UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

Dottorato di Ricerca in Ingegneria dell'Informazione

ING-INF/03 – Telecomunicazioni

CICLO XXXV

Novel Perspectives on Physical Layer Security and Privacy in Wireless Communications

Candidato:
Dott. Marco Cominelli

Supervisore:
Ch.mo Prof. Francesco Gringoli

A Giulia

Abstract

(Italian)

In ogni ambito delle telecomunicazioni, sicurezza e privacy sono temi fondamentali la cui definizione evolve di pari passo con lo sviluppo della tecnologia. Nel caso specifico delle comunicazioni wireless, nuove tecniche di analisi del segnale radio (spesso supportate da piattaforme di calcolo parallelo) dimostrano che anche sistemi ritenuti "sicuri" in senso classico possono in realtà rivelare informazioni sensibili come l'identità o la posizione degli utenti. In questa tesi presentiamo alcuni attacchi che minano la privacy degli utenti sfruttando diverse vulnerabilità dei sistemi Bluetooth e Wi-Fi.

Nella prima parte, ci concentriamo sulla sicurezza della tecnologia Bluetooth nelle sue due varianti Classic e Low Energy. Dimostriamo in particolare che i dispositivi Bluetooth non sono in grado di preservare l'anonimità degli utenti. Infatti, utilizzando una o più software-defined radio relativamente economiche è possibile catturare tutto il traffico Bluetooth in una determinata zona e monitorare la presenza o il passaggio di dispositivi Bluetooth. È inoltre possibile re-identificare gli stessi dispositivi anche in giorni diversi, tracciando di fatto statistiche sugli spostamenti giornalieri di ogni persona. Per il caso specifico di Bluetooth Low Energy presentiamo infine l'implementazione di un sistema che, sfruttando la potenza di una scheda grafica (GPU), è in grado di effettuare attacchi di questo tipo in tempo reale.

La seconda parte del lavoro si focalizza invece su tecniche di *Wi-Fi sensing*, ovvero su innovative tecniche di analisi dell'ambiente basate sullo studio della propagazione dei segnali Wi-Fi a livello fisico. Riconosciamo che tecniche di Wi-Fi sensing potrebbero essere usate come base per sofisticati attacchi contro la privacy, per esempio consentendo di localizzare una persona in una stanza senza che abbia con sé alcun dispositivo elettronico. In questo lavoro non siamo interessati a studiare l'accuratezza di tecniche di Wi-Fi sensing, bensì a come impedirle quando possono violare la privacy di persone ignare, senza tuttavia interrompere le trasmissioni Wi-Fi legittime. Consideriamo quindi due scenari in cui si cerca di localizzare una persona in una stanza e proponiamo possibili contromisure basate su opportune manipolazioni dei segnali trasmessi oppure sull'impiego di superfici smart, o riconfigurabili.

Tutti i risultati presentati in questa tesi sono stati ottenuti con esperimenti effettuati sia con dispositivi commerciali che con prototipi basati su software-defined radio, senza utilizzare hardware altamente specializzato che potrebbe limitare la riproducibilità degli esperimenti proposti.

Abstract

Advanced analysis of wireless signals can tell much more than one would expect. Even systems deemed "secure" in the classical sense might reveal sensitive information, such as users' identities or current locations. The computing capability of modern platforms is enabling applications that have long been considered impractical or computationally infeasible. In this dissertation, we show several classes of attacks that, exploiting different vulnerabilities of Bluetooth and Wi-Fi systems, can undermine people's privacy through device re-identification and device-free localization.

In the first part of this work, we focus our attention on Bluetooth. In particular, we prove that both Classic Bluetooth and the more recent Bluetooth Low Energy fail to preserve users' privacy. Using affordable software-defined radios, we implement a system that can snoop on all the Bluetooth traffic in an area and efficiently scan for the presence of Bluetooth devices. We show that it is possible to re-identify the same devices over many days and derive the commuting patterns of target victims, e.g., by placing the system at a checkpoint. Furthermore, we show the same system can monitor Bluetooth Low Energy devices in real-time when computation-intensive tasks are offloaded to a general-purpose graphics processing unit (GPU).

We then shift our attention to Wi-Fi, and in particular to *Wi-Fi sensing*, i.e., a novel set of techniques that can derive information about the surrounding environment by analyzing the physical-layer properties of the Wi-Fi signals. Device-free localization is an example of a Wi-Fi sensing technique threatening users' privacy because it does not require the victim to carry any device with her, and it cannot be easily detected. In this work, however, we prove there are ways to disrupt privacy attacks based on Wi-Fi sensing techniques without hampering legitimate Wi-Fi communications. We consider two scenarios where an attacker wants to localize a victim inside a room and propose different countermeasures. In particular, we will show the impact of intentional distortions applied to the Wi-Fi signal by the transmitter and the effect of reconfigurable smart surfaces both on the accuracy of Wi-Fi sensing techniques and the data throughput.

Our discussion is supported by empirical results obtained using both commercial devices and software-defined radios, without the need for other specialized hardware that may limit the replicability of our experiments.

List of publications

[P1] M. Cominelli, P. Patras, and F. Gringoli. "Dead on arrival: An empirical study of the Bluetooth 5.1 positioning system." Proceedings of the 13th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH). ACM, 2019.

[P2] M. Cominelli, F. Gringoli, P. Patras, M. Lind, and G. Noubir. "Even black cats cannot stay hidden in the dark: full-band de-anonymization of Bluetooth classic devices." Proceedings of the 41st IEEE Symposium on Security and Privacy (SP). IEEE, 2020.

[P3] M. Cominelli, P. Patras, and F. Gringoli. "One GPU to snoop them all: a full-band Bluetooth Low Energy sniffer." Proceedings of the 18th Mediterranean Communication and Computer Networking Conference (MedComNet). IEEE, 2020.

[P4] M. Cominelli, F. Kosterhon, F. Gringoli, R. Lo Cigno, and A. Asadi. "An experimental study of CSI management to preserve location privacy." Proceedings of the 14th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH). ACM, 2020.

[P5] M. Cominelli, F. Gringoli, and R. Lo Cigno. "Passive device-free multi-point CSI localization and its obfuscation with randomized filtering." Proceedings of the 19th Mediterranean Communication and Computer Networking Conference (MedComNet). IEEE, 2021.

[P6] F. Gringoli, M. Cominelli, A. Blanco, and J. Widmer "AX-CSI: Enabling CSI extraction on commercial 802.11ax Wi-Fi platforms." Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization (WiNTECH). ACM, 2022.

[P7] M. Cominelli, F. Gringoli, and R. Lo Cigno. "Non-intrusive Wi-Fi CSI obfuscation against active localization attacks." Proceedings of the 16th IEEE Annual Conference on Wireless On-demand Network Systems and Services Conference (WONS). IEEE, 2021.

[P8] M. Cominelli, F. Kosterhon, F. Gringoli, R. Lo Cigno, and A. Asadi. "IEEE 802.11 CSI randomization to preserve location privacy: An empirical evaluation in different scenarios." Computer Networks 191, 107970. Elsevier, 2021.

[P9] A. Blanco, J. Palacios, M. Cominelli, F. Gringoli, J. Widmer. "Accurate ubiquitous localization with off-the-shelf IEEE 802.11 ac devices." Proceedings of the 19th ACM Annual International Conference on Mobile Systems, Applications, and Services (MobiSys). ACM, 2021.

[P10] M. Cominelli, F. Gringoli, and R. Lo Cigno. "AntiSense: Standard-compliant CSI obfuscation against unauthorized Wi-Fi sensing." Computer Communications 185, 92-103. Elsevier, 2021

[P11] M. Cominelli, F. Gringoli, and R. Lo Cigno. "On the properties of device-free multi-point CSI localization and its obfuscation." Computer Communications 189, 67-78. Elsevier, 2021

[P12] L. Ghiro, M. Cominelli, F. Gringoli, and R. Lo Cigno. "On the Implementation of Location Obfuscation in openwifi and Its Performance." Proceedings of the 20th Mediterranean Communication and Computer Networking Conference (MedComNet). IEEE, 2022.

[P13] R. Lo Cigno, F. Gringoli, M. Cominelli, and L. Ghiro, "Integrating CSI Sensing in Wireless Networks: Challenges to Privacy and Countermeasures." IEEE Network 36-4, 174-180. IEEE, 2022.

This list comprises several works whose results are not included in this dissertation because they are outside our main research line. However, we briefly report the contribution of such works here for completeness.

The work in [P1] is a study that characterizes the Direction Finding features (particularly the Angle of Arrival method, or AoA) introduced in Bluetooth Low Energy 5.1 for localizing a peer Bluetooth device. We show that AoA measurements of Bluetooth systems might suffer from severe multipath in indoor environments but can generally achieve good performance outdoors, especially when combining the measurements of multiple receivers. Moreover, in [P1], we demonstrate that an adversarial user can easily advertise a fake direction of arrival by exploiting a flaw in the AoA protocol design.

In [P9], we develop an accurate device localization system based on measurable quantities like the direction of arrival, the direction of departure, and the time of flight of Wi-Fi signals. This system is implemented on commercial Wi-Fi devices and uses advanced techniques to resolve different multipath components, achieving superior performance with respect to other state-of-the-art platforms.

The work in [P6] is somehow related to the topics presented in this dissertation. Without going into much detail, in [P6], we developed a channel state information (CSI) extraction tool for IEEE 802.11ax, or Wi-Fi 6, systems (what a CSI is and why it is useful are topics that will be addressed later in this document). This tool gives access to

an unprecedented amount of data concerning the CSI. To put things in scale, our tool can extract up to 32k data points from each Wi-Fi frame, while the most common tool used nowadays can only extract 30 data points per frame [1].

Finally, in [P12], we describe how we implemented some of the results discussed in this dissertation in an FPGA prototype. The work is based on the open-source project *openwifi* (`https://github.com/open-sdr/openwifi`) and shows that the CSI obfuscation schemes we are going to discuss later in this dissertation can actually work with "real" Wi-Fi devices.

Preface

This dissertation summarizes my work as a Ph.D. student at the University of Brescia under the supervision—and with the precious collaboration—of Prof. Francesco Gringoli. The key topics of the research lie in the broad field of wireless communications, particularly on novel security and privacy aspects of wireless systems. Wireless communications profoundly changed how people share information and access services. Unfortunately, due to the intrinsic broadcast nature of wireless channels, they also expose users to many threats undermining privacy. As we show in this dissertation, advanced analysis of wireless signals can reveal sensitive information allowing users' identification and localization. Leaking information about users' identity and location is a matter of particular concern, as it could underpin more severe threats, such as tracking, identity discovery, and pinpointing home and work premises.

The research activity discussed here focuses on two wireless communication standards: Bluetooth and Wi-Fi. However, most considerations can be broadly applied to wireless communications in general, regardless of the dissimilarities between different standards. Various ways to enforce location privacy have been investigated since the early days of cellular networks. For example, GSM systems employed a Temporary Mobile Subscriber Identity (TMSI) to identify users, using procedures that later evolved into Subscription Concealed Identifiers (SUCI) in 5G cellular networks [2]. In recent years, however, the vulnerable surface for privacy attacks has expanded significantly with the pervasiveness of mobile and sensing devices, open platforms (often running untrusted code), diverse wireless connectivity options, and the availability of powerful computing systems.

This dissertation presents several scenarios concerning wireless communications in which users' privacy is at risk. In some cases, we will be interested in proving the insecurity of the systems; in other cases, we will discuss and propose different ways to mitigate possible attacks. Chapter 1 provides a general introduction to Bluetooth and Wi-Fi, focusing on the topics of interest for our work. In Chapter 2, we introduce a framework we developed for conveniently monitoring (or sniffing) all Bluetooth connections in an area. This framework is further expanded in Chapter 3 to enable the real-time operation of our Bluetooth sniffer. The results presented in Chapters 2 and 3 show that Bluetooth communications cannot safely preserve users' privacy against state-of-the-art sniffing platforms. Then, we shift our attention to Wi-Fi in Chapters 4 and 5. We will primarily deal with device-free localization using Wi-Fi sensing, i.e., with advanced

analysis techniques of the Wi-Fi signals that allow attackers to localize people in a room even if they do not carry any electronic device with them. In this case, our interest is to prevent unauthorized sensing applications effectively; hence, we devise several different techniques—depending on the assumptions we make about the attacker's capabilities—to obfuscate users' location without disrupting Wi-Fi service. Summarizing the scope of this contribution, the work presented here is an overview of novel physical-layer threats that modern wireless communications must face. For every threat, we evaluate different aspects of its severity and applicability. Finally, we validate all the threats and possible mitigations through empirical experiments that reproduce typical attack scenarios. In every chapter, we detail the hardware and software tools used to run the experiments and provide a link to the open-source code we developed to facilitate the reproducibility of the results.

Marco Cominelli
November 18, 2022

Contents

Chapter 1

Introduction

Wireless technologies are the foundation of a considerable part of the modern telecommunications ecosystem, from personal fitness devices to local access to the Internet, large-scale sensor networks, and cellular networks. Their impact is not expected to decline any time soon; smart cities and the Internet of Things (IoT) foster a societal model that gives wireless systems a unique role, entangling common physical assets with advanced communication systems. Therefore, securing wireless communications is paramount.

Traditional research in network security has been focusing on guaranteeing essential security features such as authentication, confidentiality, and integrity protection, citing a few. However, modern processing systems—usually powered by general-purpose graphics processing units (GPUs) with massive computational capabilities—facilitate novel types of security and privacy attacks. The main argument of this work is proving the possibility of jeopardizing users' security and privacy with attacks that have been rarely considered in the literature but are now worth investigating. In particular, our goal is to show that advanced analysis of communication signals may reveal sensitive information—such as the identity or the location of unaware victims—and to propose suitable countermeasures when possible. These attacks find their roots in the outstanding computational power of modern processing systems, which can collect and process unprecedented information.

In this work, we consider two wireless technologies in particular: Bluetooth and Wi-Fi. Concerning Bluetooth, we are primarily interested in showing that today is pretty easy to identify and track any Bluetooth connection due to its inadequate anonymization mechanism. Clearly, this is a severe privacy issue. In Chapter 2, we present a system that, using a combination of signal processing techniques and iterative inference, can quickly identify specific connections and enable reliable user tracking. We go even further in Chapter 3, where we shift our focus on Bluetooth Low Energy and implement a similar system that can perform the same attack in real time with the aid of a GPU. On the other hand, for what concerns Wi-Fi, we will focus on the version standardized by the IEEE 802.11ac specification [3], also known as Wi-Fi 5. In this dissertation, we are not interested in Wi-Fi as a communication standard *per se*, but as a support for an "environment sensing" framework. In particular, we will focus our attention on device-

free localization systems, i.e., systems that can localize people in indoor environments thanks to the physical interaction of human bodies with the propagating wireless signals. In Chapter 4 and Chapter 5, we will consider several privacy-preserving frameworks to hamper unauthorized sensing without disrupting legitimate Wi-Fi communications.

Nevertheless, before delving into the core part of this dissertation, we must first lay some foundations and present the related work. In this chapter, we introduce key concepts about Bluetooth and Wi-Fi technologies that will be used in the following chapters. It is worth noticing that we will omit many crucial aspects of these technologies, mainly for two reasons. First, this chapter does not mean to provide an introduction to the basic elements of Bluetooth and Wi-Fi technologies but only to essential features and tools that will support our analysis. Second, both Bluetooth and Wi-Fi specifications amount to more than 3000 pages at this point [4, 5], and a detailed description of their features is an effort that goes beyond the scope of this work. After this brief introduction, in Section 1.3, we review the existing literature on monitoring Bluetooth communications and hampering Wi-Fi sensing applications to better frame the scope of this contribution.

Research questions

To summarize the major topics of this dissertation and help the interested reader locate relevant information, we explicitly outline the key research questions addressed here.

- What kind of resources are needed to track Bluetooth devices reliably? Is it possible to improve the performance of state-of-the-art systems using general-purpose programmable hardware? (Sections 2.2, 3.1)
- While tracking Bluetooth devices, what is the tradeoff between the available computational power and the expected tracking performance? (Sections 2.5, 3.3)
- What kind of techniques can we employ to restrain privacy attacks based on Wi-Fi sensing? (Sections 4.2, 5.2)
- To what extent can we protect privacy against Wi-Fi sensing without impairing legitimate communication throughput? (Sections 4.3.3, 5.5.2)

1.1 Bluetooth

The latest version of the Bluetooth Core Specification (version 5.3) defines Bluetooth as a short-range wireless communications system whose key features are robustness, low power consumption, and low cost [4]. Today, Bluetooth is one of the most pervasive wireless technologies, embedded in almost every phone, tablet, and personal computer and in many cars, headsets, keyboards and mouses, gaming consoles, and wearable devices. Annual Bluetooth device shipments are expected to exceed 5 billion in 2022 and are projected to reach the unprecedented target of 7 billion units shipped annually by 2026 [6]. These numbers show Bluetooth's central role today, despite being more

than 25 years old. Therefore, auditing its security with respect to modern attackers' capabilities becomes a matter of utter importance not only for the research community but for the entire user base at large. In this chapter, we will briefly review the basic concepts and the general architecture of Bluetooth to set some common grounds for our subsequent analysis.

The Bluetooth Core Specification describes two different types of systems: Bluetooth Basic Rate, also called Classic Bluetooth (BT), and Bluetooth Low Energy (BLE). Despite sharing the same name and several commonalities, these two systems are not interchangeable, meaning that BT and BLE cannot communicate with each other. To overcome this issue, most devices implement both systems to maximize compatibility across the possible use cases. Today, BT remains the *de facto* standard for short-range connectivity, especially for voice and audio streaming applications for wireless headsets and car infotainment systems. On the other hand, BLE is designed for applications that require a lower data rate and minimal energy consumption. However, even if BLE mainly targets low-power devices in the IoT, it is now rapidly gaining traction as a full-fledged replacement of BT.

Both types of Bluetooth transceivers—BT and BLE—work in the unlicensed ISM band[1] at 2.4 GHz and employ a frequency-hopping spread spectrum (FHSS) technique to combat interference and fading. What differs between the two systems are the number of physical channels, the supported data rates (with optional error correction coding), the allowed type of traffic, and the channel access schemes.

Bluetooth systems consist of a *Host* and one or more *Controllers*. These are logical entities that separate the management of high-level information (done by the Host) from the control of the time-critical radio interface (done by the Controller). The Host-Controller interface (HCI) is an abstraction layer defining a standard set of commands and events to standardize serial communication between the Host and the Controller. The Host and the Controller could be implemented in the same integrated chipset. Still, the HCI ensures that single implementations of Host and Controller can work together with little to no adaptation, even if from different manufacturers. It is worth noticing that the Host is not entirely agnostic with respect to the Bluetooth radio type. Indeed, the set of services that the Host implements differ between BT and BLE, but due to the "software nature" of the Host protocol stack (meaning that it deals with high-level abstractions), the Host is usually considered to be a hardware-independent entity.

In this dissertation, we will primarily deal with the lowest layers of the Bluetooth stack, which incidentally are also the ones that differ more between BT and BLE. Thus, we find it more appropriate to introduce the two types of Bluetooth radios separately. In the following discussions, we will not explicitly mention whether something is related to the Host, the Controller, or the HCI.

[1]ISM bands are portions of the radio spectrum reserved for industrial, scientific, and medical (ISM) applications. In general, low-power radio communication devices are allowed to operate in ISM frequency bands without a license, but they must adopt some techniques to avoid or limit the interference generated by other electronic devices.

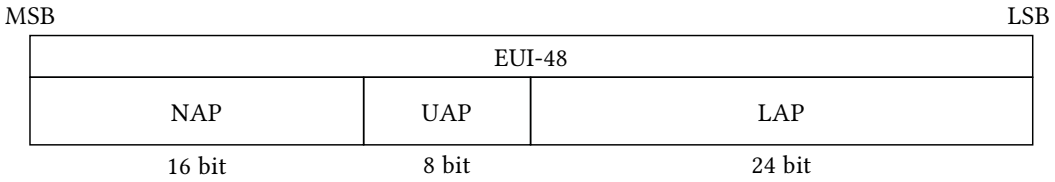


Figure 1.1: Structure of a Classic Bluetooth address BD_ADDR. Usually, the NAP and the UAP identify a specific vendor, while the LAP is assigned by the vendor itself.

1.1.1 Classic Bluetooth

BT transceivers can operate in two different modes. Basic Rate (BR) radio operation uses a binary Gaussian frequency-shift keying (GFSK) modulation to minimize the radio complexity with a symbol rate of 1 Msym/s, supporting the bit rate of 1 Mbit/s. Enhanced Data Rate (EDR) radio operation can achieve bit rates of 2 Mbit/s and 3 Mbit/s using $\pi/4$-DQPSK (differential quadrature phase-shift keying) and 8-DPSK modulations respectively. Classic Bluetooth radios are required to support at least BR operation, while EDR support is optional. In this work, we will consider BR transceivers only; therefore, every time we refer to BT radio transmissions, we are implicitly talking about BR operation.

Bluetooth Clock and Address

Every BT device has a native clock derived from a free-running reference clock. The BT clock is not only used to trigger time-critical events in BT devices but is essential for the proper functioning of BT communications, as we will see later. The BT clock is a 28-bit counter that wraps around at $2^{28} - 1$, and the least significant bit (LSB) CLK_0 ticks every 312.5 µs. The BT clock has a cycle of approximately one day, but it should be noted that it has no relation to the time of day and can be initialized to any value.

Every BT device is assigned a unique 48-bit Bluetooth device address (BD_ADDR) that is an extended unique identifier, or EUI-48.[1] As shown in Fig. 1.1, the BD_ADDR is divided into three parts: a 3-byte Lower Address Part (LAP), comprising the least significant bits of the BD_ADDR; a 2-byte non-significant Address Part (NAP), comprising the most significant bits of the BD_ADDR; and finally the Upper Address Part (UAP), which consists of the remaining byte in the BD_ADDR between the LAP and the NAP. The LAP and UAP will play a fundamental role in the following discussions; in particular, the UAP is the target of a privacy attack we describe in the next chapter.

Physical Channels

Concerning the usage of the radio spectrum, BT subdivides the ISM band at 2.4 GHz into 79 orthogonal narrow-band channels separated by 1 MHz. BT channels are accessed

[1] EUI-48 are unique identifiers administered by the IEEE Registration Authority. Manufacturers and vendors have to buy blocks of unique EUI-48 identifiers before assigning them to the devices [7].

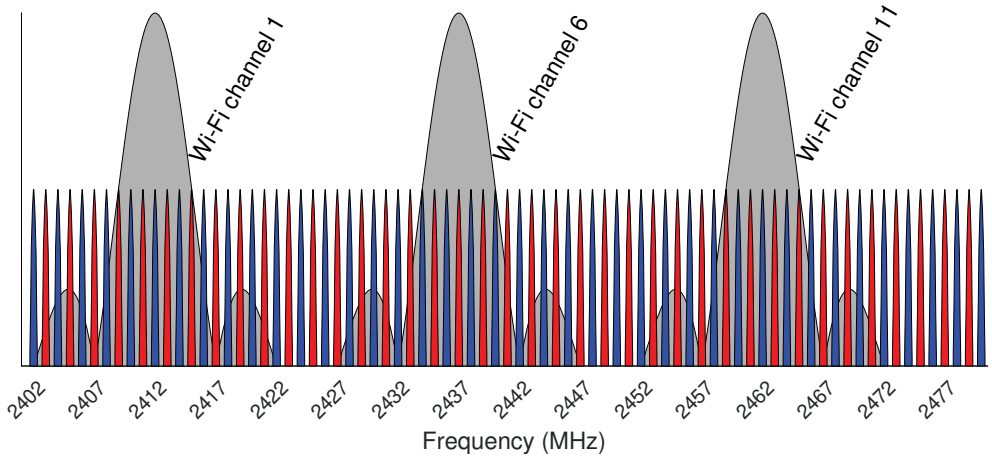


Figure 1.2: Layout of the 79 BT channels defined in the 2.4 GHz ISM band (in alternating blue/red colors), overlayed to the three most common Wi-Fi channels in the same band.

according to a pseudo-random hopping scheme to reject interference from co-located wireless standards. For example, in Fig. 1.2, we show that many BT channels overlap with the three most common Wi-Fi channels 1, 6, and 11. In case of persistent interference with the statically-allocated Wi-Fi channels, some BT channels can even be excluded altogether from the hopping scheme.

During regular operation, a physical radio channel is shared by a group of devices synchronized to a common clock and frequency hopping pattern. One device, the *Central*, provides the synchronization reference and controls the channel access; all the other devices are called *Peripherals*. The group of devices synchronized in this fashion is said to form a *piconet*. In Fig. 1.3, we show a sample piconet comprising five devices (one Central and four Peripherals). In principle, an unlimited number of Peripherals can join a piconet; however, in any piconet, there can be only seven "active" Peripherals that the Central can directly query due to design limitations. All the other Peripherals are in a "parked" state and cannot actively join the communication. The specific hopping pattern used in a piconet is determined by the Bluetooth address and clock of the Central and consists of a pseudo-random ordering of the 79 BT channels.

The basic piconet physical channels are also subdivided into time units called *slots*. Data is transmitted in packets aligned with these slots, while frequency hopping is performed between the transmission or reception of packets. Communication between the Central and the Peripherals is bi-directional and follows a time-division duplex (TDD) scheme. Since the number of BT channels is limited and there might be many piconets in a particular area, BT transmissions can *collide* if two or more piconets end up sharing the same physical channel temporarily. To avoid any cross-talking effects between different piconets, BT packets contain a sequence of bits—called Access Code—that univocally determines the target piconet. In the next section, we see how BT packets are formatted and how this Access Code is generated.

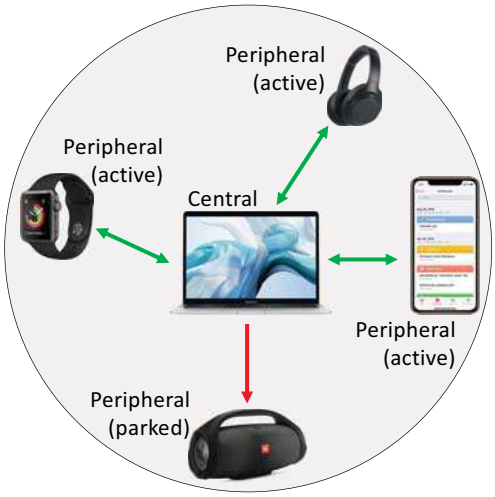


Figure 1.3: Piconet example. There is no limit to the number of devices that can join the piconet, but the Central can only address up to seven "active" Peripherals.

