequences

3.3.2.1 Wazabee with an nRF52832 chip

The first implementation of the attack was carried out on the *nRF52832* chip. This chip offers great flexibility in the configuration of the embedded radio component

BLE 5.0 (and, in particular, is compliant with the *LE 2M PHY* layer). Its radio *API* is quite close to the *nRF51* one. This *nRF51 API* is well known to the security community for having been massively hijacked in recent years in order to develop offensive tools dedicated to *BLE* and Enhanced ShockBurst (*BTLEJack*, *radiobit*, ...). The prototype was implemented on a development board proposed by *AdaFruit* integrating this chip, the *Adafruit Feather nRF52 Bluefruit LE*.

We present the main steps of the implementation of *WazaBee* attack on this chip. The first step consists in writing an initialization function for the configuration of the radio component to set up the attack in transmission and reception modes. The first relevant register to examine is the *FREQUENCY* register. It expects an integer value in MHz to which 2400 MHz is added to select the right frequency.

The data rate could be controlled through the register *MODE*. It is possible to select the *LE 2M PHY* mode using the constant named *RADIO_MODE_MODE_Ble_2Mbit*.

The detection pattern selection is an important step: in order to maximize the number of frames detected, it is appropriate to use the PN sequence coded in *MSK* corresponding to the symbol 0000, repeated 8 times to be compliant with the preamble of an 802.15.4 frame. The chip can be configured in Big Endian, the following table was generated from PN sequences coded in *MSK* (each having been prefixed with a bit to 0 so it can easily be represented with a 4 bytes word):

```
static uint8_t SYMBOL_TO_CHIP_MAPPING[16][4] = {
{0x60,0x77,0xae,0x6c},  // 11000000111011111010111001101100 (0)
{0x4e,0x07,0x7a,0xe6},  // 10011100000011101111101011100110 (1)
// [...]
{0x78,0x85,0x19,0x31}}; // 11110001000010100011001001100001 (15)
```

The detection pattern selection is carried out via registers *BASE0* and *PREFIX0*, allowing to provide the first value of the table *SYMBOL_TO_CHIP_MAPPING* as *Access Address*.

Registers *PCNF0* and *PCNF1* allow a global configuration of the radio component. For example, they allow to deactivate the *whitening*, to configure a pattern of 4 bytes (3 address bytes + 1 preamble byte) and to configure the recovery of raw blocks, with a maximum size (i.e. 255 bytes) at the output of the demodulator.

Integrity checks are disabled via the *CRCCNF* register, using value 0.

The reception and the decoding of 802.15.4 frames are implemented in the *RADIO_IRQHandler* interrupt handler corresponding to the radio component. These two features apply an algorithm based on the Hamming distance using the table previously described in order to find the symbols initially transmitted. Similarly, the transmission primitive (corresponding to the *send* method) builds the sequence to be transmitted by means of the correspondence table, then transmits the data to the modulator.

3.3.2.2 Wazabee with a CC1352-R1 chip

The second implementation was carried out on the *CC1352-R1* chip manufactured by *Texas Instruments*. The main motivation was to test the approach on a chip offering less configuration possibilities than the *nRF52* chip. The chip natively supports several protocols, including *BLE* and 802.15.4. However, only the Bluetooth API was used for the implementation. This API being common to several chips from *Texas Instruments*, the implementation of the attack should be similar on other systems from the same manufacturer.

The configuration of the radio component can be done through several commands. The command *CMD _BLE5_RADIO_SETUP* is used to initialize the communication with the radio component. We used the default configuration, with the exception of the *pRegOverrideCommon* property, which allows overwriting the default configuration of certain registers of the radio component: it is used here to allow the reception of frames up to 255 bytes, limited to 37 by default. This modification has been, among others, used by Sultan Qasim Khan for the development of a Bluetooth *sniffer* named *Sniffle* [Qasim Khan 2019].

```
uint32_t pOverridesCommon[] =
{
    HW_REG_OVERRIDE(0x5328,0x0000),
    // Increases max RX packet length from 37 to 255
    // Sets one byte firmware parameter at offset 0xA5 to 0xFF
    (uint32_t)0x00FF8A53,
    (uint32_t)0xFFFFFFFF
};
rfc_CMD_BLE5_RADIO_SETUP_t RF_cmdBle5RadioSetup =
{
    .commandNo = 0x1820,
    .status = 0x0000,
    // ...
    .pRegOverrideCommon = pOverridesCommon,
    .pRegOverride1Mbps = 0,
    .pRegOverride2Mbps = 0,
    .pRegOverrideCoded = 0,
};
```

The reception configuration is carried out by means of the command *CMD_BLE5_GENERIC_RX*. Frequency selection is made via the field *channel*, which allows the selection of an arbitrary frequency (for example, the value 0x69 corresponds to 2405 MHz, i.e. channel 11 in *Zigbee* protocol). The *LE 2M PHY* layer is activated by setting the value of the *phyMode.mainMode* field to 0x1. *whitening* can be disabled by providing an initialization value of 0 in the *whitening.init* field. Packets are allowed to go up with an invalid CRC by specifying 0 in the *rxConfig.bAutoFlushCrcErr* field. Finally, the chip has to be configured in *Little*

Endian, so we provide the detection pattern 0x9b3af703 (corresponding to the PN sequence coded in *MSK* of the symbol 0000, rewritten in *Little Endian*) in the field *pParam→accessAddress*, corresponding to the Access Address. The symbol / PN sequence correspondence table used is similar to the one implemented for *nRF52*, but the sequences have been rewritten in *Little Endian* format. The decoding algorithm is similar to the one implemented on the *nRF52*.

The transmission feature is available using the command *CMD_BLE5_ADV_AUX*. During this step, an extended *advertisement* packet with the succession of PN sequences is constructed. Then, this packet is transmitted to the modulator. Its configuration is comparable to the previous command *CMD_BLE5_GENERIC_RX*.

3.3.2.3 Assessment

Channels	Valid reception rate		Valid transmission rate	
	nRF52832	CC1352-R1	nRF52832	CC1352-R1
11	100%	100%	98%	100%
12	100%	100%	100%	100%
13	100%	100%	95%	100%
14	100%	100%	97%	100%
15	99%	100%	100%	100%
16	100%	97%	90%	100%
17	98%	99%	94%	96%
18	95%	100%	91%	95%
19	100%	100%	97%	100%
20	100%	100%	100%	100%
21	98%	100%	100%	100%
22	95%	98%	100%	100%
23	97%	96%	100%	100%
24	99%	100%	100%	100%
25	100%	100%	100%	100%
26	97%	100%	98%	100%

Table 3.3: Reception and transmission primitives assessment results

Two experiments were carried out in order to assess the reception and transmission primitives previously described. The first experiment, dealing with reception, consisted in transmitting one hundred 802.15.4 frames with a payload including a counter (incremented with each frame) using a *Zigbee* transmitter (AVR RZUSB-Stick Atmel). The development board implementing the attack *WazaBee*, spaced from the transmitter by a distance of 3 meters, received and decoded the corresponding frames, then calculated the FCS corresponding to the received frame to assess its integrity. For each *Zigbee* channel, the frames were classified into three categories: not received, received with integrity corruption, received without integrity

corruption. The results are shown in table 3.3.

It can be seen that the reception primitive of *WazaBee* has a very satisfactory reception rate for the two implementations on all channels, with an average of 98.625 % of the frames received without integrity corruption for the *nRF52832* and 99.375 % for the *CC1352-R1*. In both cases, there is a slight decrease in the reception rate for channels 17, 18, 21, 22 and 23, which can be explained by the interference with WiFi channels 6 and 11, used in our experimental environment. It can also be observed that the *CC1352-R1* seems to present a more stable reception than the *nRF52832*, without any integrity corruption on the frames received while the *nRF52832* missed 0.6875 % of the frames.

The transmission primitive was assessed under similar conditions: the development board implementing *WazaBee* was configured to transmit one hundred frames including a counter, and a 802.15.4 receiver (the RZUSBStick) was placed 3 meters away. Each transmitted frame could also be classified into three categories: not received, received with integrity corruption and received without integrity corruption. The experiment was performed on all *Zigbee* channels, and the corresponding results are shown in the table 3.3.

In both cases and for all channels, the rate of transmitted frames received without integrity corruption by the RZUSBStick is very satisfactory, with an average of valid received frames of 97.5% for frames emitted by the *nRF52832* and of 99.438 % for the ones emitted by the *CC1352-R1*. We observe a similar phenomenon to the one observed during the assessment of the reception primitive for channels 17 and 18, related to the simultaneous use of WiFi channel number 6 in our experimental environment. The rate of frames received with some integrity corruptions is also slightly higher for nRF82832 (with an average of 0.8125 % while the *CC1352-R1* did not miss any frame).

3.3.3 Impact

We believe the impact of *WazaBee* attacks is critical, especially considering the IoT context. Indeed, we demonstrated that it is possible to implement two transmission and reception primitives on a *BLE* chip, allowing reliable communication with multiple wireless protocols based on the *802.15.4* standard. Indeed, the attack has been mainly evaluated by targeting the *Zigbee* protocol, however *Thread* and *6LoWPAN* are also affected as they are based on the same lower layers.

Basically, *WazaBee* allows an attacker to perform a large variety of active or passive attacks targeting these protocols, from a *BLE* chip. As many embedded systems integrate a Bluetooth chip, this dramatically increases the attack surface for systems using *802.15.4* protocols.

Currently, as the attack requires low level access to the radio component, this limits the number of devices that could easily be used to implement it. However, we managed to implement a subset of *WazaBee* attack primitives on an *Android* phone, as described in [Cayre 2021c], resulting in a considerable increase of the number of potentially exploitable devices. A *jamming* primitive could be implemented too,

expanding the number of possible attacks.

With that in mind, two main offensive scenarios seem particularly critical:

- **Pivoting attacks:** *WazaBee* allows an attacker to target a *802.15.4* device from a compromised *BLE* device. For example, an attacker could compromise an employee's *BLE* smart watch in the street. Then, they could use it to perform pivoting attacks targeting *802.15.4* devices in its professional environment.
- **Covert-channel attacks:** *WazaBee* could be used by an attacker after compromising a *BLE* device in order to exfiltrate sensitive data using unmonitored protocols.

3.4 Compatibility conditions

Wazabee is not the only possible attack that leverages similarities between modulations. To improve the coverage of the attack surface of a network using wireless protocols, it can then be interesting to study the reception and emission communication capabilities of the devices outside their designated range.

In this section, we present theoretical compatibilities between usual classes of modulations, and the impact on security that could ensue.

3.4.1 Frequency Shift Keying emitters

3.4.1.1 FSK to PSK

The Frequency Shift Keying is one of the most common modulations among IoT devices, especially its GFSK variant, used for example in Bluetooth, BLE, or ANT[1]. As shown in Wazabee, this type of modulation can be used to mimic an MSK, and its equivalent O-QPSK, given that its modulation index is close enough to 0.5. This is due to the fact that the GFSK uses two rotation directions, and that if the modulation index is close to 0.5, two consecutive samples are separated by a $\frac{\pi}{2}$ angle.

This could be generalised by analysing which frequencies would be needed to control the phase shifts between two samples, in order to reproduce other phase modulations. For example, to reproduce a QPSK, one needs to be able to do rotations of either $\frac{\pi}{2}$, π, $\frac{3*\pi}{2}$, or $2 * \pi$ between two samples to replicate all needed transitions. For example, and as shown in figure 3.4, it can be done with a 2FSK if 1) its symbol rate is four times the symbol rate of the QPSK, and 2) its modulation index is approximately equal to 0.25. Indeed, in this case, the 2FSK allows us to do four rotations of either $\frac{\pi}{4}$ or $\frac{-\pi}{4}$ when the QPSK would transmit a symbol. This way, it is possible to reproduce π with four rotations in the same direction, 0 with two times each direction, and $\pm\frac{\pi}{4}$ with three rotations in one direction and one in the other.

[1] ANT is another protocol of the IoT, often found in connected sportswear

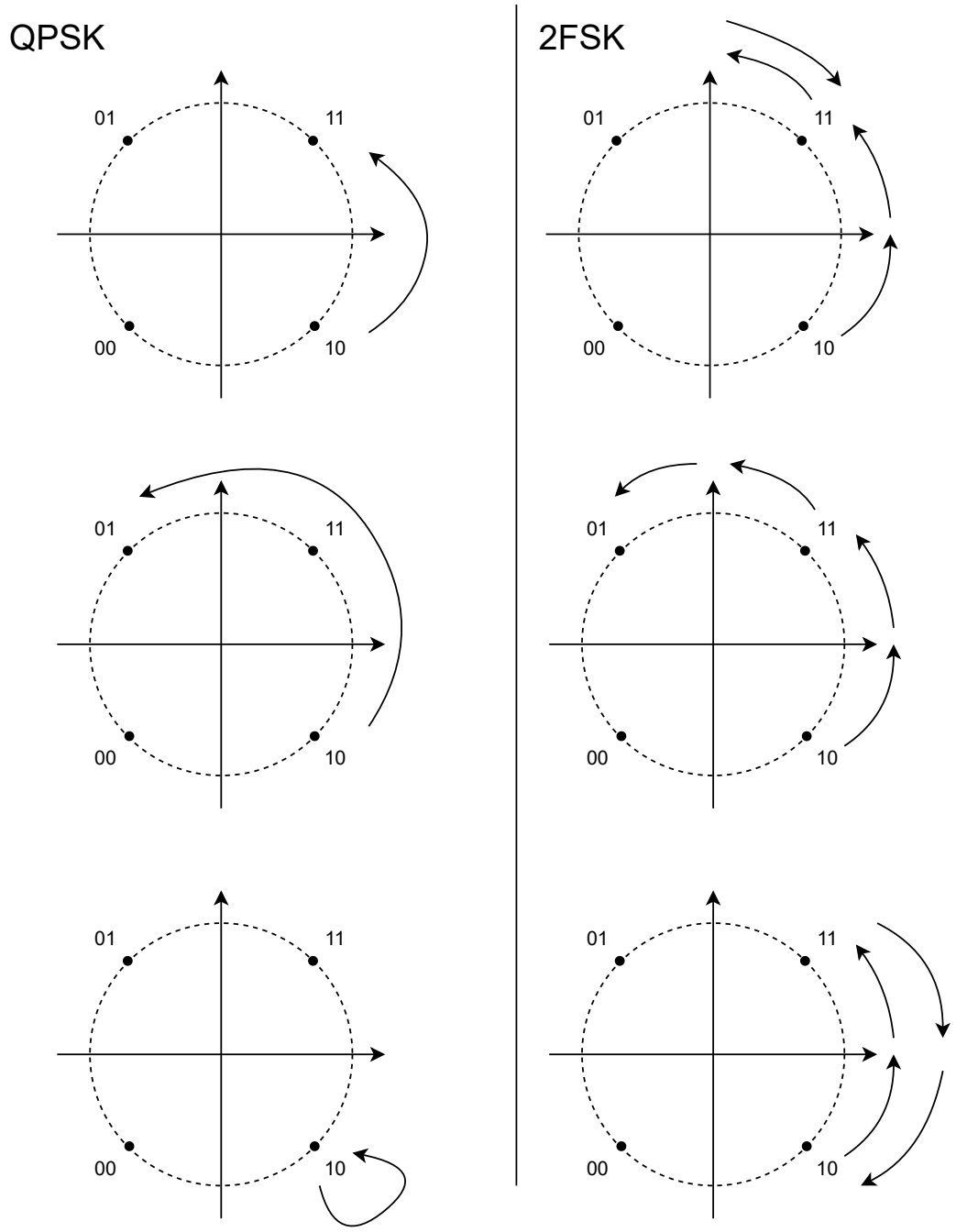


Figure 3.4: Transition equivalences between a QPSK and a 2FSK

In a more realistic setup, however, this attack suffers from several drawbacks, that hinder its applicability:

- The FSK needs to use fairly high sample rates, compared to the sample rate used by the QPSK. Using the technique described above, it must precisely be a multiple of four times the QPSK's sample rate

- The FSK also needs to use a modulation index of 0.25, or even lower when working with higher sample rates, which would allow for more symbols during the modulation of a single QPSK one
- A wrong synchronization between the periods of the QPSK and the groups of four periods for the 2FSK could be problematic. For example, imagining the symbols we want to send in QPSK would result in a rotation of π, a rotation of $-\frac{\pi}{2}$, and finally a rotation of $\frac{\pi}{2}$. We could code it with the sequence "0000-0001-1110" in 2FSK. However, should the QPSK receiver be desynchronized with our emitter by exactly half a QPSK symbol period, it would receive "...00-0000-0111-10...", meaning a rotation of π and directly afterwards a rotation of $\frac{\pi}{2}$. To reduce the impact of this phenomenon, it could be possible to use smaller steps than $\frac{\pi}{4}$, and do series of "01", i.e. u-turns, once the QPSK symbol we targeted is reached. However, this results in 1) a significantly higher sample rate, as we are using even more 2FSK symbols during a QPSK symbol, and 2) a significantly lower modulation index, making the two frequencies of the 2FSK even closer, risking to confuse them in presence of noise

3.4.1.2 FSK to ASK

It would also theoretically be possible to use a Frequency Shift Keying modulation to emit a signal for an Amplitude Shift Keying receiver, by overlapping one of the two frequencies of the FSK with the carrier frequency of the ASK receiver. Two issues are then to consider: the frequency variations introduced by the pulse-shaping filter, which could bring the frequency outside the range of a potential filter on the ASK receiver, and the interference with the other frequency of the FSK, if it is too close and enters the range of the ASK receiver. By definition, these two issues are incompatible with each other, and thus can be managed separately. However, both depend on the window in which the ASK receiver is listening.

The first one depends on the protocol in use, because it often determines the shape of the pulse-shaping filter. For example, BLE emitters always use a gaussian filter. However, the exact shape of the gaussian filter, especially its width, is often not fixed by the specifications. It is then necessary to study, for each specific emitter, whether the variations between two identical bits would go outside the listening range of the target receiver.

