# Wireless Modulation Identification: filling the gap in IoT networks security audit

Florent Galtier[1], Guillaume Auriol[1,2], Vincent Nicomette[1,2], Paul L. R. Olivier[1], Romain Cayre[3], and Mohamed Kaâniche[1]

[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400
firstname.lastname@laas.fr
[2] Université de Toulouse, INSA, LAAS, F-31400
[3] Eurecom, 450 Route des Chappes, F-06410
firstname.lastname@eurecom.fr

**Abstract.** The massive deployment of IoT devices comes with the creation of many wireless communication protocols designed to support various applications. However, while some of these protocols are public and widely used, such as Bluetooth Low Energy or Enhanced ShockBurst, the specifications of other protocols are only partially, if at all, publicly available, making it difficult for security researchers to study them, especially for auditing purposes. In this paper, we address this issue by presenting an innovative, easy-to-use and protocol-agnostic toolkit to analyze unknown network communications. The toolkit is designed to automatically infer their physical layer characteristics and and to extract the binary content of their frames. We conducted experiments to identify the modulation parameters for various wireless communications used by off-the-shelf devices as well as for randomly generated protocols, considering both FSK and ASK modulations. In addition, our results show that the proposed toolkit is capable of successfully detecting covert channels in wireless environments. We also conducted a case study on an undocumented proprietary wireless mouse protocol.

**Keywords:** IoT, Intrusion Detection, Radio Frequency Identification, Secure Communication, Cyber-Physical Systems, Security and Privacy

## 1 Introduction

Industrial and home networks are increasingly relying on wireless networks, either to ease communication, save space or reduce costs. However, such networks are 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 inject malicious transmissions among legitimate ones. In contrast, such attacks on wired networks are mitigated by using robust communication systems and by monitoring anomalies in transmission frames. However, auditing and monitoring wireless networks is hindered by the large number and heterogeneity of

protocols. They can have radically different physical layers and use specific channels to communicate, sometimes adding channel hopping. Moreover, an attacker willing to stealthily exfiltrate data could communicate through a custom crafted protocol designed for escaping the scrutiny of potential monitoring measures.

Extracting binary streams from known protocols is possible with available tools [1, 2], but such methods are lacking for more generic use cases. Auditing unfamiliar wireless protocols involves laborious tasks to identify their modulations and parameters, then to recover the binary streams and higher-layer protocols.

With the diverse protocols in IoT and numerous proprietary ones, auditing wireless communications becomes increasingly difficult. Many IoT devices have flaws that make them vulnerable to attacks. Furthermore, proprietary protocols often rely on security through obscurity, further complicating audits. Securing wireless networks is therefore a pressing challenge. Developing new tools to analyze protocols in a broader way is crucial for improving wireless security audits.

In this paper, we propose to address this challenge by designing and implementing a novel modulation identification approach and analysis module suitable for different types of modulations, and easily extensible. The proposed approach also allows to retrieve the binary payload of known protocols, and identify fields of interest in unknown ones. More precisely, this approach with the designed toolkit is able to detect transmissions and infer their parameters and content, without making any assumption. It combines reception over a wide frequency band (for example higher than 10MHz) and different modulation detection algorithms 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. It only requires for the operator manual parameterization and estimation of the noise level beforehand, and is entirely automated afterwards. Furthermore, the source code of the tool and the experiments will be made public to allow reproducibility.

Overall, the following contributions can be highlighted in this paper:

- A novel approach to identify and analyze modulations in signals, infer their parameters and retrieve basic information about the higher-layer protocols.
- An open-source tool implementing this approach.
- The validation of the approach and test of the tool on well-known as well as on randomly generated protocols, by demonstrating its capability to detect covert channels, and describing a full case study on a wireless mouse.

The remainder of this paper is structured as follows. Section 2 discusses the state-of the art about wireless protocol identification and transmission detection in wireless networks. Then, Section 3, briefly describes the theoretical background behind our tool. The implementation is presented in detail in Section 4, and its validation on different application use cases is addressed in Section 5. It also presents a case study on a wireless mouse using an undocumented and proprietary protocol, and shows how our tool can help a reverse-engineer. The limitations of and possible improvements are discussed in Section 6. Finally, Section 7 concludes and presents future directions for our work.

## 2 Related works

Automated detection of transmissions has always been a major concern in security because of its importance in the design of security monitoring systems.

Several steps are needed to retrieve the information of interest from the signal: first, detecting when and at which frequency frames are transmitted; then, identifying the modulation type, identifying its parameters; and finally, demodulating and analyzing the binary stream to extract information on the protocol's frame format, and on the higher-layer content of the communications. 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 [3] or [4] focus on specific modulations and their unique characteristics to detect them. The main issue with these approaches is their high false positive rates, as they ignore the other modulations. 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 [5] and [6], 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 paper, 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. However, this hypothesis is, in general, met with protocols using pulse-shaping filters to increase the spectral efficiency, which are predominant.

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. Some approaches focus on a single protocol, and their goal is rather to detect attacks on upper layers [7]. Others take a broader perspective, considering different possibilities and performing a case disjunction based on protocol-dependent heuristics to identify the type of the emissions [1, 2]. However, they do not allow detecting unknown protocols, especially outside pre-defined channels. This exposes them to covert channel attacks that exploit the knowledge of the monitored channels.