BT Packet Format

The structure of a generic BT packet transmitted inside a piconet is shown in Fig. 1.4. The packet starts with a fixed 4-bit preamble, followed by a 64-bit Sync Word and an 18-bit header. The payload is generally optional because some BT packets are only used for discovery and other control functions. The Sync Word is obtained starting from the 24-bit Central's LAP and appending a Barker sequence of 6 bits (`001101` or `110010`, depending on the last bit of the LAP) to improve the auto-correlation properties [8] of the Sync Word [9]. Based on the LAP and the selected Barker sequence, an expurgated (64,30) block code [9] is computed as a bit-wise XOR of a 64-bit pseudo-random sequence. Only 34 bits of the resulting codeword are kept and prepended to the LAP; this procedure preserves the LAP, while the extra coded bits guarantee a large Hamming distance between Sync Words of different piconets. In some cases, a trailer (consisting of another fixed pattern of `0`'s and `1`'s) can follow the Sync Word for improved DC compensation. The preamble, the Sync Word, and the trailer (when present) form the Access Code. As already stated, the Access Code does not only help to identify the beginning of a new BT packet; it is also crucial to determine which piconet the transmitted packet belongs to, preventing cross-talking between piconets that might accidentally end up on the same physical channel while hopping. It is essential to notice that the Access Code is transmitted *as is*, i.e., it is not subject to any further encoding. This represents a design flaw in protecting users' privacy since one part of the Central's BD_ADDR is broadcasted in the clear with every BT packet transmitted in a piconet.

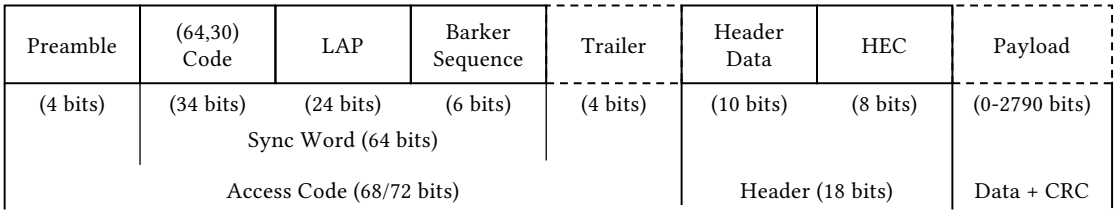


Figure 1.4: BT packet format. Only the "Payload" field can be eventually encrypted.

BT Packet Header

Not just the LAP of the Central device is transmitted inside every BT packet; the Central's UAP and clock are encoded as well in the packets exchanged in a piconet; in particular, they are used in the generation of the BT packet header.

The BT packet header in Fig. 1.4 comprises two fields: the header data, which contains 10 bits of information, and the 8-bit header error check (HEC) code. The Header Data field is divided into other subfields; specifically, 3 bits are used to address the seven possible active Peripherals in the piconet,[1] 4 bits to specify the packet type, and the last 3 bits are reserved for three 1-bit flags.

Figure 1.5 sketches the header generation process. The HEC is first generated using the linear-feedback shift register (LFSR) shown in Fig. 1.6a. The HEC LFSR has a state of 8 bits that is first initialized with the Central's UAP and then updated using the Header Data bits (starting from the LSB). The final state of the register is the HEC itself. Then, the 18-bit header resulting from the concatenation of the Header Data and the HEC is whitened[2] before transmission using the LFSR shown in Fig. 1.6b. The state of the whitening LFSR is initialized with six bits of the Central's clock (CLK_{1-6}) and the last bit (bit 6) set to `1`. The header is then whitened serially, XORing its bits with the last bit of the self-updating LFSR.

Given the Header Data content, different UAPs generate different HECs, while different master clock values produce different whitened sequences. Recovering the correct UAP and CLK is arguably computationally expensive since an exhaustive search of these values would require up to 2^{14} iterations for every eavesdropped packet. In the past, the HEC computation and the whitening procedure were expected to effectively protect the UAP against passive eavesdropping. We will debunk this assumption in Chapter 2.

Inquiry and paging procedures

The discovery of another BT device can happen through an *inquiry* procedure. BT defines 32 inquiry channels that are accessed in time slots. The procedure starts with an unconnected Peripheral entering the *inquiry scan* state. In this state, the Peripheral

[1] Address `000` is reserved for broadcast transmissions. The Central device is not addressable.

[2] Whitening is a linear transformation that decorrelates sequence of bits to improve the spectral properties of the transmit signal. Its goal is to equalize the power spectral density of the signal over the allowed frequency range, one of the distinctive properties of white noise, and hence the name.

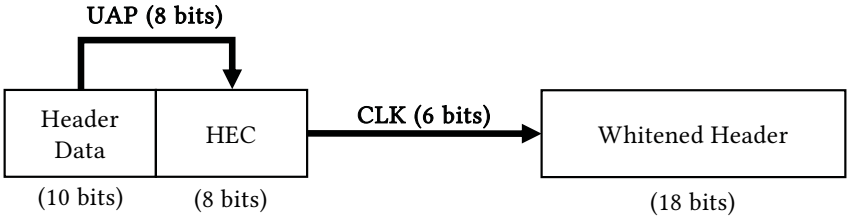
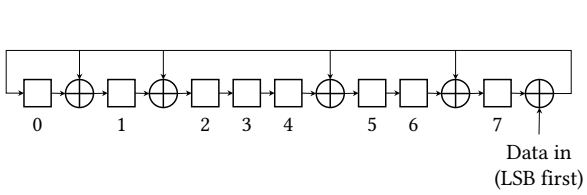
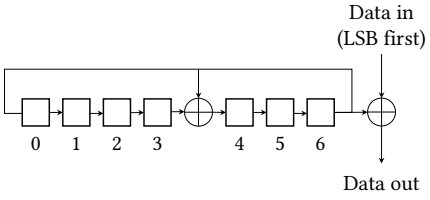


Figure 1.5: BT packet header generation process: the Central's UAP is used to derive the HEC, while the Central's clock CLK is used to whiten the header before transmission.



(a) LFSR for computing the HEC. It is initialized with the UAP and then updated using Header Data bits. The final content of the register is the HEC.



(b) LFSR for header whitening. Bits 0 to 5 are initialized with bits CLK_{1-6}; bit 6 is always initialized with 1.

Figure 1.6: LFSRs used to generate the BT packet header.

must listen on one of the 32 inquiry channels for 16 slots (10 ms) and change the inquiry channel every 2048 slots (1.28 s). A Central that wants to discover a new Peripheral, enters the *inquiry* state, selecting a subset of 16 channels out of the 32 possible inquiry channels. During every even-numbered slot, the Central transmits two packets on two different channels (in the two halves of the same slot), waiting for the response from a Peripheral in the very next slot. This is repeated for 16 slots (10 ms) until all the 16 channels are covered, and this 16-slot sequence is repeated 256 times (2.56 s) before switching to the other set of 16 inquiry channels.

A similar procedure, called *paging*, is used to establish a new BT connection. Again, the Central and the Peripheral hop on a subset of 32 BT channels with the same timing. This time, if the Peripheral responds to the message of the Central, the Central transmits a special packet (indicating the frequency hopping sequence) in the next slot and waits for an acknowledgment from the Peripheral. If this acknowledgment is received, then the two devices are connected: they start hopping using the selected hopping sequence and the Sync Word derived from the LAP of the Central. A representation of this procedure is illustrated in Fig. 1.7.

1.1.2 Bluetooth Low Energy

BLE has been introduced in the Bluetooth Core Specification starting from version 4.0. It is better suited than BT for devices with low data rate requirements and strict constraints on power consumption. Like BT, BLE operates in the unlicensed 2.4 GHz

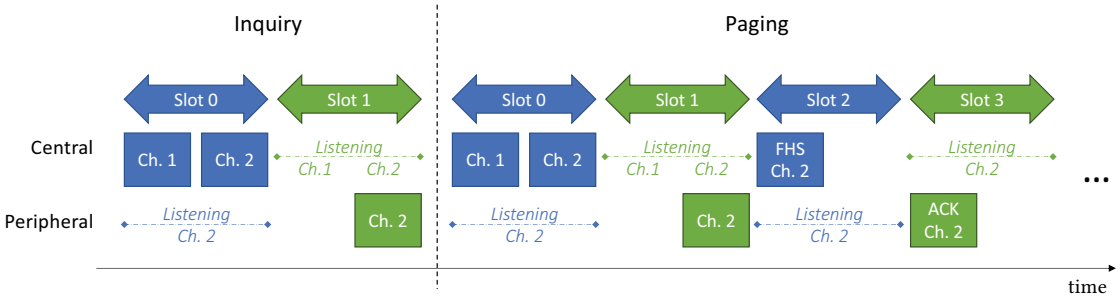


Figure 1.7: Representation of the inquiry and paging procedures in Classic Bluetooth. Inquiry is used to discover new devices, while paging is used to start a new connection.

ISM band, employs a FHSS technique for accessing the physical medium, and uses a binary GFSK modulation with different modes of operation that depend on the physical layer (PHY). The only mandatory PHY is the Uncoded LE 1M, supporting 1 Mbit/s. Optional PHYs are the Uncoded LE 2M (2 Mbit/s) and the long-range LE Coded PHYs S=2 and S=8, achieving respectively 500 kbit/s and 125 kbit/s (S indicates the number of symbols per bits).

In general, we identify the same two challenges we have already highlighted for BT when we want to snoop BLE communications: (i) the central frequency of successive transmissions is not fixed but is rapidly switched between a set of 40 narrow-band channels, according to a pseudo-random hopping sequence known only by the transmitter and the receiver; and (ii) multiple data sessions (i.e., connections between different devices) can be active at the same time on different channels. Therefore, the only way to reliably track all BLE sessions in a target area is to capture a wide-band 80 MHz signal corresponding to the entire 2.4 GHz ISM band and then recover the BLE traffic.

BLE Address

Unlike BT, BLE supports different address types that offer better privacy features. A BLE device address is a string of 48 bit and can be *public* or *random*. A public address is assigned exactly like BD_ADDR and represents a EUI-48. On the other hand, random addresses are pseudo-random sequences of bits that can be changed to ensure superior privacy, and (according to the BLE terminology) they can be either *static* or *private*. Static random addresses are random sequences of 46 bits (the two most significant bits (MSBs) must be `11`) that can be changed by the device, e.g., at every power cycle. Obviously, when a static address is changed, the address stored in peer devices is no longer valid, and the ability to recognize the device is lost. To avoid this issue, BLE devices can use private addresses, which are composed of two parts. In private addresses, half of the address is a random 24-bit number p_{rand} (with the two MSBs set to `01`) generated by the device itself, e.g., every 15 minutes. The remaining 24 bits of the address are deterministically derived from p_{rand} and a key K_I using a hashing function $h(p_{rand}, K_I)$. When two BLE devices are paired, they can exchange their respective K_I.

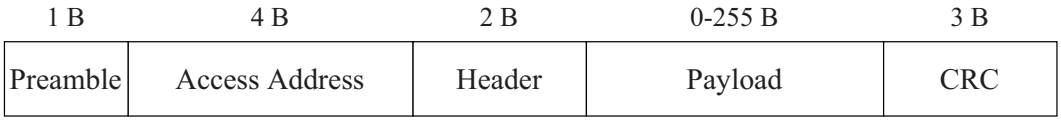


Figure 1.8: Structure of a generic BLE packet. All the fields except the preamble are whitened with a function that only depends on the BLE channel number.

In this way, even if the private address changes, a BLE device can resolve the identity of a peer device by verifying that, given the first half of the private address p'_{rand}, the second half of the address matches $h(p'_{rand}, K_I)$.

BLE Packet Format

The structure of a generic BLE packet is reported in Fig. 1.8. The BLE preamble is a short sequence of eight alternating symbols (`0`'s and `1`'s) that helps the receiver detect the beginning of a new packet. The preamble is followed by the Access Address (AA), a sequence of 32 bits used to identify a specific data session. Its function is identical to the Access Code for BT; however, unlike BT, the AA does not contain any information about the devices' addresses. In BLE, the AA can be arbitrarily chosen by the Central when establishing a new connection, and it is a pseudo-random sequence of bits. The Header field is long 2 B, the first one containing information about the packet (like the packet type or a 1-bit sequence number for data flow control) and the second one encoding the length of the payload (hence the maximum payload length is 255 B). The payload is optional and is the only field in a BLE packet that can be encrypted. Finally, the header and the (encrypted) payload are protected by a 24-bit cyclic redundancy check (CRC) code.

BLE Advertising and Connection Procedures

BLE jointly employs two multiple access schemes, frequency-division multiple access (FDMA) and a time-division multiple access (TDMA).

Concering the FDMA scheme, BLE defines 40 narrow-band channels in the 2.4 GHz band, each separated by 2 MHz. Three of them are Advertising channels used for device discovery and to broadcast information that does not require the setup of a connection. The remaining 37 BLE channels are the Data channels used by Central and Peripheral in a hopping fashion once they form a piconet (conceptually like BT, but the specific hopping algorithms differ). Data channels are numbered from 0 to 36, while channels 37, 38, and 39 are the Advertising channels. We report the layout of the BLE channels in Fig. 1.9. It is worth noticing that Advertising channels are spread throughout the entire spectrum to avoid interference with the most common Wi-Fi channels 1, 6, and 11.

Each physical BLE channel is subdivided into time units called events. Inside these events, data is exchanged between the devices in a piconet according to a TDMA scheme.

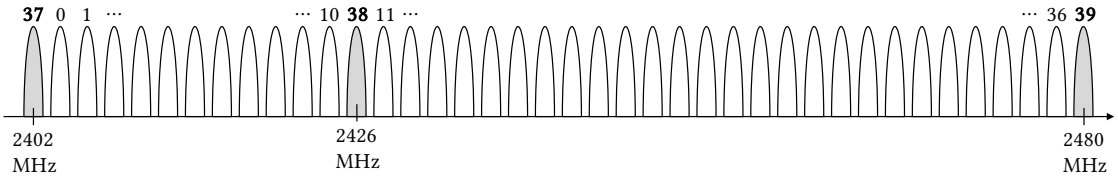


Figure 1.9: Layout of the 40 BLE channels in the ISM 2.4 GHz band. Channels highlighted in gray (37, 38, and 39) are the Advertising channels.

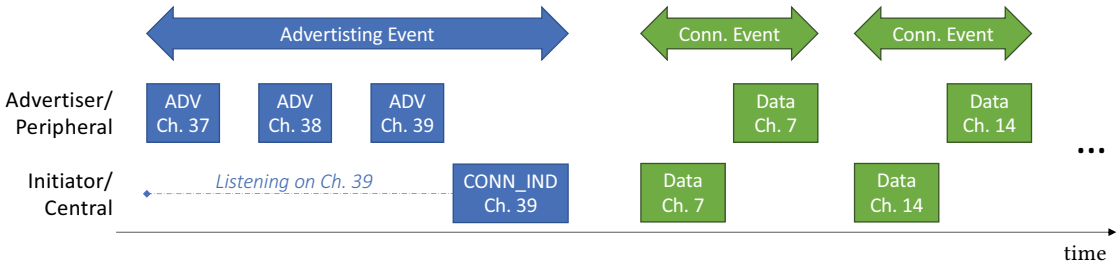


Figure 1.10: Setup of a new BLE connection. Every Advertising packet (in blue) uses the default AA (`0x8E89BED6`). The CONN_IND packet "creates" a new BLE connection identified by its own AA and hopping pattern.

Different types of events exist, the most important ones being Advertising and Connection events. During an Advertising event, one BLE device—the Advertiser—starts the event by transmitting a packet on one of the Advertising channels. Another BLE device—the Scanner—that intercepts an Advertising packet can send a request to the Advertiser on the same physical channel, which the Advertiser immediately answers. This procedure can be repeated within the same Advertising event on each of the three Advertising channels. All the transmissions on the Advertising channels use the default AA `0x8E89BED6`.

Unidirectional broadcast transmissions (from the Advertiser to one or more Scanners) could be entirely fulfilled using Advertising events only. However, BLE devices can also form a bi-directional connection following the procedure depicted in Fig. 1.10. To start a new connection, the Scanner (called Initiator in this case) sends to the Advertiser a Connection Indication (CONN_IND) packet that contains all the connection parameters. In particular, a CONN_IND specifies the AA of the connection, the hopping parameters (like the map of allowed BLE channels and the channel selection algorithm), and a shared initialization value for the CRC. From that point on, the Initiator becomes the Central and the Advertiser the Peripheral of a new piconet. They exchange data in a client-server fashion, where every packet of the Central (either a command or a request) is followed by a packet from the Peripheral (an ack or a reply). All the packets in the Connection events use the new AA and are exchanged while hopping on the available data channels. This implies the Advertiser is not "discoverable" anymore on the Advertising channels after the connection setup.

1.2 Wi-Fi Sensing

The second part of this dissertation deals with Wi-Fi technology. However, we are not interested in Wi-Fi as a communication standard here, so we will not detail many fundamental aspects of Wi-Fi communications. Of much more interest for the scope of this work, as we will see, are the physical properties of Wi-Fi signals. Today, research on *Wi-Fi sensing* is thriving both in academia and industry. Wi-Fi sensing refers to the practice of exploiting Wi-Fi signals to derive some information about the environment around Wi-Fi communication. Typical applications are motion detection [10], activity and gesture recognition [11, 12], and device-free localization [13], to name a few. Gaining some knowledge about the environment using electromagnetic waves is indeed not novel, but doing it as a side task of wireless communications is indeed both novel and compelling.

Exploiting Wi-Fi signals to "sense" the surrounding environment seems a natural choice for two reasons. First, Wi-Fi communications are ubiquitous. Nearly everyone today owns at least a Wi-Fi device, and almost every wireless local area network (WLAN) for accessing the Internet is based on Wi-Fi. Second, orthogonal frequency-division multiplexing (OFDM) systems like Wi-Fi systems have to estimate the wireless channel's characteristics to properly equalize the received signal. The estimated frequency response of the wireless channel measured at the receiver is called channel state information (CSI). The refinement of novel CSI-based equalization techniques in 802.11be (branded as Wi-Fi 7 by the Wi-Fi Alliance) should allow up to 16 spatial streams and a data rate of 46 Gbit/s [14]. However, the CSI also enables quite naturally several "sensing" applications, as it directly reflects variations in the channel's properties.

About ten years ago, researchers started looking at the CSI not only as a way to equalize the frequency-selective behavior of wideband Wi-Fi channels but also as a complex piece of information embedding "knowledge" about the physical environment through which the signal propagated. Indeed, the intricate combination of reflections and refractions of the wireless signals—known as multipath—creates unique patterns of interference that also depend on the position and the motion of objects in the environment, as we show in Fig. 1.11. Many applications were enabled by CSI-based sensing, which is often called with the evocative name (albeit technically improper) of *passive radar*. Among all possible applications, we think one of the most compelling is device-free localization. The attention to CSI-based localization was brought by early works [15–18] a decade ago, immediately proving that CSI-based localization techniques can outperform traditional received signal strength indicator (RSSI)-based methods. After these initial works, the topic flourished, with proposals exploring, e.g., the usage of massive multiple-input multiple-output (MIMO) systems [19, 20], or Bayesian estimators [21].

Recent years witnessed the explosion of artificial intelligence and machine learning techniques applied to this topic [13, 20, 22–30], which achieve astounding results using different classification or analysis techniques, often involving deep learning or reinforcement learning. None of these works have ever discussed the following questions:

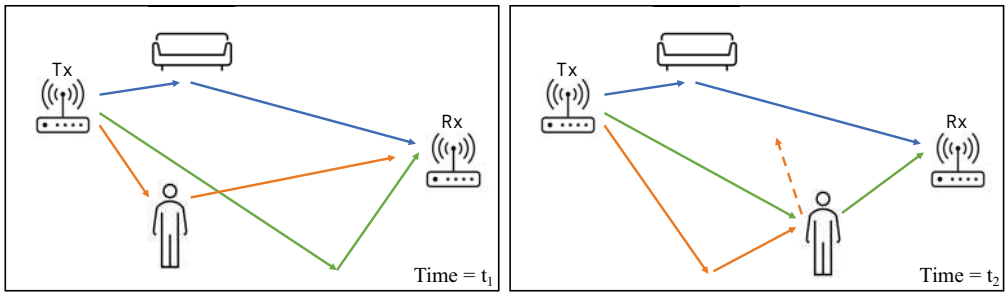


Figure 1.11: Schematic representation of the multipath effect, e.g., in a living room. As a wireless signal propagates through the environment, multiple copies of the same signal can impinge on the receiver's antenna, each with a different amplitude and phase delay. The specific interference pattern depends on the layout of the environment and the movements of people inside.

(i) how ethical or intrusive is CSI-based localization? (ii) Is it possible to prevent unauthorized use of CSI-based localization? (iii) What is the cost of communication performance we have to pay to prevent CSI-based localization? These are precisely the questions we address in our work. The answer to the first one is clear: CSI-based localization is highly intrusive, and tracking people without their consent is unethical. Furthermore, security problems can arise if an attacker can tell that people are inside a room, where they stand, or if they move. Answering the other two questions is more challenging, and it will be the primary goal of Chapters 4 and 5.

The technology is particularly invasive because the attack can be either passive or active, and the victims would be completely unaware of the attack: they do not need to wear any device to be located and have no means to detect the attack. In a passive attack, attackers capture frames transmitted by sources in known positions, like almost all the Access Points (APs) we usually have at home or work. Attackers do not need to control such transmitters; they only have to place a receiver somewhere in the same room or just outside it for precise, Cartesian localization or classification-based positioning. In an active attack, instead, the attacker controls both a transmitter and one or more receivers. The attacker has more freedom and power in the attack: the transmitter and receiver can be placed in strategic positions. Also in this case, if the attacker has no easy access to the target room, he can put the devices outside to avoid easy spotting.

1.2.1 OFDM transmissions

In this work, we will only consider 80-MHz OFDM transmissions with a single spatial stream, i.e., transmitted by a single antenna.[1] The transmission type we use is detailed in the Very High Throughput (VHT) part of the Wi-Fi specification, and the corresponding

[1]The same principles apply verbatim to other standard bandwidths, from 20 MHz to 160 MHz, the only difference being the number of OFDM subchannels in the resulting Wi-Fi frame. In the case of multiple spatial streams, we could treat every stream as a different single-stream transmitter-receiver communication.

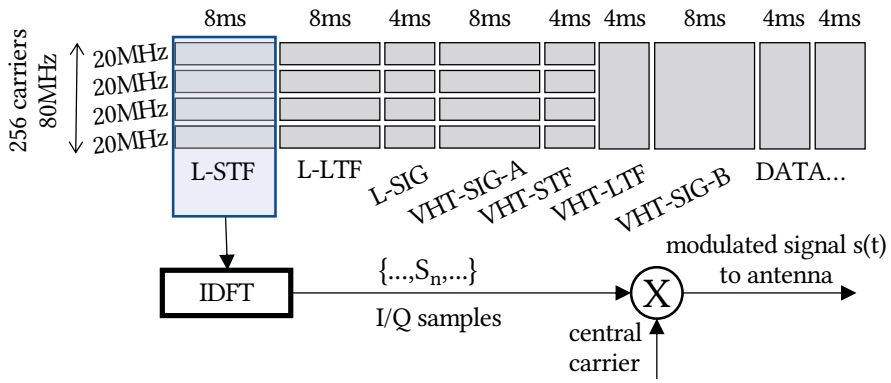


Figure 1.12: Format of an OFDM Wi-Fi frame: initial symbols are known and can be used to derive the CSI at the receiver.

physical layer is called VHT-PHY. VHT is not only an extension of the original 20-MHz physical layer (called OFDM-PHY) but also adds many features for new MIMO capabilities, enhanced modulation rates, and support for up to eight spatial streams. For a complete description of OFDM systems and how OFDM is employed in Wi-Fi, we refer interested readers to the standard [3] and classic literature [31].

OFDM divides the transmitted signal over many equally-spaced subchannels—256 in the case of an 80-MHz 802.11ac frame, including some empty guard subcarriers at both ends—keeping the subcarriers orthogonal by construction. The transmitter "builds" OFDM symbols in the frequency domain, mapping data symbols onto every subcarrier using the appropriate modulation, and then generates the corresponding sequence of I/Q samples S_n in the time domain using an inverse discrete Fourier transform (IDFT). Figure 1.12 depicts this procedure, with the blue box representing a single OFDM symbol. This operation is repeated for every OFDM symbol until the entire frame is transmitted. In the figure, this would correspond to the IDFT block taking successive OFDM symbols, moving from left to right. The resulting sequence of IQ samples represents the OFDM signal that is finally upconverted to the central frequency of the Wi-Fi channel and transmitted.

To help the receiver decode the signal, the first OFDM symbols are defined by the standard and contain information about the encoding process. Figure 1.12 details all these symbols. The first three legacy symbols (L-STF, L-LTF, and L-SIG) can also be decoded by OFDM-PHY receivers to support interoperation between legacy 20-MHz Wi-Fi channels and the newer 80-MHz VHT channels. They are used: (i) to configure the adaptive gain control (AGC) procedure and estimate both time and frequency offsets (Legacy Short Training Field, L-STF); (ii) for fine frequency tuning (Legacy Long Training Field, L-LTF); and (iii) for determining the duration of the frame (Legacy Signal, L-SIG), so that even legacy receivers can understand the channel is busy. After the legacy symbols, there are four VHT headers: (i) the VHT Signal A (VHT-SIG-A) carries essential information describing the following VHT data (but it is still transmitted as four adjacent symbols at 20 MHz); (ii) the VHT-STF is used to improve AGC

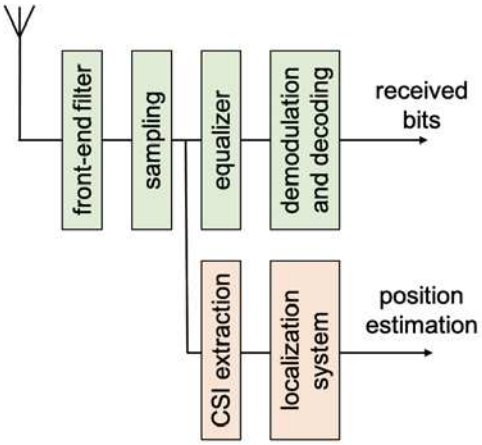


Figure 1.13: Wi-Fi receiver for sensing applications. The CSI data is "extracted" from the radio front end and further processed.

performance; (iii) the VHT-LTF is used to measure the CSI; and (iv) the VHT-SIG-B which is always present but only valid for multi-user MIMO transmissions. After the preamble, DATA symbols are transmitted.

Without going into much detail, at the receiver a correlator first detects the beginning of a Wi-Fi frame (i.e., detects the L-STF symbol) and synchronizes the radio front end to correctly receive the signal; then, demodulation operations are repeated in the opposite order until the original OFDM symbols are recovered back in the frequency domain. Information from preamble symbols is used to recover the clock, reduce the carrier frequency offset (L-STF, L-LTF), and equalize the DATA symbols before decoding their content (thanks to the CSI computed from the VHT-LTF). As we said, a good CSI is crucial to enable high-throughput communications in modern Wi-Fi, allowing fast and precise equalization of the received signal, but it can also be used to infer some information about the environment through which the signal propagated.

1.2.2 CSI extraction overview

Understanding how the CSI is used to enable Wi-Fi sensing applications is crucial for comprehending our CSI manipulation techniques later. In Fig. 1.13 we sketch the diagram of a single antenna receiver that can also perform device-free localization by using the CSI. It is rarely the case that Wi-Fi chipsets allow users to read the current value of the CSI; in general, this information is embedded in the core of the chipset and handled by a dedicated microcontroller unit. When the CSI is not provided by the Wi-Fi radio itself, some firmware modification is required in order to access it. Several tools, named *CSI extractors*, have been developed to "extract" the CSI from the Wi-Fi chipsets of different manufacturers, like Intel [1], Qualcomm Atheros [32], and Broadcom [33]. In our work, we adopt the NEXMON CSI extraction tool [33, 34], which allows us to extract the CSI from commercial access points using Broadcom Wi-Fi chipsets like the Asus

RT-AC86U.