The second issue is closely related to the bandwidth of the emitter's protocol, and the modulation index it uses. The larger the bandwidth and the modulation index, the larger is the distance between the two frequencies, limiting the risks of interference from one on the reception of the other by an ASK receiver.

Overall, it could be extremely difficult to control precisely the amplitude that is received by the ASK receiver, especially if it uses an unknown threshold to differentiate its possible amplitudes. For example an ASK could theoretically count as a 0 any amplitude falling under 75% of the maximum amplitude. Let us note that this case is rare, and a 2ASK will often consist in an On-Off Keying, or OOK, meaning that one bit is coded by an emission, and the other by an absence of emission.

3.4.2 Amplitude Shift Keying emitters

Amplitude Shift Keying emitters use a single sine wave at the carrier frequency, modulating the transmission power. It is then difficult to use such modulation to obtain precise phase shifts, as the phase offset of the sine wave is neither known nor controllable, and the phase at a given moment is only given by this offset and the frequency.

However, it is possible to combine several ASK-OOK, meaning modulations where the binary input controls periods of emissions, to generate signals on several frequencies, only emitting on the one we want to use for the FSK. This presents two main difficulties: 1) the sampling rate for ASK emitters is often significantly lower than the one from FSK emitters, as high rates are not needed to demodulate correctly the signal, and 2) it is then necessary to control k ASK emitters to emit an kFSK. Moreover, unlike the FSK, the ASK is not widely used in devices that can be linked to a network or a computer, often being simple wireless devices with a remote control. This makes ASK emitters unreliable as pivots for attacks.

3.4.3 Phase Shift Keying emitters

As we explained previously in Wazabee, there exist a specific kind of phase modulation called the Gaussian Minimum Shift Keying that can be assimilated to a Gaussian Frequency Shift Keying. More generally, if the phase remains continuous, shifting from a phase to the same phase $\pm\frac{\pi}{2}$ allows to mimic a 2FSK with a modulation index of 0.5, which could also be extended to other modulation index by varying the phase shift.

This means that with a QPSK and an MSK, since both use four points around the trigonometric circle, it is possible to build an equivalent sequence of QPSK states to reproduce the MSK, and by extension a 2FSK if the phase remains continuous in the QPSK. Similarly, if the 2FSK would have used a modulation index of 0.25 for example, it would have been possible to reproduce it with an 8PSK, i.e. a PSK with 8 equidistant phases, hence separated by $\frac{\pi}{4}$ gaps.

3.4.4 Compatibility between OFDM and QAM

The modulation used with OFDM to build its symbols is often, if not always, a Quadrature Amplitude Modulation, or QAM. Indeed, it allows to combine phase and amplitude modulations, without modifying the frequency if the transitions are not continuous. This allows each channel to remain where it should be, without interfering with its neighbours by breaking their orthogonality.

Thus, it should be theoretically possible to emit an OFDM with multiple QAMs, or to emit QAMs with an OFDM. However, both methods are subject to heavy limitations.

First, to emit an OFDM with multiple QAMs, one would need a QAM per channel in OFDM. However, as explained in section 2.5, this number can easily rise up to thousands of channels. Moreover, OFDM also uses reference symbols, which will

need to be reproduced on each corresponding channel. The use of corrupted QAMs to attack an OFDM becomes then highly unrealistic, even without considering the issue of synchronizing the multiple QAMs between themselves, and the impact of their different positions on the received signal.

Then, to emit QAMs with an OFDM, one would need to ensure that channels around each QAM channel are set to 0, to prevent interferences. Indeed, the OFDM receiver is able to recover single symbols on individual frequencies by looking at the entire spectrum at the right sample rate, during the right period, and using the reference symbols to correct the distortions introduced by the channel on the different frequencies. Meanwhile, the QAM receiver will simply try to find QAM symbols on a single frequency, with a sample rate that could break the orthogonality between the channel in use and its neighbours. Thus, one should ensure the transmitting channels are far enough from each other in this case, thus needing a low-level control on the OFDM emitter to completely shut down several channels, even the reference symbols if necessary. Another issue to cover in this case is the fact that the OFDM will send a single QAM symbol during the entire duration of one of its blocks, putting further restrictions on the symbol rates of the QAMs to attack.

3.5 Conclusion

In this chapter, we presented several theoretical equivalences or compatibilities between modulations used in wireless communications, known as Cross Technology Communication in the literature. However, these mechanisms are often disregarded by or unknown to the security community. We then also showed the practical applicability of Cross Technology Communications from an offensive perspective with our new attack, Wazabee, between well-known protocols of the IoT.

The different insights we have given on the similarities between usual modulations, even if mostly impractical, show that other attacks of the same kind could be developed, should the conditions be met. Moreover, even if the corrupted transmitter cannot communicate with other devices to realize a pivotal attack, it could also be used as a covert channel, by sending seemingly noisy data with its modulation, when in reality it is exfiltrating sensitive data using another modulation.

From a more general perspective, we think this type of attacks could become critical due to their capabilities to pivot from a protocol to the other, possibly breaking what would be otherwise thought as an isolation between two networks with different security levels.

This shows the importance of taking such potential attacks into account when analysing the attack surface of a wireless environment, and either designing protection systems able to detect communications from different protocols compatible with those present in the environment, or protocol-agnostic systems.

In the next chapters, we present examples of methods to monitor or analyse IoT communications based on their physical emissions rather than higher layers of the protocols in use, first to identify the emitting devices, and then to analyse the

communications without prior knowledge on their content and used protocol.

Chapter 4

Wireless identification - PSD-based fingerprinting

Contents

In this chapter, we present a spectrum-based fingerprinting method for IoT devices, with the objective to be less expensive than existing techniques, and to reduce the amount of information on the communications it needs. First, after briefly outlining the context and motivations of this approach, we present other methods and related works about device identification in IoT networks in section 4.2. Then, we explain, generally in section 4.3 and more into detail in section 4.4, the principles and architecture of our initial approach, and the various experiments we ran to validate it in section 4.5. We then present in section 4.6 some improvements that were added to this first approach to make it more adapted to realistic conditions,

notably for real time execution. Finally, in section 4.8, we first discuss the experiments we have carried out to apply this method to detect rogue base stations in mobile telephony, as well as the various challenges posed by OFDM to spectrum-based approaches such as ours in section 4.7, and then we address more broadly the limitations and possible further improvements for future work .

4.1 Context

From a security perspective, wireless communications bring several new threats specific to the nature of the transmission medium. Indeed, the increased link's availability to anyone equipped with a compatible transceiver greatly simplifies eavesdropping, overshadowing and injection attacks.

However, along with this increase in attack surface, this new industry suffers from a lack of maturity compared to the more traditional software industry and wired networks, resulting in poorly protected protocols and a low use of security measures, such as encryption or signatures.

This leads to the re-emergence of attacks that are known for a long time in other computer science fields, such as spoofing attacks. Those rely on identity theft, by injecting illegitimate communications while impersonating a legitimate device. Unfortunately, such attacks are easy to perform in most short-distance wireless networks, often only needing a change of MAC address on a standard transceiver for the protocol to use, without being detected as an anomaly by the actors of a communication.

As such, specific monitoring systems are needed to address these attacks, and especially means of uniquely identifying devices. More specifically, one of the most relevant defensive strategies to detect this type of active attacks is fingerprinting. Indeed, this technique makes it possible to identify legitimate devices or communications based on their characteristics, be it from the behaviour of a suspicious node or identifying features in the emissions on the lower layers.

4.2 Related works

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 [Hall 2005, Ur Rehman 2012, Köse 2019]. Another relevant work by Boris Danev's [Danev 2011, Danev 2012] 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 [Brik 2008] 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 [Chatterjee 2019], 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 [Ramsey 2015], 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 in [Nouichi 2019], or in [Hasse 2013]. 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 [Laperdrix 2019, Eckersley 2010, Boda 2011].

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.

4.2.1 Choice of the PSD-based fingerprinting

In this chapter, we present a novel method for fingerprinting IoT devices from their Physical Layer communications that aims to address these limitations. Unlike the existing solutions from the same category, 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.

Overall, this approach presents several benefits:

- *Low cost:* it doesn't require high quality receivers as it focuses on entire signals
- *Generic:* it supports the main modulation schemes traditionally used in smart environments, with a minimum of information about the protocols in use

4.3 Approach overview

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

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, ... A possible generalization of our approach to IoT environments including mobile objects, on which we started working, is discussed in Section 4.6. 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.

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 4.1 and 4.2. 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.

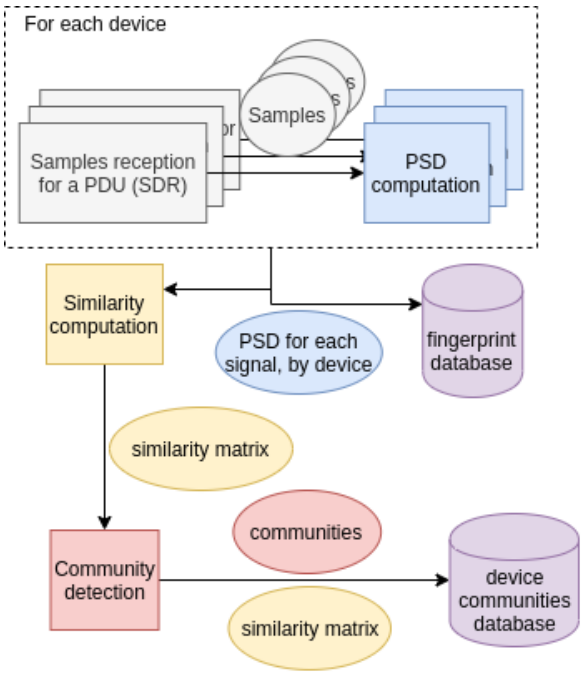


Figure 4.1: Fingerprint creation and cluster computation

4.4 Detailed description of the approach

In this section, we detail the different steps of our approach and their integration. Subsection 4.4.1 explains the motivations for choosing the PSD for the creation of fingerprints, and how these fingerprints are computed. Subsection 4.4.2 describes the algorithms used to measure the similarity between PSDs and the methodology to compute PSD clusters. Finally, subsection 4.4.3 describes how the intrusion detection is performed for each incoming signal, based on these clusters and the PSDs stored in the database.

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

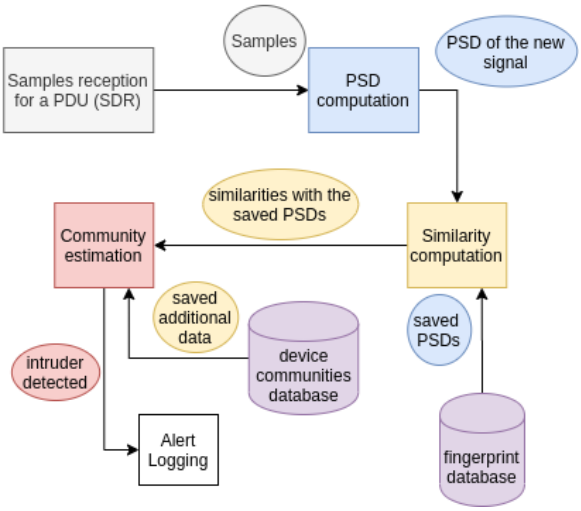


Figure 4.2: Intrusion detection

4.4.1 Fingerprint creation

4.4.1.1 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 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 [hac 2022] and the LimeSDR Mini [lim 2022]. 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. They 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.

4.4.1.2 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 "$\cdot$" 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 4.3, 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.

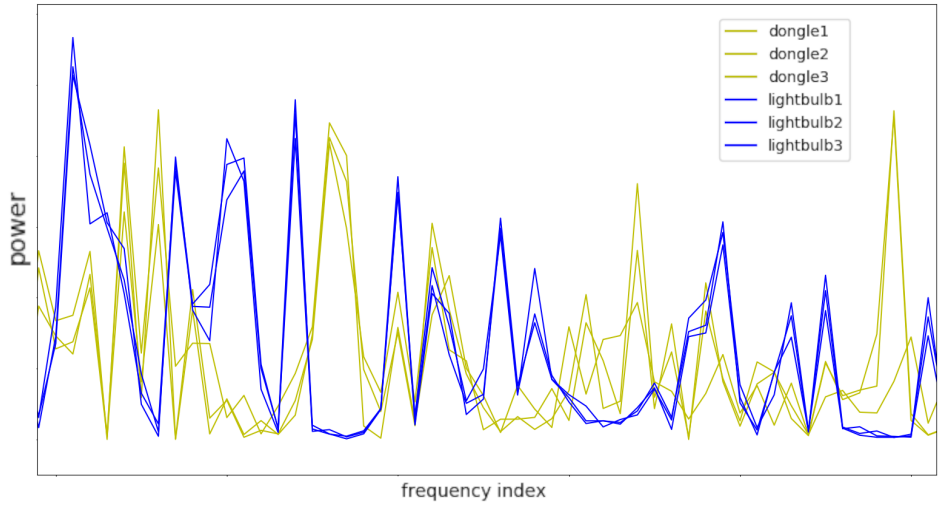


Figure 4.3: PSD for two distinct BLE devices (3 PDUs each)

[2]We discuss the experimental efficiency of our approach in Section 4.5.

4.4.2 PSDs similarity analysis and clustering

4.4.2.1 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 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:

$$D(PSD1, PSD2) = \max((PSD1(f) - PSD2(f))^2) \tag{4.1}$$

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:

$$S(PSD1, PSD2) = [1 - D(PSD1, PSD2)]^{amp} \tag{4.2}$$

Parameter *amp* is evaluated empirically for each protocol.

4.4.2.2 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 [Pons 2006], 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 [Kaufmann 1987] 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 [Girvan 2002] or the fast-greedy [Clauset 2005] (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 4.5), 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.

4.4.2.3 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:

$$\overline{S}(PSD, cluster) = \frac{1}{|cluster|} \sum_{PSD_i \in cluster} S(PSD, PSD_i) \tag{4.3}$$

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:

$$\overline{\overline{S}}(cluster) = \frac{1}{|cluster|} \sum_{PSD_i \in cluster} \overline{S}(PSD_i, cluster) \tag{4.4}$$

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:

$$threshold = \overline{\overline{S}} - 3 * \sigma_{\overline{S}} \tag{4.5}$$

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 4.4.3.

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.

4.4.3 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 4.4.1.1.
- The PSD of the signal, noted P_s is compared to the other ones (by computing the similarity measure defined by equation 4.2).
- The average similarity for each PSD in each cluster is computed (except cluster -1) as defined in equation 4.3.
- This average similarity $\overline{S}(P_s, cluster)$ is compared to the average $\overline{\overline{S}}$ (as defined in equation 4.4) 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 4.5. 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.

4.5 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 4.5.1. Section 4.5.2 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 4.5.3 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 4.5.4, 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 4.1. 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 [dat 2020].

Table 4.1: Parameters of the different experiments

experiment	number of devices	sets per device	set size fingerprint	set size testing	number of signals per fingerprint	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

4.5.1 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 and the Lime SDR Mini. The HackRF One is cheap and easy to set up, therefore good for prototyping, while the Lime SDR Mini is a little more expensive but offers better capture precision and stability. Our implementation uses Python3 and the igraph[igr 2022] 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, 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.

4.5.2 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 4.1), 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 4.4, 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.

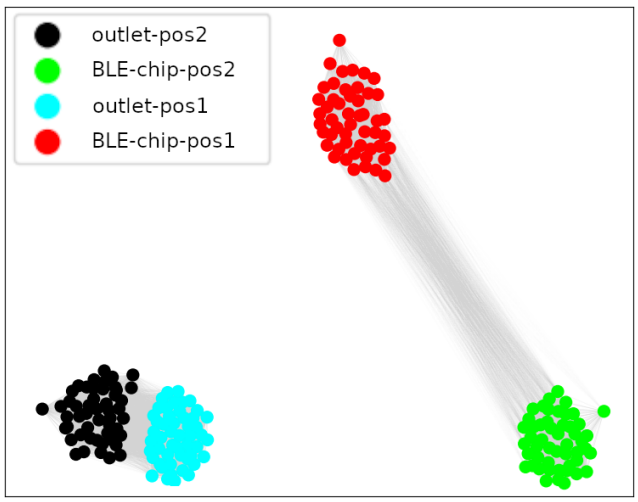


Figure 4.4: First experiment results visualisation - Setup B-1

In the second setup (B-2 in 4.1), 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 4.5, again showing a clear separation between the different devices.

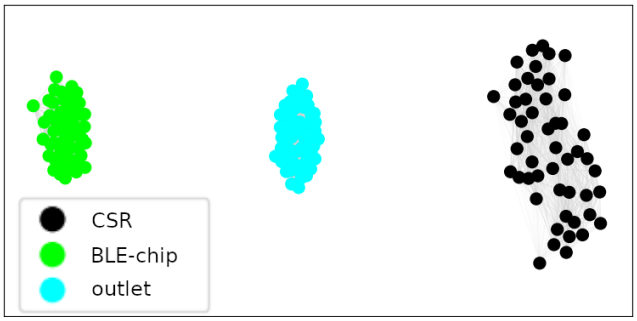


Figure 4.5: First experiment results visualisation - Setup B-2

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).

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 4.2. 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 4.2: 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.

4.5.3 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 4.6. 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 4.3.

Table 4.3: Different devices, BLE - results

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

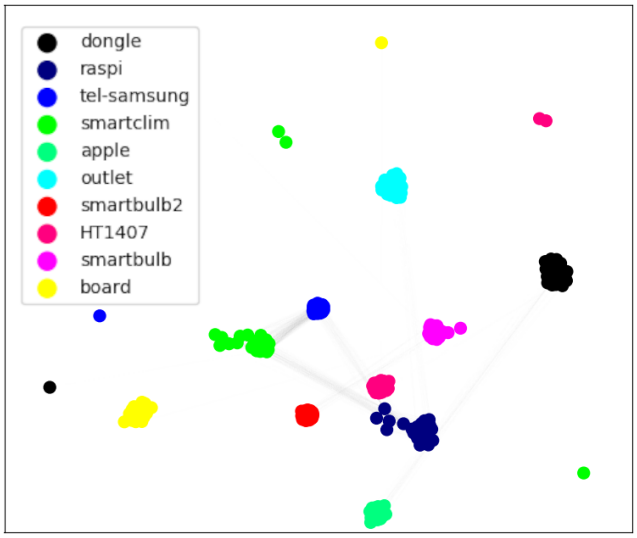


Figure 4.6: Visualisation of the data from the different BLE devices

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. 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 stays high, with a recall rate over 98%.