Other works focus on specification-agnostic anomaly detection. For instance, in RIDS [8], the authors trained an auto-encoder in an attack-free environment to detect unusual activities in the monitored frequencies. However, these approaches suffer from a trade-off between detection accuracy and the amount of processed data. This 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 paper aim at addressing the limitations of both of these approaches in the context of a more generalized covert channel detection.

Finally, a high and growing number of works focus on the use of Neural Networks-based classification systems to identify the modulation type, for example [9–13]. However, such methods require huge resources for their training, and suffer from a lack of explainability because of the black-box aspect of neural networks. Moreover, none of those methods allows reversing the modulation parameters and recovering the binary stream, unlike the method we propose.

Our approach can be applied 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.

Once the modulation parameters have been identified and the frame demodulated, another important aspect of the analysis of the emissions is the inference of the packet format. As stated by J. Duchêne in [14], a large majority of the state of the art field-identification approaches are based on the PI Project [15], an approach from 2004 based on the use of bioinformatics algorithms, Needleman-Wunsch [16] and UPGMA [17, 18]. In the case of unknown protocols, we also use them in our approach. These methods often use specific field delimiters that are known in advance [19, 20], or make some assumptions on the protocol to simplify the analysis [21]. However, the goal of our approach is primarily to extract a maximum number of information from the protocol without intervention and without making assumptions on the packet format. To our knowledge, one of the most advanced approaches is Netzob [22]. It is also based on bioinformatics algorithms to align and delimit fields, and then on applying the L* language inference algorithm [23] to estimate the protocol's grammar. However, this tool is relatively complex, and lacks recent updates, pushing us towards implementing a simpler and lighter approach. Finally, we can also cite some approaches using the Voting Experts algorithm [24] to segment packets into fields. For instance, in IPART [25], the authors introduce a specialized version of the algorithm that allows them to do a basic reverse-engineering of industrial protocols. However, the assumption behind this algorithm is that all packets should come from the same protocol, which makes it more complex to implement in our case.

## 3 Analysis workflow

Figure 1 illustrates the three main parts of our approach: 1) detecting where the packets are in the raw signal received from an SDR (Software Defined Radio), 2)

analyzing the physical layer to demodulate, and 3) finally extracting information from the demodulated stream.

### 3.1 Emission detection

The first step, represented by the **Emission detection** block, is to find where in time and frequency the signals are emitted. We first use a simple amplitude-based detection, called a *squelch*. This isolates parts of the signal with an amplitude above a certain threshold. The squelch threshold is based on the SDR's noise profile, and must be evaluated beforehand on a signal containing only noise. Then, we compute the Fourier Transform of each signal isolated this way, to detect the presence of amplitude peaks that highlight the frequencies in use. For each estimated central frequency, we then keep a "baseband" copy of the frame, i.e., a version that we have re-centered on the identified central frequency.

To demodulate and isolate the different channels, we need to estimate the bandwidth for each detected frame. We first make a coarse estimate using the previous Fourier transform, by identifying the width of the corresponding peak. This estimate is then used to pass the signal through a state-of-the-art Butterworth low-pass filter to avoid interferences from neighbouring channels. Since our estimate is not accurate enough at this point, the considered cut-off frequency used for the low-pass filter is twice the distance to the edge of the peak in the Fourier transform. This results in including more noise, but prevents accidentally removing interesting parts of the signal. Note that if this cut-off frequency is too large compared to the sample rate, and in this case the Shannon criteria would not be met, no filter is applied.

### 3.2 Physical Layer analysis

In this section, we explain the different steps of the Physical Layer Analysis, as shown on Figure 1. This process is separated in two distinct workflows, one in the general case, and another on the On-Off-Keying (OOK) modulation, which has the particularity of using non-emission periods to code information.

**General case - modulation identification** In the general case, we test the isolated signals against different types of modulation, as represented in the **Modulation analysis** block.

To precisely estimate the bandwidth, we first identify the symbol period. For different modulation types, we implement a first function that extracts the physical quantity corresponding to it. For example with an amplitude modulation, the amplitude is extracted from the signal. Then, we implement a second function that projects all symbols on a single one. For instance, we take the absolute value if the symbols are opposites.

Second, we compute the autocorrelation on this projection to highlight the presence of periodicity, which would come from the presence of regular symbols in the stream. Indeed, the autocorrelation is the correlation between the signal