The main difference between the various CSI extraction tools, apart from the fact they are developed for chipsets of different manufacturers, lies in their capability. For example, the Linux 802.11n CSI Tool for the Intel 5300 Wi-Fi chipset [1] extracts the CSI as a real-valued array corresponding to only the magnitude of a 20-MHz channel's frequency response. This array has only 30 elements, meaning that the tool just reports one value for every couple of OFDM subcarriers because 20-MHz Wi-Fi channels have 56 OFDM subcarriers in total. On the other hand, NEXMON CSI [33] can extract complex-valued vectors (reporting both the magnitude and the phase shift) with as many elements as the number of OFDM subcarriers. This means that for 80-MHz Wi-Fi frames we have access to the full channel's frequency response with 256 OFDM subcarriers. In what follows, we will use this tool to extract the CSI from incoming Wi-Fi frames and feed them to a device-free localization system based on a neural network.

1.2.3 Data-driven techniques for localization

Localization with Wi-Fi signals has a long story, from early RSSI fingerprinting techniques to novel mixed and fusion methods. We refer the reader to the survey in [35] for a complete overview of such methods. Here, we focus on techniques based on CSI analysis, which have been proven powerful for indoor localization [17, 18]. In particular, we are interested in methodologies based on machine learning (ML) [26], and especially on deep learning (DL) [13, 25, 28, 36].

As described in Section 1.2.2, the extracted CSI is fed to a neural network (NN)-based localization system. Among the different classes of neural networks, convolutional neural networks (CNNs) have been widely adopted for many applications thanks to their ability to capture spatial and temporal correlations in pattern recognition tasks. Authors of [27] suggested that also a device-free localization system can benefit from the use of a CNN to learn the mapping between the location of a person in a room and the corresponding CSI fingerprints. To this end, they proposed a CNN-based localization system named PILC. In this section, we briefly present the main features of a localization system inspired by PILC [27] and will serve us as a benchmark in the following chapters. Further details about this system's design and implementation are found in [37].

The localization system works by feeding CSI data, encoded as a sequence of complex values (IQ samples), to a CNN that was previously trained with other labeled CSI data collected when a person was standing in the target positions. The general framework is illustrated in Fig. 1.14. Usually, we can distinguish two phases in the operation of the system. First, there is an offline stage to collect labeled *fingerprints*, i.e., we assume the attacker (eventually with the help of an assistant) can collect many CSI data with a person standing in each of the allowed target locations. The first phase ends with the training of the localization system itself. Second, there is the online stage (operation) when the victim is alone in the room. The system continues to collect CSI samples that are now fed to the system to estimate her location.

In general, before feeding CSI measurements into the neural network, some pre-processing steps are required. The most important operations performed by the system

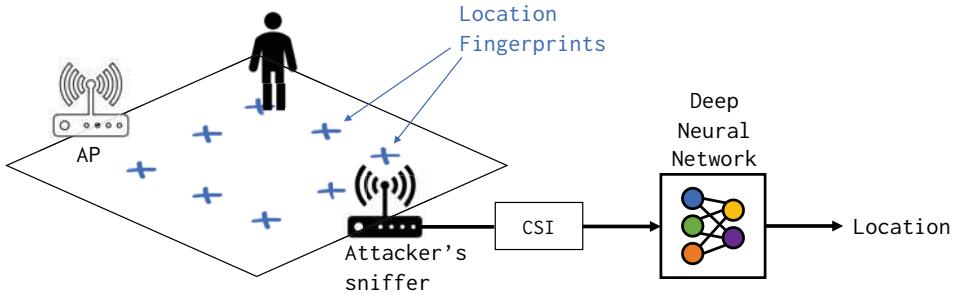


Figure 1.14: Architecture of the localization system. First, the attacker trains the NN by providing both CSI data and the corresponding location label. Then, the system outputs a position for every CSI it is presented.

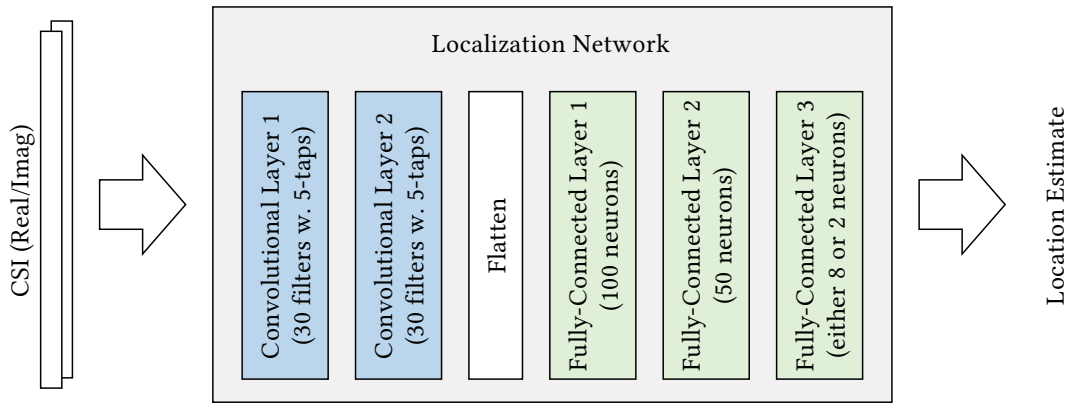


Figure 1.15: Architecture of the neural network implementing the localization system.

are the following:

Subcarriers selection. Not all the OFDM subcarriers are used by the system. More precisely, we discard the subcarriers corresponding to the guard bands, the suppressed main carrier of the Wi-Fi signal, and the pilot subcarriers. In fact, the amplitude of the pilots reported by the specific CSI extractor we used is inaccurate in the case of multi-stream transmissions [33], so we decided to discard them. In the end, the localization system uses 234 out of 256 subcarriers.

Normalization. Successive Wi-Fi frames might be received with different AGC configurations, depending on external conditions or caused by time-frequency offsets between the transmit and receive radios that cannot be controlled. If this happens, the digitalization of IQ samples results in CSI data that could have different amplitudes even if they derive from the same transmission. For this reason, CSI are usually normalized to the element with maximum magnitude.

The neural network architecture, visualized in Fig. 1.15, is the following. First, two convolutional layers extract meaningful features from the CSI data. These two layers exploit the spatial correlation between adjacent frequencies to build more complex

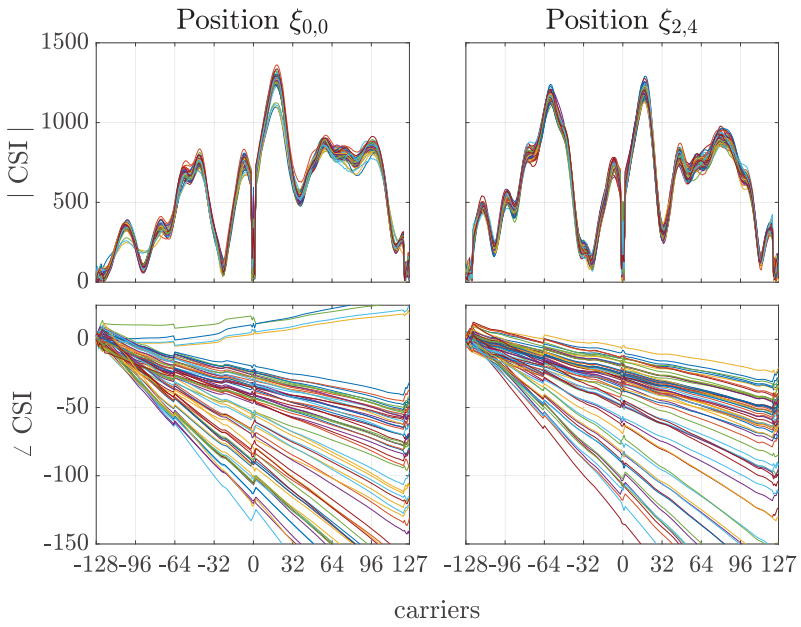


Figure 1.16: Plots of the amplitude (top row, arbitrary chipset units) and phase (bottom row, radians) versus the carrier number with a person in two different training spots.

"descriptors" of the raw data that should better fit the localization task. The output of the second convolutional layer is then reordered to produce a vector of values, an operation known as *flattening*. Then, three fully-connected layers combine the extracted features to estimate the target's position. In our work, the output layer can take two different forms, depending on the localization task of interest. In one case, we consider a traditional classification task; hence, the number of output neurons equals the number of target positions we want to estimate. Each output neuron contains the likelihood that the person is in that position. In another case, we want to estimate the output position as a regression problem in which we produce a "continuous" estimate of the position as a pair of (x,y) coordinates (i.e., not discretized over a set of fingerprints). Of course, the latter case requires collecting CSI data over a sufficiently dense and regular grid to produce valid estimates. It is crucial to notice that in our implementation, the output layer type is embedded in the network and must be selected *a priori*. This means that the NN must be retrained every time we consider a different output layer because we are technically considering a different network architecture tuned for a specific task. In the following, the context should be enough to understand which of the two implementations we are considering. However, we will try to state our output layer whenever needed to avoid confusion.

The number of layers, as well as the hidden neurons for each layer, were initially obtained heuristically with experiments starting with only one layer and increasing the number of layers as well as the number of hidden neurons until the network was able to describe the relationship between the CSI and the corresponding position of the user.

The loss function uses a categorical cross-entropy or the Euclidean distance to rate the network's performance, depending on the type of the output layer, as they provide an intuitive understanding of the error. We choose the common rectified linear unit (ReLU) as the activation function and the adaptive momentum (ADAM) [38] algorithm to adjust the weights based on the corresponding labels. The output layer uses the softmax activation function when considering the classification task.

Figure 1.16 show the magnitude and phase of 70 different CSI data points collected with a person in two distinct locations. The NN is fed precisely with this kind of data (appropriately preprocessed as described before), so whatever the NN "learns" to discriminate different positions, it should already be present in this data. It is clear that the CSI is remarkably constant across different packets for the same position, so the learning procedure is feasible. Interesting features that we can notice right now are the amplitudes' peculiar shapes and the phase discontinuities. However, the notch on the central carrier and the fading signal at both sides of the spectrum are due to the particular structure of Wi-Fi frames, and they do not help estimate the target positions (indeed, they are removed during the preprocessing phase).

1.3 Related Work

1.3.1 Bluetooth tracking

The unquestionable success of Bluetooth technology led to the development of many tracking systems that differ in complexity, cost, and sniffing capabilities. Here, we briefly review some of the most popular solutions and highlight their shortcomings.

Connection-oriented Bluetooth tracking was already proposed in [39] for room-level indoor localization of users to search for friends and optimize building heating. However, the proposed system only worked if target devices deliberately performed a one-time registration. Early indoor tracking techniques without explicit user consent exploited the BT inquiry process [40, 41]; however, since BT devices become "undiscoverable" after pairing, the practicality of these early approaches is questionable. The feasibility of eavesdropping on undiscoverable devices was pioneered by the work in [42] and was based on single-channel sniffing and CRC analysis. However, many BT packets do not contain a CRC code (or sometimes it is encrypted). Therefore, also the practicality of this approach is dubious, as it leads to a high false positive detection rate.

The most common device used to study Bluetooth communications in recent works is the Ubertooth One [43]. This tiny USB dongle, albeit originally designed for eavesdropping on BT, is also suitable for BLE analysis. For example, a system based on the Ubertooth One has been proposed in [44] for forensics and surveillance purposes exploiting the active inquiry scanning process of BT. Another dual-radio Ubertooth One framework is used in [45] to jam and predict hopping sequences of BT and BLE devices. The Ubertooth One has also been used to eavesdrop on unencrypted BLE data sessions [46]. A few other devices that work only with BLE, e.g., the Micro:bit [47] or the Adafruit Bluefruit LE sniffer [48], have been proposed in literature both for passive and

active attacks, such as connection hijacking [49]. All these devices, including Ubertooth One, share the advantage of being inexpensive and are well supported by the community with many open-source applications. Their main disadvantage is that they can only listen on a single Bluetooth channel (either BT or BLE). Therefore, it is hard to discover the parameters of existing piconets, and definitely impossible to track more than one connection at a time. Moreover, since they all use a commercial Bluetooth transceiver embedded in the hardware platform, they cannot be upgraded to support more recent versions of the standard. Indeed, all these platforms use Bluetooth transceivers that do not support the novel data rates and coded physical layers introduced in BLE in 2018 with version 5.0 of the Bluetooth Core Specification.

As of today, specialized commercial platforms, such as the Ellisys [50] or the Frontline packet analyzers [51], are capable of monitoring all types of Bluetooth traffic, storing the packets into capture files. However, we identify three major problems even in these powerful platforms: (i) they are costly, and their price might be prohibitive for many research labs; (ii) they are closed-source, implying that researchers can only use the functionalities provided by the vendor; (iii) they are designed for debugging specific applications (e.g., automotive connectivity) and their capability to monitor general Bluetooth traffic is undocumented.

It is clear that, given the difficulty of monitoring Bluetooth connections, not many solutions have been proposed to mount privacy attacks on Bluetooth technology. Some works highlighted the simplicity of detecting BLE devices by just snooping on Advertising channels [52]; however, even in this case, the system fails to discover existing connections. Other works showed that, although the BLE standard introduces address randomization, over 200 BLE devices under test revealed their identity to adversaries [53]. To overcome the limitations of all the systems described above, we consider developing a novel open-source Bluetooth monitoring framework with software-defined radio (SDR) platforms at the radio front end. SDRs offer superior performance and flexibility with respect to traditional single-channel sniffers; at the same time, they are more affordable than specialized commercial hardware. On the other hand, one shortcoming of SDR-based applications is that they are demanding in terms of computational power. Still, as we will see, modern computers can swiftly process and decode multiple Bluetooth packets in parallel. A framework for sniffing wireless protocols in the 2.4 GHz ISM band was presented in [54]; however, the proposed framework cannot operate in real time, it requires storing large amounts of data before processing, and does not implement any privacy attack. To our knowledge, no implementation capable of decoding all Bluetooth channels (or even a subset of them) concurrently in real-time has been developed to date.

1.3.2 Countermeasures against Wi-Fi sensing

In Section 1.2.3, we have introduced a few Wi-Fi sensing techniques, especially those related to device-free localization applications. However, in this work, our interest is more focused on possible countermeasures *against* Wi-Fi sensing *to favor* users' privacy; thus, we are only marginally concerned about the pros and cons of the different sensing strategies. A good survey on many CSI-based sensing techniques available today—but

without a perspective on security and privacy—can be found in [55].

The first countermeasure against Wi-Fi sensing attacks has been implemented in [56] for preventing gesture recognition; similarly to what we propose in Chapter 5, this system relies on an additional component placed in the environment and acting as a relay. From this work, we have inherited the term *obfuscation* with the meaning of distorting the information imprinted on a frame by the environment, contrasted to the more common *jamming* that instead tries to generate a strong interference signal to prevent sensing applications. However, the artificial reflection techniques we propose in Chapter 5 are different, and [56] is focused on gesture recognition rather than localization. Furthermore, in our opinion, [56] is a pioneering work in this research field, and our contribution adds novel insight into this fascinating topic.

The idea in common with [56] is the potential use of dynamically changing reflective surfaces or active devices that randomly change the electromagnetic (EM) environment. There are two other works that, even if they do not explicitly mention location privacy, are in some way close to what we will discuss in Chapter 5. The first one presents a simple yet effective device implementing a passive reflector with different delays corresponding to 0, $\frac{\lambda}{4}$, and $\frac{\lambda}{2}$ additional paths (λ represents the wavelength), which can be selected randomly [57]. The authors are more concerned with the challenge of using their system to enhance communications; however, their proposal can swiftly be adapted to obfuscate Wi-Fi signals and prevent sensing. The second one takes a "networking perspective" to smart EM spaces, called *programmable wireless environments* [58], and focuses more on potential goals and services rather than on physical layer details. The work is extensive, but the authors explicitly mention the goal of privacy protection, even if referring to eavesdropping rather than localization.

Finally, we can conceive reactive jamming devices that "kill" the frames used for localization attacks, adapting techniques like [59] or [60]. To the best of our knowledge, such approaches have never been proposed in the literature, so it is difficult to state how effective they can be. Moreover, they would require knowing that a localization attack is underway, and the jamming device must recognize the illegitimate traffic and try to intercept only those frames.

To close this overview and the chapter, we want to stress again that the security and privacy implications of Wi-Fi sensing have received less attention than they deserve. Indeed, even if we can imagine different ways to hinder Wi-Fi sensing, depending on the attacker model we have in mind, only a few options among those available have been explored so far. To our knowledge, there is only another work considering pre-distorting transmitted Wi-Fi signals to emulate the frequency response of a "fake" channel [61]. However, the proposed tool can implement limited distortions equivalent to a 3-tap finite impulse response (FIR) filter. The authors of [62] proposed to manipulate the CSI to avoid radiometric fingerprinting of the devices but did not consider sensing applications. In Chapter 4, we will investigate a similar obfuscation technique applied to localization attacks.

Chapter 2

Full-band de-anonymization of Classic Bluetooth devices

Despite the weak cryptographic foundations for protecting devices' addresses that we have reviewed in the previous chapter, there is a common belief that BT is somewhat resistant to tracking attacks. This is in part due to: (i) the perceived difficulty of capturing and analyzing 79 BT channels over about 80 MHz of the radio spectrum; (ii) the fact that BT devices hop on a different channel with a frequency of 1600 Hz (transmitting on each channel for less than a millisecond and excluding some of the possible channels); (iii) the fact that the Central's BD_ADDR is not sent entirely in the clear but encoded inside a whitening mechanism that also depends on its clock. Nevertheless, as per the initial design, assigned BT addresses continue to be immutable today.

In this chapter, we show that there are methods today that can achieve high de-anonymization accuracy in real-time, even using relatively inexpensive hardware and at a great distance from targets. To make things clear, we denote with the term *de-anonymization* the process of univocally determining a BD_ADDR; this result will then enable us to perform fast *re-identification* in the wild of BT devices. We can, of course, conceive fair uses of such methods, such as profiling vehicular traffic for planning purposes or studying Bluetooth co-existence with other wireless technologies, privacy implications are significant. Notably, while countermeasures such as address randomizations in BLE and in Wi-Fi, or concealed identifiers in cellular networks, help protect users from tracking attacks, billions of BT-powered devices without such features are already deployed and it appears impossible to patch the security flaws of these devices with newer BT version. In fact, no plans to address the privacy problems of BT are in sight and it is even unclear whether such evolution would be technically feasible. Worse yet, BT de-anonymization is a passive attack and it is impossible to discover eventual attackers. In the following, we show that affordable technical solutions fully breaking BT privacy within short observation windows are feasible.

Due to the ever-growing popularity of BT technology, several solutions have been already developed for analysis and debugging purposes indeed. Professional high-end

products enable full-band BT traffic analysis [50, 51]; nevertheless, they are very expensive and built on proprietary software that cannot be modified by users, and their tracking ability is unspecified. Few open-source alternatives exist, such as Ubertooth One [43]. Albeit cheap, these solutions are limited to capturing traffic on a single channel at a time and cannot follow multiple connections concurrently. SDR solutions that employ inexpensive radio front-ends, such as HackRF or LimeSDR, have wider bandwidth and are more flexible, yet are still limited in functionalities [63]. At best, these can extract only the Central device's LAP.

2.1 Threat model

We begin the analysis with a brief presentation of the capabilities we assume a general attacker has, and an overview of plausible adversarial scenarios that would be enabled with the fast de-anonymization of BT devices.

2.1.1 Attacker's capabilities

We assume an attacker can only control portable computers and SDR front ends tunable on the 2.4 GHz band, e.g., HackRF, LimeSDR, or USRP. These could be either battery-powered or attached to power supplies in covert locations (rooftops, balconies, tunnels, power buses, or cars). The attackers should be within the reception range of the victim devices (this range could be potentially extended with directional antennas). The attacker does not have access to other specialized high-end hardware to carry out the attack.

2.1.2 Adversarial scenarios

We identify three main types of attacks that are enabled by exploiting BT de-anonymization. Namely, these attacks are user tracking and surveillance, stalking and espionage, and compromising physical assets.

User tracking. It is conceivable that policing agencies and other surveillance entities could deploy BT sniffing and de-anonymization tools on public transport and in key transport hubs (airports, train stations, bridge crossings, tunnels, etc.) to (i) gauge footfall or traffic flow; (ii) identify movement patterns of groups of individuals; or (iii) track the precise whereabouts of a sensitive asset. BT device identity could be linked to individuals via sales databases, car plate recognition software, or CCTV and face recognition algorithms. Likewise, commercial actors would use similar infrastructure in theaters, cafés, shopping malls, etc. to monitor customers and orchestrate targeted marketing campaigns. On the other hand, city councils could hinge on knowledge of citizen flows to improve the provisioning of public services (including waste management, transportation, and lighting), admittedly at cost of users' privacy.

Stalking and espionage. As SDR hardware is increasingly more affordable and open-source tools abound, crowd-sourced stalking systems would be straightforward to design if the identity of a BT device could be reversed from overheard frames. For

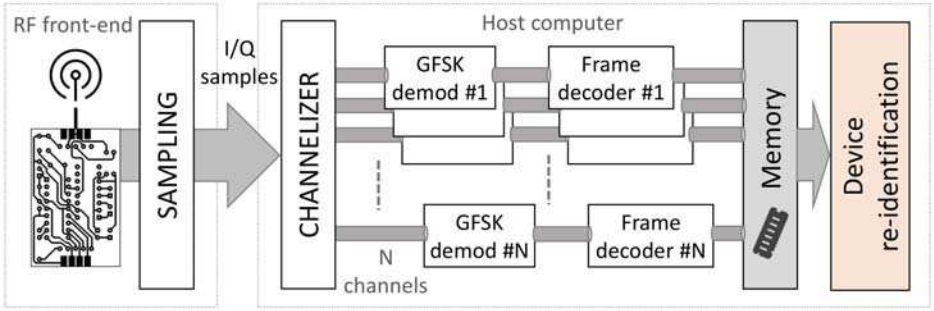


Figure 2.1: Architecture of our full-band BT sniffing system. Raw signals are captured using the SDR front end; channelization, demodulation, and frame decoding are performed on the host computer. We can re-identify BT central devices by reversing the header generation process.

instance, a malicious user would post the BT identifier of an ex-partner or celebrity to a community-controlled sniffing network, in order to know their whereabouts and cause emotional distress. Similarly, competing businesses or rival states could send victims allegedly free-replacements of BT-powered gadgets (headphones, smartwatches, etc.) with known identifiers, which would be subsequently tracked via crowdsourced location-sensing, to infer undisclosed locations of the victims.

Compromising physical assets. De-anonymization of BT devices can also underpin man-in-the-middle (MITM) attacks that can have severe consequences on victims, without necessarily being immediately obvious. For example, an attacker could fake the presence of a specific person near some kind of personal asset. This would for example enable unlocking smart locks, vehicles, or access to a computing infrastructure.

2.2 Full-band Bluetooth Sniffer

This section presents the SDR-based full-band BT sniffer we developed, describing both the data acquisition process and the processing pipeline.[1] Figure 2.1 depicts in a diagram the architecture of the entire system. The system relies on an SDR front end for capturing raw wireless signals in the 2.4 GHz ISM band. The front end is connected via USB 3.0 to the host computer. Our design is sufficiently flexible to work with different SDR platforms and different bandwidths, e.g., a single USRP N310 capable of capturing the entire 79 MHz bandwidth used by BT, two synchronized USRP B210, each one covering 40 MHz of spectrum, or even other arbitrary configurations.

The raw IQ samples captured by the radio front end are then processed on the host computer. First, they are passed to a channelizer, i.e., a system that separates the wideband signal into many narrow-band signals corresponding to a given set of individual BT channels. The maximum number of BT channels that can be processed depends on the bandwidth of the radio front end, the computational capabilities of the host, and

[1]The source code of the system is available at `https://github.com/bsnet/btsniffer`.

whether the operation in real-time is required. The narrow-band signals derived for every BT channel are then processed by GFSK demodulation blocks, which convert the stream of IQ samples into a bitstream. A separate module correlates the resulting bitstream with the blueprint of a BT preamble to identify the beginning of BT frames. As we can see in Fig. 2.1 structure of the system is inherently modular, and the channel processing pipeline on the host is designed to be implemented in parallel.

Depending on the number of channels selected for capture and the computing power, the BT frames acquired are stored in RAM or on PCI Express solid-state drive (SSD), together with a timestamp, to aid frame sequence reconstruction. Data acquisition and processing can either work at two different times (*offline* operation) or concurrently if the time required to process a sequence of IQ samples is less than the time needed to acquire the same sequence (*online* operation). The latter is largely dependent on the computational power of the host, and it can be achieved for example with a double-buffer implementation where one buffer is filled with new data while data in the other one is processed. We will detail a real-time implementation of a full-band Bluetooth sniffer in Chapter 3, with particular emphasis being put on BLE analysis.

A separate module, placed at the end of the diagram in Fig. 2.1 and described describe later in Section 2.3, fetches sniffed frames from memory and exploits weaknesses in the BT header generation process to de-anonymize BT Central devices in generic piconets.

2.2.1 Data acquisition

The first task performed by the system is recording a wideband signal covering the full 79 MHz of bandwidth used by BT. Different SDR platforms are suitable for this operation; for our experiments, we adopt two USRP B210. The USRP B210 SDR supports full-duplex operation with up to 56 MHz of real-time bandwidth, which is not enough to capture all BT channels. Hence, the need for deploying two such boards. We tune the central frequency of the two boards on 2421.5 MHz and 2460.5 MHz respectively and configure the receive bandwidth on both of them to 44 MHz. We allow a small overlap (5 MHz) between the bandwidths of the two receivers, to facilitate output trace synchronization without using an expensive external clock.

Synchronization across all BT channels is paramount in our system since estimating the BT Central clock is a crucial part of the de-anonymization process. Hence, frames captured by the two different SDRs must have a common time reference. This is also for debugging purposes if we want to correctly order the sequence of messages transmitted over different channels at different time instants. Hence, we devise a synchronization method with µs-granularity that does not require coherent capture. In essence, we configure an external BT radio to transmit every 1 s on a BT channel that is captured by both SDRs a synchronization packet containing a unique sequence number. It is even possible to transmit this synchronization packet with one of the SDRs used for capture, given their full-duplex capabilities. The synchronization packet shall be received by both SDR boards and therefore appear in both traces. We can safely ignore propagation delays between the two SDRs if they are close enough, and assume the reception of this packet happens at the same time on both of them. Hence, by using these periodic

synchronization packets, we can quickly compensate for time offsets between the traces captured by the two SDRs in our radio front end.

Finally, it is worth pointing out that the IQ samples captured by the two SDRs with a frequency of 44 MHz are quantized with 2 B per IQ sample (1 B per sample component). If we consider the entire system using two radios, it means that the host system must support an input data rate of 176 MB/s. This can be managed by writing to RAM or to an M.2 SSD, which supports at least twice the required rate.