4.5.4 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 4.4, and the visualisation of the PSDs similarities for Zigbee devices is presented in Figure 4.7. Moreover, we can conclude that the LimeSDR gives indeed better results than the HackRF in our Zigbee experiment, while they remain comparable for BLE.

Even though in both cases, our visualisation highlighted a potential collision between two emitters (as shown in Figure 4.7 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

Table 4.4: 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%

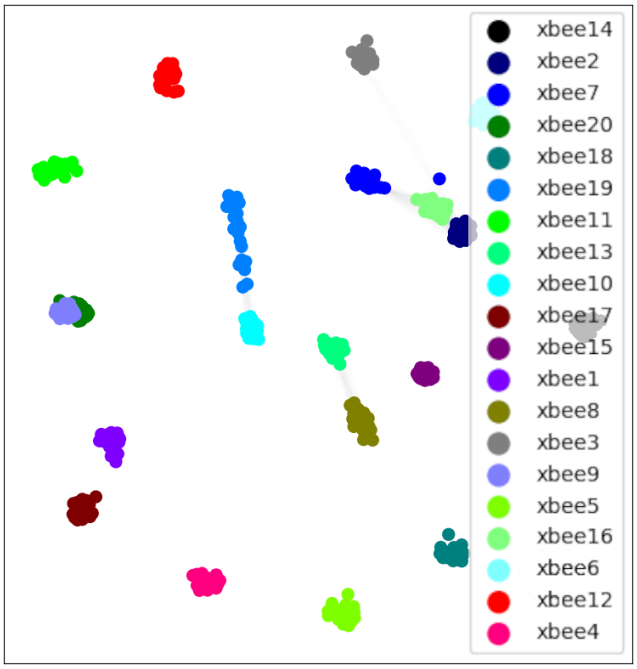


Figure 4.7: Visualisation of the data from identical ZigBee devices

know or guess precisely the 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.

4.5.5 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 4.8 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.

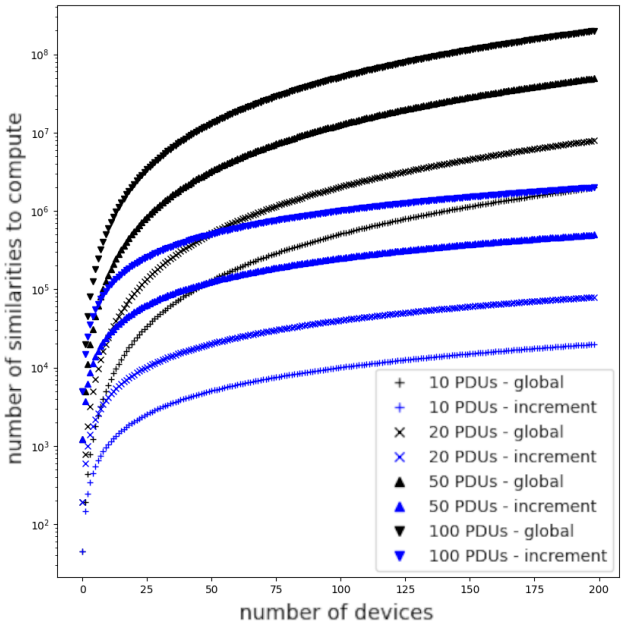


Figure 4.8: Number of similarity computations estimation

The selection of the optimal number of PDUs should be based on the detection efficiency of the proposed approach. Table 4.5 presents the results of our intrusion detection algorithm in the case of the experiment with BLE devices presented in Section 4.5.4, considering different numbers of PDUs per device, varying from 10, 20, 50 and 100, respectively.

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

Additionally, Table 4.6 presents the average times, over 100 samples, to compute the fingerprints measured in our different experiments with different numbers of

Table 4.5: 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%

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 4.6: 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 4.6, 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.

4.6 Extension to dynamic environments

The approach as previously presented suffers from some drawbacks, we then decided to work on improvements to tackle them. This approach is still a work in progress, and no publications were currently made on them.

The main additions we made are:

- **dynamic updates**, or on-line updates, i.e. updates of the fingerprint database while the detection intrusion is running by replacing the oldest fingerprints by new ones if they are close enough from the corresponding cluster
- **full real-time re-implementation** in C to make full use of this new feature, and to allow for real-life intrusion-detection experiments, and finally
- **metrics and detection measures improvements** from the signal processing point of view, to improve our detection accuracy

4.6.1 Real-time and accuracy improvements

In this section, we expose the different improvements made to our initial approach, in order to address its main issues, namely its limitation to static environments because of fingerprint shifts and its desynchronization over time for the same reason, and to improve the detection accuracy.

4.6.1.1 Dynamic updates

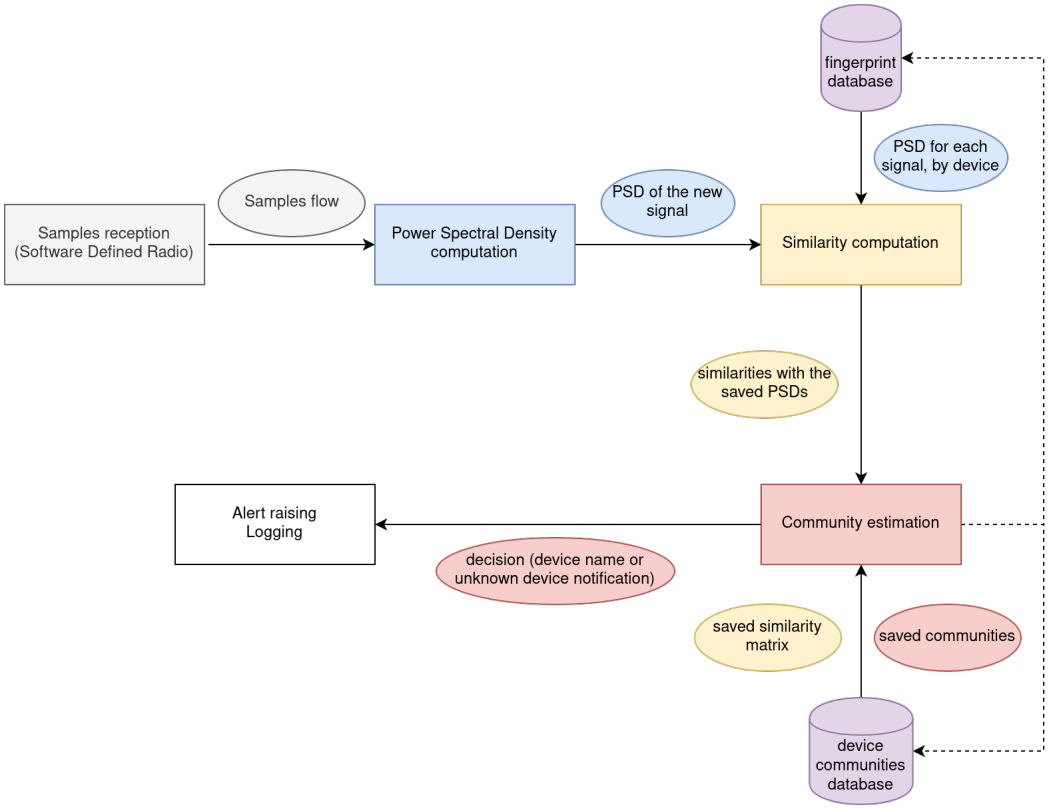


Figure 4.9: Loopback for dynamic fingerprint updates

During our experiments, we realized the legitimate signals were progressively shifting away from the initial fingerprints. This can be due to different factors in the environment and the devices themselves, such as variations in temperature, a drift in the frequency due to a clock desynchronization, or even addition and removal of devices. This issue could, after some time, result in a gap between the initially computed fingerprint and the device's emissions sufficient to detect it as an intruder.

As this shift occurs progressively, it can be compensated by following the modifications in the emission profiles during real-time reception. Thus, we modified our approach so that the fingerprint database would be updated during our intrusion detection phase, registering the variations in the device's emissions.

We do so by evaluating our confidence in negatives, i.e., frames not detected as intruders, to check if we are almost sure that the frame is indeed from the corresponding legitimate emitter. If so, we then remove the oldest PSD registered in the corresponding cluster, and replace it with the new one, while updating the similarity matrix accordingly. This mechanism as we implemented it is illustrated in figure 4.9, updating the two databases of our approach during the intrusion detection phase depending on the result of community estimation.

A possible issue with this mechanism would be an attacker trying to artificially shift the fingerprint of a device in a direction of their choosing, by replaying frames close enough from the current cluster repeatedly until they control entirely the content of the cluster, for example to make an illegitimate emitter considered as legitimate. However, to do so, the attacker would need two prerequisites: 1) a prior knowledge of the content of the cluster at the moment they want to attack, and 2) the ability to create successive signals that are considered close enough from the original cluster by the system, and that shift progressively in the direction they want. Please note that filling the second condition would also mean that the attacker would have already been able to bypass the system by creating signals falling into the legitimate cluster.

While the first condition could realistically be met if the attacker uses a receiver similar to the one used by the Intrusion Detection System (IDS) to create similar fingerprints from their end, the second is significantly harder to meet. Indeed, to perform this attack, the attacker needs to first generate an array of signals progressively shifting to the position they want from the current position of the cluster, and then emit the elements of the array one after the other in order to compensate for the possible legitimate packets that are still sent, or emit less while blocking the transmissions from the legitimate object. While it seems easy to perform in a simulated environment, a simple interpolation between the starting PSDs and the target ones allowing to create the array, it is in fact highly challenging to perform in real conditions, first because of the possible mistakes made on the starting point estimation, but also and above all because of the interferences the emitting hardware adds to the signal. The attacker would then need to estimate the correction to apply to the central frequency and the modifications on the spectrum of the frames sent to compensate for this effect, additionally to the initial array generation. To our knowledge, such attack would then require both high-end equipment and a high level of expertise from the attacker.

4.6.1.2 Real-time implementation

Another limitation to our previous works was the fact that our Proof-of-Concept implementation was not able to perform in real-time, the processing speed being far too low compared to the frame reception rate. This was mainly due to the use of Python, which we chose for its ease-of-use, but introduces a significant overhead to the computations. We then re-implemented the initial Proof-of-Concept, with the addition of the other improvements presented in this Section, in C. This allowed us

to reduce the computing overhead enough to allow for a real-time use of the IDS, and to use the dynamic fingerprint updates previously presented.

As we present further in section 4.6.2, the tool now follows a more modular architecture, separating each unique function in a single process, each process communicating with the others with specific Unix FIFOs depending on their function.

4.6.1.3 Detection improvements

This new implementation of our approach includes multiple improvements, to better receive the packets, to improve the detection accuracy, and to better separate the different parts that make a fingerprint unique.

In an ideal world, the emitter and the receiver have their oscillators perfectly synchronized, and emit or receive exactly on the right frequency for the channel in use. However, those devices always exhibit inaccuracies in their oscillator frequencies. This results in a frequency offset in the received signal on our receiver's end. Interestingly, a given emitter presents a specific shift profile compared to our receiver, characterized by its average and variance.

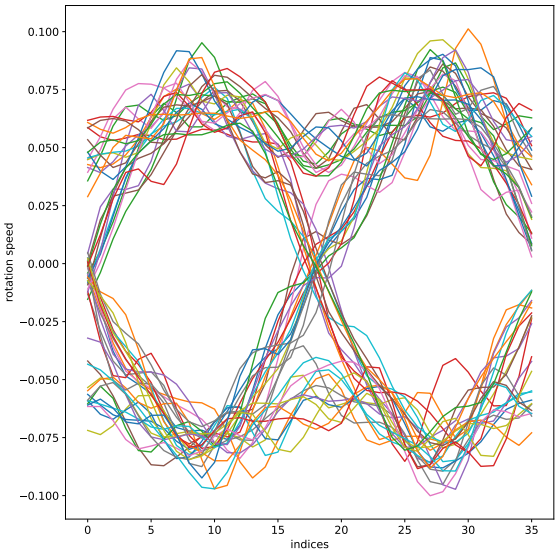
Our first approach didn't make full use of this error, even if it was appearing in the resulting PSDs, the frequency shift resulting in a shift of the PSD. However, the variability of this offset could also make the same object's fingerprint quite inaccurate, taking into account multiple shifted versions of similar PSDs. We then decided to detect this frequency offset, by identifying the frequency offset compared to our reception frequency, in order to 1) correct it before comparing the different PSDs, and 2) take it into account in our similarity measure.

The identification of the central frequency is done by computing the signal's Fourier Transform (FT), and then getting the average of the frequencies, weighted by the corresponding FT amplitudes, as follows:

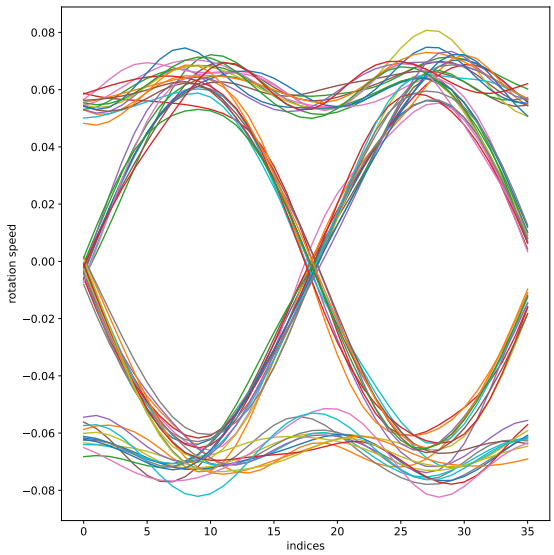
$$\hat{f}_c = \frac{\sum_f f * S(f)}{\sum_f S(f)} \tag{4.6}$$

With $\hat{f}_c$ the central frequency estimate and S the FT of the starting signal.

This allowed us to correct the frequency of the signal prior to computing its PSD, in order to minimize its impact on the shape comparison of our algorithm. However, even if it was introducing inaccuracies in this measure, the frequency error remains an identifying feature for the emitters, we then needed to take it into account in our similarity measure. To do so, we also registered, along with the device's fingerprint, the different corresponding frequency estimations, in order to check, while receiving, whether the frequencies estimated for the received object correspond to the ones that were registered. More precisely, we maintain a cyclic buffer of frequency estimations for a given recognized cluster during the intrusion detection phase, and once the buffer contains as many frequencies as the registered one corresponding to the cluster, we compare the two distributions by using their



(a) Wrong filter



(b) Matching filter

Figure 4.10: Examples of cross-correlations with different filters

cross-correlation. This gives an additional feature that could give away an attacker that passed the PSD similarity test, but with a different frequency error repartition. The choice of the cross-correlation to measure this similarity was mainly due to the possible presence of frequency hopping, that did not allow for a simple comparison of the actual values of the frequency errors.

We then also decided to change entirely the similarity computation with a new version, also replacing the initial distance computation by a correlation computation, to minimize the impact of a possible remaining shift in the frequency domain. The new similarity measure is then defined as the maximum of the cross-correlation between the two PSDs $S = \max(R_{PSD1,PSD2})$, with $R_{PSD1,PSD2}$ the cross-correlation between $PSD1$ and $PSD2$.

Finally, the demodulator of our initial approach was simply "demodulating" the stream of IQ samples coming from the receiver trying to find valid data, using a basic majority vote algorithm. For example, if the sample rate is three times the symbol rate of a binary protocol we want to demodulate, we take the bits three by three, and then check which bits are in majority in each group of three. However, this algorithm loses a high number of packets, possibly allowing an attacker to send attacks spread in time and possibly not be detected.

We then decided to improve the algorithm, by implementing a filter bench containing various common pulse-shaping filters with different parameters. The cross-correlation with each filter gives, for each index in the signal, a value whose amplitude is higher if the signal contained a shape near the one of the filter at this index. Moreover, the amplitude is also lowered if the signal had not the right shape. One interesting thing is that, the cross-correlation being linear, if we choose a pulse-shaping filter matching the one used for the emission, we get amplitude peaks where the symbols are from the original signal, and reduce the impact of the additive noise on the resulting signal since it doesn't follow the shape of the filter. We then represent the different cross-correlations as eye-diagrams, that we presented in Chapter 2, to analyse the resulting signal-to-noise ratio. Examples of eye-diagrams for a wrong filter and matching filter on a noisy signal can be seen in figures 4.10a and 4.10b. The signal-to-noise ratio can be determined by computing the average and variance of the eye's amplitude at each index, a high average and low variance meaning that the signal-to-noise ratio is high, and vice-versa. Since a high noise can produce a high average amplitude, this average can't be used alone. We finally choose the filter that gave, for one of the indices in the corresponding eye-diagram, the maximal signal-to-noise ratio, and take the index at which this maximum was reached as the moment of decision for demodulating. This method, used with common pulse-shaping filters, allows for a better demodulation of most protocols, without prior knowledge on the filter's specification.

4.6.2 Tool architecture

Finally, we worked on a cleaner and more maintainable and customizable architecture for our tool.