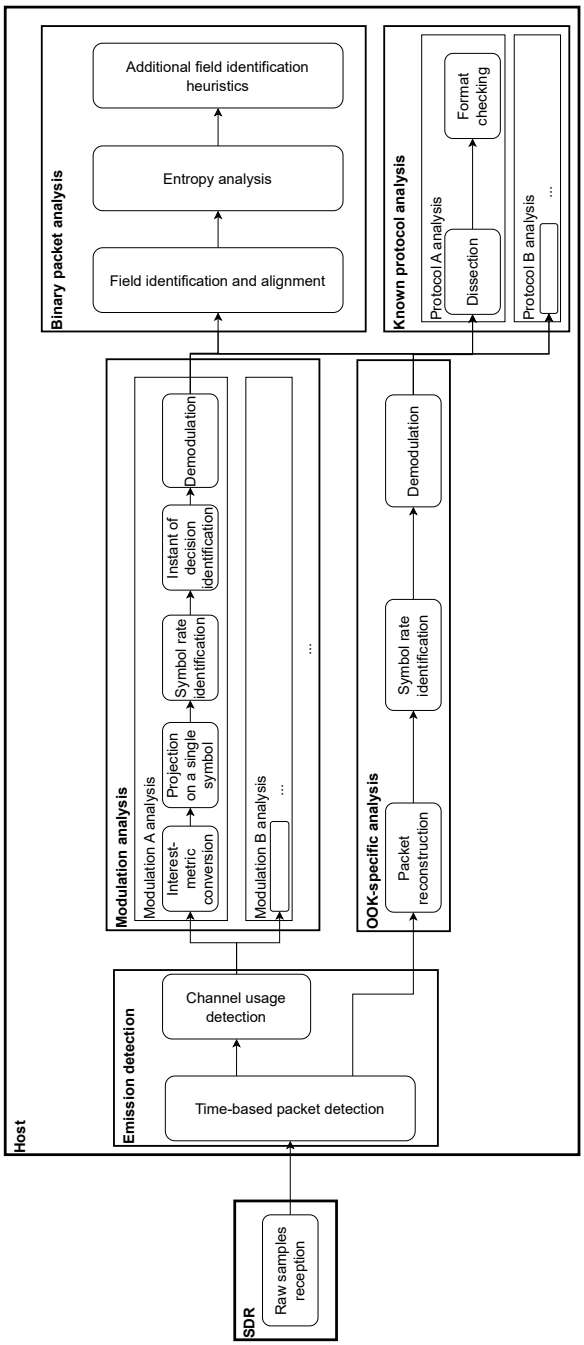
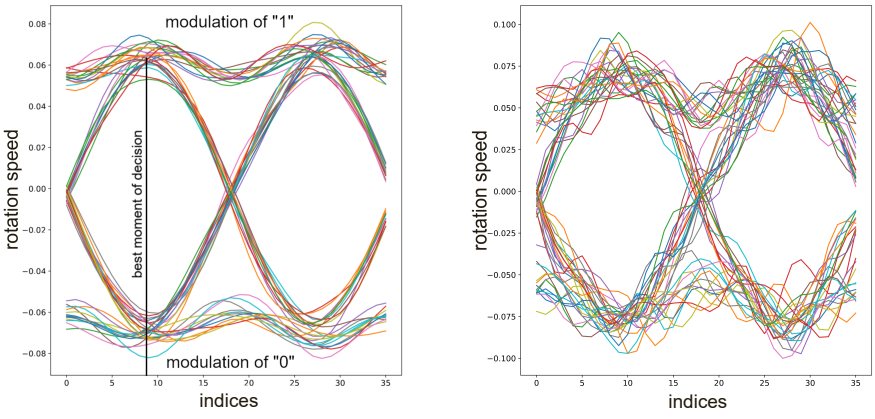



Fig. 1: Approach architecture overview

and shifted versions of itself. Therefore, if the signal contains recurring patterns at regular intervals, it correlates better with itself shifted by multiples of this interval, resulting in regular peaks in the autocorrelation. When such periodicity is found, we deduce the number of samples per symbol and bandwidth.

**General case - binary stream extraction** After identifying a modulation and filtering the signal, it is possible to automatically recover the binary stream.

All signals are made of series of symbols shaped with a pulse-shaping filter, that transforms pulses corresponding to binary data into a modulated signal. Accordingly, we build a bench of different usual filters, and correlate each of them with the received signal. If the chosen filter is similar to the original pulse-shaping filter that was used, the noise correlates significantly less than the original signal, partially reducing the noise's impact. The quality of the correlation is evaluated thanks to eye diagrams : all pairs of two successive symbols are overlapped in order to see when the symbols are the farthest from each other. Eye diagram examples in a practical case can be found in Figure 2. The correct moment of decision for demodulation is selected where the eye-shape is the most open, thus maximizing the distance between symbols. The correct filter is chosen where the ratio between the height of the shape, linked to the signal power, to the width of its lines, linked to the noise impact on the signal, is kept minimal, as it maximizes the signal-to-noise ratio. Additionally, the eye diagrams are used to estimate the moment of decision to demodulate the signal, which is the instant where the signal-to-noise ratio is maximized.




(a) eye diagram - correct filter (b) eye diagram - less adapted filter

Fig. 2: Eye diagrams for GFSK, with both the right filter and a less adapted one

The Figures 2a and 2b represent eye diagrams for a Gaussian Frequency Shift Keying modulation, applying a matching filter and a less adapted one. The eye diagram then represents the different possible transitions in rotation speed between symbols representing binary 0 or 1. The moment of decision is then chosen as the ninth sample of each period, because, as seen in Figure 2a, the symbols for 0 and 1 are concentrated each in a single spot at this moment, meaning it minimizes the risks of demodulation errors. Additionally, as seen in Figure 2b, a less adapted filter keeps a significant amount of noise in the signal, reducing the margin between the two symbols.