2.2.2 Data processing

The processing system on the host can be divided into several phases:

Channelization. To separate the spectral components of the wideband signal acquired previously and distinguish the frames transmitted on the different 79 BT channels, the first component of the data processing chain we implement is a channelizer. This comprises a digital down-converter scheme that (i) shifts the complex input signal to the baseband, (ii) applies FIR filter with 1-MHz bandwidth, and (iii) decimates the output signal to reduce the data rate [64]. By tuning the local oscillator to the central frequency of each target BT channel, we can extract the corresponding narrowband IQ symbols. The channelizer's speed can be greatly increased by using polyphase filters to directly separate all the narrowband BT channels from the wideband signal [65].

Demodulation. All BT channels are subsequently processed by 79 demodulation and frame decoding blocks, fed with the corresponding baseband signals. The IQ samples for every channel are fed into a GFSK demodulation block that outputs the corresponding binary data. GFSK is a digital frequency modulation technique where symbols are first filtered by a Gaussian filter and then used to modulate the carrier signal. Gaussian filtering helps avoid abrupt changes in the instantaneous frequency of the output signal.

A well-known technique for demodulating GFSK signals is based on measuring the phase difference $\Delta\phi$ between two consecutive samples of the corresponding baseband signal. Assuming that the channelizer has already filtered out the high-frequency components, $\Delta\phi$ will have the same sign as the frequency deviation of the signal from the carrier and will satisfy the relation $-\pi < \Delta\phi < \pi$. Given two successive IQ samples (I_1, Q_1) and (I_2, Q_2), the phase difference between them can be measured using simple trigonometric computation. However, since what is relevant to our task is only the sign of $\Delta\phi$, we can skip any trigonometric operation by verifying that, for $-\pi < \Delta\phi < \pi$, the following holds:

$$\operatorname{sign}(\Delta\phi) = \operatorname{sign}[\sin(\Delta\phi)] = \operatorname{sign}(I_1 Q_2 - I_2 Q_1). \tag{2.1}$$

Thanks to Eq. (2.1), we can replace the expensive computation of a trigonometric function with two products and one subtraction (which are usually much faster and easier to compute in parallel on modern hardware) and finally evaluate the sign of the result to determine the value of the next bit in the output bitstream.

Frame decoding. Once the demodulation step is completed, we can finally detect and decode BT packets. The bitstream of every BT channel is correlated with the

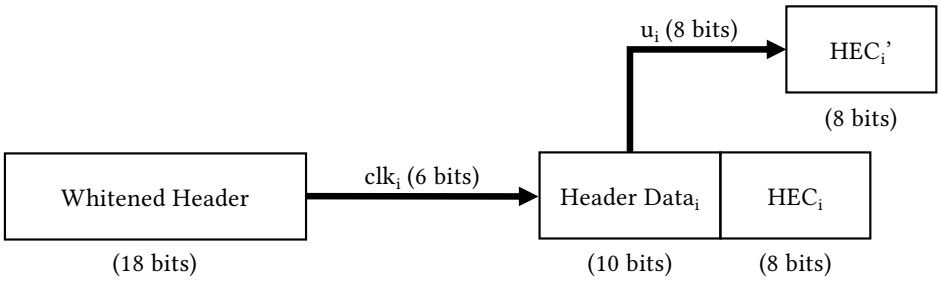


Figure 2.2: Processing logic for inferring the UAP of a BT Central from the whitened header.

possible preambles of a BT packet (eventually including the sync words that follow) to detect the boundaries of BT packets with high confidence. In the following, we examine the extracted packets for re-identifying Central devices in target piconets.

2.3 Re-identifying Bluetooth devices

BT has been long considered to provide good user privacy because devices (i) stop responding to Inquiry packets after establishing a connection, (ii) change channels every 625 µs, following a pseudo-random hopping pattern that is only known to devices in the piconet, and (iii) are not directly identifiable, since BT packets only contain half of the Central's BD_ADDR (the LAP). Moreover, the BT packets' headers are obfuscated using a per-frame "pseudo-key" that depends on the Central's UAP and part of its clock. Our full-band sniffing system presented in Section 2.2 breaks the first two identity protection features, as it enables adversaries, which supposedly neither know the channel nor the pseudo-key, to capture frames in a target session. We show that it is possible to de-anonymize Central devices by exploiting weaknesses in the HEC computation and header whitening procedures. These flaws enable us to derive also the Central's UAP.[1]

To find the UAP of a device, we need to (i) identify the 6-bit value of the Central clock ($clk = \mathrm{CLK}_{1-6}$) used to whiten a frame header, and (ii) infer what UAP value u produces a HEC$'$ that matches the HEC in the de-whitened header. We illustrate this logic in Figure 2.2. we recall from Fig. 1.6a that the HEC is produced using a LFSR that is initialized with the UAP. By construction, given any 6-bit value for the Central clock (clk), only one UAP u derives a HEC$'$ matching HEC. With this in mind, we first employ Algorithm 1 to identify the (UAP, CLK) pairs that could be valid.

In general, for each whitened header wh we capture, we look for the different (u_i, clk_i) pairs, with $0 \leq i < 64$, such that the following holds:

$$\begin{aligned} clk_i &= i \\ wh &= [hd_i \mid \mathrm{HEC}(hd_i, u_i)] \oplus \mathrm{w}(clk_i) \end{aligned}$$

[1]We recall that the 2-byte NAP is never used, but merely present for compliance with EUI-48 standards.

Algorithm 1 Identifying plausible (UAP,CLK) pairs.

```
1: set good_list = []
2: for clk = 0:63 do
3:     header = dewhiten(whitened_header, clk)
4:     for u = 0:255 do
5:         HEC' = computeHEC(header_data, u)
6:         if HEC' == HEC then
7:             push (u, clk) into good_list
8:         end if
9:     end for
10: end for
```

where hd_i is the de-whitened header data corresponding to clk_i, the symbol | represents the concatenation operator, $\mathrm{HEC}(hd_i, u_i)$ is the map generating the 8-bit HEC code, $\mathrm{w}(\mathrm{clk_i})$ is the whitening sequence generation function, and the symbol $\oplus$ represents the XOR operation. In the following, we will express the latter as $\mathrm{w}(clk_i) = \mathrm{w'}(clk_i \oplus 2^6)$, where w' is the linear map implemented by a LFSR that is identical to the one that generates the whitening sequence, but without the static initialization of the internal state's MSB, and 2^6 makes such initialization explicit. We also introduce notation for this mapping's upper and lower parts, i.e., $\mathrm{w}' = \mathrm{w}'_{hd}|\mathrm{w}'_{hec}$, which whiten respectively the HeaderData and the HEC that are concatenated in the above equation.

Let (UAP, CLK) be the actual UAP and clock value CLK_{1-6}, and hd the correct HeaderData. We can use the following equations to compute the other valid (albeit wrong) UAP values from the (also wrong) corresponding clocks and the associated (bogus) HeaderData values:

$$\mathrm{HEC}(hd_i, u_i) = \mathrm{HEC}(hd, UAP) \oplus \mathrm{w}'_{hec}(clk_i \oplus CLK) \quad (2.2)$$

$$hd_i = hd \oplus \mathrm{w}'_{hd}(clk_i \oplus CLK). \quad (2.3)$$

After introducing $\overline{clk}_i = clk_i \oplus CLK$ and reworking the equations, we obtain

$$\mathrm{HEC}(0, u_i) = \mathrm{HEC}(\mathrm{w}'_{hd}(\overline{clk}_i), UAP) \oplus \mathrm{w}'_{hec}(\overline{clk}_i). \quad (2.4)$$

Equation (2.4) shows that the candidate values u_i for the UAP that our search algorithm computes can be obtained by taking all possible values of the clock $0 \leq \overline{clk}_i < 64$. They only depend on the correct value of the UAP, and they do not change over consecutive transmitted HeaderData, which are likely to change every time. We also note that the UAP values for clocks that are each the 1's complement of the other are the same. After observing that the 1's complement of clock $\overline{clk}_i$ can be written as $\overline{clk}_i \oplus (2^6 - 1)$ (remember that these are 6-bit values), the above follows from the following equality:

$$\mathrm{HEC}(\mathrm{w}'_{hd}(2^6 - 1), 0) = \mathrm{w}'_{hec}(2^6 - 1).$$

Finally, let u be the correct UAP and $CLK(n)$ the sequence of actual clock values. The search algorithm cannot distinguish them from incorrect candidates u' and $CLK'(n)$

that verify the following equalities:

$$\begin{aligned} CLK'(n) &= CLK(n) + 32, \\ \mathrm{HEC}(0, u') &= \mathrm{HEC}(\mathrm{w}'_{hd}(32), u) \oplus \mathrm{w}'_{hec}(32). \end{aligned}$$

The dewhitened HeaderData corresponding to the candidate can be easily determined from the "correct" one, as $hd' = hd \oplus \mathrm{w}'_{hd}(32) = hd \oplus \texttt{0xC0}$. This means that, after dewhitening, the incorrect candidate would have a different MSB in the packet type and a different flow control bit. This could be later used for discriminating such incorrect candidates from the real UAP.

One key observation is that in this list of 64 pairs, only 32 different UAPs. From this list, we can remove wrong candidates by executing Algorithm 2 on successive frames with the same LAP until the list is reduced to u, u'. This algorithm verifies the consistency between the timestamps of the frames and the clock values associated with a candidate UAP. It is also worth noting that the execution of Algorithm 2 on the first two frames received allows us to discard at least the 1's complement of the clock for any candidate UAP, unless the time difference between the received packets is precisely $64 \cdot 625$ s. Thus, with only two frames, we can effectively reduce our search space from 256 to 32 unique pairs (UAP, CLK) or less. When only two possible clock values (with the 32 tick delay as demonstrated above) remain (and hence two possible UAPs), further ambiguity can be resolved only by examining the Header data de-whitened with the two possible clocks and keeping the UAP for which the inferred Header makes sense.

Algorithm 2 Removing implausible UAPs from candidate list.

```
1: set t1, t2 the times in s when consecutive frames with same LAP were received
2: ΔT = round((t2 − t1)/625)
3: for (UAP, CLK) ∈ good_list do
4:     CLK' = (CLK + ΔT)  mod 64
5:     Header = DeWhiten(WhitenedHeader, CLK')
6:     HEC' = ComputeHEC(HeaderData, UAP)
7:     if HEC' == HeaderHEC then
8:         update (UAP, CLK) = (UAP, CLK')
9:     else
10:        remove (UAP, CLK) from good_list
11:    end if
12: end for
```

2.4 Experimental testbeds

We implement the designed full-band BT sniffing system using two Ettus USRP B210 SDR boards, connected to the same antenna using a radiofrequency splitter and to the host via separate USB 3.0 controllers. The host runs a GNU/Linux operating system

and is equipped with a quad-core Intel Xeon W-2123 CPU, 32 GB of memory, and an NVMe SSD.

To evaluate the capability of the system to intercept BT traffic and compromise users' privacy through re-identification, we consider three different testbeds: a controlled indoor multi-device testbed, a controlled BT connection set-up, and an environment in the wild. Next, we briefly discuss each of these testbeds.

2.4.1 Controlled multi-device testbed

We run the first set of experiments using 26 Raspberry Pi 3 (RP) embedded boards having an integrated Broadcom Bluetooth chipset. Moreover, we connect a Bluetooth USB dongle with a CSR chipset to 24 of them. In total, we have 50 different BT interfaces that we can connect in pairs. This allows us to establish 25 simultaneous BT connections which we seek to monitor in this testbed. We develop scripts to enable dynamic control of the connections and traffic exchanged between peers. With this testbed, we are able to evaluate the performance of our system against known ground truth, thereby setting a baseline for the system's performance. A full-band trace is recorded using two USRP B210 SDRs and processed on our workstation.

2.4.2 Controlled single-connection testbed

The second testbed serves to investigate the success of sniffing a single connection while varying the distance between the *attacker* and *target*. For this testbed, we consider two representative use cases, namely (i) a Ford S-Max car (equipped with CarPlay streaming functionality) that connects to a mobile phone (Apple iPhone SE) and (ii) another mobile phone (Huawei P20) streaming to a headset (Sony WH1000-XM3). Also in these scenarios, a full-band trace is first recorded using two B210 SDRs and processed on the same workstation as before.

2.4.3 Experimenting in the wild

Lastly, we measure the performance of our system in the wild. We deploy a smaller version of our system with a single SDR that is capable of processing up to 16 channels *in real-time* on the 4^{th} floor of a building (about 14.5 m of elevation), connected to a Yagi-Uda antenna with 13 dB gain and pointed at a traffic junction for a total of 5 days. The goal is to estimate how many distinct cars equipped with BT can be observed to infer user commuting patterns.

Disclaimer on data collection and users' privacy: In our in-the-wild experiment, we do not store any information that could identify individual users (e.g., the actual discovered BT address). Instead, we compute a hash of this information on the fly, which is kept in RAM for as little as necessary and only to generate the statistics presented here. We also disposed of radio frequency (RF) recordings to prevent malicious use of the traces in the future, e.g., via physical interface fingerprinting.

2.5 Experimental evaluation

In this section, we comprehensively evaluate the BT sniffing system described before. We begin by assessing in our controlled environment the time required to detect all the devices in an area, with and without BT traffic. We compare the performance of our system in terms of detection time and accuracy against Ubertooth [43], the only existing open-source commodity platform for BT sniffing. We then study the impact of the distance from the target on the detection time with the experiments from the second testbed involving CarPlay and a wireless headset system. We also use the second testbed to investigate how limiting ourselves to different subsets of BT channels affects the sniffing performance. Finally, we investigate the severity of attacks on users' privacy in the wild, assessing how many connected cars we can re-identify over several days, thus inferring people's commuting patterns.

2.5.1 De-anonymization time

In the controlled multi-device environment with 25 BT sessions (Section 2.4.1), we can configure each session to generate traffic using `iperf`. Hence, we are going consider two cases: (i) with no traffic, so that only BT keep-alive frames are transmitted, and (ii) with traffic between peers (similar to streaming applications). We position the sniffer's antenna at approximately 2 m from the targets (all RP devices are placed next to each other on a 1-m wide board). We perform full-band sniffing for 30 s and process each captured trace starting from three different points in time, separated by 2 s. We then measure the time required to detect any session. The results are shown in Fig. 2.3, where we report the empirical cumulative distribution function (ECDF) of the time required to retrieve the LAPs and identify the UAPs for each connection. When we find a new UAP, we discard it and restart de-anonymization from scratch. In this way, we obtain a statistical characterization of de-anonymization times by running our technique on a single trace.

Observe that when traffic is present, our system can detect and de-anonymize 80% of the BT sessions within less than 1 s of sniffing. In the absence of data traffic, we still require only 2 s traffic to detect 80% of sessions. Every connection can be identified in less than 4 s when traffic is present and in less than 7 s if only keep-alive BT packets are exchanged.

2.5.2 Impact of the distance from the target

Next, we evaluate the de-anonymization time as a function of the distance from a target connection. For this purpose, we work with the controlled single device testbed described in Section 2.4.2, considering both an outdoor (CarPlay) and an indoor (headset) scenarios.

We report in Fig. 2.4a the performance of our system when we seek to intercept and de-anonymize a connection between the vehicle and the phone, increasing the distance from the sniffer while measuring (i) the time required to compute the target UAP (top

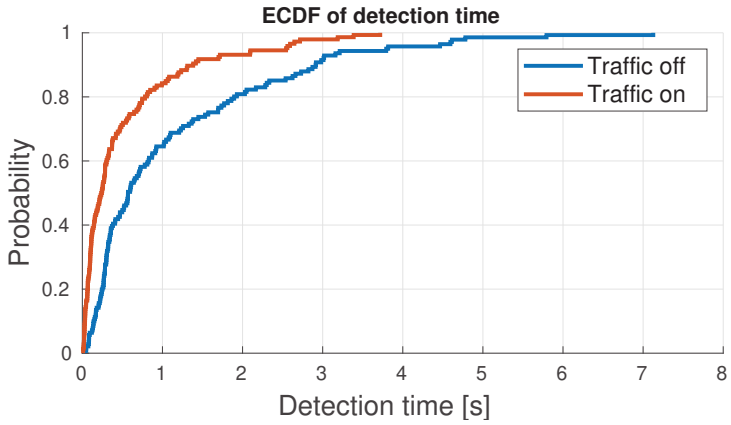
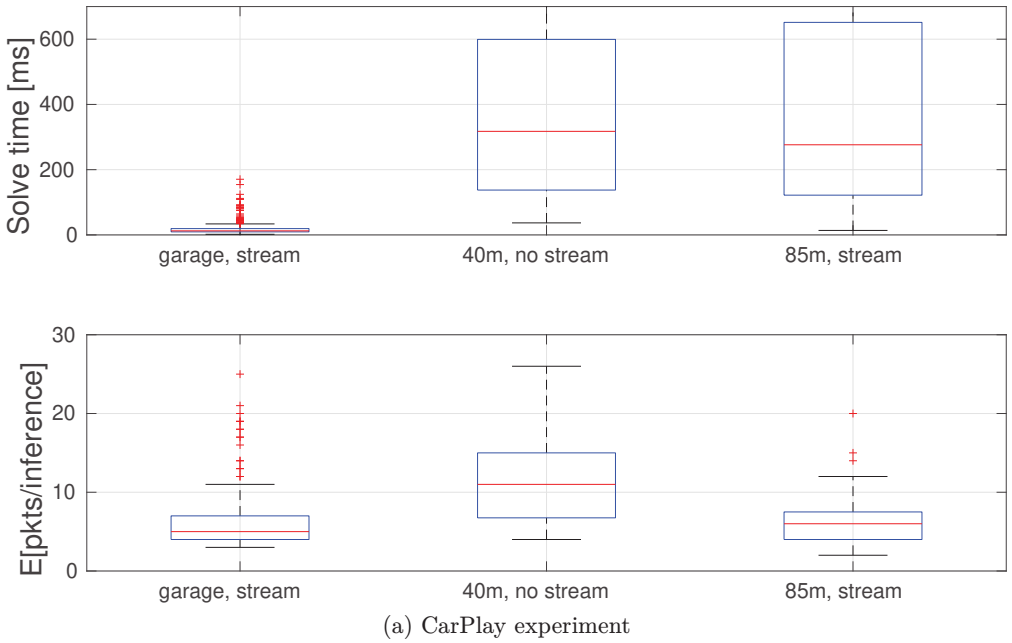


Figure 2.3: ECDF of the detection time (i.e., active sniffing until UAP detection), with and without traffic. Experiments with the multi-device setup (25 connections).

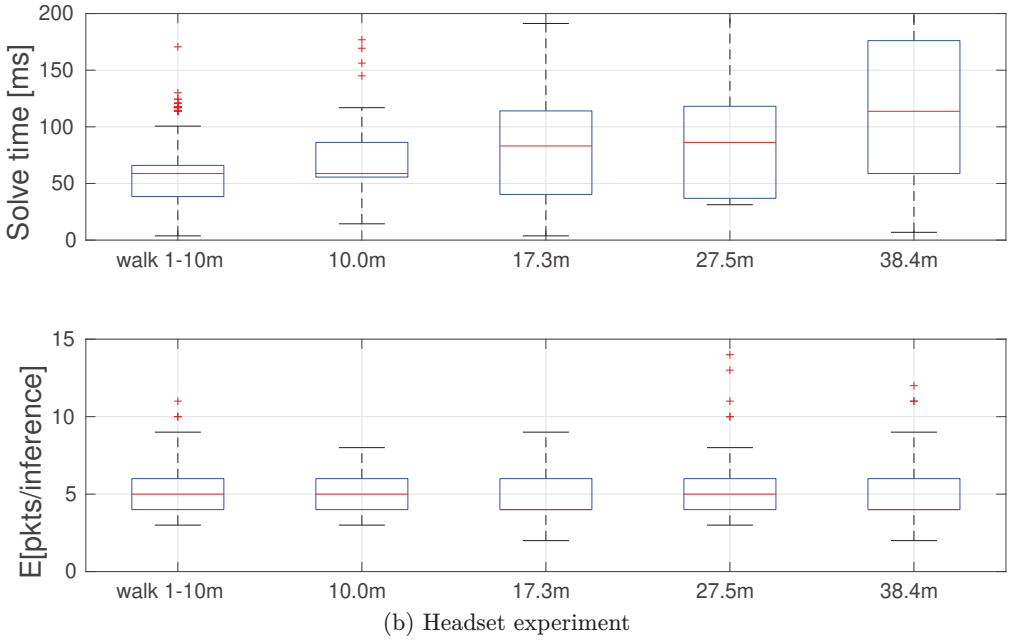
sub-plot) and (ii) the average number of packets needed to determine the target UAP. Received frames are assumed to be correct if the received Access Code is valid and the 1/3 forward error correction (FEC) decoding of the Header does not need error correction (i.e., we observe sequences of 3-bit groups, in which every bit of the group has the same value). We compute the statistics when the target connection is 40 m away from the sniffer not performing any audio streaming, and when it is at a distance of 85 m, with streaming on. To put things in perspective, we also consider a "garage" scenario where the car is very close to the sniffer, and the phone streams music to it.

The first thing to notice is the notable difference between the indoor (garage) scenario and the outdoor one in terms of time required to successfully solve a target UAP (bottom). This can be because RF signals attenuate with the distance and the medium becomes more susceptible to noise and external interference. The median time to solve UAPs outdoors is about 300 ms. We also note that, in general, less than twelve correct BT packets are enough to recover the UAP, with even better results when there is active audio streaming between the two BT devices.

In Fig. 2.4b, we show the results of similar experiments conducted in the headset scenario, i.e., where a connection between a mobile phone that streams music to a wireless headset is targeted at different distances. In the "walk" experiments, the target is moving within a 10 m range from the sniffer, while the rest of the measurements correspond to cases where the user is in a fixed location at the indicated distances. It is clear that the performance of our system depends on the distance to the target connection, i.e., we can solve the target UAP faster if the BT devices are closer. However, in all cases, less than six packets are often sufficient to re-identify the Central's UAP. This is consistent with the CarPlay experiments in which the phone was streaming music.



(a) CarPlay experiment



(b) Headset experiment

Figure 2.4: Success of privacy attacks as a distance to target increases in the two single-connection setups. Boxplots of the time required to solve the UAPs (top) and the number of packets needed (bottom).

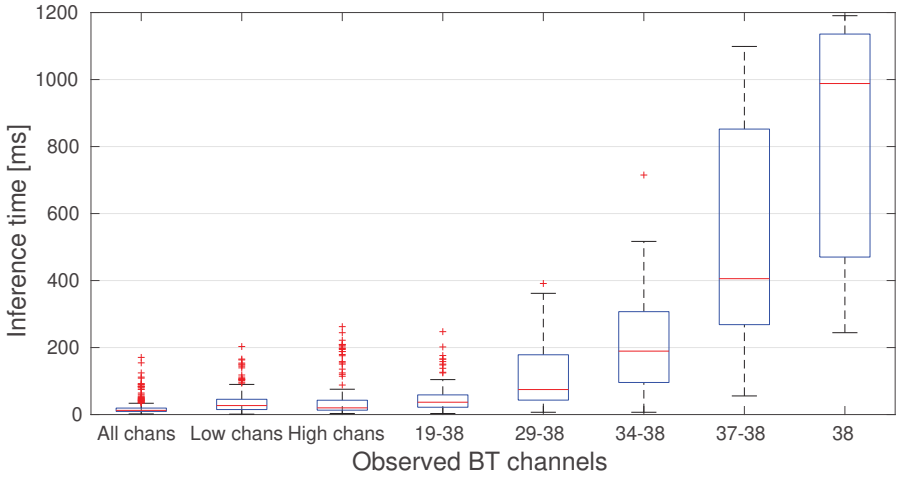


Figure 2.5: Time required to solve the target UAP in the CarPlay scenario, as the number of sniffed BT channels varies.

2.5.3 Impact of the number of channels sniffed

Undoubtedly, the number of channels employed for sniffing impacts the accuracy of the sniffing and re-identification system and its feasibility. Sniffing fewer channels at a time would make real-time surveillance possible, but can also miss some potential targets. To understand how monitoring only a portion of the spectrum used by BT affects the success of our attacks, we conduct new experiments in the CarPlay scenario where the phone streams music, whilst we sniff frames on all channels, half of them, then 20, 10, 2, and respectively 1 channel(s). Results are summarized in Fig. 2.5.

It is no surprise that we are able to infer the UAP of the target connection within milliseconds if all channels are observed. The performance degrades only marginally if we listen on either the lower or the upper part of the spectrum. The target UAP can still be determined within tens of milliseconds if twenty or ten channels are observed. The performance then degrades rapidly though, as with two channels the median solving time is approximately 400 ms, while a single channel yields 1 s median solving times.

2.5.4 Surveillance attacks

Up to this point, we ran all the experiments in controlled environments. In what follows, we present results in the wild (see Section 2.4.3), whereby we use our system to demonstrate its surveillance capabilities.

With our sniffing and re-identification system pointing at a one-way road segment ahead of a traffic junction, we first count the number of vehicles that have BT technology on board and which can be de-anonymized by an attacker during typical working hours (9 AM to 7.30 PM). We can detect cars up to a distance of 114 m, as confirmed by measurements with a car we control. We illustrate the statistics gathered in Fig. 2.6, where we plot the number average number of cars observed every 15 minutes. On

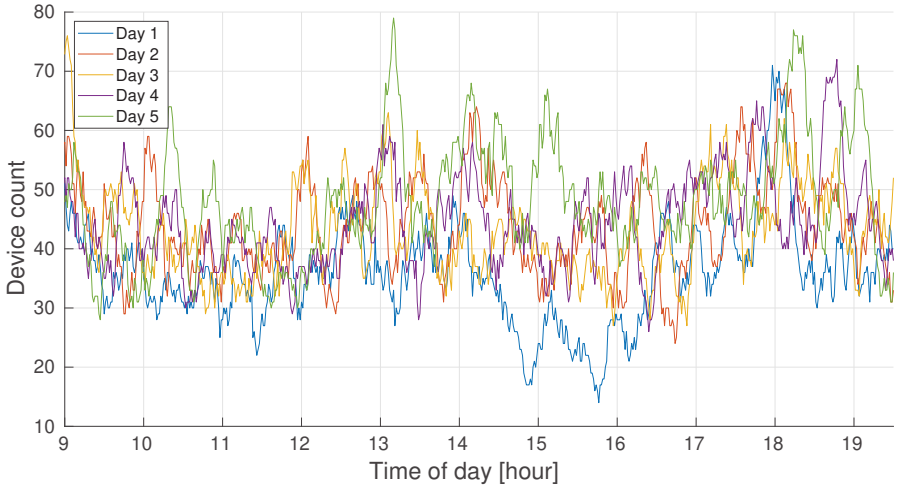


Figure 2.6: Number of unique UAPs found in the wild for 5 days, averaged every 15 mins.