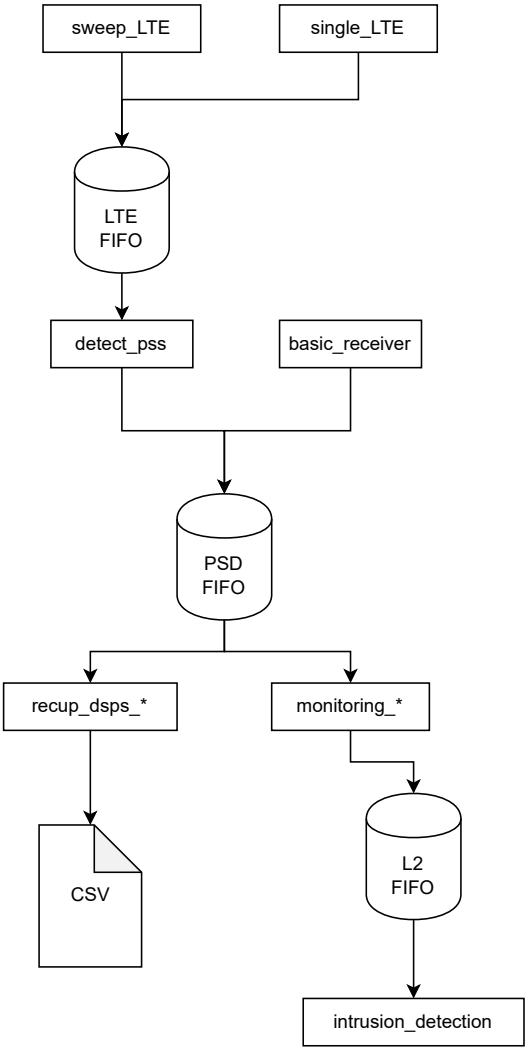


Figure 4.11: Modular tool architecture

One of the main issues that prevented a real-time use of our algorithm was the initial implementation in Python, with eased greatly the development process, but introduced a significant overhead due to the fact that Python is an interpreted language. We then decided to do a new implementation in C/C++, allowing us to drastically reduce this overhead and optimize from a very low-level perspective the signal processing primitives needed for our algorithm.

This new implementation, illustrated in figure 4.11, works in different levels, communicating by the means of UNIX FIFOs, the three main levels being:

- **Packet detection and isolation:** the modules on this level, *detect_pss* and *basic_receiver* in the current state of the implementation, detect the presence of packets in a stream. They then isolate the corresponding IQ samples, compute their PSD, and store both the temporal signal and spectrum in a FIFO called *PSD FIFO*
- **Detection of data of interest:** the modules on this level take the data sent by the previous one, and analyse it depending on the protocol to monitor. They are split in two categories, one saving the signals and PSDs in CSV files to compute the fingerprint databases afterwards, the other forwarding them in another FIFO called *L2 FIFO* for real-time intrusion detection
- **Intrusion detection:** this level takes the data forwarded by the previous level in *L2 FIFO* to compare it with an existing fingerprint database, in order to detect if the signal that was received corresponds to a known emitter

Additionally, the remaining level in 4.11, with *sweep_LTE* and *single_LTE*, corresponds to specific processing steps made for fingerprinting 4G base stations, that use the LTE technology. We discuss in more detail the experiments that were run with 4G base stations in the next section.

4.7 Application to mobile telephony

As explained in 2.5, the recent mobile telephony technologies use a specific modulation scheme, the Orthogonal Frequency Division Multiplexing. Due to its nature, this scheme mixes the useful signal that carries data with the entirety of the spectrum, making it difficult to extract the impact of imperfections from data-carrying samples without removing the data from it, hence the need for specific preprocessing.

In our case, we made use of synchronization signals that are sent periodically by 4G base stations, called the *Primary Synchronization Signals*, or PSS. This signal is used by 4G users to synchronize with specific time slots in the 4G emissions, and have 3 possible shapes depending on the base station's ID. To detect them, we correlated the flow received from a base station with each of the 3 shapes, and tried to detect a peak in the correlation's amplitude. The presence of regular peaks higher for one of the three shapes than for the others means than this specific PSS

was used, and allows us to isolate the sections of the signal containing the PSS, and removing it by hand to reduce its impact. Those isolated sections were then used as packets after being sent to the PSD FIFO.

We then ran a campaign of captures from various base stations in the surroundings of our laboratory, to try to re-run the previous experiments on them. The objective was to build a system that would be able to identify legitimate base stations, and embark it in a portable equipment to detect rogue base stations.

However, this resulted initially in our approach only separating the base stations in three different clusters, seemingly depending only on the PSS they used. Running manually a second time our algorithm on one of these clusters alone allowed for better results, but still with collisions between base stations, not allowing to fully identify them.

Our hypothesis relatively to this issue is that our measures were too impacted by a well-known phenomenon called *channel-selectivity*. This phenomenon translates the different impact a transmission medium has on different frequencies, resulting in a deformation of the spectrum, non-trivial to revert. The presence of those deformations could suffice to modify the PSS enough for it to still be present in the signal after having removed the theoretical one, but just not enough for us not to detect it.

This issue is generally corrected by the introduction of reference symbols at specific time and frequency slots in the OFDM flow, to which the receiver compares what was received to estimate the overall deformation and revert it. At the time of writing, the introduction of the correction to our algorithm depending on the 4G reference symbols is still a work in progress.

4.8 Conclusion

In this chapter, we presented a novel, low-cost approach to uniquely identify wireless emitters in a context of IoT networks, with minimal assumptions on the protocols in use to maximize its coverage on the heterogeneous ecosystem of the IoT. This method, using full packets as input to reduce the sample rate needed to get a significant amount of data, has still shown excellent results in different experiments, being able to differentiate emitters of the same model from the same manufacturer.

Our initial 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, it still had issues with progressive desynchronization between the receiver and the various emitters. Thus, we developed the improved version with dynamic fingerprints, which is currently still under testing. This version, re-developed in C, also aims to work in real-time, and introduce modularity to facilitate the modification or addition of individual components for new protocols. The Proof-of-Concept implementation is currently near completion, but requires validation experiments to compare it with the previous version, and ensure it still works well in dynamic

environments.

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.

Chapter 5

Wireless protocol audit automatization

Contents

In this chapter, we present an approach for modulation and protocol identification and analysis, along with a tool implementing it and the experiments we ran using it to assess its performances.

We first discuss the general state-of-the-art in the fields of modulation and upper-layer protocol identification and analysis in section 5.2, and how it motivates the creation of this new approach. We then explain the theoretical principles behind each step of our approach, from the modulation analysis to the protocol format inference, in section 5.3. Subsequently, we present a Proof-of-Concept tool we built to implement this approach in section 5.4. Finally, we expose the results of the different validation experiments we ran with this implementation in section 5.5, and discuss them and the possible improvements for both our approach and tool in 5.6.

5.1 Context and objectives

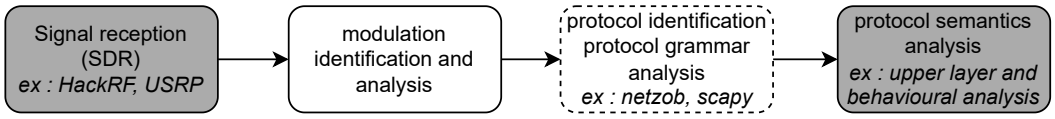


Figure 5.1: The different steps of a full wireless protocol analysis

Industrial and home networks are increasingly relying on wireless networks, either to ease communication, to save space or reduce costs. However, such networks are, by design, more exposed to security attacks than traditional wired networks, due to the use of unconfined and open radio transmissions.

In such systems, attackers could easily eavesdrop on communications or even create their own to inject malicious transmissions among legitimate ones. In wired networks, such attacks can be mitigated by using robust communication systems and by monitoring transmissions to detect potential anomalies. However, auditing and monitoring wireless networks is hindered by the large number and heterogeneity of protocols, which can have radically different Physical Layers and use specific channels to communicate, sometimes with the addition of channel hopping. Moreover, an attacker willing to stealthily exfiltrate data could communicate through a custom protocol escaping the scrutiny of potential monitoring measures.

Because of this heterogeneity, auditing unknown wireless protocols requires tedious work for recognizing the modulation used, as well as its parameters, in order to be able to recover the binary streams from the transmissions and perform the analysis of the protocol.

This entire process is composed of four main steps, represented in figure 5.1, that consist in:

- **Signal reception:** capturing the raw signal corresponding to the protocol to analyse; this can be done with Software Defined Radios (SDR) and corresponding software
- **Modulation identification and analysis:** identifying which modulation the signal uses, and its different parameters in order to demodulate it
- **Protocol identification and syntax analysis:** identifying the Link Layer protocol in use from the format of the demodulated binary stream
- **Protocol semantics analysis:** identifying the semantics of the protocol, namely the meaning of its different packets in a complete communication and their use for upper layers

The first step is already covered by the hardware available, and the fourth step depends more on what is the objective of the analysis. The third step is covered by the literature, but only partially, as we will precise further in the following section. Finally, the second step is poorly covered, especially regarding generic approaches.

In this chapter, we focus on the second and third steps, proposing a tool that aims to address the lack of a simple approach to automatically analyze wireless protocols, by designing:

- a modulation identification and analysis module, suitable for different types of modulations and easily extensible
- a dictionary of usual protocols and their modulations, for easier recognition of known communication systems
- a link-layer binary analysis module, to extract useful fields from unknown binary streams

More precisely, the tool we propose is able to detect transmissions and infer their parameters and content, without making any assumption, by combining a wideband receiver and different modulation detection algorithms able to estimate the parameters used in the transmission. This tool can then be enriched with grammar estimation modules to become an assistant for wireless protocol reverse-engineering, or a covert channel attack detection system to identify exfiltrated data.

5.2 Related works

5.2.1 Modulation detection and analysis

Automated detection of transmissions has always been a major concern in security because of its importance in the design of monitoring systems. Existing solutions take full advantage of partial to complete knowledge of a set of protocols they want to monitor in order to ease their detection and collect additional metadata about the transmissions. They range from approaches that focus on a single protocol, and whose goal is rather to detect attacks on upper layers, such as [Helluy-Lafont 2021], to broader approaches that study different possibilities and perform a case disjunction based on protocol-dependent heuristics to identify the type of the emissions, such as [Li 2015] or [Lakshminarayanan 2009]. These approaches are optimized for the analysis of specific types of emissions, and are therefore the best choice for an upper layer monitoring system. However, they do not allow detecting unknown protocols, especially outside pre-defined channels, making them vulnerable to covert channel attacks based on knowledge of the monitored channels.

Other works focus on specification agnostic anomaly detection, such as [Gimenez 2021] which performs an analysis based on an auto-encoder trained in an attack-free environment to detect unusual activities in the monitored frequencies. However, to our knowledge, these approaches suffer from the constraint to make a trade-off between the accuracy of the detection and the amount of data to be processed, which does not allow performing a deeper analysis of the anomaly beyond the identification of its occurrence time and associated frequencies. The method and tool presented in this chapter aim at addressing the limitations of both

of these types of approaches in the context of a more generalized covert channel detection.

Our approach is composed of several main steps in order to retrieve the information of interest from the signal: 1) detect when and on which frequency frames are transmitted, 2) identify the modulation type, 3) retrieve information about the modulation's parameters and finally 4) analyse the binary demodulated stream to extract information based on possible protocols for the studied modulation. The first step (detection of frame transmission) is well-documented in the state of the art, and one of the most common methods, which we also use, is based on an amplitude threshold. Regarding the second step (identification of the modulation type), several modulation detection techniques exist in the state of the art. Some approaches such as [Chen 2008] or [Mototolea 2020] focus on specific modulations and their unique characteristics to detect them. The main problem with these approaches is that other modulation types are ignored, leading to false positives. Also, as with other protocol detection methods, their focus on specific modulations makes them unusable in a multiprotocol monitoring context, unless a number of them are combined. Some other works are based on the comparison of the received signals with models of the different modulations. For example in [Azim 2013] and [Hao 2019], probabilistic models of the modulations are built, and then compared to signals, using goodness-of-fit tests such as Kolmogorov-Smirnov. The main limitation of this type of approach is the potential difficulty of building a model that fits a specific modulation. In this chapter, we describe an approach based on the detection of symbols in the signals by computing the autocorrelation of different representations fitting different modulation types, allowing the addition of extensions for new modulation types if needed. This method is simpler than building models for goodness-of-fit tests, but it strongly depends on the hypothesis that a stable periodicity can be highlighted by the autocorrelation of the signal with the correct modulation representation.

Our approach applies to a wide range of classic digital modulations, such as standard Frequency Shift Keying, Phase Shift Keying or Amplitude Shift Keying.

Compared to the state-of-the art, our detection approach makes minimal assumptions about the signal, relying on state-of-the-art frame and channel detection and a dictionary containing various modulation types and protocols to identify the Physical Layer used and extract the raw bitstreams. Furthermore, its genericity allows for the easy addition of new modulations and protocols, and the extracted metadata can be used as a set of features for auditing unknown signals or detecting covert channel emissions.

5.2.2 Protocol grammar inference

As explained by J. Duchêne in [Duchêne 2018], to audit properly a protocol, or to understand what is the purpose of a suspect emission, only getting the modulation information isn't enough: even if it allows a monitoring system to detect abnormal emissions, it won't give enough information to an expert to understand a potential

attack, or to audit properly a protocol under study.

It is then needed to go further into the analysis, and to try understanding the meaning of the packet content. From the modulation, we obtain the binary stream corresponding to a packet, which means to get a full understanding of the protocol we then need to study:

- the packet format, meaning its fields, their positions and possible values
- the syntax and semantics of the protocol itself, of a communication containing several packets

However, each of these steps is difficult to perform in its own ways. The first one requires the ability to distinguish the borders of the different fields, and identify the type of values they took over a sample of packets. A large majority of those are based on PI Project[Beddoe 2004], an approach from 2004 based on the use of bioinformatics algorithms, Needleman Wunsch[Needleman 1970] and UPGMA[Sokal 1958, Nei 1983], to detect similar portions of messages, estimated as corresponding to similar fields. As we also made use of them, we will further describe those two algorithms in 5.3.2.

Some of those works use specific field delimiters, known in advance, to segment properly the fields without needing to estimate their positions and alignment with Needleman Wunsch, such as Discoverer[Cui 2007], which separates the fields with those delimiters before running specific heuristics to detect field interdependencies, or ReverX[Antunes 2011], which uses the different values it found for the fields to generate a finite automaton describing a language covering the packets it saw, taking into account the possibility of loops in the automaton (to extend the possibilities of the language). Our objective is to extract a maximum of information from the binary streams got from the modulation identification part, without any prior hypothesis on the protocol's structure. Thus, we cannot integrate those approaches into our work, since they would need to identify potential delimiters, which could even not exist in the protocol currently analyzed.

We can also cite Roleplayer[Cui 2006], a tool to replay communications it registered, but in another environment, thus needing to adapt some fields in the packets, such as the addresses. As a consequence, it needs to estimate with a high level of precision where such fields are, and which values it should give them depending on the environment. To do so, it first identifies the dynamic fields in the received transmissions, then the different roles, such as addresses or arguments, and finally which values those fields should have in the new environment. However, this approach makes several hypotheses on the protocol, and requires the operator to provide additional information on the communications, such as the sending addresses in the registered packets. It is therefore also incompatible with our hypotheses.

Finally, in [Bossert 2012b, Bossert 2014], Bossert et al. describe Netzob, currently one of the most advanced tools in this domain. It uses a variant of Mealy state machines to model the protocol language, including information on the time between messages, and the L* algorithm[Angluin 1987] to estimate the protocol's grammar.

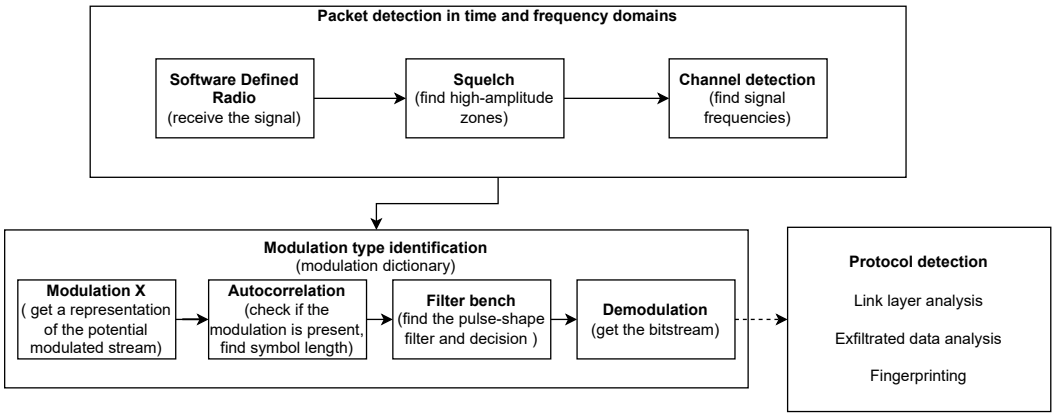


Figure 5.2: Architecture of the Physical Layer identification

The authors released the tool as an open-source project on GitHub[Bossert 2012a]. However, the tool's complexity and the lack of recent updates pushed us towards a simpler and lighter approach.

5.3 Theoretical components

Our approach is split into three parts: analysing the Physical Layer, identifying known protocols, and finally finding interesting fields in the binary stream if the protocol is unknown.

5.3.1 Physical Layer Identification

The first step of our approach is based on a Physical Layer identification module, which detects and analyses signals on the medium. This module runs on signals received from a Software Defined Radio, or SDR, listening on a wide bandwidth (and hence with a high sampling rate) in a specified band. The global architecture of this Physical Layer identification is presented in figure 5.2.

In the remainder of this subsection, we explain more thoroughly the different parts that compose it, namely the detection of emissions on the wireless medium, the identification of the modulation type, and the estimation of its parameters.