**OOK-specific analysis** The amplitude-detection method has the drawback of not detecting correctly packets sent with an OOK, since this modulation uses sequences of periods with and without emissions to code zeros or ones. More precisely, a period without emission and the "emission" of a sequence of zeros are similar, making it difficult to detect when a packet begins or ends with amplitude only. Thus, we added an OOK-specific workflow, based on a "fusion" algorithm, that builds aggregates of packets previously detected based on their amplitudes that are near enough. More precisely, series of packets that are separated by periods of time of the same order of magnitude as the length of individual packets are "fused" together as a new packet. Note that the pre-fusion packets are still present, the new "fused" packet being treated as a new one altogether. This step is represented in Figure 1 by the **Packet reconstruction** sub-block in the OOK-specific processing.

To estimate the symbol rate in this case, for a given packet, we estimate the minimal length of an emission or non-emission period. This allows us to demodulate the packet by estimating, for each of those periods, the number of times they contain the shortest one. However, this method relies on the assumption that the packet contains at least an isolated "1" or "0" bit, meaning the shortest period divides all others. To be able to demodulate packets that do not meet this condition, we also test integer divisors of the minimal size. Then, using a score on symbol length stability (meaning how much the lengths deviate from multiples of the estimated symbol length), we choose the most probable option.

### 3.3 Link Layer analysis

The Link Layer analysis was also split in two parts : one for analyzing unknown protocols to extract information from the packet format, the other to dissect packets according to known protocols that correspond to the received bitstream.

To facilitate the addition of new protocols, the known protocols are regrouped in a dictionary. Each element representing a protocol is a set of functions allowing to identify whether a binary stream corresponds or not to this protocol.

First, the received bitstream is tested against a dictionary of known protocols, by trying to dissect the data according to the protocol's specification, and then evaluating if the packet is valid. Note that, by default, the protocols are chosen depending on the modulation they should use. However, it is also possible to

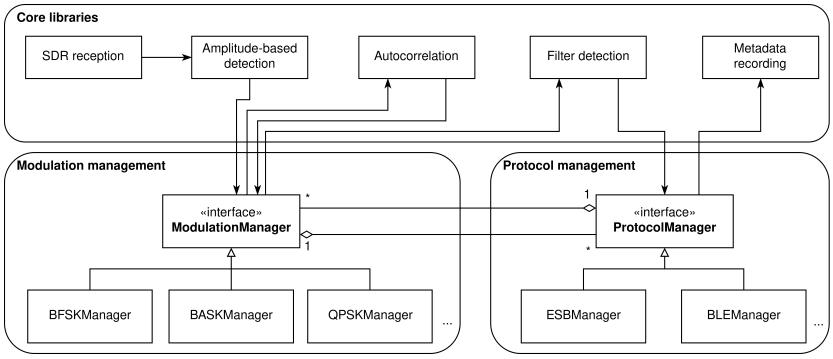



Fig. 3: Tool architecture overview

test any protocol for any modulation, in case an attacker would use a protocol with an unintended modulation.

Then, the bitstream is analysed to try to find fields of interest. To do so, we first need to identify fields that are at different positions due to variable-length fields. This was done using the Needleman-Wunsch alignment algorithm, that aligns similar fields, as used in some works of the state of the art such as Netzob [22], a tool to assist in simple protocol reverse engineering.

Once fields are aligned, we compute their entropy to put them in three classes:

- **fixed-value fields**, that could correspond to addresses
- **low-entropy variable fields**, for instance length fields or unencrypted data
- **high-entropy variable fields**, or **pseudo-random fields**, that could correspond to encrypted data

Finally, variable length fields with low entropy are tested against different heuristics to identify more precisely their types. In the current state of our works, two heuristics are tested : correlation between fields and packet length to identify length fields, and linear evolution of a given field for sequence numbers.

## 4 Implementation

We implemented the approach into a new tool, mainly developed in C and Python, and based on Scapy[4] and the SoapySDR[5] library. It consists of three main components: a set of core libraries, a modulation dictionary and a protocol dictionary for each modulation[6]. This architecture is represented in Figure 3. Please note that this illustration doesn't cover the particular ASK-OOK processing, which is implemented via a set of independent methods in the tool.

[4] https://scapy.net/

[5] https://github.com/pothosware/SoapySDR

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

### 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 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 estimated symbol period 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 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 4.3) that are linked with the current Modulation Manager, along with the analysis for eventual unknown protocols.

Once the Protocol Managers have analyzed the binary stream, all the extracted metadata is then saved by the *Metadata recording* block in a structure for further use by different applications, such as the one presented in 5.3.

### 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 result from `get_mod` on two opposite symbols to discern the moment of decision in eye-diagrams

All Modulation Managers must then be added to the modulation dictionary of the tool to be used by the core libraries. As of now, only binary Frequency Shift

Keying and Amplitude Shift Keying are implemented in the tool. However, it is possible to easily add new modulations by implementing those four functions.