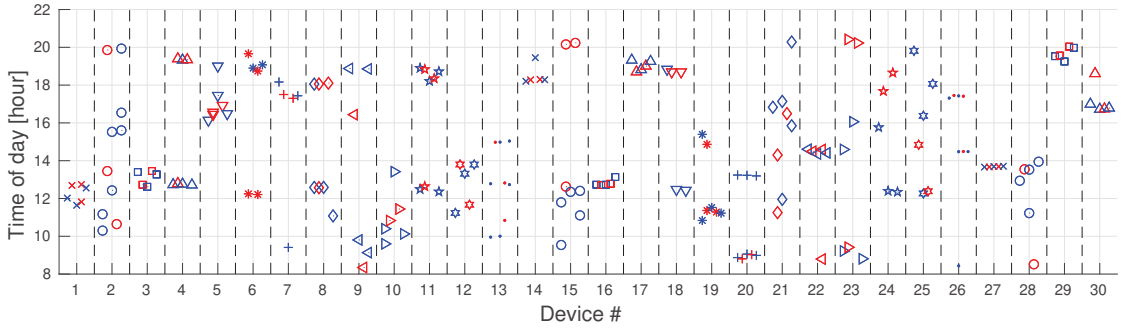


Figure 2.7: Commuting patterns of 30 recurring users identified over 5 days of capture. Symbols are in alternate blue/red colors to discriminate consecutive days better.

average, we detect forty to fifty devices every 15 minutes. As expected, we note heavier traffic around 9 AM, noon, and 6 PM, which are the typical hours of many work shifts and lunch times.

We further examine the commuting patterns of the BT cars discovered. In particular, we record when a de-anonymized device has been seen by our sniffing system during each of the days when we collected measurements. We show the results in Fig. 2.7, revealing the serious privacy issues to which BT users are exposed. Arguably, one may infer information about a user's personality, routine, and behavior from the observed commuting patterns. For instance, we note the precise commuting times of cars 1, 3, 16, 20, 26, and 27.

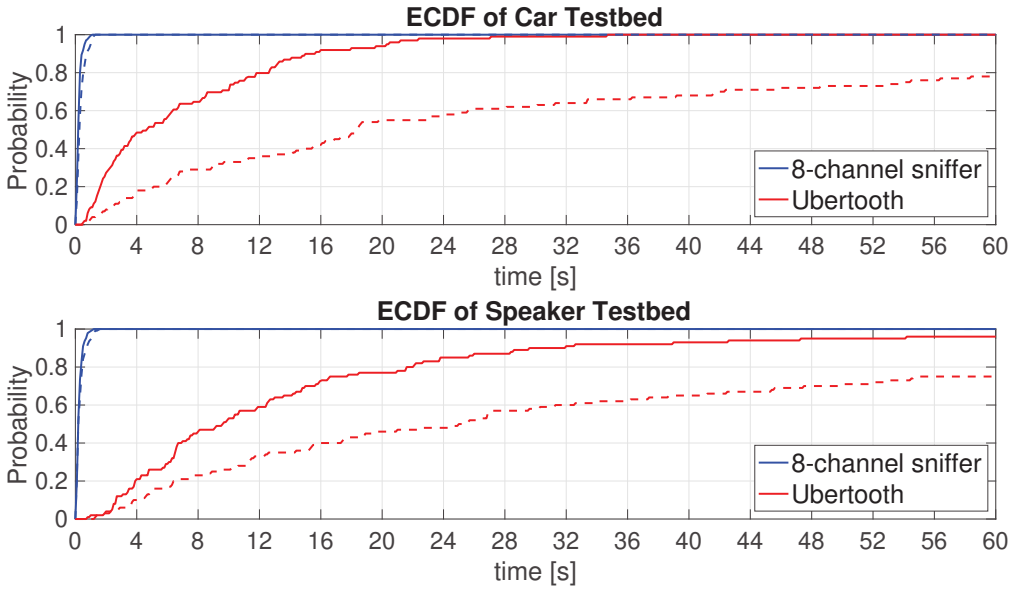


Figure 2.8: ECDF of the time needed to resolve UAPs in two different testbeds. Continuous lines: data traffic on. Dashed lines: no traffic.

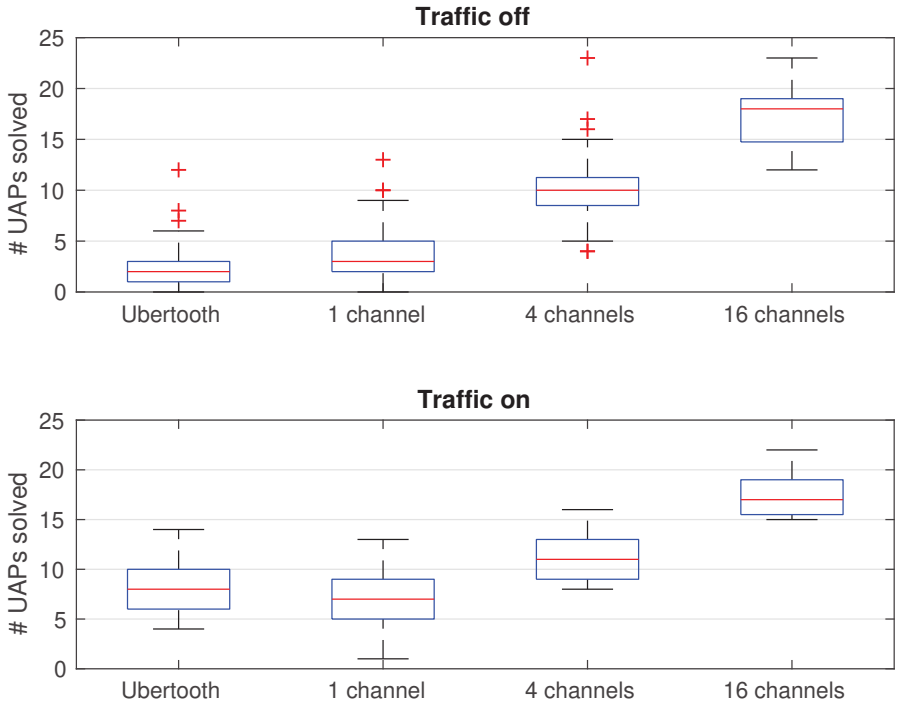


Figure 2.9: Comparison in terms of UAPs resolved in 10 s of sniffing between Ubertooth and our system limited to different numbers of channels.

2.6 Comparison against existing solutions

Ubertooth One is an "open source wireless development platform suitable for Bluetooth experimentation". It connects to hosts via USB and handles the MAC and PHY layers through custom firmware that controls a CC2400 transceiver. Its main advantage is the low price tag, which comes with the drawback of only being able to capture a single channel at a time. To discover ongoing sessions, the platform either stays on a set channel (which can be helpful when trying to detect multiple active sessions) or hops "randomly" (to increase the chances of meeting a session). Being hardware-based, the platform cannot be updated, and the CC2400 radio can only work with BT frames encoded at the Basic Rate (i.e., 1 Mbit/s).

Next, we give a detailed performance comparison between Ubertooth and our system, considering different scenarios; namely, we examine the time required by each platform to discover the UAP of an Android Auto session and the UAP of a connection between a BT loudspeaker and a smartphone, with and without active traffic on the BT link. Figure 2.8 shows the ECDF of these measurements. In both cases, while our approach takes less than a second to discover the UAP in over 95% of the experiments, irrespective of whether traffic is present, Ubertooth struggles to solve the UAPs. When no data is exchanged, it requires more than a minute in 20% of the experiments conducted. This makes it incompatible with high mobility scenarios where observation times can be brief.

We also assess how many devices could be discovered within 10 s in the multi-device testbed described in Sec. 2.4.1 and report the results obtained in Fig. 2.9. Note that the performance of our system in the single-channel operation mode is comparable to that of Ubertooth (fixed on the same channel); Ubertooth detects on average one more UAP than our system when traffic is exchanged, but our system again performs better in the absence of traffic. We examined its sniffing code to understand the reason behind Ubertooth's behavior discrepancy. Unlike what is reported in the documentation, Ubertooth does not use timestamps of consecutively captured frames in recovering target UAPs, as we do in our system. Instead, it removes implausible UAPs iteratively, based on some sanity checks on the frame's payload. While this is pretty effective when payloads are present in the frames, the approach does not work when only NULL or POLL frames are exchanged to keep sessions alive. In these situations, Ubertooth has to wait a considerable time before collecting a proper data frame. For instance, in the previous Car Testbed experiment, such frames carry instructions for updating the car's display. To make things worse, only a fraction of these frames can be captured since the hopping sequences followed by Ubertooth are different than that followed by the BT devices. In addition, it is worth noting that while our solution only identifies valid LAPs, Ubertooth might exhibit false positives in these scenarios.

Even though we relied on USRP B210 SDRs for implementation and testing, it would be straightforward to port our system to other platforms, as long as they support IQ sampling. For instance, the HackRF One is SDR platform that provides up to 20 MS/s and synchronizes multiple devices through clock daisy-chaining. With a single HackRF One and limiting the capture to just 8 or 16 channels, we anticipate it is possible to

achieve excellent performance at a low price. We leave such experimentation for future work, but we remark that the open-source nature of our system and the flexibility it offers (compared to proprietary 'black-box' commercial platforms) lowers the entry barrier for attackers and future research into BT security alike.

2.7 Discussion

Privacy Implications. The BT vulnerability we uncovered has profound privacy implications as users rely more and more on connected devices. Arguably, the most severe risk is that of long-time surveillance, where a user's location can be tracked with concealed wireless equipment once their identity is linked to that of a BT device they own. For instance, road video footage could be processed with plate recognition software to identify the car's owner and BT address. It would then be possible to track the person without using a ubiquitous video surveillance infrastructure, but only by deploying cheap BT eavesdroppers in a few key locations around the city.

Potential benefits. As Bluetooth wearables adoption grows [66], it is also possible to adopt this technology for monitoring and measuring anonymized passenger flows in urban settings. This would require deploying a BT sniffing infrastructure at strategically selected hubs/checkpoints, such as airport terminals, train stations, road tunnels, or bridge crossings. For instance, only twenty-one main bridges and sixteen tunnels connect Manhattan to other New York City and New Jersey boroughs. Deploying a system like the one presented here at such locations could offer insights into commuter flows.

Possible Mitigations. Preserving BT device anonymity would require a new revision of the Bluetooth standard. While it is unreasonable to expect the whitening and HEC generation procedures to be modified, given the number of BT devices already on the market, full address or UAP randomization should be feasible. Cryptographically generated addresses similar to those used in BLE or proposed for IPv6 [67] could be used, by which the 64-bit device identifier would be created with a cryptographic hash of information exchanged by peers during pairing.

Chapter 3

Monitoring Bluetooth Low Energy devices in real-time

In the previous chapter, we described the design of a full-band BT sniffer and examined its implications for users' privacy. However, during the experiment at the road junction, we had to drop the full-band sniffing requirement and favor real-time operation (see Section 2.4.3). Behind that choice stood a critical tradeoff between the computing capabilities of the system and the storage available on the host computer. On the one hand, when considering *offline* (non-real-time) functioning, the system proposed in Chapter 2 can collect full-band traces, but they have a large footprint on disk and take more time to process. On the other hand, when considering *online* functioning, the bandwidth must be limited, and the sniffer can only process a subset of the 79 BT channels.

In this chapter, we shift our attention from BT to BLE to prove it is possible to evade the tradeoff. In particular, we present an implementation of a BLE sniffer[1] that works *on the entire Bluetooth band in real time*. The core idea is to offload heavy computing tasks to a separate piece of hardware, namely a general-purpose GPU, to speed up the detection of BLE packets. We believe the results shown in the previous chapter suffice to show the severity of a tracking attack mounted on Bluetooth technology; therefore, we are not going to repeat an evaluation "in the wild" with BLE devices. Instead, in this chapter, we focus on verifying empirically (in a controlled environment) that it is actually possible to achieve real-time operation with a BLE sniffer, irrespective of the number of target devices.

Albeit BLE provides a generally safer architecture for users' privacy than BT, we argue that it is inherently less complex to quickly and reliably detect BLE connections over the air. First, the packet generation process is much more straightforward in BLE than in BT. The BLE address (which can even be randomized) is not involved anymore in the encoding of BLE packets as it was in the case of BT. In addition, BLE defines only 40 channels, spaced 2 MHz each, in the ISM band at 2.4 GHz instead of the 79 channels defined by BT.

[1] Source code available at `https://github.com/marcocominelli/ble-sniffer-realtime`.

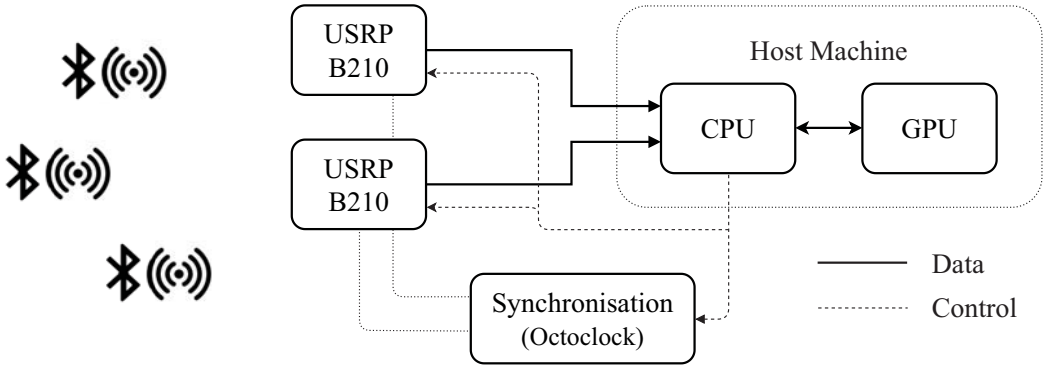


Figure 3.1: Overview of the proposed BLE sniffer architecture.

3.1 Revisited full-band Bluetooth sniffer

We borrow part of the system architecture from the BT sniffer we have already discussed in Section 2.2 and apply some substantial modifications. In particular, the system has been extended to support full-band operation in real-time, with a relatively small latency due to buffering and other delays in the processing chain.

The system diagram is depicted in Fig. 3.1 and can be logically divided into three distinct parts: (i) the radio front-end comprising multiple SDRs that together can capture the entire 2.4 GHz ISM band; (ii) the processing back end, running on the host and powered by a general-purpose GPU to channelize the 40 BLE channels and decode BLE packets in every channel; (iii) the synchronization mechanism that is used to derive coherent timestamps on BLE channels acquired by different SDRs. In the following, we will focus on each of these three subsystems.

3.1.1 Radio front end

The radio interface comprises two Ettus USRP B210 boards. Each USRP B210 can sample up to 56 MHz of instantaneous bandwidth from 70 MHz to 6 GHz and can acquire IQ samples with double floating point resolution (64 bits per component). A single USRP B210 is not sufficient to capture the entire 2.4 GHz ISM band; therefore, we need two of them. We tune them respectively to 2420 MHz and 2462 MHz and set an IQ sampling rate of 40 MHz on both boards with a resolution of 8 bits per sample component. This accounts for a total data rate towards the host machine of approximately 640 Mbit/s for each SDR. To support the required data rate, the boards of the radio front end are connected to the host machine using two USB 3 controllers.

3.1.2 Processing back end

We develop the BLE sniffing system on a computer running Ubuntu 16.04 with an Intel Core i7-7700K and 16 GB of memory. The host computer is also equipped with an

Nvidia GTX 1080 GPU with 8 GB of memory. The real-time functioning of the entire system is achieved thanks to the parallel architecture of the GPU.

In general, GPU-based applications show outstanding performance in terms of computational power; however, performance worsens in case of repeated data transfers to and from host memory, mainly due to the high latency of the interface. To limit this problem, samples are first stored in a temporary buffer (e.g., storing 1 s 2 s of traces) and then transferred in batches to the GPU. Keeping in RAM samples from the wide-band signals can require a large amount of memory—up to a few GB, depending on the size of the temporary buffer. On the other hand, since everything is kept in memory, the footprint on the hard drive is minimal; indeed, we are going to store only the detected BLE packets into a capture file. This is a great advantage with respect to the work in [54], which can only work offline and requires a considerable amount of storage available.

3.1.3 Time synchronization

Finally, we must ensure that the SDRs in the radio front end can capture time-aligned IQ samples. The timestamps from every board must be coherent if we want to correctly record BLE packets received over the entire frequency band. To synchronize the operation of the SDRs in the time domain, we use an external clock distribution module, namely the Ettus OctoClock. This module provides both a 10 MHz reference clock signal and a 1 PPS (pulse per second) signal, both of which are fed to the two B210 boards and used to synchronize the IQ sampling.

Using a clock distribution module is the best solution in terms of accuracy, but admittedly can prove relatively expensive; therefore, we also design a cheaper (less accurate) procedure to achieve time synchronization. In this case, we slightly increase the bandwidth of the two SDRs so that the receiving bands overlap at the center of the 2.4 GHz ISM band. We can then program an affordable BLE transceiver (e.g., as a Nordic nRF52840) to transmit beacons at regular time intervals on a fixed frequency that is captured by both radios. Using the timestamp of such beacons, we can synchronize the timestamps of all the BLE packets while processing the trace.

3.2 BLE processing chain

The system's primary goal is processing *all* the BLE sessions in a particular area *in real-time*. The goal is achieved by offloading the most demanding operations of the signal processing chain to the GPU. We show an overview of the processing chain in Fig. 3.2. We can ideally split the processing into two phases based on the type of data the system is processing. During the first phase, the system takes in input the (quantized) wideband IQ samples from the SDR interface, separates the content of the different BLE channels, and demodulates the stream of narrowband IQ samples in every BLE channel to obtain a sequence of bits, or bitstream. During the second phase, the system processes the bitstream extracted for evert BLE channel to detect BLE packets and log them to a capture file that can be stored or further analyzed. In the following, we discuss some

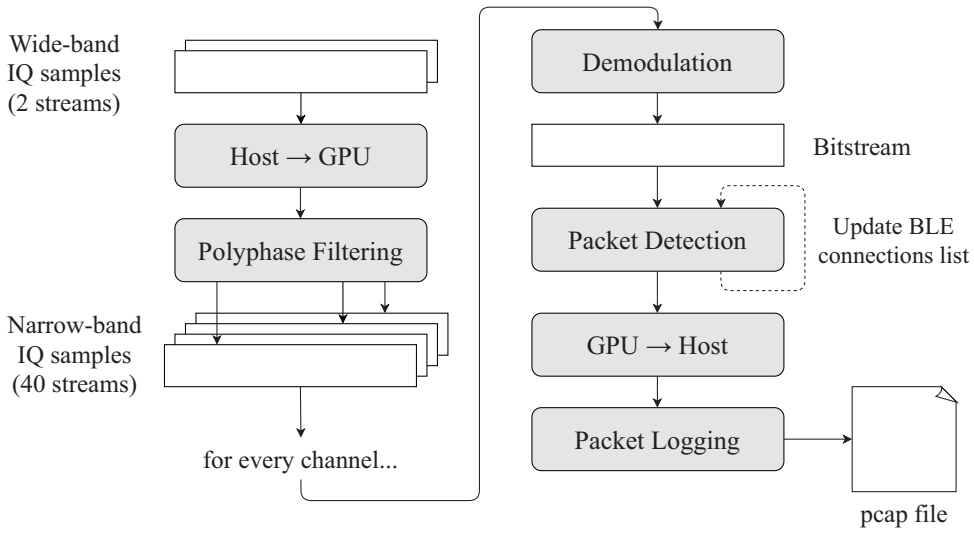


Figure 3.2: Diagram of the processing chain. Operations are executed in parallel by assigning a different GPU core to every IQ sample or bit in a single channel stream. Each task is then repeated for every channel.

details regarding these two phases.

3.2.1 Channelization and demodulation

IQ samples captured by each SDR are temporarily stored in a buffer in RAM on the host computer and then moved in batch to the GPU's memory. The batch size is a parameter that must be carefully chosen. The interface between the Central Processing Unit (CPU) and the GPU has very high latency, so it is impractical to transfer small batches of IQ samples to and from the GPU. At the same time, larger batches require more memory and increase the system's latency due to buffering. In our implementation, we decided to process the signals in batches of one second each. Using a double-buffer implementation, we ensure a continuous flow of samples from the SDRs to the processing back-end.

A polyphase filter is used to channelize the wideband signal from each SDR into 20 narrowband signals, thus accounting for all the 40 BLE channels. The structure of polyphase filtering is well-suited for the highly-parallel architecture of the GPU, and its implementation is reasonably efficient [68].

Since also BLE transmissions use a binary GFSK modulation, we can implement on the GPU a demodulation scheme identical to the one proposed in Section 2.2.2. For every channel, we convert the sequence of IQ samples into the corresponding bitstream[1] by discriminating the phase difference between successive samples using the results already

[1]We anticipate that every single bit of every bitstream is encoded as a `uint8_t` value. This choice might seem suboptimal at a first glance, but it will become useful later.

discussed in Eq. (2.1). In this case, we can design all the operations to be performed in parallel on the GPU, boosting the overall decoding speed.

3.2.2 Bitstream processing and packet logging

At this point, the system has to process 40 bitstreams—one for each BLE channel—looking for new BLE packets. As discussed in the introduction, not all 40 BLE channels are equal. Three advertising channels are reserved for discovering new devices and broadcasting information; the other 37 data channels are used to exchange data between two or more connected devices. However, at this time, the procedure for detecting new BLE packets is the same on every channel and is based on correlating the bitstreams with known fields of the BLE packet.

We cannot rely only on the packet preamble to detect BLE packets because it is very short (only one byte), and the number of false positives would be very high. Therefore, we must correlate the bitstream with the preamble and the AA to be sufficiently confident that we have detected a packet. Once a new packet is detected, we can validate its correctness by checking that the received CRC matches the one computed from the packet content. This is straightforward for Advertising packets, where the AA and the CRC initialization values are known quantities specified by the standard. However, detecting Data packets is more difficult in general because these parameters are pseudo-random values shared by the devices in the piconet only at the beginning of the connection. Therefore, we must devise a different method to discover BLE data packets.

The sniffer can discover Data packets and track existing BLE connections in two distinct ways. In the first case, the sniffer is already active and intercepts a `CONN_IND` packet on one Advertising channel. This packet, sent by the initiator to start a new BLE connection, contains in the clear the AA and the CRC that will be used during the connection phase. These parameters are immediately added to a list of valid addresses, and the discovery of subsequent Data packets works as for Advertising packets. In the second case, a target BLE connection was already active before starting the sniffer; hence, retrieving the connection parameters shared by the BLE devices in the `CONN_IND` packet at the beginning of the connection requires a little more effort.

To discover an existing connection, we exploit the fact that connected BLE devices shall continuously transmit packets—even empty ones—to stay synchronized while hopping. Data packets from the Central are usually followed by a response from the Peripheral on the same channel after a short time. This implies that the same 40-bit sequence (preamble and AA) appears twice on the same Data channel in a very short time (we recall that neither the preamble nor the AA are encrypted or whitened like in BT). The system can validate every BLE packet reception, even if the payload is encrypted, thanks to the final CRC field, which is never encrypted. To recover the correct CRC initialization value, the system must find which one out of the possible 2^{24} initialization values matches the CRC received. Despite being an arguably computationally expensive method, the GPU's architecture can speed up the process by evaluating many different candidate values in parallel. Obviously, with this method, we get the correct initialization value only if BLE packets are received without errors; however, one can

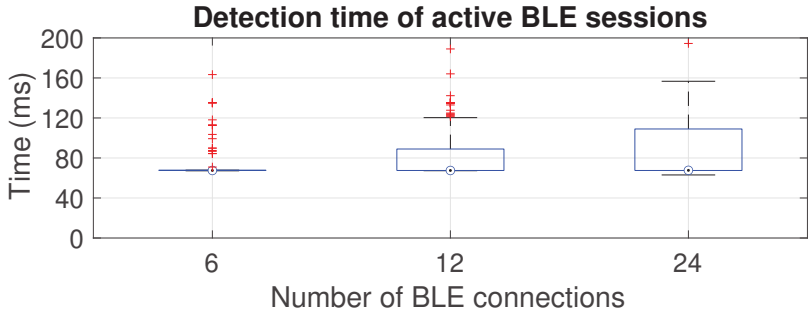


Figure 3.3: Boxplot of the time required to identify all the BLE connections within the range of the sniffer in the considered testbed.

become more and more confident about the correct CRC initialization value if many packets agree on the same result.

When we have identified all the BLE packets in a trace, we still have to decode them and store them on disk. At this point, we cannot yet write data to a file because all the information is kept in the GPU memory. Therefore, when the GPU part of the system identifies a valid BLE packet in the bitstream of a BLE channel, it marks the first bit of the packet with a flag. Note that every bit in the bitstream is encoded as `uint8_t`, and we only use the least significant bit to represent its value. Therefore, we have room for up to 7 binary flags if we use a one-hot encoding for the control information and metadata. With this method, when the 40 single-channel bitstreams are moved back to the host's main memory, the CPU can quickly skim through them, looking only for the specific flags (the identification and correct reception of the BLE packets has been already validated). The only task for the host—quite easy at this point—is to decode the BLE packets detected and write a capture file with all the packets properly reordered.

3.3 Performance evaluation

We set up two testbeds to evaluate the performance of the system. In the first one, we implement a BLE transmitter using a Nordic nRF51 development kit. The transmitter emulates a target BLE connection hopping on all the channels, sending 1000 packets on each channel. The packets have the same AA (which we know) and embed a unique sequence number in the payload. With this setup, we can count how many packets the system detects on every channel and extract statistical information about the detection rate. Every packet must pass the CRC verification to be counted as "correctly received." Empirical results show that the average packet detection rate across all the BLE channels is 99.65%. Specifically, the detection rate is 100% on 23 different channels, and 97% is the worst detection rate measured on one channel. It is essential to notice that interference due to Wi-Fi and other Bluetooth transmissions could not be removed from our testbed; hence, some BLE packets might have been detected but discarded during the CRC verification stage.

The second testbed evaluates the system's ability to detect existing BLE connections without capturing `CONN_IND` packets. We set up multiple BLE connections in a controlled environment, using 24 Raspberry Pi 3B and 24 BLE adapters. We repeat the experiment three times with 6, 12, and 24 active BLE connections in total. In every experiment, the sessions are created before activating the sniffing system, which does not know the AAs used. The goal is to discover the AA and the CRC initialization value of every connection. Even if we do not show this, it is straightforward to recover also the correct hopping sequence of every connection because of the full-band capture.

In Fig. 3.3, we report the statistics about the time needed to discover all the target connections. While this depends on the number of existing connections, even in the most challenging scenario with 24 active sessions our system detects all the connections in less than 200 ms. We note that the distribution gets wider as the number of connections increases. This might be explained by the fact that since more BLE sessions are active, they have higher chances to collide and originate spurious packets rejected by the system. Finally, it is interesting to observe that the median of the detection time is almost constant in all the considered scenarios. This means that, regardless of the number of BLE devices, the majority of the time, the system detects all the existing connections in less than 80 ms.

3.3.1 A consideration on the privacy of BLE

In the introduction, we have briefly argued that BLE is more secure than BT from the standpoint of preserving users' privacy, mainly because the former provides anonymized addresses supported by cryptographic primitives. However, we have shown that BLE connections can be easily discovered and tracked in real-time, even with general-purpose (programmable) hardware. In principle, a system like ours could be used again to track users, as we already discussed in the previous chapter for the case of BT. More importantly, there is no way to change the AA of a BLE connection on the fly; the only way to achieve this is by dropping the connection itself and starting from scratch from the advertising phase. Indeed, we noticed that personal devices like fitness bands, smartwatches, and smartphones might stay connected for a long time.