5.3.1.1 Packet detection in time and frequency

First, we need to isolate the samples of interest in the whole signal, to reduce the amount of data to be analyzed and identify the periods during which a signal was emitted. To detect them, we use a state-of-the-art squelch to find rising and falling edges in amplitude, as shown on Figure 5.3. The threshold used to detect an above-noise amplitude needs to be estimated from a *noise profile* for the receiver and environment in which it is used. This estimation can be done automatically, by making a hypothesis on periods that contain only noise, for example using a

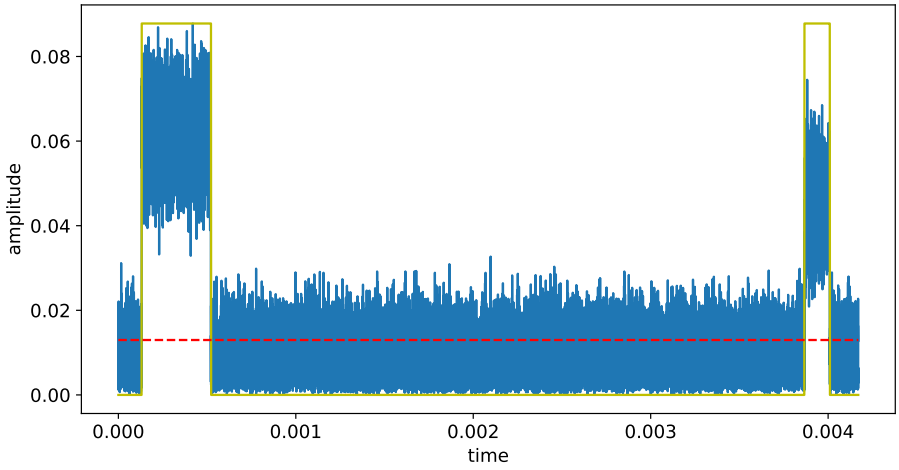


Figure 5.3: squelch example (with threshold)

buffer at the start of the reception as reference by considering it improbable to get a signal at this specific time, but we preferred to keep this part manual, as it is fast and easy to perform, and more reliable this way since a human expert can more easily identify noise in a small reception period before running our approach.

This amplitude detection step can be done at two different levels: either directly on the host computer, integrated with the rest of the tool, or in the acquisition hardware itself, for example by implementing it on the FPGA of an SDR. The first option is simpler to implement and maintain, but imposes limits on the reception sample rate, and therefore on the bandwidth, to allow the computer to process all the incoming samples. A possible compromise, which is used in the current version of our implementation, is to analyse pre-acquired signals offline, which allows the full capabilities of the computer to be used to receive the signal, and thus acquire it at the maximum supported sample rate. However, it obviously means that the implementation as is can't run in full-real time, and needs a delay to get full buffers of samples before running the analysis modules.

Once we identified the time periods in which packets were sent, we need to estimate the frequencies where the emissions occurred during each of these periods, and the bandwidth they used.

First, we identify the presence of the different emissions, and the corresponding central frequencies, by identifying significant peaks in the amplitude of the Fourier Transform (FT) of a packet. A peak is defined as a prolonged sequence of high amplitudes in the FT, and an amplitude being considered as "high" when it exceeds a threshold determined from the noise profile of the acquisition hardware.

For each peak, we then estimate its width depending on the first and last amplitude above our threshold, and apply a given factor to it to ensure that no part of the signal are cut afterwards. The central frequency is estimated from each peak by

computing a weighted average of the frequencies that constitute it: each frequency is weighted by the corresponding amplitude in the FT, and the sum is then divided by the sum of those amplitudes.

After correcting the central frequency, meaning that we re-centred the spectrum around the identified frequency for a given emission, and thus put it in *baseband*, we need to isolate it from the other simultaneous emissions. To do so, we use the estimated bandwidth as a parameter for a state-of-the-art low-pass filter to remove all other emissions from the signal. In our implementation, we used an order 5 Butterworth filter, and took twice the width estimated with the threshold as first bandwidth estimation.

5.3.1.2 Modulation type identification

The next step of our approach is to identify the modulation that was used in each of the emissions that were isolated. We then study, for each physical quantity that can modulate information, namely phase, frequency and amplitude, the possible presence of symbols for different modulations.

To verify that the signal uses a specific modulation, we chose the following conditions:

- **condition 1:** the signal must be composed of a specific set of valid symbols for the modulation, that are of equal length in time
- **condition 2:** the signal must contain at least two of such symbols, in significant proportions

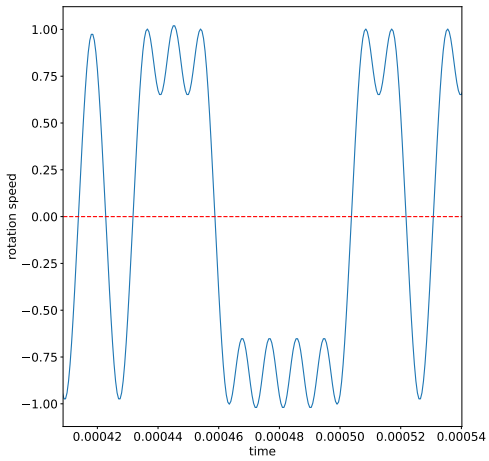
The first condition ensures that the signal contains data modulated in the physical quantity we are studying. For example when we try to find a Binary Frequency Shift Keying (BFSK) modulation, it ensures that there is a sequence using two different symbols of equal length embedded in the signal's instantaneous frequency, as shown in the example in Figure 5.4a.

To find such symbols, we need a representation of the signal corresponding to the modulation under study. For example, for the binary versions of the three basic modulations (amplitude, frequency and phase), these representations $r(t_i)$ are defined as functions of the signal $s(t_i)$ as follows:

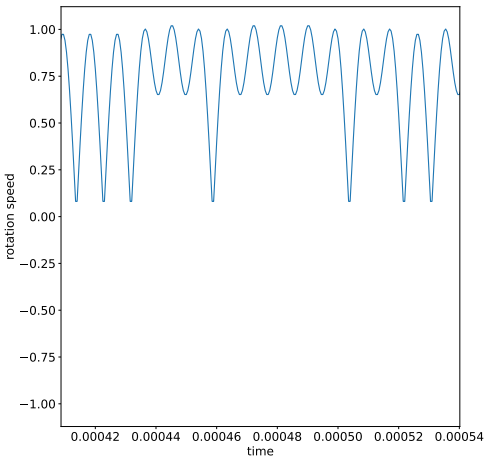
- **BASK:** the signal amplitude $r_{BASK}(t_i) = |s(t_i)|$
- **BPSK:** the phase of the signal $r_{BPSK}(t_i) = \arg(s(t_i))$
- **BFSK:** the instantaneous rotation speed $r_{BFSK}(t_i) = \arg(s(t_i)) - \arg(s(t_{i-1}))$

Our implementation uses a modulation dictionary, containing Python classes implementing a method to get a representation for the corresponding modulation.

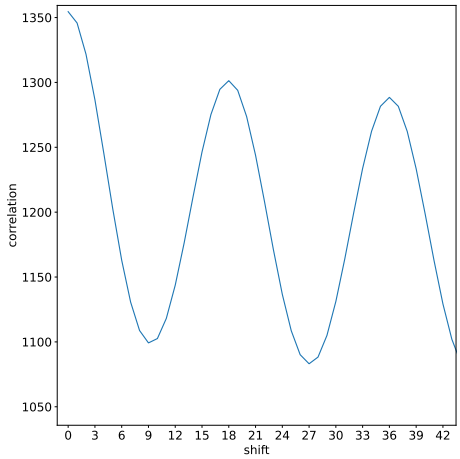
We then need to estimate precisely the symbol period to be able to demodulate. The method we chose to do that is to detect periodicity in the modulated flow by



(a) example of symbol view



(b) example of symbol projection



(c) example of symbol rate identification via autocorrelation

Figure 5.4: Physical layer identification steps for a GFSK with 18 samples per symbol

using an autocorrelation: the signal is compared to shifted versions of itself, and the autocorrelation returns higher amplitudes for shift values that resulted in high similarity, thus highlighting a possible periodicity. By checking the presence of periodic amplitude peaks in the autocorrelation, which can be found by running again an autocorrelation on it, we can find out if there are repeated symbols in the signal. Indeed, if the signal is periodic, the autocorrelation periodically shows high amplitudes, when the corresponding shift aligns symbols; then, a new autocorrelation

highlights the periodicity of the first one, which already "flattened" the noise since it didn't correlate with anything, to obtain a clean first peak value at the symbol size.

Depending on the modulation, we apply a transformation to the representation given by the corresponding class in order to represent in the same way all possible symbols. In our example in Figure 5.4b, a BFSK usually uses two opposite rotation speeds to modulate 0 or 1 bits, so when considering this modulation we study the absolute value of the previously computed rotation speed[1]. Again, the different classes implemented in our implementation contain a method to perform this transformation for the three main modulation classes. The architecture of the approach allows to easily integrate other modulations by computing the corresponding representation. We can then finally compute the autocorrelation of the resulting signal to evaluate if it contains symbols of the modulation, and if so, estimate more accurately its symbol period and bandwidth.

In the example in Figure 5.4c, the autocorrelation has a maximal value (excepting the value at 0, corresponding to a comparison between the signal and itself without shift) at offset 18, corresponding to the number of samples per symbol that we used to generate it.

The second condition ensures that we really received a signal from the modulation we are studying, and not another modulation which results in a constant emission of the same symbol for the physical quantity under study. By default, we consider that if 99.9% of the signal is modulating the same symbol, the second condition is not met.

Finally, to demodulate the signal correctly, we need to identify the moment of decision and reduce the impact of the noise. To do so, we used a set of usual pulse-shaping filters, such as gaussian or semi-sinusoidal shapes with various parameters, to correlate with the signal. If the signal was created using a given filter, the noise is attenuated in the correlation with it, while keeping the useful parts of the signal. Otherwise, if the filter is too far from the right one, it reduces the useful signal's part[2] and lets more noise and less signal out. Moreover, a good knowledge of the filter that was used is an identifying feature for the signal, and allows for a more accurate estimation of the modulation parameters.

Additionally, we add metadata to the demodulated stream, in order to use it later as a feature for communication detection. The final object corresponding to a packet for a specific modulation contains:

- the timestamp at which the packet was detected
- the central frequency at which it was detected

[1]For clarity, the example shown is a GFSK, which corresponds to a BFSK where the symbols have a gaussian shape, and which is the standard type of BFSK used in protocols such as BLE or ESB.

[2]Examples of eye diagrams for a noisy signal can be found in figures 4.10b, with the correct filter, and in figure 4.10a, with a less adapted filter that lets more of the noise and less of the signal out

- the estimated bandwidth
- the demodulated binary stream
- the modulation index (depending on the modulation, it characterises the distance between the different symbols, for example it characterises the distance between the two modulating frequencies in a BFSK)[3]

5.3.2 Link-layer identification

After having analysed the Physical Layer, we get a set of demodulated streams along with physical-layer metadata. The next step is to analyse together streams that might use the same protocol in order to check if it corresponds to a known IoT protocol, and if not to infer a maximum of information to simplify further analysis by an expert.

5.3.2.1 Identification of known IoT protocols

Before analysing further the binary stream, we first need to know if it can already be dissected as a known protocol. To do so, we built a protocol dictionary, in the same way as we built the modulation dictionary previously described.

This dictionary contains, for each possible modulation from the previous step, a list of protocols that could correspond with their dissectors, and methods to detect if the dissected packet is valid or not. Then, for each of the packets that we isolated, we pass them through the dictionary that matches its modulation to dissect it with every possible protocol. If a protocol matches, its dissected version, along with the protocol, are added to its metadata and no further analysis is done.

However, even if bases of dissectors for various protocols can be found, for example with Scapy[sca 2022] in Python, it is more difficult to assess whether a dissected packet is valid or not just from the dissector: indeed, it is possible to get an impossible value for a given metadata field without impacting the rest of the dissection, for example with opcodes, which could make the rest of the dissection meaningless. It is then needed to implement, along with our approach, validation heuristics based on the specifications of the different protocols to estimate whether the packet is valid or not.

Therefore, we also considered the case where there was a dissector for a known protocol, but no validation method, or only partial ones. In that case, the dissected versions for each protocol that can be considered valid with the basic validation methods, or for the protocols that don't have validation methods, are added to the metadata for the expert to check, and the packet is still forwarded to the rest of the analysis in case none of the protocols were right.

[3] other definitions of the term exist, in this manuscript we focus on the distance between symbols

5.3.2.2 Detection of similar packets

To infer the possible fields of the unknown protocols that remain in our packet list, we first need to regroup the packets by protocol, or at least estimate which packets use the same one. A first step of this isolation is done thanks to the physical-layer metadata, separating packets that use different modulations, or use different communication channels. Let us note that this approach has the drawback of also separating in several classes packets from a protocol implementing channel-hopping. However, we still chose this approach because it would be easier to merge similar protocols afterwards than analysing a list of packets that don't use the same one, hence interfering with our field detection.

The following parts of this step were heavily inspired by the works in the Netzob tool[Bossert 2012b, Bossert 2014]. As in their tool, we first decided to use a bioinformatics algorithm called Needleman-Wunsch, used originally to align DNA sequences between themselves.

This works by finding similar sequences at different positions in a packet, allowing us to align fields whose value doesn't change greatly over time in a packet even if they are separated by variable-length fields. The similarity between packets is determined thanks to an edition distance, similar to the Levenshtein distance algorithm in the sense that it uses penalty scores for different edition types, replacement, addition or deletion of a character.

It is then possible to regroup the different packets in clusters based on their similarity with the score returned by the algorithm. As used in [Bossert 2012b, Bossert 2014], we chose to use the UPGMA[4] algorithm: each packet is first used to constitute its own group, then we iteratively regroup the two closest clusters as a single one, the distance between two clusters being defined as the average of the distances between elements from each other. The algorithm stops iterating once the closest clusters have a distance above a certain threshold. In our case, we used as maximal distance between two clusters half the maximum of the distances between packets.

5.3.2.3 Field classification by entropy

Once the packets are regrouped by similarity, we try to infer information on their link-layer protocol, assuming they use the same one.

The first step is to classify the fields depending on their variability, which first gives some high-level information on the packet structure, and allows optimizing further analysis by eliminating impossible field types.

We define three main categories for the fields:

- **Constant fields:** fields whose value remains the same in all packets. It could for example be a preamble, an address if only one object is emitting, or a constant measure from a sensor

[4] Unweighted Pair Group Method with Arithmetic mean

- **Pseudo-random fields:** fields whose entropy over time exceeds a certain threshold, that is used as a parameter for the approach. It could for example be an encrypted part of the packet, or highly variable applicative data.
- **Variable fields:** fields that vary in value between packets, but not enough to fall in the previous category. It could for example be a sequence number, a size field, or applicative data.

This step is simply done by looking at the aligned versions of the packets from each group, and for each byte index, if the value remains the same, it is labelled as constant, otherwise we compute its entropy to determine whether the field is to be considered pseudo-random or not.

5.3.2.4 Field-by-field heuristics

We then implemented a variety of heuristics allowing for a better estimation of the possible fields in each part of the packets. Let us note that the following list is not exhaustive, and we give some insights for retrieving more information than the current state of our approach in 5.3.2.5.

Length fields:

A first field that can be estimated easily is the length field, if present. Indeed, when packets have variable-length fields, they also need to have another field containing information of said length. We use the fact that the evolution of the field's value is proportional to the length of the field it describes, thus being correlated with the packet size in the case where there are not two non-encapsulated variable-length fields. If all variable length fields are encapsulated in each other, at least the length field for the encapsulating field is correlated with the total length. Let us note that, to our knowledge, there is no widespread IoT protocol using several non-encapsulated variable-length fields in the same packet.

To find such length fields, we build vectors with the evolution of each field's value, and another one with the evolution of the packet's lengths. Then, we compute the correlation of each of the first vectors with this last one. The correlation allows seeing the similarity between a vector and shifted versions of a second one. This way, the correlation with shift 0 gives a high value for a field that would correspond to a length field.

Sequence number fields:

Another type of fields that can be quickly estimated are the sequence numbers. We considered that such fields can only stay at the same value, or be incremented by a small amount between two consecutive packets. As such, we analysed the evolution of each variable field, and tagged as a possible sequence number all of

those that followed such an evolution, taking the possible overflows into account (for example, on a 1-byte field, a value going from 255 to 0 is considered as being incremented by 1).

5.3.2.5 Heuristics evolution perspectives

Some other heuristics were thought of, even if not implemented yet in our works. The first one is the detection of the packet header, thanks to the high-entropy and length fields. This would work by estimating where the data begins, or at least a maximum index for the data by finding the first high-entropy field in the packet format, if there is any, hypothesizing that the header won't contain such kind of fields. Then, we also can find a minimum position for the end of the header by making the hypothesis that the length is always in the header.

The first hypothesis is justified by the fact that a header contains metadata on the packet (sequence numbers, length, address...), which shouldn't change pseudo-randomly over time. However, there are exceptions: if the CRC is inside the header, it can show pseudo-random evolutions if the data is variable, and, depending on the protocol, the flags could also show this kind of behaviour, especially if we look at a few disconnected packets. Those kinds of fields then need to be detected before using this heuristic on the data position. At the moment, we only know of ways to detect and reverse the CRC by testing several CRC sizes on several sub-parts of the packets, trying to determine by brute force the polynomial that is used, but this approach is quite long and fastidious. Therefore, in the current state of our approach, we cannot use this heuristic.

The second hypothesis, however, is coherent with what can be found in usual protocols, as the length field needs to be before the variable-length fields in order for a parser to be able to read them. Some protocols may be designed to be able to read variable length data by putting a recognizable suffix to it, and using the length field for another purpose, but this would impede the presence of the suffix in the data, meaning pre-processing on it to remove the forbidden symbols. To our knowledge, such protocols where the length field is after a variable-length fields don't exist, at least in commercial protocols.

5.4 Implementation

As shown in figure 5.5, our tool consists of four main components: a set of core libraries, a modulation dictionary, a protocol dictionary for each modulation[5], and finally a protocol structure analysis module. The protocol and modulation dictionaries are composed of Protocol or Modulation Managers, which are classes implementing the required primitives for a given protocol or modulation.