### 4.3 Protocol Managers

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. Two methods must be implemented:

- `get_label`: a function similar to the one for Modulation Managers, that also needs to return a uniquely identifying string for the protocol
- `check_protocol`: a function checking whether a valid frame for the protocol is present in the binary stream

More precisely, the `check_protocol` function 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 class that corresponds to the frame type to compute later the corresponding scapy packet, and the truncated binary stream. If not, it returns a boolean with the value False and two empty values (*None* in Python).

For each Modulation Manager, we associate corresponding protocols[7] 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 is then augmented with metadata returned by `check_protocol`. 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.

As for the Modulation Managers, this dictionnary is user defined and must be completed beforehand by implementing the needed ProtocolManagers.

### 4.4 Link Layer analysis

In addition to the previously mentioned Protocol Managers, all frames identified for any given modulation are analyzed following the workflow described in 3.3. The resulting assumptions on the different fields are then also saved as metadata linked to the corresponding signal.

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

## 5 Experiments

This section describes the experiments conducted to validate our signal analysis approach, and to implement basic defensive algorithms based on the results. The source of our tool, along with the code for the experiments, is openly available on https://gitlab.laas.fr/fgaltier/wireless-analysis-toolset.

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.

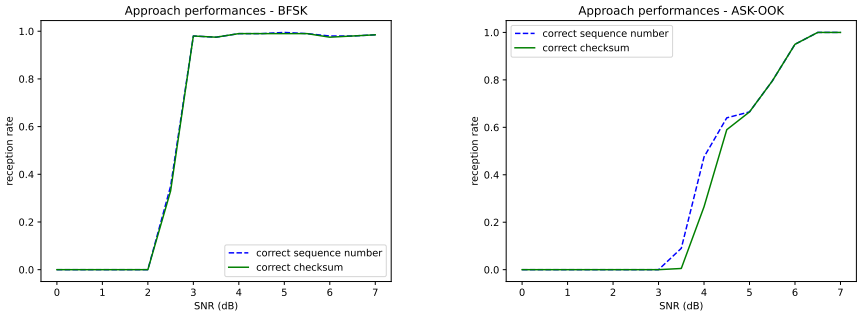
All our experiments were conducted using an Ettus USRP B200mini as receiver, and when needed, we used a LimeSDR USB to emit the signals.

### 5.1 Validation on known protocols

We first tested our approach against well known wireless protocols: 1) Bluetooth Low Energy[8], or BLE, often used by sensors or for short wireless commands, and integrated in most modern Bluetooth chips, and 2) Enhanced ShockBurst[9], or ESB, used by several wireless keyboards. We also tested it on IEEE 802.15.4 communications in 2.4GHz O-QPSK, using a GFSK demodulator to simplify the approach. Indeed, as it was shown in various works [26, 27], there is a functional equivalence between this version of O-QPSK and a GFSK, requiring minor changes on the demodulation process. These protocols were chosen because of the availability of their modulation parameters, both using GFSK in the 2.4GHz ISM band. In fact, this combination of band and modulation is one of the most frequent in the IoT. The goal of this experiment is to validate whether our approach is able to detect correctly emissions from well-known and widespread protocols, without significant performance degradation compared to a traditional receiver. As such, we compare the number of frames received correctly by our approach and by a traditional receiver, and evaluate the errors in the estimation of the different parameters of the emission.

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 10 seconds. Note that, even if the 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 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 collected several hundreds of BLE signals from the everyday environment

[8] https://www.bluetooth.com/specifications/specs/core-specification-5-4/
[9] As described in nRF24L01+ Product Specification v1.0




(a) Performances of our approach on a 2-FSK

(b) Performances of our approach on a 2-ASK (OOK)

Fig. 4: Performances versus SNR in 2-FSK and 2-ASK

of our laboratory, including phones, computers, and some smaller devices such as smartwatches. The ESB experiments were conducted only with a dongle and a corresponding mouse, the devices using this protocol being limited to mice, keyboards and dongles, and also included several hundred received frames. The IEEE 802.15.4 communications were generated using ZigBee devices, as ZigBee uses this protocol as MAC layer. We used a XBee device, a Philips Hue Bridge and a Philips Hue lightbulb. The ratio of frames correctly decoded by our approach over the frames correctly decoded by the hardcoded receiver is 99.7% (955 correctly received with our approach versus 958 with the standard demodulator) for BLE and 98.8% for ESB (89 correctly received with our approach versus 90 with the standard demodulator). For ZigBee, we compared the results of our approach with a GFSK demodulator with hard-coded carrier frequency and binary rate. In this case, we received 85 out of 85 frames with our approach. These results show the relevance of our approach: the demodulation performances on known protocols are very close to those with hardcoded values.

### 5.2 Validation on random protocols

We then tested our approach on unknown protocols, generated with random modulation parameters. To test the efficiency of our approach on different modulations, we chose the BFSK and ASK-OOK modulations. The goal of this experiment is to evaluate, on emissions with non-standard parameters, whether our approach would still be able to detect and analyze the received frames.