During our experiments, we found some AA identifying the same connections over multiple consecutive days. This is clearly a flaw in the privacy-preserving mechanism of BLE. Suppose an attacker could resolve the identity of the devices in a BLE connection. In that case, he can track those devices for as long as the link is active, and the address randomization features would be useless in protecting users' privacy. The simplest solution to this issue would be to consciously drop the BLE connection every time a reasonable amount of time has passed and then immediately restart the connection.

Chapter 4

CSI obfuscation techniques against passive Wi-Fi sensing

Wi-Fi is another wireless standard that, like Bluetooth, pervades most of the spaces in which we live today. In the previous chapters, we have discussed how modern technology enables novel threats to Bluetooth security and privacy. To prove this, we developed a system that exploited some flaws in the design of Bluetooth communications and showed that it is feasible to collect sensitive information at the physical layer. Apparently, these vulnerabilities cannot be removed entirely, but we briefly presented some ways to mitigate them. For Wi-Fi, we follow a different approach. Indeed, several privacy attacks exploiting vulnerabilities in the physical layer of Wi-Fi have been discussed in the literature (see Section 1.2.3). Even if they require a lot of resources and their applicability on a large scale is still debatable, there is no doubt that widespread Wi-Fi sensing represents a looming menace to users' privacy. Hence, we focus not on the attack but on protecting users' privacy by deploying appropriate countermeasures. To the best of our knowledge, minimal effort has been made in this sense. This work tries to frame the problem and, more importantly, possible solutions for the case of Wi-Fi in a systematic way.

The topic of precise localization of devices or people has long been of great interest from both a research and an industrial perspective. In particular, indoor localization can still be considered an open research topic because GPS systems do not work indoor. One specific field of indoor positioning is device-free localization, i.e., the localization of human beings (but sometimes also objects) that do not carry any active or passive communication device. Clearly, camera-based localization or anti-intrusion systems are part of this latter field, but they are outside the scope of this work. We focus instead on systems based on Wi-Fi, as wireless communications signals are less detectable by users in common scenarios. In addition, such systems can take advantage of the widely deployed Wi-Fi infrastructure, thus paving the road to ubiquitous surveillance systems.

In this work, we deal with device-free localization based on the CSI, whose variations can be associated with changes in the physical environment, and in particular to the location of a person in a room, as we discussed in Section 1.2. The authors in [16]

pioneered this field by proposing to use advanced MIMO technologies and signal processing typical of radar systems with an SDR module to reveal the position, and even gestures, of a person behind a wall. Clearly, such a system would pose huge privacy concerns, as the localization can be obtained even from outside the room or building the person is in. This exposes the person to the risk of being tracked without having the possibility to avoid it, or even be aware of it. A possible countermeasure against Wi-Fi sensing attacks has been implemented in [56] to prevent gesture recognition; however, the proposed system relies on an additional device that acts as an active signal relay in the propagation environment. Moreover, obfuscation performance strongly depends on the position of such a device. If no countermeasures are available, the only solution to stop such attacks would consist in jamming or disabling all Wi-Fi communication, which is not desirable. In this chapter, we show how it is possible to solve this problem by arbitrarily altering the CSI without hampering communications.

4.1 Threat model

Tracking a person without her consent is illegal in many countries and knowing the location and whereabouts of a person may reveal a lot about her behavior and activities. Let us imagine a scenario in which an attacker wants to infer the location of a person in a room; for example, an employer that wants to keep under surveillance one employee inside a laboratory. We assume the presence in the room of a common Wi-Fi AP providing Internet access in the laboratory. The attacker (i.e., the employer) positioned a hidden Wi-Fi receiver in the laboratory—in our case a second AP, but in general any device capable of extracting the CSI—and uses the NN-based localization system described in Section 1.2.3. A couple more pragmatic assumptions are needed in our model: first, that the attacker can train the localization system beforehand; second, that the attacker can monitor the CSI extracted from the eavesdropping device. This setting is very general and can be easily replicated in hotels, offices, and private homes.

4.1.1 Passive and active localization attacks

We call the attack considered in this chapter a **passive** localization attack. Passive attacks are characterized by the collection of frames sent by a legitimate AP over which the attacker has no control. Passive attacks are harder to mount because they must rely on legitimate (uncontrolled, from the perspective of the attacker) Wi-Fi traffic. However, the severity of these attacks is higher because they are completely passive, that is, possibly undetectable.

In Chapter 5, we are going to consider an **active** localization attack instead. In active localization attacks, the attacker controls both the transmitter and the receiver. The attacker has complete control over the transmission chain, and the only way to interfere with his intrusion is by actively disturbing transmitted frames on the channel. We will discuss later the scope of active attacks and how it is possible to contrast even active attackers.

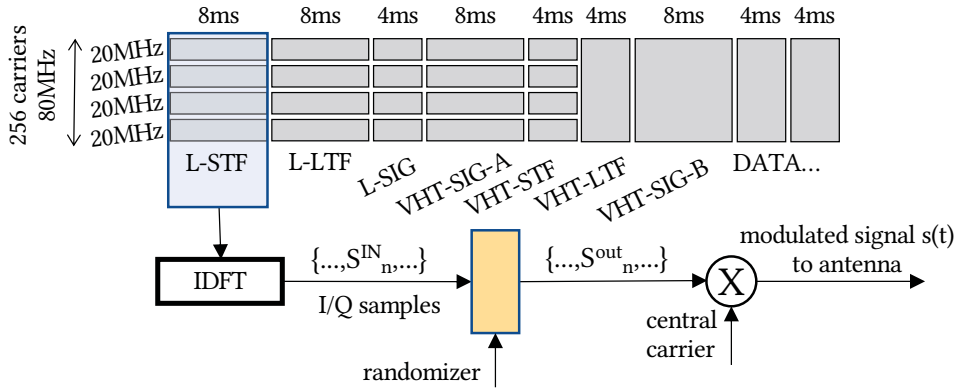


Figure 4.1: Schematic representation of the proposed anonymization system with respect to the process of creating new OFDM symbols.

4.2 CSI obfuscation

To safeguard location privacy—i.e., to prevent a localization system to identify the position of a person—we can apply a time-varying random distortion to the transmitted signal in such a way that a receiver can still decode the Wi-Fi signal while any localization effort based on CSI is prevented.

With reference to Fig. 4.1, we decided to apply the appropriate distortion at the output of the IDFT block, right before the radio front end, in the orange block that injects a *randomizer* modifier in the figure. This position may be sub-optimal, but it makes the technique easily understandable and it can also be implemented outside Wi-Fi radios, allowing the realization of specialized, privacy-preserving devices without the need to develop a new chipset from scratch. Introducing location privacy protection in commercial devices can be done at different levels, from a pure software implementation (as hinted above) to full integration at the physical layer. In principle, it is also possible to develop a sublayer of the standard protocols supporting many new functions, like a negotiation between transmitter and authorized receivers of the *randomizer* sequence to facilitate the frame decoding.

With *randomization* we refer to a manipulation of the transmitted signal so that the actual CSI computed at the receiver contains a pseudo-random combination of distortions like sudden peaks, notches, or phase jumps. The disturbance must change over time, but preliminary evaluations show that the manipulation should neither change too quickly (e.g., resembling white noise) nor too slowly (e.g., a constant distortion imposed on the signal). In fact, in both cases, the NN would be able to easily filter out artifacts and identify again location-dependent features. While some theoretical work is required to devise randomization techniques whose effect cannot be learned at all by the NN, the first goal is to propose a proof-of-concept with simple, intuitive techniques. Furthermore, the disturbance introduced by the randomization process must not distort the signal too much, otherwise it could hamper communication performance.

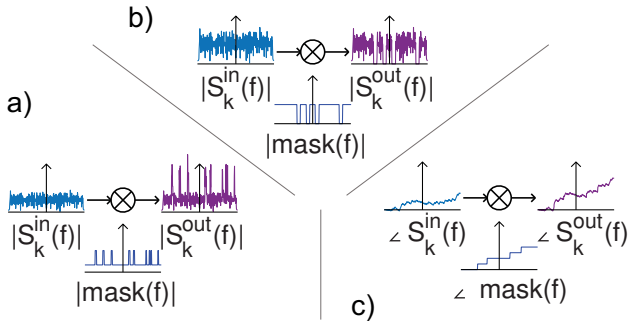


Figure 4.2: The three different manipulations we test to obfuscate the position: (a) random peaks, (b) random notches, and (c) random phase jumps are introduced in the channel response.

The actual manipulation carried out in the randomizer block depends on the disturbances that we introduce, shown in Fig. 4.2, where $S^{IN}(\cdot)$ is the signal at the output of the IDFT, and mask$(\cdot)$ identify the type of distortion. We apply the disturbance by filtering the sequence of IQ samples in the frequency domain. Since in this case the modulation format is fixed to VHT-PHY at 80 MHz with Long Guard Interval we can easily extract the samples forming each OFDM symbol and, knowing the structure of each symbol, we can: (i) invert the encoding process, going back to the coefficients assigned to each of the 256 subcarriers; (ii) modify them according to the desired randomization scheme; and (iii) recreate the "tampered" OFDM symbols.[1] We tested several different types of filters to explore the effect on localization given by basic filtering functions: introducing peaks, notches of phase jumps, where the number and positions of the peaks, notches and phase jumps are random functions. With reference to Fig. 4.2, we tested the randomization technique applying a random number of peaks, notches, or phase jumps. At every iteration, the system places a random number of phase jumps and peaks or notches that is comprised between five and ten. Such limit values have been selected empirically to maintain the packet delivery ratio at acceptable levels; the theoretical analysis of the optimal randomized filtering is left for future investigation.

From a preliminary performance analysis it turned out that notches and phase discontinuities affect the considered localization system only marginally. On the other hand, interesting results were obtained for the filter with random peaks. Localization results for one sample point under different types of randomization are shown in Fig. 4.3. From this example, it is clear that only the distortion with random peaks can trick the localization system to estimate a wrong location with high confidence. Phase shifts and random notches are reducing the precision of the estimate, but an attacker could still make a sensible guess about the user's location. Therefore, we start our analysis by

[1]This sequence of steps might seem strange at a first glance, but it depends on the actual implementation. The software we use generates a Wi-Fi signal as IQ samples ready to be transmitted from an SDR; hence, we have first to go to the frequency domain, alter the signal, and finally get back to IQ samples.

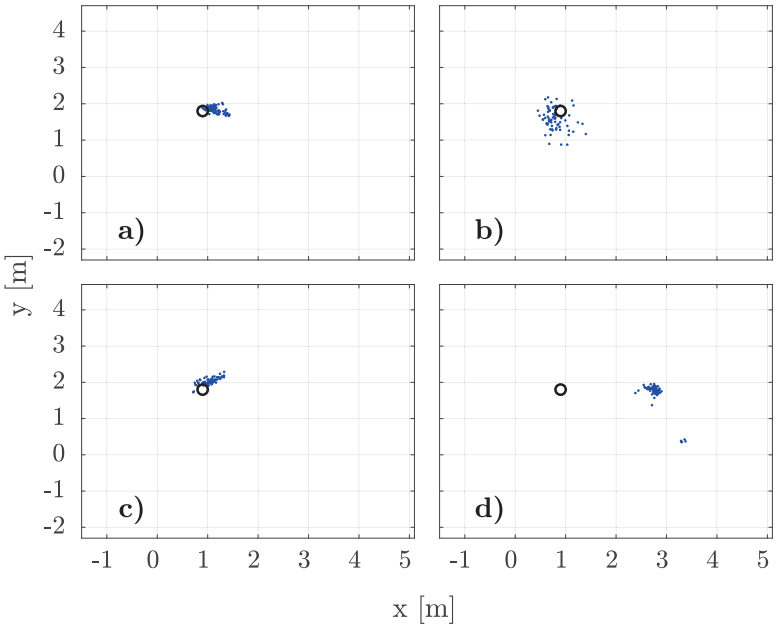


Figure 4.3: Localization results for one target point (black circle) when different randomization schemes are applied: a) without CSI modification; b) with phase discontinuities; c) with randomly-placed notches; d) with randomly-placed spikes.

using randomly-placed peaks only.

4.2.1 Obfuscation Metrics

Measuring the performance of a CSI obfuscation method is not straightforward. In fact, in some cases, the localization system might estimate the wrong location but still get enough information if its guess is not too far from the correct one. To gain the best possible insight into the performance of both localization techniques described in Section 1.2.3 and the obfuscation method introduced in Section 4.2, we use three different metrics, which are best suited in different contexts and situations.

The first metric we consider is the classification accuracy, measured as

$$A_c = \frac{N_c}{N_l}\,, \tag{4.1}$$

where N_c is the number of samples correctly classified, and N_l the total number of samples taken. The overall accuracy is usually accompanied by other metrics, which help interpreting the results, such as the per-class accuracy, precision, and recall. Sometimes, confusion matrices are very useful to present the results because they summarize all the accuracy metrics in a natural way. However, in the specific case of localization, this approach has some limitations. First, when the number of classes (target locations in our case) is large, confusion matrices become very large and difficult to read and

understand. Moreover, they blur the sense of spatial proximity between different target points. Indeed, not all localization errors are equally bad, and guessing a position adjacent to the correct one is not as wrong as guessing a position at the opposite side of the room. In Section 4.3.1 we use a concept of spatial proximity that holds for the particular case considered and helps the interpretation of the results.

The second metric is a Euclidean distance measure, but the classical mean square error of the distance is not appropriate for our goals. The localization NN can be trained as a classifier, but it can also be designed to output a position in the 2D Cartesian space if it is trained on a regular grid with dense training spots ξ. However, the NN outputs a (x, y) position in a plane, while a human body occupies a fairly vast space in 3D, so that it is indeed not possible to univocally define the distance between the body and the (x, y) estimate. Let us call ρ a radius around the point estimate (x, y) of the NN, so that the circle of radius ρ and center (x, y) can be considered the estimated projection of the human body onto the 2D plane. The parameter ρ indicates the desired accuracy of the location estimation; if not otherwise stated, we usually consider $\rho = 0.25$ m, which corresponds to a high accuracy since the projection of a human body is hardly smaller than a circle with such a radius. On the other hand, $\rho = 1.0$ m corresponds instead to a fairly loose requirement in localization precision.

Given the estimate $e = (\hat{x}, \hat{y})$ made by the NN, and the coordinates (x_c, y_c) of the actual position p_c where the person stands, we construct a localization accuracy index $\mathcal{L}_{\text{R}}$ as follows

$$\mathcal{L}_{\text{R}} = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathcal{I}_d(i) \ ; \ \mathcal{I}_d(i) = \begin{cases} 1 & \text{if } d_i < \rho \\ 0.5 & \text{if } \rho \leq d_i < 2\rho \\ 0.25 & \text{if } 2\rho \leq d_i < 3\rho \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where $d_i = \sqrt{(\hat{x} - x_c)^2 + (\hat{y} - y_c)^2}$ is the Euclidean distance between the position estimate $(\hat{x}, \hat{y})$ and the coordinates of the position where the person is when the i-th sample is taken. Estimates are made assuming that the person is standing still in the same position p_c.

Clearly, $\mathcal{L}_{\text{R}} \in [0, 1]$, converges to 1 when all the estimates (one for every received frame) are within ρ from the true position, and converges to 0 when all estimates are at least three times ρ from the true position. Note that averaging over multiple position estimates would not help the attacker because, ideally, when the obfuscator is on, each position estimate could be any random point in the room, and the result of averaging the result over many frames would always return as most likely position the center of the room, which is ultimately a worthless guess.

Figure 4.4 graphically illustrates the relevance of the considered metric. The idea is that an attacker is interested in understanding where a person is with some level of accuracy described by ρ. When the target position is always estimated correctly (i.e., all samples lie within the desired accuracy), then the attacker is satisfied and $\mathcal{L}_{\text{R}} = 1$. On the other hand, if the estimated position is not only always outside the circle with radius ρ, but it lies so far away that even the two circles of radius ρ, centered in the estimate

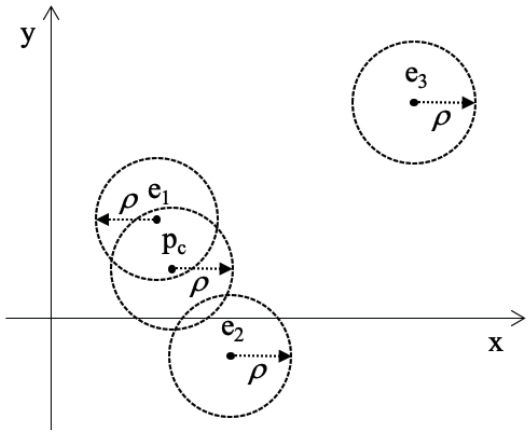


Figure 4.4: Examples of the $\mathcal{L}_{\mathrm{R}}$ metric: for estimate $e_1 \rightarrow \mathcal{L}_{\mathrm{R}} = 1$, for $e_2 \rightarrow \mathcal{L}_{\mathrm{R}} = 0.5$ and for $e_3 \rightarrow \mathcal{L}_{\mathrm{R}} = 0$, thus if the position is estimated from these tree samples $\mathcal{L}_{\mathrm{R}} = \frac{1.5}{3} = 0.5$.

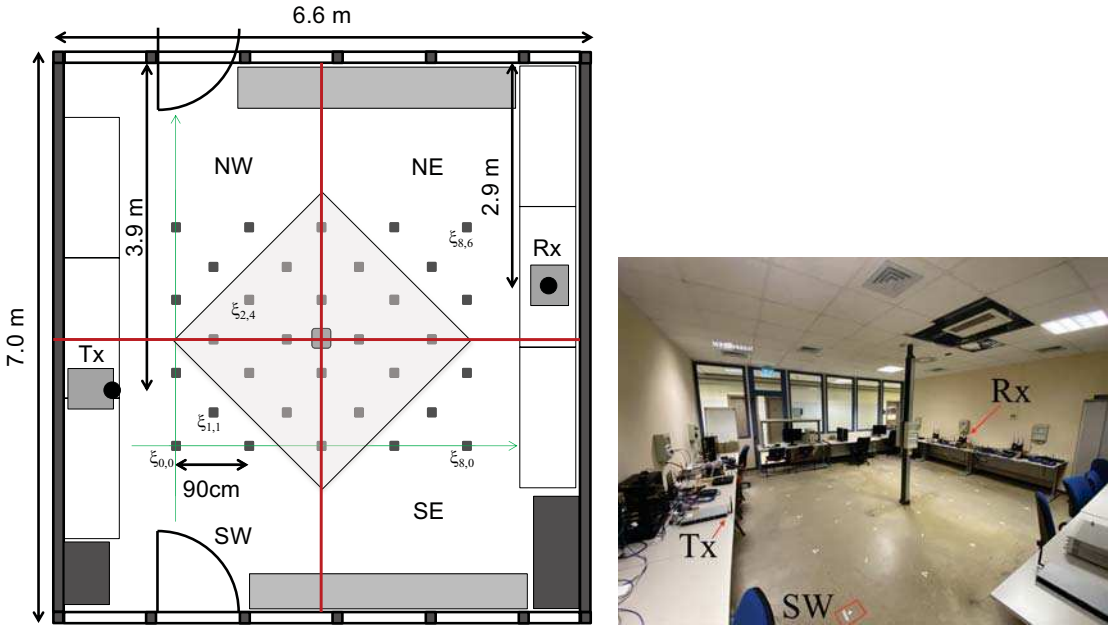
$e = (\hat{x}, \hat{y})$ and in the true position $p_c = (x_c, y_c)$ respectively, are separated by more than ρ, then the estimate is useless, hence $\mathcal{L}_{\mathrm{R}} = 0$. All the other cases lie in between. One could argue that also the true position of a person cannot be measured with absolute precision, so we assume that when a person stands on (covers with his feet) a specific location (x_c, y_c), which we assume to be his "exact" position.

To evaluate the quality of the location obfuscation technique, we can measure the accuracy $\mathcal{L}_{\mathrm{R}}$ of a random guess within the area in which the target is allowed to stay, i.e., the entire room minus the area covered by tables and furniture. Let us call this "useful" area in the room A_u; then, the $\mathcal{L}_{\mathrm{R}}$ for a random guess as a function of ρ is given by

$$\mathcal{L}_{\mathrm{R}}^{\mathrm{rand}} = \min\left(1, \frac{15\pi\rho^2}{4A_u}\right) \tag{4.3}$$

To compute Eq. (4.3) we considered all the point estimates $e \in A_u$, neglecting the border effect. With the term "border effect" we refer to the fact that by applying Eq. (4.2) for uniformly distributed guesses within A_u we end up considering also portions of the room that are outside of A_u. This effect is marginal for small values of ρ. For larger values of ρ, this effect causes an underestimation of the actual localization accuracy. Therefore, $\mathcal{L}_{\mathrm{R}}^{\mathrm{rand}}$ is a lower bound for the localization accuracy and gives a good reference to measure the quality of the randomization techniques we propose.

Finally, the third metric is focused on a privacy breaching scenario in which the attacker is not interested in determining the exact position of the target, but rather in which area of the room it is located. This metric measures the capability of the system to localize a person with high reliability, but with relaxed precision. The subject of interest does not stand still in a specific location, but he is moving around in one of the four quarters of the lab. Equation (4.4) defines the accuracy as a-posteriori probability, where P_l is the empirical probability that the attacker successfully infer the position of the person in the lab quarter where he actually is, N_l is the total number of position



(a) Plan of the lab.

(b) Photo of the lab.

Figure 4.5: Experimental setup, the square dots (white marks in the picture) on the floor are the training spots (ξ); the space is divided into four quadrants (`SW`,`NW`,`NE`,`SE`).

estimates, and $\mathcal{I}_l(i)$ is an indicator function that tells if the i-th location estimation is correct ($\mathcal{I}_l(i) = 1$) or wrong ($\mathcal{I}_l(i) = 0$).

$$P_l = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathcal{I}_l(i) \tag{4.4}$$

It is immediate to notice that if P_l tends to 1, also the time spent by the target person in the different parts of the lab is revealed to the attacker.

4.3 Experimental activity

We perform the experiments in the laboratory of the Advanced Networking Systems (ANS) research group at the University of Brescia. Figure 4.5a shows a plan of the laboratory with the position of the transmitter (Tx) and receiver (Rx) and the spots ξdesigning the target locations. The Euclidean coordinate system origin is set on the `SW` corner of the grid, as shown by the thin green axes. The training spots $\xi_{x,y}$ are numbered starting from the axis origin so that $\xi_{0,0}$ is in the origin and $\xi_{8,6}$ is the one in the `NE` corner.

The goal of the experiments is twofold: (i) to validate the results presented in [37] to guarantee that they can be reproduced with an independent implementation; (ii) to measure the actual capacity of the randomization technique presented in Section 4.2 to

obfuscate the actual position of a person. It is worth mentioning that our models do not consider several features that we deem marginal for our preliminary analysis. For example, we do not consider the body orientation of the victim to have a relevant effect on device-free localization. Similarly, also the type and the fabric of the victim's clothes are deemed irrelevant to the localization problem. However, to exclude any effect of such factors altogether, the data collection is performed with the target always facing in the same direction and wearing the same clothes, so that these factors can be safely excluded from our analysis.

Finally, besides position fingerprinting (i.e., classification over the different ξspots) and position estimation in terms of (x, y) coordinates (using the Euclidean distance setting), we also consider a third scenario in which the goal of the attacker is to identify which working areas (e.g., desk, workbench) are used by the staff at the laboratory. For instance, the attacker could be interested in determining the fraction of the work time dedicated to different tasks, corresponding to different locations in the lab. To this end, we divide the map in Fig. 4.5a into four sectors: `NW`, `NE`, `SE`, `SW`, delimited by red lines. Figure 4.5b shows a photo with the actual setup of the laboratory. The shaded square of 2 m edge at the center of the room in Fig. 4.5a is not considered for localization purposes in the third scenario, as it is an area where a person would not normally stay, but simply transit moving between the quadrants of the lab. As a side note, consider how simple it is to set up such an attack: the presence of an AP in a laboratory is very likely, a small CSI eavesdropper can be hidden easily, and the training can be done when nobody else is present. Given all this, then the attacker can very easily tell how much time the person spends in every part of the lab.

Implementation details

The localization system is implemented on commercial devices, namely the ASUS RT-AC86U routers. The firmware on the Wi-Fi chipset of the router is replaced by the one provided by the Nexmon CSI project [69]. With this tool, we can access all the 256 subcarriers of every Tx-Rx stream for Wi-Fi transmission with 80 MHz bandwidth. The received frames are saved to a capture file together with the corresponding CSI. During the post-processing phase, we extract CSI data from such packets and we feed the NN. On the transmit side, we use a USRP N300 SDR and the MATLAB WLAN toolbox to generate the frame and apply the CSI randomization scheme.[1] Note that no other modifications are required at the receiver to make the transmissions decodable.

At the transmitter, an infinite MATLAB loop generates Wi-Fi frames with a random payload, obfuscating the corresponding raw signal at every iteration as explained in Section 4.2. First, the MATLAB WLAN toolbox converts the packet into a sequence of IQ samples according to the VHT-PHY modulation with one spatial stream and 80 MHz bandwidth. Second, it parses the vector and separates the VHT-PHY symbols. For each of them, the software applies a specific procedure to isolate the 256 IQ samples of every OFDM symbol. Third, it runs the discrete Fourier transform (DFT) to get the

[1]The source code of the system is available at `https://github.com/ansresearch/csi-murder`

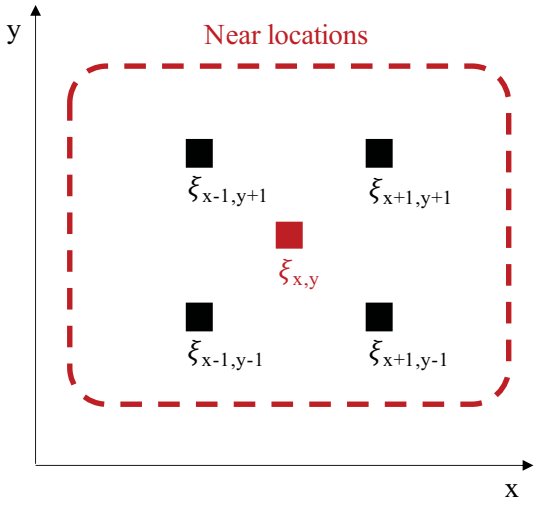


Figure 4.6: Visualization of the *near* locations for a generic point $\xi_{x,y}$ on the localization grid defined in Fig. 4.5a. All the other points on the grid are considered *far* from the point $\xi_{x,y}$

OFDM coefficients assigned to each carrier. Once in the frequency domain, the software multiplies each coefficient of the OFDM symbol by the value of the randomization filter at the same frequency. Finally, it generates a new sequence of 256 IQ samples by applying the IDFT. The correct structure of the symbol can ultimately be recovered by adding also inserting the Guard Interval.