[5]BASK: Binary Amplitude Shift Keying, BFSK: Binary Frequency Shift Keying, QPSK: Quadrature Phase Shift Keying (4 equidistant phase values)

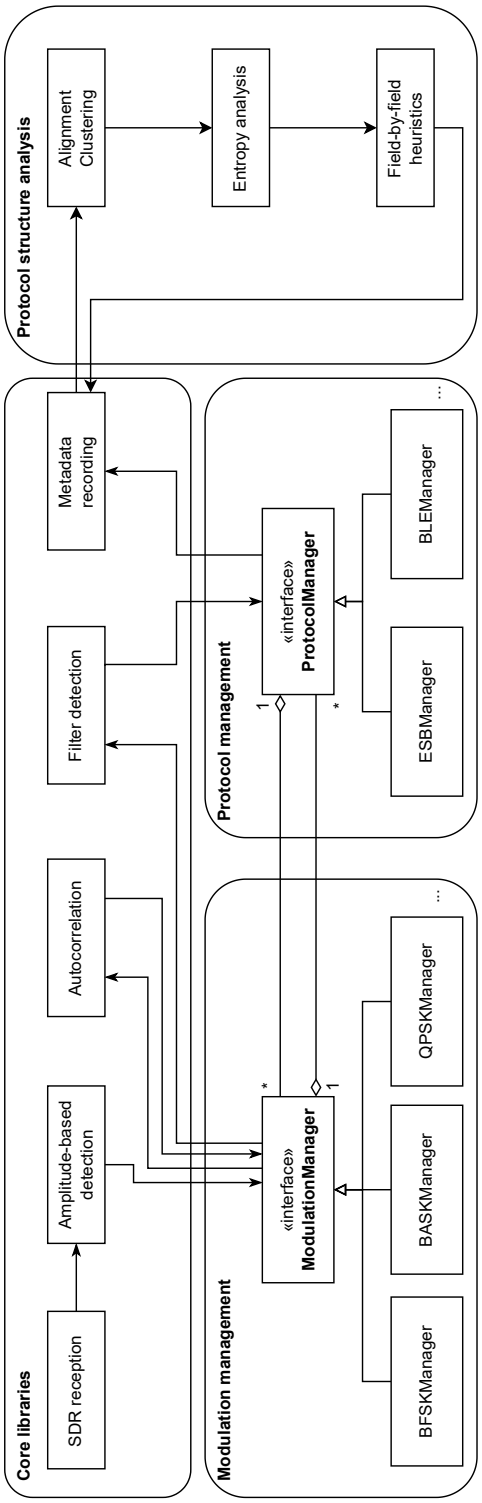


Figure 5.5: Tool architecture overview

5.4.1 Core libraries

The core libraries of our implementation are split into five blocks. The first block (*SDR Reception*) is a reception module implemented in C, using the SoapySDR library for SDR control. This allows our code to be easily used with any SDR compatible with this library. The module receives sample buffers before sending them to the next block.

The following block (*Amplitude-based detection*) isolates by amplitude the received frames. The IQ samples corresponding to the frames are analysed to isolate the frequency channels in use, before being saved in a file for further offline processing. The signal is then analysed by different Modulation Managers, presented in more detail in Section 5.4.2.

Each Modulation Manager then passes the representation of the signal to the *Autocorrelation* block, that aims at identifying the symbol period of a possible frame that it would contain. If the symbol period that is estimated is stable enough, the Modulation Manager then sends the signal representation along with the symbol period to the *Filter detection* block.

The *Filter detection* block passes the signal representation through the filter bench described in Section 5.3 to identify the filter shape minimizing the noise and get the moment of decision to demodulate the signal. The demodulated stream that results from this block is then sent to the Protocol Managers (as described in 5.4.3) that are linked with the current Modulation Manager.

Once the Protocol Managers have analyzed the binary stream, all the extracted metadata are then saved by the *Metadata recording* block in a structure for further use by different applications, such as the one presented in 5.5.3.

5.4.2 Modulation Managers

To be able to transparently manage different modulations, and to allow for an easy enrichment of our tool with new modulations, we defined an interface called "ModulationManager", that all Modulation Manager classes must implement. To implement this interface, and be able to interact with the rest of the tool, those classes need to implement four functions:

- **get_label**: a function without arguments that returns a string, which must uniquely identify the modulation
- **get_mod**: a function that takes as input raw IQ samples, and returns its representation depending on the corresponding modulation
- **get_modulation_index**: a function that takes as input the representation from **get_mod** and returns a measure of the distance between the symbols for the modulation, for further use as an identifying feature
- **eye_diagram_precomputing**: a function that projects the representation from **get_mod** on two opposite symbols to discern the right moment of decision in the eye-diagrams.

All Modulation Managers must then be added to the modulation dictionary from the tool to be used by the core libraries.

5.4.3 Protocol Managers

In order to validate the reception of frames from known protocols, we needed to be able to analyse the binary streams respectively to different possible protocols, for each modulation. We then created, similarly to the "ModulationManager" interface, a "ProtocolManager" interface that all Protocol Manager classes must implement. Those classes need to implement two methods, **get_label**, which is similar to the one for Modulation Managers, and that also needs to return a uniquely identifying string for the protocol, and **check_protocol** which takes as inputs the raw binary stream extracted from the signal, the symbol rate that was identified and the frequency on which the signal was detected. If the binary stream contains a valid frame for the protocol, the function must then return a boolean with the value True, a scapy [sca 2022] class that corresponds to the frame type and that can be used later to compute the corresponding scapy packet, and the truncated binary stream, in the form of a byte array. If not, it must then return a boolean with the value False and two empty values (*None* in Python). In further evolutions of our tool, we plan to introduce the possiblity of setting the boolean to False but still set the other values, meaning that the packet could be dissected but it cannot be said that the packet is really using this specific protocol. This will be later used to decide whether to try to reverse the packet structures or not, as we will explain further in 5.4.4.

For each Modulation Manager, we associate corresponding protocols[6] in individual protocol dictionaries. Once a modulation is identified and a binary stream extracted, the stream is tested against all available protocols for the modulation. If the binary stream matches a given protocol, its metadata are then augmented with metadata returned by **check_protocol**. Further work will include an automated reverse engineering module to identify specific fields in binary streams that appear to originate from the same communication, respectively to the already available metadata. This will allow an almost fully automated reverse engineering of unknown wireless protocols, and may also aid in the detection of spoofing by identifying abnormal variations in the value of a specific field for example. Let us note that, if a protocol that has not been implemented in scapy is added, it is possible to add user-defined scapy dissectors to the tool in a dedicated directory.

5.4.4 Protocol structure analysis

In the case where the metadata doesn't contain a clearly matching protocol, we then need to run an analysis of the protocol structure, to get more information on its fields.

[6] we have chosen to link modulations with protocols that use them, but our approach allows linking any protocol manager to any modulation manager

Our protocol structure analysis runs in three steps:

- **Alignment and clustering:** using the same Needlman-Wunsch and UPGMA algorithms as in [Bossert 2012b, Bossert 2014], as we explained in 5.3.2.2, we align similar fields at the same indexes in similar packets, to get clusters of packets that supposedly come from a same protocol, and the fields they contain
- **Entropy analysis:** we compute the entropy of each field that we detected across a same cluster of packets, to classify the field into our three categories: constant, high-entropy or variable
- **Field-by-field heuristics:** we then analyse the fields depending on their category, by passing them through various functions that test pre-defined heuristics; this part is modular and easily improvable by the user

The metadata and binary stream for each packet that none of the protocol managers considered as valid with enough certainty are passed through this module, and the resulting metadata on the different fields is sent back to the metadata recording module.

5.5 Experiments

This section describes the experiments conducted to validate our signal analysis approach, and to implement basic defensive algorithms based on the results.

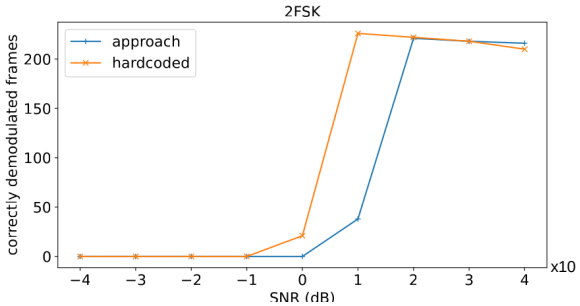
We first checked that the approach works as intended on well known wireless protocols. We then evaluated the accuracy of the parameter estimation on randomly generated wireless protocols. Finally, we demonstrated the usability of our approach in defensive applications, by building on top of it a whitelist and blacklist-based covert-channel detection.

All our experiments were conducted using an Ettus USRP B200mini as receiver, and when needed, we used a LimeSDR USB to emit the signals.

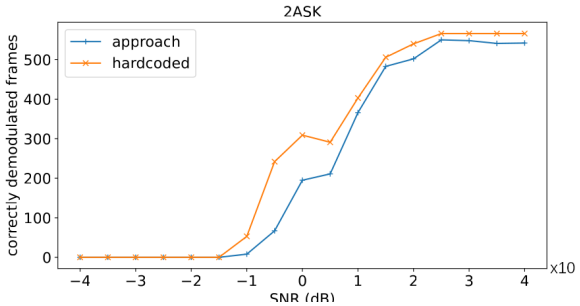
5.5.1 Validation on known protocols

We first tested our approach against well known wireless protocols: 1) Bluetooth Low Energy [Woolley 2019], or BLE, often used by sensors or for short wireless commands, and integrated in most modern Bluetooth chips, and 2) Enhanced ShockBurst [Nordic-Semiconductors 2008], or ESB, used by several wireless keyboards. These protocols were chosen because of the accessibility of their modulation parameters, as specified in table 5.1.

For this experiment, we sent frames with known parameters with both protocols in a realistic environment with other emission sources, including BLE and Wi-Fi communications. The signal was received by an USRP B200mini listening at 2405MHz with a sample rate of 10Msps, during 20 seconds. Note that, even if the



(a) Comparison of our approach against hard-coded 2FSK



(b) Comparison of our approach against hard-coded 2ASK

Figure 5.6: Results of our approach against hard-coded receivers in GFSK and 2-ASK

duration of the capture is quite short, the trace collected actually contains several hundreds of frames and is sufficiently rich to assess the relevance of our tool.

We first computed which frames a standard receiver could demodulate with a correct CRC, to serve as a reference. We then compared the results of our approach with these, by evaluating the difference between the number of frames that we correctly received with the number of reference frames.

For BLE experiments, we received several hundreds of BLE signals from the everyday environment of our laboratory, including phones, computers, and some smaller devices such as connected outlets or smartwatches. The ESB experiments were conducted only with a dongle and a corresponding mouse, the devices using this protocol being limited to mouse, keyboards and dongles, and also included several hundred received frames. The ratio of frames correctly decoded by our approach over the frames correctly decoded by the hardcoded receiver is 94.83% for BLE and 104.03% for ESB. These results show the relevance of our approach: the demodulation performances of known protocols are close to those of the hardcoded values. Our approach even allows to correctly demodulate some ESB frames that the hardcoded receiver was unable to demodulate because these frames were too far from the hardcoded values, either in frequency or in bitrate.

Table 5.1: Modulation parameters for BLE and ESB

protocol	BLE	ESB
modulation	GFSK	GFSK
band	2.4GHz ISM	2.4GHz ISM
bit rate	1/2Mbps	1/2Mbps
channel spacing	2MHz	1/2MHz

5.5.2 Blind estimation of random protocols

Table 5.2: Parameters for the random protocol generation

modulation type	GFSK	2ASK
samples per symbol	7-20	200-400
number of configurations	25	25
frames per configuration	100	100

We then tested our approach on unknown protocols, with random parameters. We generated frequency and amplitude modulations by randomizing the central frequency in the band we were listening to, the bit rate and the modulation index, registering the values for each emission. We generated 25 such configurations, and 100 frames for each of them. We finally simulated their emission in a random order, with different Signal to Noise Ratio (SNR) values. The frequencies ranged from 431.42MHz to 436.42MHz, and the signal was at a constant 5Msps sampling rate, with between 7 and 20 samples per symbol for the GFSK, covering classic values of bit rate for this modulation, and between 200 and 400 for the 2ASK needing longer symbols. For the GFSK experiment, the ratio between the inter-frequencies distance and the bandwidth was between 0.25 and 0.75. For the 2ASK experiment, the ratio between the inter-amplitude distance and the amplitude was between 0.5 and 1. Those parameters for each of the two experiments are summarized in table 5.2.

The metrics used to assess the quality of the reception are defined as follows:

- **reception accuracy:** proportion of frames that were demodulated correctly
- **frequency error:** error on the carrier estimation
- **bit rate error:** error on the bit-rate estimation

We simulated continuous emissions, where the signals were not emitted at the same time. The results for an SNR of 20dB are presented in table 5.3. The confidence intervals are computed at a 95% confidence level, assuming the distribution of each value is a gaussian centred on the value's mean. The results for all SNR values tested between -40 and 40 dB are presented in figures 5.6a and 5.6b. Overall, the results show a very good reception accuracy of our approach both for FSK and ASK modulations. The frequency error and bit rate error are also very good since

their value is at the maximum several Kb/s whereas the bandwidth and bitrate are in the MHz range (which represents less than 1% error). As a consequence, the desynchronization due to the bitrate error only appears in frames long enough for the successive shifts to become significant, and the frequency error being negligible compared to the bandwidth, it is not enough to hinder the demodulation process. This is further demonstrated by the good reception accuracy, which shows that these errors don't generate significant bitflips.

Table 5.3: Results of our approach for randomly generated protocols (without superposition, SNR=20dB)

modulation type	FSK	ASK
reception accuracy	107.35%	92.96%
frequency error	5089±58 Hz	4716±7335 Hz
bit rate error	365±1 bits/s	436±2058 bits/s

```
#status         type          label   freq                                    bitrate
AUTHORIZED      Modulation    2ASK    *                                       *
ALERT           Modulation    QPSK    *                                       *
AUTHORIZED      Protocol      BLE     [2402e6,2404e6,2406e6,2408e6,2410e6]    1e6
AUTHORIZED      Protocol      BLE     [2402e6,2404e6,2406e6,2408e6,2410e6]    2e6
AUTHORIZED      Modulation    2FSK    [2402e6]                                1e6
AUTHORIZED      Modulation    2FSK    [2402e6]                                2e6
ALERT           Protocol      ESB     *                                       *
```

Figure 5.7: Allowed and forbidden communication flows

5.5.3 Covert channel detection

We implemented a covert channel detection mechanism on top of our implementation as a real-life example of a defense mechanism. Covert channel attacks are critical as they may take several forms, while being quite easy to deploy and difficult to detect in absence of a large-band monitoring system such as the one proposed in this paper. For example, an attacker could visit a company and try to exfiltrate confidential information using a device communicating with a protocol not used by the company legitimate devices and thus likely not monitored. This scenario could for example be deployed if an employee's smartwatch is compromised outside their company, then used when they return to work to collect and exfiltrate sensitive data from their workplace.

For this experiment, we added a module that takes as input the metadata from our approach, and a configuration file to specify what should raise an alert. The configuration file uses "AUTHORIZED" and "ALERT" rules for a set of a modulation or protocol, a set of frequencies and a bit-rate, as shown in Figure 5.7. The rules are evaluated sequentially on each frame until the conditions for one of them are met, or if none of them matches the frame, a default rule is applied.

We used this configuration file for an experiment in which we emitted ESB frames with a commercial emitter (supposed to be illegitimate), along with BLE

frames (supposed to be legitimate) in several sequences of 10 seconds in the first 10MHz of the 2.4-2.5GHz band.

The results of this experiment confirm the relevance of our tool: out of 2446 ESB frames that were sent, 2116 frames raised alerts, thus with a detection rate of 86.5%. Even if some ESB frames were not correctly identified, this detection rate is sufficient to actually identify the covert channel. Note that as our approach allows to provide the demodulated binary streams corresponding to the frames transmitted, it can also be used as a means of simultaneous detection of malicious payloads in various protocols.

5.6 Limitations and discussion

Some modulation schemes use closely overlapping channels, like OFDM. This situation introduces new technical difficulties in the demodulation process, that need to be addressed by specific mechanisms integrated in the specifications. Thus, our approach, without prior knowledge of the communication scheme that was used and the associated mechanisms, or their parameters, is not suitable for such schemes.

Several approaches have been studied to detect OFDM communications and estimate their parameters, as presented in [Oularbi 2011], which separates them in four main categories: identification based on the inter-frame spacing, with the help of the cyclic prefix[Oner 2007, Yucek 2007] or not[Bouzegzi 2010, Punchihewa 2011], techniques modifying the emitters to add detectable information[Sutton 2008, Maeda 2007], and techniques based on the pilot symbols used by the scheme[Socheleau 2011, Jung 2006]. However, since our objective is analysing the modulation without making any assumptions on the signal or modifying the communicating objects, we cannot use any of those methods in our protocol-agnostic approach. Moreover, all these approaches do not allow recovering the bitstream from an OFDM signal in realistic conditions with unknown protocols. Indeed, it would require to correct the channel selectivity of the wireless medium, which is done by knowing the values sent as pilot or reference symbols, thus introducing an assumption on the transmitted data. To our knowledge, none of the existing approaches allows estimating the original versions of those pilot or reference symbols. However, OFDM signals still carry specific characteristics, such as the fixed-lengths blocks separated by cyclic prefixes or guard intervals, or reference symbols. Such characteristics could be detected by the use of methods relying on cyclostationarity or autocorrelation to detect periodic repetitions, knowing usual sample rates and OFDM block sizes. Therefore, an OFDM-specific detection module could be built and integrated in our approach, even without going as far as to demodulate the original data, for example for covert channel detection purposes.

Besides this specific point, we have identified several areas for future work. During our experiments, we were limited in the wideness of the band we listened to by our processing speed. However, as explained in section 5.3, it would have been possible to maximize the efficiency of the hardware used by implementing the first

steps isolating the frames and channels in the SDR's FPGA, which would allow to only send to the computer host the isolated frames, reducing the risks of overflows in the communication buffers between the two. Note also that this approach is not designed to run fully in real-time, since we need to receive the entire frames to analyse them, especially for the steps using the autocorrelation of the frame's signals. The performance of our approach can be improved by implementing the frame and channel isolation into the SDR's FPGA, which would reduce the amount of data to send to the computer to allow for a larger band simultaneous reception.

Finally, we also plan to improve the protocol estimation with a basic reverse-engineering module that can identify fields of interest even in unknown protocols, to go further in the direction of an automated audit tool.