This was done in two steps :

1. testing the approach on simulated emissions with controlled noise, to estimate our sensitivity to the Signal to Noise Ratio (SNR)
2. testing the approach on real signals with randomly chosen parameters, to estimate the reception quality in more realistic conditions

Table 1: Results of our approach for randomly generated modulation parameters

| | FSK | | ASK | |
|---|---|---|---|---|
| | Average | Standard deviation | Average | Standard deviation |
| **reception accuracy** | 94.67% | / | 96.14% | / |
| **frequency error** | 14.1 kHz | 286 Hz | 9.93 kHz | 116 Hz |
| **binary rate error** | 0.227 bits/s | 1.04 bits/s | 919 bits/s | 808 bits/s |

In the case of the simulated emissions, we ran our algorithm on 10 samples of 100 packets for each modulation, each time with SNR values between 0 and 7 dB. For both modulations, the packets used central frequencies randomly chosen between 433 and 435MHz, and the simulated reception was made at 434MHz with a sample rate of 4Msps. The BFSK packets were using between 4 and 16 samples per symbol, and had a modulation index between 0.4 and 0.6. The ASK-OOK packets were using between 33 and 64 samples per symbol. The results are shown on Figures 4a and 4b.

In the case of real-life emissions, we sent packets from a YARD Stick One (radio dongle with pre-implemented BFSK and ASK-OOK modulations) and received them with an USRP B200 mini. The performances of our approach are shown in Table 1. We included a checksum in the payload of the protocol, and used it to calculate the reception accuracy by comparing the number of packets with a correct checksum with the number of packets actually received (estimated from sequence numbers in the first and last received packets). In the case of ASK-OOK, we received 281 packets, among which 274 with a valid checksum. In the case of BFSK, we received 242 packets, among which 231 with a valid checksum.

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

- **reception accuracy :** proportion of frames with valid checksum
- **frequency error :** error on the carrier estimation
- **bit rate error :** error on the bit-rate estimation

Overall, the results show a very good reception accuracy of our approach for both frequency and amplitude modulations. The frequency error and bit rate error are also very good since their value is at the maximum several kHz for the frequency and a few bits per second for the bit rate, whereas the bandwidth and bitrate are in the MHz range. 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 do not generate significant bitflips.

```
#status         type           label   freq                                    bitrate
AUTHORIZED      Modulation     2ASK    *                                       *
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     *                                       *
```

Fig. 5: Allowed and forbidden communication flows

### 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 monitoring system listening on a large frequency band, 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 can even be deployed by compromising for instance an employee's smartwatch to make it exfiltrate sensitive data.

This experiment was carried out in a realistic office environment, with numerous legitimate communications in BLE and WiFi. We did not take any particular measure to reduce the noise implied by those communications, so that the experiment took place in real-life conditions.

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. 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, either raising an alert for all unknown packets or for none of them.

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 768 ESB frames that were sent, 659 frames raised alerts, thus with a detection rate of 85.8%. No false positives were detected during this experiment. 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 providing the demodulated binary streams corresponding to the frames transmitted, it can also be used for simultaneous detection of malicious payloads in various protocols.

### 5.4 Case study - Wireless 2.4GHz mouse

To illustrate the full workflow of our tool, this section presents the analysis of a wireless mouse using a proprietary protocol. We used our tool to 1) identify the

modulation of the physical layer, and 2) reverse the resulting binary streams. This experiment was carried out by making the mouse communicate with a computer while performing different actions. We then captured the mouse's emissions using an USRP B200, and transmitted them to our tool.

First, we looked at several frequencies of the 2.4GHz band to detect emissions from the mouse, in a realistic environment. Doing so, we identified several frequencies on which the dongle associated with the mouse was emitting, notably 2400MHz and 2480MHz. The mouse chooses a channel among those each time it is turned on. Our tool quickly identified that the modulation in use was a GFSK, at 1Mbps, allowing it to extract the corresponding binary streams.

The entropy analysis performed using our tool allowed us to identify that all packets followed a same format : first, a series of `0xAA` bytes, followed by four fixed bytes, a variable quantity of variable fields, and finally a postamble of `0xFF`. We hypothesized that the fixed field at the beginning corresponds to an address. Then, by observing that the two last bytes before the postamble had an entropy reflecting the entropy of the previous variable fields, we assumed it to be a CRC. Our tool also identified that the packet did not contain a size field.

Finally, some of the longest packets contained numerous `0x5A` bytes, or near values. This could highlight the use of whitening, a common practice in wireless communications consisting in masking the data with a xor to even out the distribution of 1 and 0 bits in the packet. However, we could not immediately check the integrity of our reception, to help validate our hypotheses on valid packets, as we could not find the exact parameters of the CRC algorithm.

In parallel, we found that the dongle was exposed as a `MosArt Semiconductor Corp. 2.4G INPUT DEVICE`, indicating that the device might use the "MosArt" protocol identified by M. Newlin in MOUSEJACK [28]. Using the reversed parameters, we could validate the use of whitening with a repeated `0x5A` byte, along with the use of a `XMODEM-16` CRC. We also validated that the format did not contain any size field, as our tool indicated. This also allowed us to find out some differences between our mouse's protocol and the one reverse-enginered.