4.3.1 Localization results with a fingerprinting approach

As we described in Section 4.2.1, measuring the overall accuracy of a classification algorithm in localization tasks has several shortcomings. In Fig. 4.6, we define a concept of spatial proximity in terms of *near* locations that can be used to further distinguish the severity of classification errors when trying to localize the victim. Points on the border of the grid can have less than four neighboring locations. For example, according to the definition in Fig. 4.6, a point on the border of the grid has only two neighbors, and the four vertices of the measurement grid ($\xi_{0,0}$, $\xi_{0,6}$, $\xi_{8,6}$ and $\xi_{8,0}$) have only one neighbor.

Localization results with and without the application of our CSI randomization technique are detailed in Table 4.1 for every target point ξ. The average accuracy over the 32 target positions of the chosen localization system is 78% when there is no obfuscation of the CSI. When our randomization technique is applied, the accuracy drops to below 5%, effectively disrupting CSI-based localization. Results are not uniform across all target positions. However, we think this is due to the inherent random effects induced by the obfuscation procedure rather than some intrinsic properties of the positions. It is remarkable that the CSI randomization scheme makes the localization procedure classify positions mostly far from the true one, thus not allowing even approximated estimates.

Table 4.1: Classification accuracy over the 32 target positions, with and without applying our CSI randomization technique. Classification errors for each position are divided between *near* and *far* locations, as defined in Fig. 4.6.

Without CSI randomization																
	$\xi_{0,0}$	$\xi_{2,0}$	$\xi_{4,0}$	$\xi_{6,0}$	$\xi_{8,0}$	$\xi_{1,1}$	$\xi_{3,1}$	$\xi_{5,1}$	$\xi_{7,1}$	$\xi_{0,2}$	$\xi_{2,2}$	$\xi_{4,2}$	$\xi_{6,2}$	$\xi_{8,2}$	$\xi_{1,3}$	$\xi_{3,3}$
Correct	100	92.9	5.7	90.0	0.0	100	97.1	100	98.6	58.6	100	60.0	57.1	95.7	100	75.7
Near	0.0	0.0	1.4	0.0	0.0	0.0	2.9	0.0	0.0	0.0	0.0	0.0	42.9	4.3	0.0	0.0
Far	0.0	7.1	92.9	10.0	100	0.0	0.0	0.0	1.4	41.4	0.0	40.0	0.0	0.0	0.0	24.3
	$\xi_{5,3}$	$\xi_{7,3}$	$\xi_{0,4}$	$\xi_{2,4}$	$\xi_{4,4}$	$\xi_{6,4}$	$\xi_{8,4}$	$\xi_{1,5}$	$\xi_{3,5}$	$\xi_{5,5}$	$\xi_{7,5}$	$\xi_{0,6}$	$\xi_{2,6}$	$\xi_{4,6}$	$\xi_{6,6}$	$\xi_{8,6}$
Correct	92.9	100	22.8	94.3	100	100	25.7	100	100	51.4	55.7	70.0	88.6	100	98.6	78.6
Near	1.4	0.0	52.9	1.4	0.0	0.0	2.9	0.0	0.0	7.1	0.0	0.0	11.4	0.0	0.0	0.0
Far	5.7	0.0	24.3	4.2	0.0	0.0	71.4	0.0	0.0	4.3	44.3	0.0	0.0	0.0	1.4	21.4
With CSI randomization																
	$\xi_{0,0}$	$\xi_{2,0}$	$\xi_{4,0}$	$\xi_{6,0}$	$\xi_{8,0}$	$\xi_{1,1}$	$\xi_{3,1}$	$\xi_{5,1}$	$\xi_{7,1}$	$\xi_{0,2}$	$\xi_{2,2}$	$\xi_{4,2}$	$\xi_{6,2}$	$\xi_{8,2}$	$\xi_{1,3}$	$\xi_{3,3}$
Correct	0.0	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0	2.9	0.0	0.0	0.0	0.0	40.0	0.0
Near	0.0	0.0	0.0	0.0	0.0	0.0	47.1	77.1	0.0	0.0	0.0	0.0	0.0	85.7	0.0	0.0
Far	100	100	0.0	100	100	100	52.9	22.9	100	87.1	100	8.6	100	14.3	60.0	100
	$\xi_{5,3}$	$\xi_{7,3}$	$\xi_{0,4}$	$\xi_{2,4}$	$\xi_{4,4}$	$\xi_{6,4}$	$\xi_{8,4}$	$\xi_{1,5}$	$\xi_{3,5}$	$\xi_{5,5}$	$\xi_{7,5}$	$\xi_{0,6}$	$\xi_{2,6}$	$\xi_{4,6}$	$\xi_{6,6}$	$\xi_{8,6}$
Correct	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Near	22.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	48.6	0.0	0.0
Far	77.1	100	100	100	100	100	100	100	100	100	100	100	100	51.4	100	100

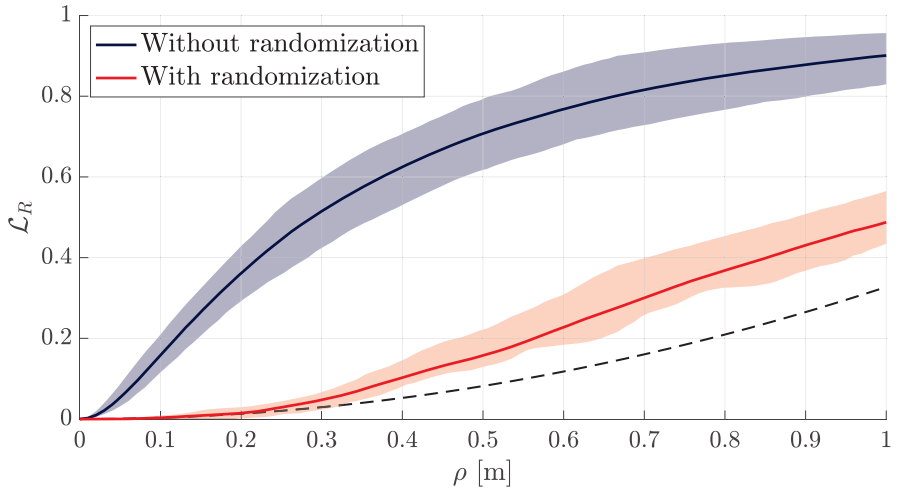


Figure 4.7: Localization performance according to metric $\mathcal{L}_{\mathrm{R}}$ for ρ ranging from 0 to 1. Solid lines report the average result, while the shaded areas are the envelope of all measures including using different antennas and positions of the receiver. The dashed line represents the theoretical result for uniformly distributed random guesses.

4.3.2 Localization results in the Cartesian space

The first insight in this case is focused on evaluating the localization capabilities of an attack using the metric described by Eq. (4.2) when the output layer produces an estimate in terms of (x, y) coordinates. Once again, we compare the performance obtained with and without the randomization procedure. In both cases, we train the NN with 700 packets for each one of the 32 positions highlighted in Fig. 4.5a and we test the localization on a different set of measures consisting of 150 packets per position collected at a different time. We capture CSI data from each of the four antennas available at the three receivers in the lab (visible in the picture of Fig. 4.5b) for a total of 12 CSI feeds. Interestingly, results for the three receivers are similar and it also turns out that training the NN with data from one single antenna or any combination of the four antennas for each of the three receivers does not have any significant impact on the results. Figure 4.7 reports the average results obtained considering the metric $\mathcal{L}_{\mathrm{R}}$. The solid line is the average for the 32 positions computed by considering all the CSI (average over 12 antennas). We also show the shaded regions between the worst and best-performing antenna, obtained again by averaging over the 32 positions. The localization reliability increases with ρ, but the most interesting cases for in-room localization are the ones with small ρ, i.e., values between 0.3 and 0.6. In particular, for $\rho = 0.3$, the average $\mathcal{L}_{\mathrm{R}}$ score is above 0.5 for the localization system but drops below 0.05 when CSI randomization is active. The benefit of using our randomization system is evident from the fact that the curves obtained while obfuscating the CSI are much closer to the black dashed line corresponding to the lower bound for uniformly distributed random guesses.

With respect to the last metric defined in Section 4.2.1, which represents the prob-

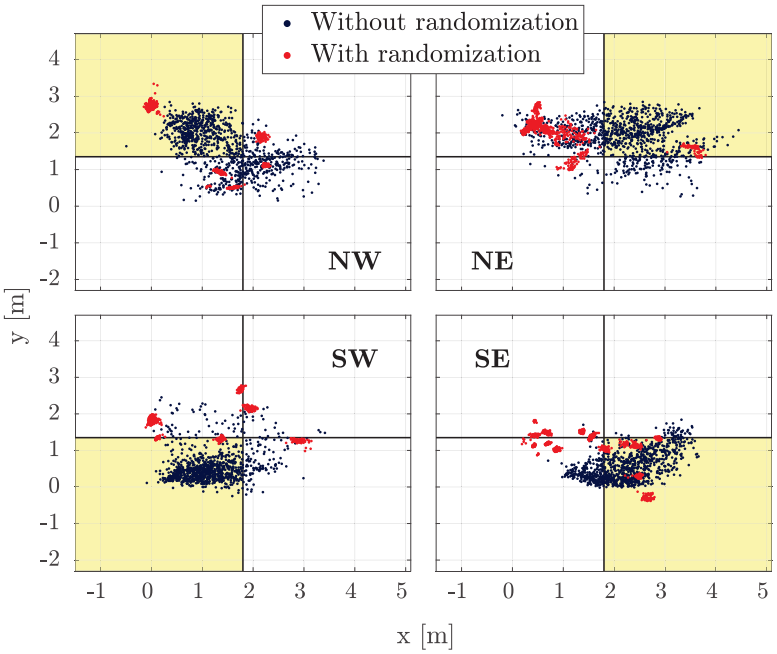


Figure 4.8: Positioning results considering different quarters of the lab. Each dot represents the position estimated by the attacker for each packet received. We report the estimates performed with and without CSI randomization using orange and purple dots respectively.

ability of an attacker locating the victim at least in the correct area of the lab, we subdivided the lab into four quadrants and we measured the performance of our system by comparing the value of P_l when the randomization is active or not and when the user is moving outside the shaded area in Fig. 4.5a. Without CSI randomization, the NN predicts positions that fall in the correct quadrant with 66% probability. Despite the result appearing quite low, from Fig. 4.8 we can clearly identify clusters of points in the correct quadrant that would help the attacker make a more sensible guess. However, we can see from the same figure that this analysis is not useful when randomization is active: in this case, in fact, the probability that the estimated position falls in the right quarter of the lab drops to 30%, just slightly above the random guess accuracy (25%).

To better assess the performance of the system with and without randomization, we ran a second experiment where we collected CSI data when the user was sitting at four different desks located at each corner of the lab as indicated in Fig. 4.9. In this experiment, the user can slightly move, i.e., by rotating over the chair's vertical axis, or by moving arms and hands on the desk. The only constraint is to stay within the circles reported in the figure. It is clear that without CSI obfuscation the NN predicts the positions with very high accuracy (they almost always fall within their circle); however, when randomization is applied the predicted position is almost always wrong.

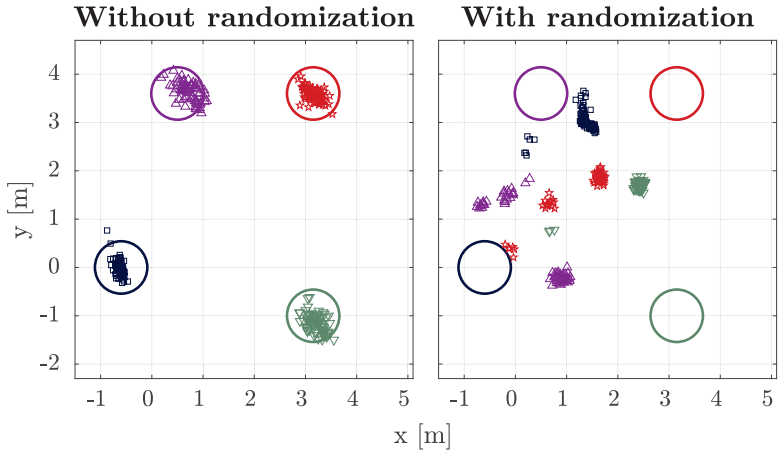


Figure 4.9: Localization results when the user is working at four different desks placed in the corners of the lab and not moving.

4.3.3 Impact on throughput

So far we have examined only CSI-based localization and the possibility of obfuscating the location information with an arbitrary manipulation of the OFDM symbols. In the experiments, we transmitted with the lowest-order modulation and coding scheme (MCS) (i.e., MCS0), which uses a binary phase-keying modulation (BPSK) and is very robust. However, it is important to investigate the communication performance of higher-order MCSs because they are more susceptible to channel errors. Therefore, we compute the packet delivery ratio (PDR) for all the MCSs transmitted with 80 MHz bandwidth (using the usual VHT-PHY and one spatial stream): we report in Fig. 4.10 the PDR for the three receivers when randomization is off (Clean) and on (Phase, Notches, Peaks). For completeness, we also measure the impact of phase- and notch-based manipulation even if they have not been used to obfuscate the victims' locations because their obfuscation performance was suboptimal.

It is clear from the figure that the three receivers have very good reception performance for all the MCSs with clean Wi-Fi frames, i.e., without CSI randomization. However, only robust MCSs retain acceptable PDR when the randomizing filter is applied. As the MCS increases, the modulation becomes more sensitive to distortions, and systematic errors introduced by the CSI randomization process may impair the correct decoding of frames (like Rx3) or completely stop reception (like Rx1). This further deteriorates when the MCS increases: MCS8 and MCS9 cannot be received in almost any location when the distortion is based on peaks. Instead, when the manipulation is based on random phase discontinuities or random notches in the transmit signal, it seems that the PDR remains higher. These preliminary results derive from a proof of concept rather than from a complete performance analysis, but we believe they represent an interesting starting point for future research on different CSI obfuscation schemes.

Let us analyze the reasons and discuss how this problem can be addressed, as it is

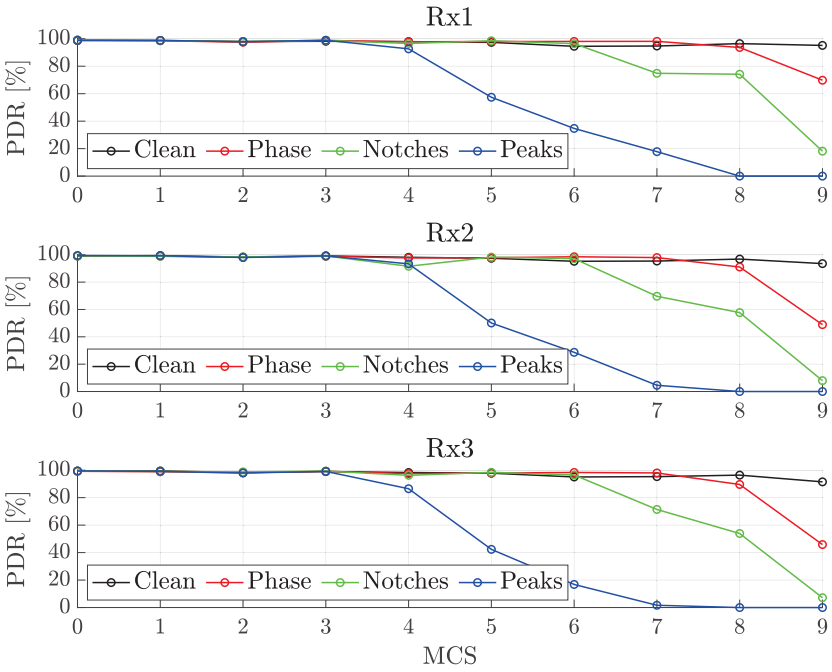


Figure 4.10: Impact of different CSI randomization techniques on the PDR at three different receivers for different values of the MCS.

clear that a localization obfuscation method should not disrupt communication capabilities. Eq. (4.5) describes the mathematical model of the signal at the receiver, where $S(F)$ is the signal spectrum at the output of the IDFT of the 802.11ac transmitter, $M(f)$ is the filtering function used for obfuscation (the "mask" in Fig. 4.2), and $H(f)$ is the channel frequency response, including attenuation, distortion, and multipath fading. All the quantities are considered in the frequency domain and complex-valued. The symbol $\odot$ represents the Hadamard product, i.e., the element-wise multiplication of two matrices of the same size.

$$R(f) = H(f) \odot M(f) \odot S(f) \tag{4.5}$$

From Eq. (4.5) it is clear that to preserve the communication performance, the channel response $H'(f) = H(f) \odot M(f)$ should satisfy some regularity constraints and cannot be any arbitrary channel response. The theoretical analysis of the properties of the "best" randomizing filter satisfying such properties is beyond the scope of this work and is still subject to investigation.

Chapter 5

Countermeasures against active Wi-Fi sensing

We have seen that we can conceive two different types of Wi-Fi sensing attacks whose difference lies in the assumptions we make about the attacker's capability. In particular, we refer to the ability of the attacker to control or not Wi-Fi transmissions: if he can control the transmission of Wi-Fi frames, the attack is *active*; otherwise, if he must rely only on traffic generated by legitimate devices, the attack is *passive*. The attacker might eventually use more than one receiver, presumably improving the localization performance by correlating the position estimation. However, such analysis is beyond the scope of this work, which focuses on countermeasures against Wi-Fi sensing instead.

In some sense, a passive system is more accessible as only a specialized receiver is needed to perform the localization. However, the frames used for localization must come from a transmitter in a fixed location (not necessarily known) because the change in the CSI determined by the moving transmitter will taint the collected fingerprints. This is not a problem in most cases, since APs are usually fixed and generate most of the traffic; thus, the localization device only needs to filter frames transmitted by the AP to achieve its goal. An active localization system, instead, allows the attacker to both transmit and receive Wi-Fi signals.

We already tackled the problem of localization obfuscation against *passive* eavesdroppers in the previous chapter. The obfuscation techniques presented so far are all based on some kind of manipulation of the CSI at the transmitter; therefore, we could imagine the existence of "secure" networks in which all nodes transmit obfuscated Wi-Fi frames. However, even such networks would not help against CSI sensing performed by active attackers. Indeed, while active attacks are more complex and detectable (they require the transmission of Wi-Fi frames that do not belong to a legitimate Basic Service Set (BSS)), there is no way to hinder the localization by manipulating the transmitted frames, because the transmitter is not controlled by legitimate users but by the attacker himself. In this chapter, we investigate a different system that can safeguard users' privacy against both passive and active attacks thanks to an adjustable signal reflector.

The localization technique we adopt in this chapter is the same as in the previous one.

We have already discussed the NN implementation in Section 1.2.3, a design that was inspired by the work in [27] and refined in [37]. The CNN-based localization system can work both with active and passive attacks, as long as the transmitter—either controlled by the attacker or by some other user depending on the case—stays in a fixed position. In both cases, the attack is mounted in two phases: first, the attacker trains the CNN using data collected with the help of another person standing in the target positions; then, when the victim enters the room, the attacker can use the trained model to associate the received CSI to the victim's position. The range and scope of the localization system are indeed relatively flexible. We have seen that the system can work in different settings both as a classifier of target locations (fingerprinting approach) and as a fine-grained estimator of $(x, y$ coordinates in the room. Localization accuracy becomes more fragile when the precision requirement is increased; thus, to validate the obfuscation technique in this chapter we decide to consider only a robust classification approach based on eight target locations.

5.1 Reflector design and requirements

The goal of the obfuscator is to blur location information without impairing communication. Differently from what we have already seen in Chapter 4, here we are interested in designing an active device that can prevent unauthorized localization against all types of attacks. The device must be able to randomly change the channel response "reflecting" the incoming signal with adequate amplification, delay, and phase distortion. This reflecting device should behave as an additional time-varying feature of the propagation environment so that the localization system cannot identify the person's position based on the CSI fingerprint. This can be achieved if such a device effectively modifies the propagation of Wi-Fi signals in the environment. We will call this device the *obfuscator* or equivalently the *reflector* since it conceptually reflects the Wi-Fi signal. The reflector can be any repeater mimicking a reflective surface or, in a more futuristic scenario, a reflective intelligent surface (e.g., a smart space [58, 70]) changing its properties under the control of a proper obfuscation function.

The reflector cannot operate only on non-legitimate frames simply because the reflection delay must be well below a single symbol duration, and it is impossible to read the Medium Access Control (MAC) addresses before actuating the reflection. Moreover, given that the CSI information is embedded in preambles, it would not be convenient to stop the controlled reflection when the MAC address is processed because this would alter the channel coherence and harm the correct reception of legitimate Wi-Fi frames.

We reckon that a good obfuscation system must meet three fundamental requirements: (i) it does not hamper communication performance (ideally, with an active device, it should even enhance it); (ii) it alters the signal in ways that are compatible with people's movements; (iii) its random behavior cannot be reversed or understood in a reasonable amount of time.

The last two requirements are needed to guarantee that even sophisticated analysis cannot filter out the random distortions and void the CSI obfuscation scheme. In other

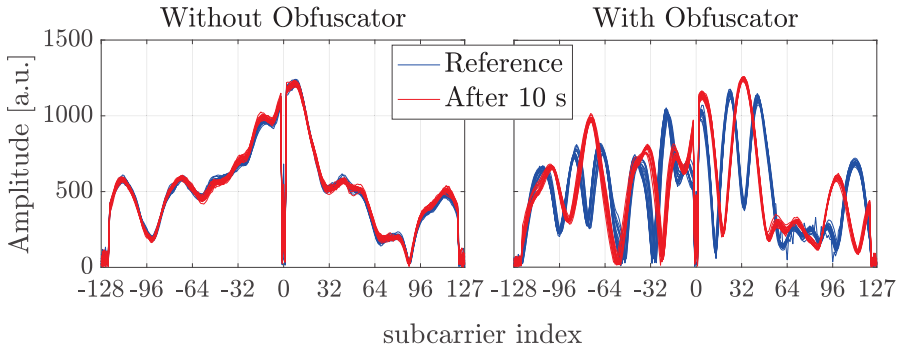


Figure 5.1: Effect of the active obfuscation on the CSI. In both cases, the victim stands still in one position; however, when the obfuscator actively relays the received signal, the channel conditions appear to change over time.

words, the attacker should not get any information that is significantly better than a random guess.

5.2 Randomized reflection strategies

The goal of the obfuscator is to add one or more "reflecting paths" into the propagation environment so that the behavior of the channel that embeds the information on the victim's location in the CSI becomes blurred and time-varying, confusing the localization system. At the same time, the channel distortion must remain plausible, allowing the equalizer at the receiver to compensate for the distortion correctly. Legitimate frames should be received without reducing communication performance.

To fix the ideas on what an active obfuscation shall achieve, consider Fig. 5.1. On the left-hand side, there are 100 CSI samples collected as reference (in blue) and 100 collected after 10 s when a person is standing still in a given position and there is no obfuscation. On the right-hand side, instead, we repeat the same experiment with the obfuscator turned on. It is clear that the obfuscator significantly alters the propagation environment (the blue lines are very different in the two plots); after just 10 s, the red lines tell a different propagation story. In any case, we can conjecture that any localization technique will have a hard time in fingerprinting and classifying positions if the CSI keep changing. Ideally, the outcome of the obfuscation at the receiver should be indistinguishable from a standard channel response, both as a distribution of attenuation and phase jumps in frequency and as a correlation in time. However, it is unclear if this is achievable because there is a lack of experimental studies that adequately characterize the stochastic properties of the CSI.

The simplest approach we can conceive for obfuscating the CSI is to let the reflector apply a pseudo-random delay that swipes between two extreme values. However, this method has the shortcoming that prolonged observations may reveal the periodicity in the random process. A better approach would be to use a delay miming a random

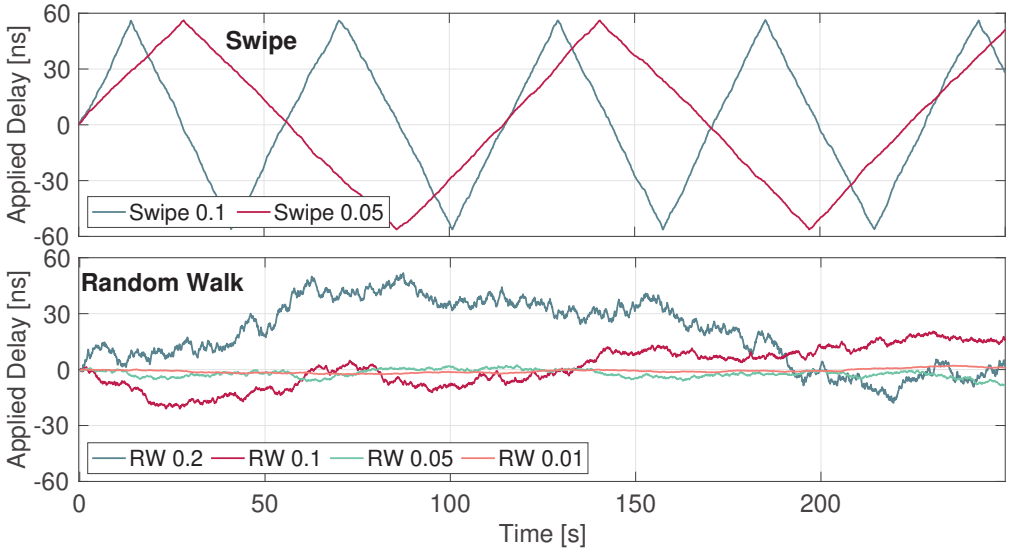


Figure 5.2: Processes driving the delay added by an active obfuscator, the processes are centered around zero, then a proper offset τ_{off} is added depending on the use cases. The upper plot shows a random delay swipe with $\delta = 0.1$ and 0.05, respectively; the lower plot shows the outcome of a Markov-Uniform process with four different δ parameters.

person's walk inside the room. Understanding what delay corresponds to each person's position in the room is generally unfeasible, as it would require the characterization of every space we want to protect and the tuning of the obfuscation algorithm. A solution that appears feasible and robust and that may mimic the movements of people in a room is a random walk of the delay itself, which can be efficiently implemented as a Markovian random process, as defined by Eq. (5.2), whose output is the delay to be added to the next reflected Wi-Fi frame.

The difference between the two approaches is clear if we look at Fig. 5.2 where the upper plot shows the output of a simple random swipe—see also Eq. (5.1)—with two different values of the maximum delay step δ while the bottom one reports the output of the Markov-Uniform random process—see also Eq. (5.2)—with four different values of δ. The processes are centered around zero; then, in our implementation, a constant τ_{off} will be added depending on the scenario to make the output reasonable. For instance, if the scenario is based on an active relay or reflector, then there is a minimum positive delay corresponding to the propagation time plus the minimum time required by the device to relay/reflect the signal. Let us define the minimum and the maximum admitted delays $\tau_{\min}$ and $\tau_{\max}$ so that, for any frame, the actual delay introduced is $\tau_{\min} \leq \tau \leq \tau_{\max}$. Setting these two limits and τ_{off} is crucial to maintain the additional delay added by the reflector within bounds that are coherent with the ambient to be protected, as already commented. Since the movement of a person in a room is based on time and not on transmitted frames, we also define a time interval Δt, used to pilot the random delay evolution; $\tau(i)$ means the delay added at the i-th time interval.