5.7 Conclusion

In this chapter, we presented a new, complete and easily extensible approach to ease the work of wireless network auditors and to detect and analyse communications on the wireless communication medium. In its current state, it is able to recognize and demodulate the three basic modulation schemes, namely amplitude, frequency and phase shift keying modulations, identify known protocols integrated in the well-known python library Scapy, and analyse unknown protocols to identify some fields of interest. Moreover, it can also identify unusual modulations, and estimate their parameters efficiently. This assists greatly in the detection of custom protocols, that can be used for example to stealthily exfiltrate data.

As stated before, this approach is, for the moment, unable to handle communications using OFDM as precisely as it could for simpler modulation schemes. However, it is still possible to integrate protocol-specific OFDM modules to allow receiving communications from known protocols using it, such as the most recent mobile telephony downlink, satellite communications or WiFi standards.

The protocol-format reverse part is also still in an early stage, for the moment being able to find length and sequence number fields. However, we worked on theoretical improvements that would allow getting more information from the packet, including header-position inference.

In the end, this approach allows, in our opinion, to lay the first foundations to fill the current lack of automated approaches covering the first phases of radio emissions' analysis, especially for the Physical Layer inference.

Conclusion and perspectives

Conclusion

In this thesis, we studied the security of IoT, and to some extent, of long-distance wireless communications, from the prism of the Physical Layer. We chose this research axis because of the relative youth of the domain, and the fact that we consider transversal approaches between security and signal processing to be beneficial to protect such networks. Indeed, the layers from Network to Application could be covered by more generic approaches, designed for traditional networks, and the Link layer can also be mostly covered by known measures. The Physical layer, however, is particularly different in this context than in wired networks, expanding the attack surface with vectors specific to these wireless communication means. However, the lack of transversal studies with wireless signal processing can lead to wrong assumptions on the risks of using such devices, resulting in insufficient securization. Moreover, the lack of maturity that can still be seen in the IoT industry, combined with the addition of more and more features, further expands the attack surface. It then becomes necessary to highlight the issues in these networks from a security point of view, and work on solutions to address them.

We first presented an attack focused nearly entirely on the Physical Layer, that allows to pivot from the widely deployed Bluetooth Low Energy to attack Zigbee networks, called Wazabee. We also showed that other modulations could be impacted by pivotal attacks, by highlighting some theoretical compatibilities between them. The numerous works in the field of Cross Technology Communications, even if they often imply cooperation from the devices, also highlight the porosity of these networks between themselves. These works, along with the practical applicability of our attack Wazabee, that could even be embarked on mobile phones[Cayre 2021c], show that one shouldn't consider colocated wireless networks with different protocols as isolated, on the sole basis that the protocols differ. Moreover, this highlights the importance of taking the common communication medium into account when analyzing the security of wireless networks, as it can have a significant impact on the attack surface.

We then designed a new, low-cost fingerprinting approach for IoT devices, working with minimal prior knowledge on the protocols to monitor. Our approach uses entire packets, instead of short periods where the components charge or discharge, to extract characteristics from the spectrum of the emissions. This method thus makes the trade-off of being able to use lower sample-rates to gain the same amount of data, but becoming more sensitive to high variations in the environment or data sent, in the absence of estimation of the part of the signal that carries useful data to remove it. We showed experimentally that, despite this trade-off, our approach is able to distinguish various emitters of standard IoT protocols, even if they are from the same model of the same manufacturer. Monitoring methods similar to

this one can be used to mitigate numerous attacks, notably spoofing attacks that could even be undetectable from the higher layers. They allow, for example, to set a whitelist of devices that communicate in the environment, and to check whether emitters pretending to be one of them is indeed a legitimate object. We believe such monitoring systems to be primordial in the surveillance of wireless networks, as the simplicity of the protocols in use can often result in undetectable attacks, that clone perfectly the few characteristics of legitimate devices of the network.

Finally, we presented an approach to identify communications on wide-band signal receptions, analyzing automatically their modulation and a few interesting features of their Link layer. This approach can be used either to detect suspicious emissions in an entire frequency range and analysing what they may be carrying, for example to detect covert channels exfiltrating sensitive data using unknown protocols. It can also be used to help in the audit of wireless transceivers, by automatically realizing the first steps of the identification of the channels, modulation and several fields of interest in the protocol in use. Against attacks like Wazabee, or other physical-layer based pivotal attacks, we believe such approaches to be primordial to monitor efficiently the frequency range in which emitters may communicate in an environment, to detect emissions on unauthorized frequencies, or suspicious emissions. Moreover, auditability is a critical property of secure systems, but studying wireless networks can quickly become cumbersome, due to a lack of generic tools, that leads to heavy development for each system. Our approach is then part of a process to create generic tools that can work with the heterogeneity of the IoT, minimizing this additional work to facilitate the audit and analysis of wireless networks.

Future works

The attack scenario we presented in Chapter 3, along with the other modulation similarities presented in the same chapter, are examples of possible pivots that could be used from an offensive perspective. However, it doesn't cover all possibilities, and several other may exist, requiring additional monitoring in critical environments. It would then be interesting to work on the theory behind those similarities to try to formalize what makes modulations compatible, and which attack scenarios should be considered when introducing a given modulation in an environment. In the current state of the IoT industry, the theoretical compatibilities presented in this chapter remain less realistic, as the parameters needed for the modulations to coincide raise some issues regarding the link reliability or the limitations of embedded systems in terms of performances. However, in the future, some transceivers might match these conditions, and allow to perform such attacks. Thus, it seems important to us to carry out this analysis to prevent potential future attacks based on such cross-technology communications.

The defensive contribution of this thesis work on two different aspects of wireless network monitoring: on one hand, the fingerprinting approach of Chapter 4

targets spoofing attacks, but needs a minimal protocol-dependant part to identify the devices to monitor (for instance, the MAC address of the devices to monitor must be manually provided). On the other hand, the automated analysis of Chapter 5 allows to extract information about a protocol's parameters, to facilitate further work by an expert to identify potential covert channel attacks or vulnerabilities in the protocol. It could then also be possible to combine both approaches, by using the fingerprinting behind the output of the automated analysis, the second giving the needed information about devices to monitor to the first one. By doing so, it could remove the need to develop additional protocol-specific modules in the fingerprinting approach, facilitating the introduction of new devices in an environment to monitor against spoofing attacks. Indeed, the approach of chapter 5 could allow to extract identifying information from the emission of a device. This would allow to launch it in parallel with the registering phase of the fingerprinting approach, to automatically register the fingerprint and be able to protect the device in the intrusion detection phase.

For the different approaches presented in this manuscript, we need to receive signal samples corresponding to frames to analyze them. To do so, we focused only on Software Defined Radios sending the raw samples to a connected computer doing all the processing. However, some parts of this processing, common to our two approaches, and present in general when trying to get the raw signal corresponding to frames in wireless networks, can be transferred directly onto the receiving chip: indeed, most SDR embed an FPGA for signal reception, often programmable by the user. We could then transfer the detection of packets and channels in the signal directly on the FPGA, and only send the corresponding samples to the computer, greatly reducing the amount of processing required on its end. This allows for a better real-time compliance, as the computer has less data to process before getting the next samples from the SDR, and also improves the performances in the parts transferred to the FPGA. Dedicated chips could also be considered, while being less practical, the chips being specifically designed for this purpose.

We are also working on versions of both approaches that would support, to a certain extent, OFDM-based protocols. Indeed, in the recent years, OFDM has become predominant in long-distance wireless communications, used in recent mobile telephony such as 4G or 5G, recent WiFi standards, and even satellite communications. However, as stated in Chapters 4 and 5, it requires some adaptations from the initial approaches.

For the fingerprinting approach, working with OFDM means being able to fully recover and remove the "useful" data in the spectrum, to try to isolate the characteristics of the environment and devices. To do so, it is then necessary to know which OFDM protocol is in use, to first recover the reference symbols, then correct the signal in accordance to their value, demodulate the signal and finally "re-modulate" the data to create an ideal version of it. Then, we apply to this ideal version the inverse of the correction we did using the reference symbols, to finally substract it from the received signal. To detect and understand OFDM-based protocols, our automated analysis approach needs to be able to correct the channel's impact on the

spectrum, by using the reference symbols of the protocol, and to know the OFDM block size. Methods exist to determine automatically the block size, for example by detecting the presence of cyclic prefixes and the distance with the copied part of the signal, but, to our knowledge, no such method is able to recover the original reference symbols protocol-agnostically. Then, it is necessary to introduce in our approach an other dictionary, containing various OFDM protocols along with their parameters and reference symbols, and testing the received signals against each entry to detect valid emissions. However, this would only serve as a wide-band receiver for those OFDM emissions, as all parameters our approach is designed to reverse-engineer will already be known, as the protocol itself was identified.

Finally, our protocol analysis approach is currently limited to a few fields that it can detect. In its current state, we are capable to detect possible length fields, sequence numbers and low or high entropy sections. This gives a minimal number of information, but is not enough to automatically determine which kind of data is extracted by a covert channel for example. To improve this approach, it could be augmented with new heuristics to find more information about the packet structure. For example, some methods exist to reverse CRC polynomials, which could allow to detect such fields in packets, and thus determine packet validity. It would also be possible to use the position of fields known to be in the header, such as the length or the sequence number, along with a low enough entropy, to detect the header's position, and by extension the payload section. The entropy analysis could also be used to detect high-entropy sections, corresponding to pseudo-random data, corresponding for example to an encrypted payload.

Overall, our defensive contributions aim to create a unified approach, with minimal interaction with an expert and minimal development cost, to counter Physical Layer attacks such as those illustrated in our offensive work, independently from the protocols in use.

Bibliography

[Angluin 1987] Angluin, D. *Learning regular sets from queries and counterexamples.* Information and computation, vol. 75, no. 2, pages 87–106, 1987. (Cited in page 83.)

[Antunes 2011] Antunes, J., Neves, N. and Verissimo, P. *ReverX: Reverse engineering of protocols.* 2011. (Cited in page 83.)

[Avizienis 2004] Avizienis, A., Laprie, J.-C., Randell, B. and Landwehr, C. *Basic concepts and taxonomy of dependable and secure computing.* IEEE transactions on dependable and secure computing, vol. 1, no. 1, pages 11–33, 2004. (Cited in page 9.)

[Azim 2013] Azim, A. W., Khalid, S. S. and Abrar, S. *Modulation classification based on modified Kolmogorov-Smirnov test.* In 2013 IEEE 9th International Conference on Emerging Technologies (ICET), pages 1–6. IEEE, 2013. (Cited in page 82.)

[Bachy 2015] Bachy, Y., Basse, F., Nicomette, V., Alata, E., Kaâniche, M., Courrège, J. and Lukjanenko, P. *Smart-TV Security Analysis: Practical Experiments.* In 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pages 497–504, June 2015. (Cited in page 30.)

[Beddoe 2004] Beddoe, M. A. *Network protocol analysis using bioinformatics algorithms.* Toorcon, vol. 26, no. 6, pages 1095–1098, 2004. (Cited in page 83.)

[Blu 2021] Bluetooth SIG. *Bluetooth Core Specification*, 7 2021. (Cited in page 34.)

[Boda 2011] Boda, K., Földes, Á. M., Gulyás, G. G. and Imre, S. *User tracking on the web via cross-browser fingerprinting.* In Nordic conference on secure it systems, pages 31–46. Springer, 2011. (Cited in page 51.)

[Bossert 2012a] Bossert, G., Guihéry, F., Hiet, G.*et al. Netzob GitHub repository.* 2012. [Online]. Available: `https://github.com/netzob/netzob/tree/master/netzob`. (Cited in page 84.)

[Bossert 2012b] Bossert, G., Guihéry, F., Hiet, G.*et al. Netzob: un outil pour la rétro-conception de protocoles de communication.* In Actes du Symposium sur la sécurité des technologies de l'information et des communications, volume 43, 2012. (Cited in pages 83, 90, and 96.)

[Bossert 2014] Bossert, G., Guihéry, F. and Hiet, G. *Towards automated protocol reverse engineering using semantic information.* In Proceedings of the 9th ACM symposium on Information, computer and communications security, pages 51–62, 2014. (Cited in pages 83, 90, and 96.)

[Bouzegzi 2010] Bouzegzi, A., Ciblat, P. and Jallon, P. *New algorithms for blind recognition of OFDM based systems*. Signal Processing, vol. 90, no. 3, pages 900–913, 2010. (Cited in page 100.)

[Brik 2008] Brik, V., Banerjee, S., Gruteser, M. and Oh, S. *PARADIS : Physical 802 . 11 Device Identification with Radiometric Signatures*. 2008. (Cited in page 51.)

[Cauquil 2016] Cauquil, D. *Btlejuice: The bluetooth smart mitm framework*. DEF CON, vol. 24, pages 4–7, 2016. (Cited in page 13.)

[Cauquil 2017a] Cauquil, D. *Radiobit, a BBC Micro:Bit RF firmware*, 2017. `https://github.com/virtualabs/radiobit`. (Cited in page 32.)

[Cauquil 2017b] Cauquil, D. *Sniffing BTLE with the Micro:Bit*. PoC or GTFO, vol. 17, pages 13–20, 2017. (Cited in page 32.)

[Cauquil 2017c] Cauquil, D. *Weaponizing the BBC Micro:Bit*, 2017. `https://media.defcon.org/DEFCON25/DEFCON25presentations/DEFCON25-Damien-Cauquil-Weaponizing-the-BBC-MicroBit-UPDATED.pdf`. (Cited in page 32.)

[Cayre 2021a] Cayre, R., Galtier, F., Auriol, G., Nicomette, V., Kaâniche, M. and Marconato, G. *InjectaBLE: Injecting malicious traffic into established Bluetooth Low Energy connections*. In 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 388–399. IEEE, 2021. (Cited in page 14.)

[Cayre 2021b] Cayre, R., Galtier, F., Auriol, G., Nicomette, V., Kaâniche, M. and Marconato, G. *WazaBee: attacking Zigbee networks by diverting Bluetooth Low Energy chips*. In 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 376–387. IEEE, 2021. (Cited in pages 2 and 29.)

[Cayre 2021c] Cayre, R., Marconato, G., Galtier, F., Kaâniche, M., Nicomette, V. and Auriol, G. *Cross-protocol attacks: weaponizing a smartphone by diverting its Bluetooth controller*. In Proceedings of the 14th ACM conference on security and privacy in wireless and mobile networks, pages 386–388, 2021. (Cited in pages 41 and 103.)

[Cayre 2022] Cayre, R. *Offensive and defensive approaches for wireless communication protocols security in IoT*. PhD thesis, INSA Toulouse, 2022. (Cited in page 15.)

[CC2 2019] *CC2652R Data Sheet*, 2019. `http://www.ti.com/lit/ds/symlink/cc2652r.pdf`. (Cited in page 31.)

[Chatterjee 2019] Chatterjee, B., Das, D., Maity, S. and Sen, S. *RF-PUF: Enhancing IoT Security Through Authentication of Wireless Nodes Using In-Situ Machine Learning.* IEEE Internet of Things Journal, vol. 6, no. 1, pages 388–398, 2019. (Cited in page 51.)

[Chebrolu 2009] Chebrolu, K. and Dhekne, A. *Esense: Communication through Energy Sensing.* In Proceedings of the 15th Annual International Conference on Mobile Computing and Networking, MobiCom '09, page 85–96, New York, NY, USA, 2009. Association for Computing Machinery. (Cited in page 31.)

[Chen 2008] Chen, Q., Wang, Y. and Bostian, C. W. *Universal Classifier Synchronizer Demodulator.* In 2008 IEEE International Performance Computing and Communications Conference (IPCCC), Los Alamitos, CA, USA, dec 2008. IEEE Computer Society. (Cited in page 82.)

[Clauset 2005] Clauset, A., Newman, M. and Moore, C. *Finding community structure in very large networks.* Physical review. E, Statistical, nonlinear, and soft matter physics, vol. 70, page 066111, 01 2005. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.70.066111. (Cited in page 58.)

[Cui 2006] Cui, W., Paxson, V., Weaver, N. and Katz, R. H. *Protocol-independent adaptive replay of application dialog.* In NDSS, 2006. (Cited in page 83.)

[Cui 2007] Cui, W., Kannan, J. and Wang, H. J. *Discoverer: Automatic Protocol Reverse Engineering from Network Traces.* In USENIX Security Symposium, pages 1–14, 2007. (Cited in page 83.)

[Danev 2011] Danev, B. *Physical-Layer Identification of Wireless Devices.* PhD thesis, ETHZ, 2011. pages 25-90. (Cited in page 50.)

[Danev 2012] Danev, B., Zanetti, D. and Capkun, S. *On physical-layer identification of wireless devices.* ACM Computing Surveys (CSUR), vol. 45, no. 1, pages 1–29, 2012. (Cited in page 50.)

[dat 2020] *Fingerprinting experiments datasets*, 2020. (Cited in page 61.)

[Duchêne 2018] Duchêne, J., Le Guernic, C., Alata, E., Nicomette, V. and Kaâniche, M. *State of the art of network protocol reverse engineering tools.* Journal of Computer Virology and Hacking Techniques, vol. 14, no. 1, pages 53–68, 2018. (Cited in page 82.)

[Eckersley 2010] Eckersley, P. *How unique is your web browser?* pages 1–18, 2010. (Cited in page 51.)

[European-Comission 2022] European-Comission. *The next generation Internet of Things*, 2022. (Cited in page 6.)

[Galtier 2020] Galtier, F., Cayre, R., Auriol, G., Kaâniche, M. and Nicomette, V. *A PSD-based fingerprinting approach to detect IoT device spoofing.* In 2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC), pages 40–49. IEEE, 2020. (Cited in page 2.)

[Gimenez 2021] Gimenez, P.-F., Roux, J., Alata, E., Auriol, G., Kaâniche, M. and Nicomette, V. *RIDS: Radio intrusion detection and diagnosis system for wireless communications in smart environment.* ACM Transactions on Cyber-Physical Systems, vol. 5, no. 3, pages 1–1, 2021. (Cited in page 81.)