Thanks to our entropy analysis, we identified the frame sequences corresponding to mouse movement and scrolling. Moreover, it allowed us to detect, depending on the input, which field corresponded to which action. Interestingly, this showed us that the payload of movement frames was different from MOUSEJACK version. Indeed, the movement coordinates were alternating between X-coordinates and Y-coordinates bytes instead of being two contiguous two-byte values. We also identified a seemingly empty field after the frames corresponding to scrolling, that was not described by M. Newlin.

Overall, this case study shows the capacity of our tool to extract information from unknown signals, to help reverse-engineering unknown protocols. It validates the results from the previous experiments, showing the capability of our tool to help analyzing devices. However, it also shows the need for further work in the higher-layer analysis, notably on the detection and reverse of CRC. To do so, we plan to integrate into our tool an error-tolerant variant of CRC RevEng[29], a CRC algorithm finder. This could allow to automatically, with

an heuristic to find CRC fields, or semi-automatically, by specifying the field manually, identify the CRC parameters.

## 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 [30], which separates them in four main categories: identification based on the inter-frame spacing, with the help of the cyclic prefix [31] or not [32], techniques modifying the emitters to add detectable information [33], and techniques based on the pilot symbols used by the scheme [34]. However, since our objective is to analyze 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 parameters. 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, and thus an assumption on the transmitted data. To our knowledge, none of the existing approaches allow 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.

Some other schemes, such as Chirp Shift Keying (CSK), used in LoRaWAN, do not directly transmit symbols as values of specific parameters of the signal. For example, this modulation uses “chirps”, symbols where the signal transitions from a frequency to another, and use temporally shifted versions of those chirps to modulate data (the shift corresponding to the value to modulate). In the case of those modulations, it is then needed to add a translation layer to the ModulationManager. In the example of CSK, it would amount to translating the signal into an array containing at each instant the corresponding CSK symbol if the chirp began at this instant. We are still integrating such modulations, and building the corresponding translation layers, in particular for CSK.

Let us also note that frequency hopping schemes are currently supported, but only as independent frames. Indeed, we did not integrate any means of identifying whole communications, so each one is analyzed on its own. We plan to add such functionality in the future, but it is not currently under development.

Finally, it is to note that our approach focuses on modulations carrying digital data, as its goal is to translate a signal into corresponding binary streams. It is therefore unsuitable for detecting transmissions using analog data.

We also identified several other 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 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 then send to the computer host the already isolated frames, reducing the risks of overflows in the communication buffers between the two. Note also that this approach is not designed to run 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.

Our reverse engineering module is, for now, limited to a few specific kinds of fields, but could be augmented with more heuristics to detect other fields of interest, in particular frequent header fields or CRC fields.

## 7 Conclusion

In this paper, we presented a tool and experiments to fill the gap in the current state of the art of wireless protocol auditing and wireless covert channel attack detection, by implementing a method for detecting and analysing the physical layer of such wireless emissions. The different experiments we carried out show that this tool is able to successfully retrieve data from known protocols, and to efficiently detect the parameters of randomly generated ones. Furthermore, this tool also allows to reverse engineer the extracted data.

This approach constitutes an important step towards the design of a unified means for reverse-engineering the various and heterogeneous protocols that can be used in the domain of IoT. Indeed, as of now, we lack such a tool that would allow saving substantial amounts of time in the audit of unknown wireless devices, and analysis of unknown emissions in a monitored environment.

For future work, we plan to investigate several topics. Regarding the physical analysis, we plan to add a PSK analysis to complement the ASK et FSK. We also plan to work on the extension of the tool, by enriching the reverse engineering capabilities from the protocol analysis point of view.

## Acknowledgements

This work has been partially supported by the French National Research Agency under the France 2030 label (Superviz ANR-22-PECY-0008). The views reflected herein do not necessarily reflect the opinion of the French government.

## References


1. A. Li, C. Dong, S. Tang, F. Wu, C. Tian, B. Tao, and H. Wang, "Demodulation-free protocol identification in heterogeneous wireless networks," *Computer Communications*, vol. 55, pp. 102–111, Jan. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0140366414003041
2. K. Lakshminarayanan, S. Sapra, S. Seshan, and P. Steenkiste, "Rfdump: an architecture for monitoring the wireless ether," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009, pp. 253–264.
3. Q. Chen, Y. Wang, and C. W. Bostian, "Universal classifier synchronizer demodulator," in *2008 IEEE International Performance Computing and Communications Conference (IPCCC)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2008. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/PCCC.2008.4745076
4. D. Mototolea, R. Youssef, E. Radoi, and I. Nicolaescu, "Non-cooperative low-complexity detection approach for fhss-gfsk drone control signals," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 401–412, 2020.
5. A. W. Azim, S. S. Khalid, and S. Abrar, "Modulation classification based on modified kolmogorov-smirnov test," in *2013 IEEE 9th International Conference on Emerging Technologies (ICET)*. IEEE, 2013, pp. 1–6.
6. S. Hao, N. Wang, R. Sun, M. Liu, M. Dong, and H. Wang, "Modulation classification using a goodness of fit test," in *Journal of Physics: Conference Series*, vol. 1169, no. 1. IOP Publishing, 2019, p. 012068.
7. É. Helluy-Lafont, A. Boé, G. Grimaud, and M. Hauspie, "Bluetooth devices fingerprinting using low cost sdr," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2020, pp. 289–294.
8. P.-F. Gimenez, J. Roux, E. Alata, G. Auriol, M. Kaâniche, and V. Nicomette, "Rids: Radio intrusion detection and diagnosis system for wireless communications in smart environment," *ACM Transactions on Cyber-Physical Systems*, vol. 5, no. 3, pp. 1–1, 2021.
9. X. Liu, D. Yang, and A. El Gamal, "Deep neural network architectures for modulation classification," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 915–919.
10. Z. Zhang, C. Wang, C. Gan, S. Sun, and M. Wang, "Automatic modulation classification using convolutional neural network with features fusion of spwvd and bjd," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 3, pp. 469–478, 2019.
11. S. Peng, H. Jiang, H. Wang, H. S. Alwageed, Y. Zhou, M. M. Sebdani, and Y. dong Yao, "Modulation classification based on signal constellation diagrams and deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, pp. 718–727, 2019.
12. N. E. West and T. O'Shea, "Deep architectures for modulation recognition," *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 1–6, 2017.
13. T. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, pp. 168–179, 2017.
14. J. Duchêne, C. Le Guernic, E. Alata, V. Nicomette, and M. Kaâniche, "State of the art of network protocol reverse engineering tools," *Journal of Computer Virology and Hacking Techniques*, vol. 14, no. 1, pp. 53–68, 2018.