[Girvan 2002] Girvan, M. and Newman, M. E. J. *Community structure in social and biological networks.* Proceedings of the National Academy of Sciences, vol. 99, no. 12, pages 7821–7826, 2002. [Online]. Available: `https://www.pnas.org/content/99/12/7821`. (Cited in page 58.)

[Goodspeed 2011a] Goodspeed, T. *Promiscuity is the nRF24L01+'s Duty*, 2011. (Cited in page 32.)

[Goodspeed 2011b] Goodspeed, T., Bratus, S., Melgares, R., Shapiro, R. and Speers, R. *Packets in packets: Orson Welles' in-band signaling attacks for modern radios.* pages 7–7, 08 2011. (Cited in page 32.)

[hac 2022] *HackRF One Official webpage*, 2022. (Cited in page 55.)

[Hall 2005] Hall, J., Barbeau, M. and Kranakis, E. *Radio Frequency Fingerprinting for Intrusion Detection in Wireless Networks.* IEEE Trans. Dependable Secure Comput., 2005. (Cited in page 50.)

[Hao 2019] Hao, S., Wang, N., Sun, R., Liu, M., Dong, M. and Wang, H. *Modulation classification using a goodness of fit test.* In Journal of Physics: Conference Series, volume 1169, page 012068. IOP Publishing, 2019. (Cited in page 82.)

[Hasse 2013] Hasse, J., Gloe, T. and Beck, M. *Forensic identification of GSM mobile phones.* In Proceedings of the first ACM workshop on Information hiding and multimedia security, pages 131–140, 2013. (Cited in page 51.)

[Helluy-Lafont 2021] Helluy-Lafont, E. *Sécurité et détection d'intrusion dans les réseaux sans fil.* PhD thesis, 2021. (Cited in page 81.)

[igr 2022] *igraph project webpage*, 2022. (Cited in page 61.)

[IOT 2018] *B-L475E-IOT01A Data Brief*, 2018. `https://www.st.com/resource/en/data_brief/b-l475e-iot01a.pdf`. (Cited in page 31.)

[Jiang 2017] Jiang, W., Yin, Z., Liu, R., Li, Z., Kim, S. M. and He, T. *BlueBee: A 10,000x Faster Cross-Technology Communication via PHY Emulation.* In Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, SenSys '17, New York, NY, USA, 2017. Association for Computing Machinery. (Cited in page 31.)

[Jiang 2018] Jiang, W., Kim, S. M., Li, Z. and He, T. *Achieving Receiver-Side Cross-Technology Communication with Cross-Decoding*. In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, MobiCom '18, page 639–652, New York, NY, USA, 2018. Association for Computing Machinery. (Cited in page 31.)

[Jung 2006] Jung, Y.-H. and Lee, Y. H. *Use of periodic pilot tones for identifying base stations of FH-OFDMA systems*. IEEE communications letters, vol. 10, no. 3, pages 192–194, 2006. (Cited in page 100.)

[Kaufmann 1987] Kaufmann, L. and Rousseeuw, P. *Clustering by Means of Medoids*. Data Analysis based on the L1-Norm and Related Methods, pages 405–416, 01 1987. (Cited in page 58.)

[Kim 2015] Kim, S. M. and He, T. *FreeBee: Cross-Technology Communication via Free Side-Channel*. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom '15, page 317–330, New York, NY, USA, 2015. Association for Computing Machinery. (Cited in page 31.)

[Köse 2019] Köse, M., Taşcioğlu, S. and Telatar, Z. *RF Fingerprinting of IoT devices based on transient energy spectrum*. IEEE Access, vol. 7, pages 18715–18726, 2019. (Cited in page 50.)

[Lakshminarayanan 2009] Lakshminarayanan, K., Sapra, S., Seshan, S. and Steenkiste, P. *Rfdump: an architecture for monitoring the wireless ether*. In Proceedings of the 5th international conference on Emerging networking experiments and technologies, pages 253–264, 2009. (Cited in page 81.)

[Laperdrix 2019] Laperdrix, P., Bielova, N., Baudry, B. and Avoine, G. *Browser Fingerprinting: A survey*. 05 2019. (Cited in page 51.)

[Laprie 1995] Laprie, J. *Guide of dependability*. 1995. (Cited in page 9.)

[Li 2015] Li, A., Dong, C., Tang, S., Wu, F., Tian, C., Tao, B. and Wang, H. *Demodulation-free protocol identification in heterogeneous wireless networks*. Computer Communications, vol. 55, pages 102–111, January 2015. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S0140366414003041`. (Cited in page 81.)

[Li 2017] Li, Z. and He, T. *WEBee: Physical-Layer Cross-Technology Communication via Emulation*. In Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom '17, page 2–14, New York, NY, USA, 2017. Association for Computing Machinery. (Cited in page 31.)

[lim 2022] *LimeSDR Mini Official webpage*, 2022. (Cited in page 55.)

[Maeda 2007] Maeda, K., Benjebbour, A., Asai, T., Furuno, T. and Ohya, T. *Recognition among OFDM-based systems utilizing cyclostationarity-inducing transmission.* In 2007 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, pages 516–523. IEEE, 2007. (Cited in page 100.)

[Millian 2015] Millian, M. C. and Yadav, V. *Packet-in-packet Exploits on 802 . 15 . 4.* 2015. (Cited in page 32.)

[Mototolea 2020] Mototolea, D., Youssef, R., Radoi, E. and Nicolaescu, I. *Non-Cooperative Low-Complexity Detection Approach for FHSS-GFSK Drone Control Signals.* IEEE Open Journal of the Communications Society, vol. 1, pages 401–412, 2020. [Online]. Available: http://dx.doi.org/10.1109/OJCOMS.2020.2984312. (Cited in page 82.)

[Needleman 1970] Needleman, S. B. and Wunsch, C. D. *A general method applicable to the search for similarities in the amino acid sequence of two proteins.* Journal of Molecular Biology, vol. 48, no. 3, pages 443–453, 1970. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/0022283670900574`. (Cited in page 83.)

[Nei 1983] Nei, M., Tajima, F. and Tateno, Y. *Accuracy of estimated phylogenetic trees from molecular data.* Journal of molecular evolution, vol. 19, no. 2, pages 153–170, 1983. (Cited in page 83.)

[Newlin 2016] Newlin, M. *MouseJack : White Paper*, 2016. `https://github.com/BastilleResearch/mousejack/blob/master/doc/pdf/DEFCON-24-Marc-Newlin-MouseJack-Injecting-Keystrokes-Into-Wireless-Mice.whitepaper.pdf`. (Cited in pages 8 and 32.)

[Nguyen 2014] Nguyen, D., Sahin, C., Shishkin, B., Kandasamy, N. and Dandekar, K. R. *A real-time and protocol-aware reactive jamming framework built on software-defined radios.* In Proceedings of the 2014 ACM workshop on Software radio implementation forum, pages 15–22, 2014. (Cited in page 14.)

[Nordic-Semiconductors 2008] Nordic-Semiconductors. *nRF24L01+ Product Specification v1.0*, 2008. (Cited in page 96.)

[Nouichi 2019] Nouichi, D., Abdelsalam, M., Nasir, Q. and Abbas, S. *IoT Devices Security Using RF Fingerprinting.* In 2019 Advances in Science and Engineering Technology Int. Conferences (ASET), pages 1–7, 2019. (Cited in page 51.)

[Okhravi 2010] Okhravi, H., Bak, S. and King, S. T. *Design, implementation and evaluation of covert channel attacks.* In 2010 IEEE International Conference on Technologies for Homeland Security (HST), pages 481–487. IEEE, 2010. (Cited in page 14.)

[Oner 2007] Oner, M. and Jondral, F. *On the extraction of the channel allocation information in spectrum pooling systems.* IEEE Journal on Selected Areas in Communications, vol. 25, no. 3, pages 558–565, 2007. (Cited in page 100.)

[Oularbi 2011] Oularbi, M. R. *Identification de systèmes OFDM et estimation de la QoS: application à la radio opportuniste.* PhD thesis, Ecole Nationale Supérieure des Télécommunications de Bretagne-ENSTB ..., 2011. (Cited in page 100.)

[Pons 2006] Pons, P. and Latapy, M. *Computing Communities in Large Networks Using Random Walks.* J. Graph Algorithms Appl., vol. 10, pages 191–218, 01 2006. [Online]. Available: http://dx.doi.org/10.7155/jgaa.00124. (Cited in page 58.)

[Powell 2001] Powell, D., Stroud, R.*et al. Malicious-and accidental-fault tolerance for internet applications-Conceptual model and architecture.* 2001. (Cited in page 10.)

[Punchihewa 2011] Punchihewa, A., Bhargava, V. K. and Despins, C. *Blind estimation of OFDM parameters in cognitive radio networks.* IEEE Transactions on Wireless Communications, vol. 10, no. 3, pages 733–738, 2011. (Cited in page 100.)

[Qasim Khan 2019] Qasim Khan, S. *Sniffle: A sniffer for Bluetooth 5 (LE)*, 2019. `https://hardwear.io/netherlands-2019/presentation/sniffle-talk-hardwear-io-nl-2019.pdf`. (Cited in page 39.)

[Ramsey 2015] Ramsey, B. W., Mullins, B. E., Temple, M. A. and Grimaila, M. R. *Wireless Intrusion Detection and Device Fingerprinting through Preamble Manipulation.* IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 5, pages 585–596, 2015. (Cited in page 51.)

[Ryan 2013] Ryan, M. *Bluetooth: With Low Energy comes Low Security.* 2013. (Cited in page 9.)

[sca 2022] *Scapy project homepage*, 2022. (Cited in pages 89 and 95.)

[Schroeder 2010] Schroeder, T. and Moser, M. *Practical exploitation of modern wireless devices.* CanSecWest, Mar. http://www.remoteexploit.org/content/keykeriki_v2_cansec_v1, vol. 1, 2010. (Cited in page 8.)

[Shintani 2020] Shintani, A. *The Design, Testing, and Analysis of a Constant Jammer for the Bluetooth Low Energy (BLE) Wireless Communication Protocol.* 2020. (Cited in page 13.)

[Socheleau 2011] Socheleau, F.-X., Houcke, S., Ciblat, P. and Aïssa-El-Bey, A. *Cognitive OFDM system detection using pilot tones second and third-order cyclostationarity.* Signal processing, vol. 91, no. 2, pages 252–268, 2011. (Cited in page 100.)

[Sokal 1958] Sokal, R. R. and Michener, C. D. *A statistical method for evaluating systematic relationships.* Univ. Kansas, Sci. Bull., vol. 38, pages 1409–1438, 1958. (Cited in page 83.)

[Speth 2004] Speth, M., Madden, M., Hammes, M. and Neubauer, A. *MLSE based detection for GFSK signals with arbitrary modulation index.* In International Zurich Seminar on Communications, 2004, pages 228–231. IEEE, 2004. (Cited in page 30.)

[Spuhler 2014] Spuhler, M., Giustiniano, D., Lenders, V., Wilhelm, M. and Schmitt, J. B. *Detection of reactive jamming in DSSS-based wireless communications.* IEEE Transactions on Wireless Communications, vol. 13, no. 3, pages 1593–1603, 2014. (Cited in page 14.)

[Sutton 2008] Sutton, P. D., Nolan, K. E. and Doyle, L. E. *Cyclostationary signatures in practical cognitive radio applications.* IEEE Journal on selected areas in Communications, vol. 26, no. 1, pages 13–24, 2008. (Cited in page 100.)

[Ur Rehman 2012] Ur Rehman, S., Sowerby, K. and Coghill, C. *RF fingerprint extraction from the energy envelope of an instantaneous transient signal.* In 2012 Australian Communications Theory Workshop (AusCTW), pages 90–95, 2012. (Cited in page 50.)

[Wilhelm 2011] Wilhelm, M., Martinovic, I., Schmitt, J. B. and Lenders, V. *Short paper: Reactive jamming in wireless networks: How realistic is the threat?* In Proceedings of the fourth ACM conference on Wireless network security, pages 47–52, 2011. (Cited in page 14.)

[Wilhelm 2012] Wilhelm, M., Schmitt, J. B. and Lenders, V. *Practical message manipulation attacks in IEEE 802.15. 4 wireless networks.* In MMB & DFT 2012 Workshop Proceedings, pages 29–31, 2012. (Cited in page 14.)

[Woolley 2019] Woolley, M. *Bluetooth core specification v5. 1.* In Bluetooth, 2019. (Cited in page 96.)

[Xu 2005] Xu, W., Trappe, W., Zhang, Y. and Wood, T. *The feasibility of launching and detecting jamming attacks in wireless networks.* In Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, pages 46–57, 2005. (Cited in page 13.)

[Yucek 2007] Yucek, T. and Arslan, H. *OFDM signal identification and transmission parameter estimation for cognitive radio applications.* In IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference, pages 4056–4060. IEEE, 2007. (Cited in page 100.)

[Zuo 2019] Zuo, C., Wen, H., Lin, Z. and Zhang, Y. *Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps.* In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 1469–1483, 2019. (Cited in pages 9 and 13.)

Abstract: We witnessed, in the last few years, the massive expansion of the Internet of Things (IoT) to multiple everyday devices, in homes, buildings and factories. The inclusion of heterogeneous wireless transceivers in potentially critical appliances and environments is then raising growing security and safety concerns. However, according to the litterature, there is still room for improvement, especially in off-the-shelf devices, often not implementing the different security measures of the protocols they use. As such, numerous works focus on the improvement of the global security and safety of the IoT, and more generally in the wireless communications often used by these devices. The subject of the Physical Layer security is often disregarded, because of its complexity or the fact that it is less critical in more traditional wired networks. However, the wireless communication medium can have a substantial impact on the attack surface because of its higher availability, since anyone in radio range can listen to communications. It also brings new vulnerabilities, that are specific to this kind of transmissions. Furthermore, the low complexity and computational power of the devices in use often results in simple protocols, making device cloning and injections more easy, and harder to detect from higher layers. Thus, in this thesis, we decided to focus on the impact of the wireless Physical Layer on network security, and highlight the importance of transversal approaches between signal processing and cybersecurity. We first show the new dimension introduced by the common wireless medium in offensive security, by showing a new attack based on similarities in Physical Layers of two protocols to break the isolation between them. This attack, Wazabee, allows Bluetooth Low Energy transceivers to communicate seemlessly with Zigbee networks. We also show that the principle behind it could be extended to other pairs of protocols, should the conditions be met. This highlights the importance of a more protocol-agnostic monitoring of the wireless medium, able to detect communications outside the legitimate protocols of the environment, and to not consider co-existing wireless protocols as isolated. Then, we propose two defensive contributions, to help secure wireless networks by analysing the communications from a Physical Layer perspective. First, we present a low-cost fingerprinting approach for wireless devices, to detect potential identity theft attacks, i.e. attacks where an illegitimate device tries to take the place of a legitimate one in the network. Then, we present an approach for automated wireless protocol audit, allowing to detect and analyze various protocols emitting in a wide range of frequencies, with minimal assumptions on their nature. These two approaches complete higher-layer security measures, to detect potential intruders or covert channels in the environment, and to ease protocol analysis for security experts.

Keywords: IoT, Internet of Things, Cybersecurity, Wireless networks, Software Defined Radio, Signal Processing

Résumé : Nous assistons, depuis plusieurs années, à l'extension de l'Internet des Objets à de nouveaux objets du quotidien, que ce soit dans les domiciles ou sur le lieu de travail. L'ajout d'émetteurs hétérogènes dans de nombreux objets et certains environnements sensibles pose de nombreux problèmes en termes de sûreté et de sécurité. Les travaux récents montrent qu'il y a encore du chemin à parcourir, notamment pour les objets grand-public, qui n'implémentent souvent pas les mesures de sécurité définies dans les standards. Partant de ce constat, de nombreux travaux s'intéressent aujourd'hui à la sécurité et à la sûreté dans l'IoT, et plus généralement dans les réseaux sans-fil, souvent utilisés par ces objets. Le sujet de la sécurisation de la couche physique est souvent mis de côté, que ce soit pour sa complexité ou par comparaison avec les réseaux filaires traditionnels, où elle a moins d'impact. Toutefois, la plus grande accessibilité des réseaux sans-fil par tout ce qui se trouve à portée radio en augmente considérablement la surface d'attaque. S'y ajoutent également des vulnérabilités propres à ce type de transmissions, et la simplicité de certains protocoles, due à la faible puissance de calcul des objets, rendant des attaques telles que le clonage d'objet légitime plus aisées, et difficiles à détecter depuis les couches supérieures. Dans cette thèse, nous nous sommes donc concentrés sur l'impact de l'aspect sans-fil de la couche physique sur la sécurité de ces réseaux, mettant en évidence l'intérêt d'approches transversales entre traitement du signal et sécurité. Nous montrons d'abord l'impact offensif du moyen de communication commun, en exposant une nouvelle attaque basée sur la similarité des couches physiques de deux protocoles. Cette attaque, Wazabee, permet à des objets BLE de communiquer avec des objets Zigbee. Nous montrons également que le principe de l'attaque pourrait être étendu à d'autres paires de protocoles. Nous soulignons ainsi l'importance de moyens de détection plus indépendants des protocoles déployés capables de détecter des communications hors des canaux légitimes, et de ne pas considérer des réseaux sans-fil différents mais colocalisés comme isolés les uns des autres. Enfin, nous présentons deux contributions défensives basées sur l'analyse de ces réseaux depuis leur couche physique. Premièrement, nous présentons une approche à bas coût d'identification d'émetteurs sans-fil, visant des attaques usurpant l'identité d'émetteurs légitimes. Ensuite, nous présentons une approche d'automatisation de l'audit de protocoles sans-fil, permettant de détecter et d'analyser le contenu d'émissions sans-fil, avec un minimum d'hypothèses sur leur nature. Ces deux approches complètent des mesures aux couches supérieures, en permettant de détecter de potentielles intrusions ou canaux cachés, et pour faciliter l'analyse des protocoles utilisés par les experts.
Mots clés : IoT, Internet des Objets, Cybersécurité, Réseaux sans-fil, Software Defined Radio, Traitement du Signal