15. M. A. Beddoe, "Network protocol analysis using bioinformatics algorithms," *Toorcon*, vol. 26, no. 6, pp. 1095–1098, 2004.
16. S. B. Needleman and C. D. Wunsch, "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, pp. 443–453, 1970. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022283670900574
17. R. R. Sokal and C. D. Michener, "A statistical method for evaluating systematic relationships." *Univ. Kansas, Sci. Bull.*, vol. 38, pp. 1409–1438, 1958.
18. M. Nei, F. Tajima, and Y. Tateno, "Accuracy of estimated phylogenetic trees from molecular data," *Journal of molecular evolution*, vol. 19, no. 2, pp. 153–170, 1983.
19. W. Cui, J. Kannan, and H. J. Wang, "Discoverer: Automatic protocol reverse engineering from network traces." in *USENIX Security Symposium*, 2007, pp. 1–14.
20. J. Antunes, N. Neves, and P. Verissimo, "Reverx: Reverse engineering of protocols," 2011.
21. W. Cui, V. Paxson, N. Weaver, and R. H. Katz, "Protocol-independent adaptive replay of application dialog." in *NDSS*, 2006.
22. G. Bossert, F. Guihéry, G. Hiet *et al.*, "Netzob github repository," 2012. [Online]. Available: https://github.com/netzob/netzob/tree/master/netzob
23. D. Angluin, "Learning regular sets from queries and counterexamples," *Information and computation*, vol. 75, no. 2, pp. 87–106, 1987.
24. P. Cohen, N. Adams, and B. Heeringa, "Voting experts: An unsupervised algorithm for segmenting sequences," *Intelligent Data Analysis*, vol. 11, no. 6, pp. 607–625, 2007.
25. X. Wang, K. Lv, and B. Li, "Ipart: an automatic protocol reverse engineering tool based on global voting expert for industrial protocols," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 35, no. 3, pp. 376–395, 2020.
26. R. Cayre, F. Galtier, G. Auriol, V. Nicomette, M. Kaâniche, and G. Marconato, "Wazabee: attacking zigbee networks by diverting bluetooth low energy chips," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2021, pp. 376–387.
27. S. Pasupathy, "Minimum shift keying: A spectrally efficient modulation," *IEEE Communications Magazine*, vol. 17, no. 4, pp. 14–22, 1979.
28. M. Newlin, "Mousejack, keysniffer and beyond: Keystroke sniffing and injection vulnerabilities in 2.4 ghz wireless mice and keyboards," *DEFCON*, 2016.
29. G. Cook, "Crc reveng website." [Online]. Available: https://reveng.sourceforge.io/
30. M.-R. Oularbi, S. Gazor, S. Houcke, and A. Aissa-El-Bey, "Ofdm system identification using pilot tone signature," in *International Workshop on Systems, Signal Processing and their Applications, WOSSPA*. IEEE, 2011, pp. 95–98.
31. M. Oner and F. Jondral, "On the extraction of the channel allocation information in spectrum pooling systems," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 3, pp. 558–565, 2007.
32. A. Bouzegzi, P. Ciblat, and P. Jallon, "New algorithms for blind recognition of ofdm based systems," *Signal Processing*, vol. 90, no. 3, pp. 900–913, 2010.
33. P. D. Sutton, K. E. Nolan, and L. E. Doyle, "Cyclostationary signatures in practical cognitive radio applications," *IEEE Journal on selected areas in Communications*, vol. 26, no. 1, pp. 13–24, 2008.
34. F.-X. Socheleau, S. Houcke, P. Ciblat, and A. Aïssa-El-Bey, "Cognitive ofdm system detection using pilot tones second and third-order cyclostationarity," *Signal processing*, vol. 91, no. 2, pp. 252–268, 2011.