A random delay swipe, including τ_{off}, is defined as

$$\tau(i) = \tau_{\text{off}} + \tau(i-1) + I(\tau) \cdot \delta U_{[0,1]} \tag{5.1}$$

where $I(\tau)$ is an indicator function that takes the value $+1$ if τ is increasing and -1 if it is decreasing, $U_{[0,1]}$ is a uniform random variable with support $[0, 1]$, and δ is the maximum allowed difference between additional delays in adjacent time intervals. Switching between increasing and decreasing behavior happens when τ reaches $\tau_{\min}$ and $\tau_{\max}$, respectively.

The Markov-Uniform delay is instead computed as

$$\tau = \left[\tau_{\text{off}} + \tau(i-1) + \delta U_{[-1,1]}\right]_{\tau_{\min}}^{\tau_{\max}} \tag{5.2}$$

where $U_{[-1,1]}$ is a uniform random variable with support $[-1, 1]$, and $[\cdot]_{\tau_{\min}}^{\tau_{\max}}$ indicates the clipping between $\tau_{\min}$ and $\tau_{\max}$. Other types of Markovian processes can be used for obfuscation, and the analysis to identify the one that guarantees optimal obfuscation is an interesting future work, with the possibility of finding a theoretical optimum that minimizes, or even nullify, the location information carried by the CSI.

In the case of the simple random swipe of Eq. (5.1), δ controls the average period of the swipe, which is $T_s = \dfrac{2(\tau_{\max} - \tau_{\min})}{\delta}$. In the Markov-Uniform case considered in Eq. (5.2), instead, δ controls the probability that the process is clipped. The plots in Fig. 5.2 show sample realizations of both the swipe and the Markov random delays for different values of δ. In all the experiments with the swipe method, we report results only for $\delta = 0.1$ because we have verified that this value has little influence on the results; hence, the parameter is not repeated every time. For the Markov case, instead, we report results for the four values $\delta = 0.01, 0.05, 0.1, 0.2$ because we want to analyze the impact of the delay variation amplitude on both the obfuscation performance and the communication performance. Of particular interest are the two extreme values $\delta = 0.01$ and 0.2, because from the bottom plot of Fig. 5.2 one might argue that $\delta = 0.01$ has too slight variations to obfuscate the location. At the same time, $\delta = 0.2$ might be too "noisy" to guarantee good communication performance.

5.3 Implementation

Depending on the considered attack scenario, we can assume the transmitter is a device on which the attacker has complete control (active attack) or no control at all (passive attack). Implementing the obfuscation mechanism in hardware is unfortunately beyond the possibilities of our lab, let alone realizing a controllable reflective intelligent surface. Moreover, we believe that such an expensive endeavor is only justified once it is clear that the proposed technique works and is tamper-proof. Thus, we again resort to SDRs to realize our proof-of-concept implementation.

Our setup consists of two SDRs—namely two USRP N300, one for the transmitter and one for the obfuscator—and a commercial AP (Asus RT-AC86U) used to collect

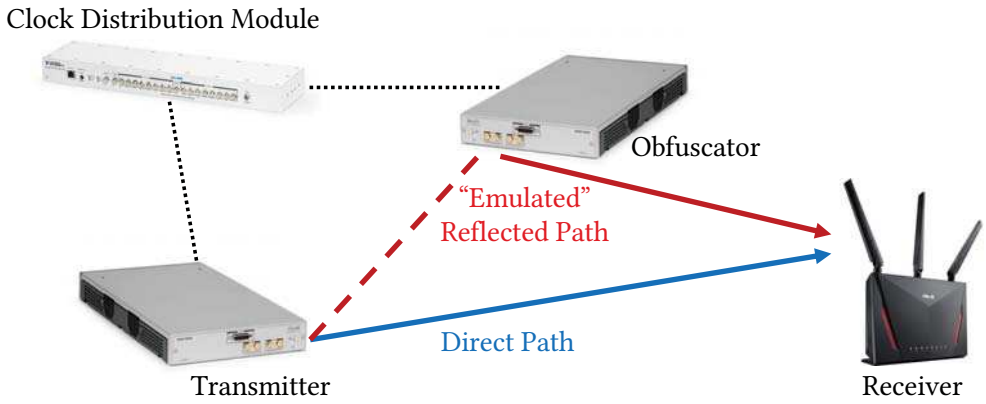


Figure 5.3: Schematic representation of the experimental setup; the obfuscator acts as a configurable reflector.

the CSI. The SDR transmitter keeps sending Wi-Fi frames generated using the Matlab WLAN Toolbox at a constant rate of approximately one frame every 10 ms. The receiver encloses a Broadcom chipset from which we can extract the CSI data points using the tools provided by the Nexmon project [33]. Finally, the localization system works offline on stored CSI data points.

Implementing the real-time Wi-Fi signal reflector in software is doable but still tricky: the latency introduced by typical SDR systems cannot meet the strict timing requirements, and some kind of hardware-accelerated processing is necessary. For this reason, we resort to a gimmick, as shown in Fig. 5.3. The two SDRs—one playing as the transmitter and the other as the obfuscator—are synchronized through a common clock source. This common clock source is provided in our case by an Ettus Octoclock-G and consists of both a 10 MHz reference and a 1 Hz pulse signal that allows almost perfect synchronization. Once the two SDRs are synchronized, the obfuscator can emulate the effects of a reflected path by re-transmitting the original signal with a delay, as discussed in Section 5.2.

The easiest way to apply a time delay to the signal transmitted by the obfuscator is to shift the sequence of IQ samples sent by the obfuscator by a certain number of IQ samples. Despite being simple, this solution has a significant limitation. Since the radio transmits samples at a fixed rate, the available bandwidth determines the time between two consecutive samples and the maximum granularity of the delay. In our implementation, the transmission rate of the USRP N300 SDRs is 125 MSample/s, which corresponds to a sampling period of 8 ns; therefore, the minimum delay corresponding to a shift of the sequence by one sample is equivalent to a path difference of approximately 2.4 m. Moreover, this method can only emulate propagation delays multiples of the minimum delay, which would be an inconvenient limitation for our obfuscation system.

A better solution is to process the sequence of IQ samples in the frequency domain. Given the digital signal $x[n]$ and assuming that all the conditions on proper sampling are satisfied, we apply the DFT to get its representation in the frequency domain $X[k]$ (Eq. (5.3)). Then, we modulate the digital frequencies by a complex exponential as in

Eq. (5.4) to obtain $X_d[k]$, which is the frequency domain representation of the delayed digital signal $x_d[n]$ obtained applying the Inverse DFT (Eq. (5.5)). The effect of these operations is to produce a new sequence of samples $x_d[n]$ representing a signal that is a copy of $x[n]$ delayed by a generic value τ. In this case, τ can also be a fraction of the sampling period, which allows an arbitrary resolution when tuning the delay introduced by the obfuscator.

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot \exp\left(-j\frac{2\pi}{N}kn\right),\ k = \{0, ..., N-1\} \tag{5.3}$$

$$X_d[k] = X[k] \cdot \exp\left(-j\frac{2\pi}{N}k\tau\right) \tag{5.4}$$

$$x_d[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_d[k] \cdot \exp\left(j\frac{2\pi}{N}kn\right),\ n = \{0, ..., N-1\} \tag{5.5}$$

As already discussed in Section 5.2, the delay τ changes over time and can vary between a minimum value $\tau_{\min}$ and a maximum value $\tau_{\max}$ as defined in Eqs. (5.1) and (5.2). The obfuscator reflects the incoming Wi-Fi signals; hence, the reflected signal in Fig. 5.3 will always be transmitted *after* the original signal. Therefore, to emulate its operation, we have arbitrarily chosen an offset delay τ_{off} of 88 ns when generating the delay processes reported in Fig. 5.2. The applied delay τ is updated every $\Delta t = 100$ ms—following either Eq. (5.1) or Eq. (5.2)—and it can range between $\tau_{\min} = 32$ ns and $\tau_{\max} = 144$ ns. All these values are arbitrary in our setup, but they can be tuned based on the expected performance of the emulated system. It is interesting to notice that the time delays we are considering are so tiny (tens of nanoseconds) that they do not affect the Wi-Fi MAC layer. However, they still have a considerable effect on the physical properties of the communication channel.

From a technical perspective, the implementation of the system does not change whether we consider an active or passive attack scenario since, in both cases, the obfuscator can act as a reflector following the same rules. However, to further explore the possibilities of our obfuscation mechanism, we would like to consider a different implementation that can only work when dealing with passive attack scenarios. Let us imagine that the transmitter and the obfuscator can cooperate in obfuscating the CSI; for instance, they can be two radio frontends driven by a single controller. We notice that in such case the obfuscator can even transmit the signal *before* the transmitter once the two devices are synchronized, i.e., τ can take negative values. For this reason, when discussing the passive attack scenario, we set $\tau_{\text{off}} = 0$ ns, $\tau_{\min} = -56$ ns, and $\tau_{\max} = 56$ ns to explore the effect of this configuration.

5.4 Scenario and measures

As in the previous chapters, we run the experiments in the laboratory of the Advanced Networking Systems (ANS) research group at the University of Brescia. The plan of the laboratory, with the positions of the transmitting and receiving nodes, is shown in

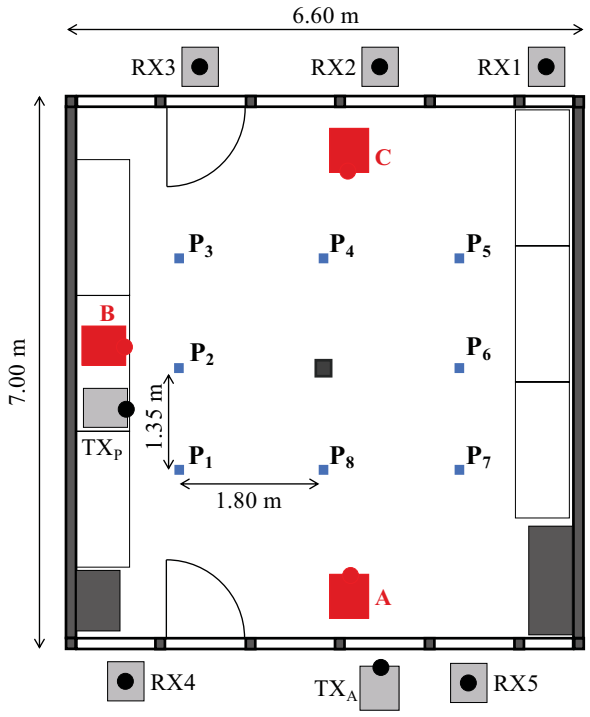


Figure 5.4: Plan of the lab in which localization experiments are performed. The small square dots represent the target locations of the victim. The red 'shadows' labeled A, B, and C are the locations of the obfuscator in different scenarios. Five different positions for the receiver are also considered, while the transmitter is inside (TX_P) for the passive attacks and outside (TX_A) for the active attacks.

Fig. 5.4. Here, we distinguish between the cases of active and passive attacks. The transmitter is outside the room when the attack is active (TX_A), and the attacker controls it. In the case of passive attacks, the transmitter is instead inside the room (TX_P), and the attacker cannot control it. All the receivers, controlled by the attacker, are placed outside the room on two opposite sides. The five different positions help explore whether the relative positions of the transmitter and receiver influence the localization and the obfuscation. For instance, RX5 should have abysmal performance in localization during the active attack, as it is too close to the transmitter (TX_A) for the EM environment inside the room to have a meaningful effect on the received CSI. We assume that the victim is standing inside the room in one of eight possible spots, indicated by small blue squares in Fig. 5.4 and enumerated from 1 to 8.

As indicated in Fig. 5.4, we consider three different positions of the obfuscator, which have a different relative positioning with respect to the transmitter depending on the attack scenario. In the active attack scenario, the transmitter is TX_A, and the obfuscator is: A) in front of the transmitter; B) at a 45° angle from TX_A and the receiver RX2; and C) in front of RX2, on the line of sight with TX_A. In the passive attack scenario, the transmitter is TX_P, and the obfuscator is: A) at a 45° angle from TX_P and on the line of sight with RX5; B) right on the side of the transmitter TX_P; and C) in front of RX2, at 45° degrees from TX_P.

The three positions of the obfuscator and the five positions of the receivers cover many different configurations, both for the case of active and passive attacks. At first glance, the position of RX4 and RX5 can seem "weird" in case of active attacks. One may think that with the receiver outside the room on the same side, the localization system cannot work, but this is not always the case, as results will show. Overall the setup consists of 15 possible configurations (three positions of the obfuscator times five positions of the receiver) for each attack type (active and passive). We use five random delay patterns for each configuration (one random swipe and four Markov processes). Considering also the measures when the obfuscator is off, overall, the experiments consist of $(15 + 1) \cdot 5 \cdot 2 = 160$ different configurations, giving good coverage of different layouts and scenarios.

We collect a few thousand samples (i.e., CSI data points associated with one Wi-Fi frame) for each target position and experiment; the exact number of samples can vary depending on several other parameters, but this is irrelevant. Overall, we use 8000 samples for the training phase and 8000 samples for the testing phase, i.e., 2000 samples for each target position. Training and testing samples are collected from two experiments with the same setup separated by several minutes to make the setup more realistic. Given the impossibility of leaving the experiments mounted overnight, we cannot say if training on one day and testing on another gives good localization results. Still, we deem that obfuscation will always work.

To assess the validity of the proposed CSI randomization technique, we compare the classification accuracy of the localization system when the obfuscator is on (in the three different positions) and when it is turned off for all five receivers. Training and testing are always performed with the same obfuscation setup, which is the most favorable case

for the attacker and the most challenging one for the obfuscator.

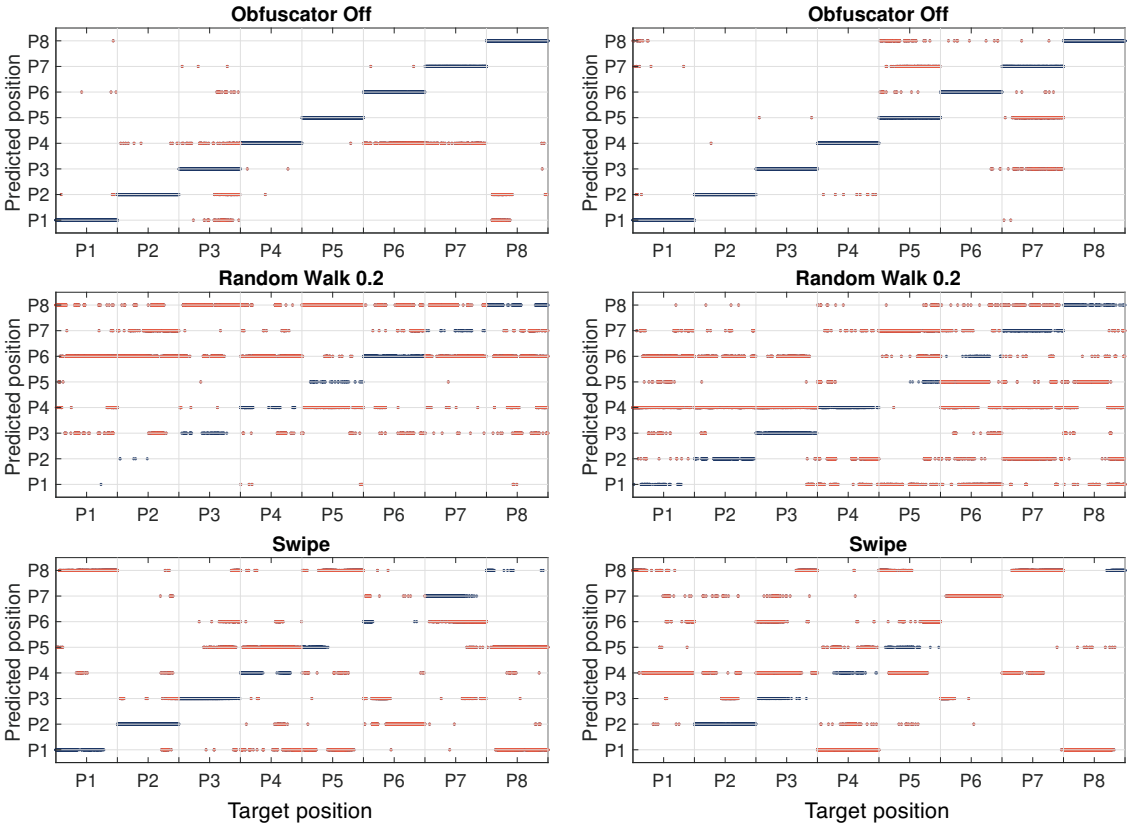
5.5 Experimental results

In this section, we summarize the properties of the obfuscation system, selecting the more meaningful ones for the following discussion.

5.5.1 Obfuscation performance

In Fig. 5.5, we present a set of plots showing the accuracy of the localization classification in several different situations; Fig. 5.5a refers to an active attack, while Fig. 5.5b to a passive one. Both figures consider RX1 and the obfuscator in position C) (see Fig. 5.4). Results for other receivers and obfuscator positions follow a similar pattern. As discussed in Section 5.3, when both the transmitter and the reflector are not controlled by the attacker (passive attack scenario), our prototype implementation allows centering the additional delay around zero. This means that, in some cases, the "reflected" copy may be transmitted before the original one. Indeed, given the setup based on two identical devices, it is impossible to state which one is acting as the transmitter and which is the obfuscator like in a real *smart space*.

Both in Fig. 5.6 and Fig. 5.7, the x-axis identifies the actual location and the y-axis the location estimated by the localization system; thus, the plots report on the diagonal, as blue dots, the correct estimates, while all the red dots outside the diagonal are misclassifications. For each position, we plot 1000 dots, i.e., position estimates. These plots represent the same information as a confusion matrix: they miss the precision of numbers, but we argue that they are easier to appreciate at a glance. The top plot refers to the benchmark of localization without obfuscation, and it is clear that localization is very effective, albeit not perfect: the accuracy is, on average, above 90%. The middle and bottom plots report the outcome when the obfuscator is active, with the Markov process and with the swipe-based delay, respectively. Obfuscation is effective in both cases, but the Markov process generally ensures a better dispersion of the misclassified position. The dispersion is not uniform, i.e., when the localization system makes a mistake, the distribution of the errors is not evenly distributed on all possible positions, as a perfect obfuscator should do. The reasons are many, but they can all be reconducted to the complexity of the scenario and the behavior of the CNN. In the end, the CNN makes a decision based on the similarity of the CSI received. Since the measures are taken in a relatively short time of tens of seconds per position, the randomness introduced has enough correlation to make the CNN decide for some positions with a higher probability. It must be noted that introducing a completely uncorrelated random delay is inappropriate because "white noise" can be filtered out given enough observations. One last remark regards the simple swipe technique. The technique works reasonably well, but the cases of positions P1, P6, and P7 in Fig. 5.5b highlight a specific behavior that can probably be used to overcome the effects of the obfuscation in the long run. The errors in these cases are almost deterministic. This is probably due to a coincidence, for



(a) Active attack.

(b) Passive attack.

Figure 5.5: Classification results for RX1 with the obfuscator placed in position C. Correct and misclassified location estimates are indicated with blue and red dots respectively. Top: obfuscator off; middle: Markov process with $\delta = 0.2$; bottom: swipe.

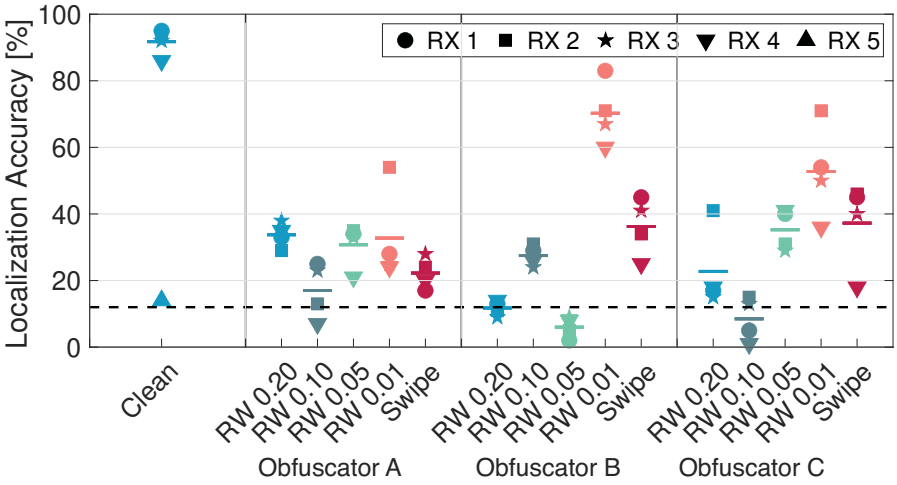


Figure 5.6: Active attack. Accuracy of the localization system for all the experiments. For each experiment, the colored markers are the accuracy at each receiver, while the horizontal line is the average. The dashed horizontal line at 12.5% marks the random guess over eight positions. RX5 is excluded from these results because of its peculiar position; the relative marker (blue triangle) for the clean experiment shows that, even in this case, the localization outcome is just a random guess.

which the random delay introduced by the reflector has a very similar effect in P4, P7, and P8 during the training phase to the one in P1, P6, and P7 during the testing phase.

Having analyzed in detail the behavior of the localization system and the obfuscation countermeasures in some cases, we can now discuss the overall results. Figures 5.6 and 5.7 report the aggregated results of all the experiments for the active and passive attacks, respectively. First, let us comment on the localization performance when the obfuscator is off, named *clean* in the figures. The average accuracy is around 90%, with minor variations depending on the receiver's position. Accuracy may vary from one position to another, but the analysis of all the detailed results, as done for a single case in Figures 5.5a and 5.5b, indicate that these variations are random from one experiment to another, most probably due to how the CNN interprets the CSI during training. Indeed, CSI-based localization has never been proven to have higher accuracy in complex scenarios; thus, these results align with the literature. One note is due for RX5 performance in active attacks. As we expected, the localization system in RX5 cannot correctly localize the target due to the powerful direct path between the transmitter and the receiver. The upward triangle for the clean experiment in Figure 5.5a shows that the localization outcome is just a random guess, so we excluded RX5 for all the active attack experiments.

When the obfuscator is on, the localization accuracy drops both for active and passive attacks, which is very interesting as it shows that a smart space with adjustable reflective devices can successfully preserve privacy in the face of different attacks. The performance of the obfuscator is still not ideal because only for a few experiments the

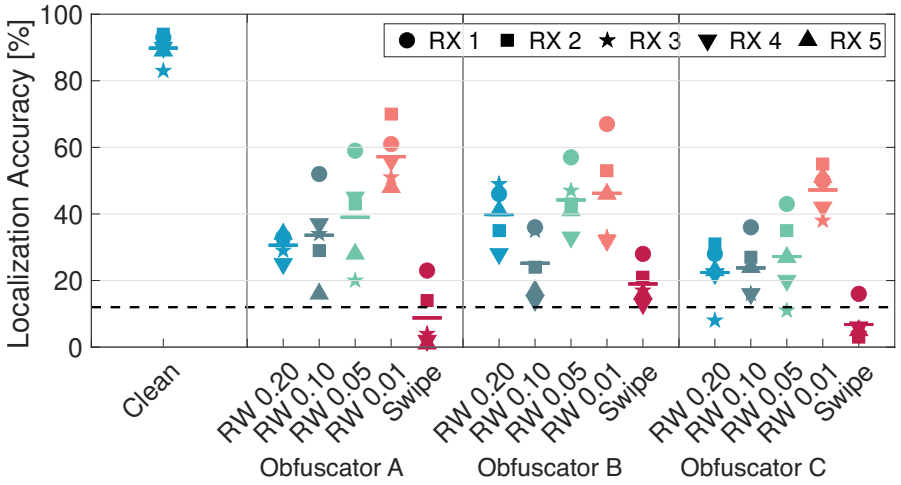


Figure 5.7: Passive attack. Accuracy of the localization system for all the experiments. For every experiment, the colored markers are the accuracy at each receiver, while the horizontal line is the average. The dashed horizontal line at 12.5% marks the random guess over eight positions.

output of the localization system is comparable to a random guess. However, for nearly all experiments, the average accuracy is below 40%, a value that makes the system hardly usable for any attacker. In both figures, the system based on the Markov process is called RW, meaning that the delay selection process is a Random Walk. Interestingly, even if it is less efficient than the other cases, setting $\delta = 0.01$ (RW 0.01) does significantly disturb the localization system, even if the delay added by the reflector is hardly detectable (Fig. 5.2).

From all the results, it is difficult to draw a general conclusion on the best possible location for the obfuscator or even the best place for the attacker to place its nodes, knowing where the reflector is. This holds both for active and passive attacks. In the case of a passive attack, the receiver's position seems to have a higher impact on the misclassification (the markers look more spread). However, there is no clear correlation between the actual positions of the receiver and the obfuscator, as the ranking between the different receivers changes from one configuration to the other. The swipe method, disregarding other drawbacks, appears to work particularly well for the passive attack. We think this is accidental, and further experiments would highlight that any of the many configurations seems to work particularly well, to change the ranking if a new configuration is tested. The fundamental observation remains that a simple reflector can protect people's privacy against unauthorized surveillance, opening one more application for EM smart spaces and granting that Wi-Fi-like systems can be safely used also in the future.

5.5.2 Impact on throughput

Protecting users' privacy is useless if legitimate services get destroyed. Since we imagine a smart surface working on all types of Wi-Fi signals, regardless of their origin, we run experiments to verify that the obfuscating node is not harming the communication throughput. To this end, we send 1000 frames from the transmitter and monitor how many of them we correctly decode at the receiver so that we can compute the PDR for different scenarios as reported in Figs. 5.8 and 5.9 for the transmitter in position TX_A and TX_P respectively. Notice that we are no longer concerned with the localization attack, as the interest is in legitimate frames. We use the exact positions of the transmitter and the receivers just for convenience.

The PDR is expected to decrease with the MCS order because frames with a higher MCS yield a high throughput but are more sensitive to noise and interference—especially in a complex environment such as our lab. This expected behavior is entirely confirmed for the clean case (obfuscator off), with the PDR dropping sharply for MCS higher than 6 in Fig. 5.8 and higher than 7 in Fig. 5.9. The difference between the two cases can be explained by the additional wall that frames received by RX1, RX2, and RX3 have to traverse. Indeed, for these cases, the PDR is much lower for these receivers than for RX4 and RX5. We also recall that receivers are placed outside the room, differently from the setup in Chapter 4. Thus, reception is more complex (we also report the presence of metallic frames around the windows boosting multipath effects). In general, higher-order MCS are known to be fragile, meaning that if there is no clear line-of-sight between transmitter and receiver, then the channel distortions are not entirely compensated for by the equalizer, leading to systematic decoding errors that the forward error correction code cannot correct. Finally, we recall that both MCS8 and MCS9 use a 256QAM modulation, which is extremely sensitive to distortions.

However, the most intriguing result is that the PDR improves when the obfuscator is active. This happens for all the transmitter and receiver positions and for all the proposed obfuscation techniques. The rationale is that the obfuscator works as a relay for the frame sent by the transmitter, creating a dominant second path and improving the quality of the link between the transmitter and receiver. In our prototype, the reflector implemented on the SDR device transmits an exact copy of the frame. This observation is generally valid for any scenario and layout: injecting two copies of the same frame separated by a sub-symbol delay, "doubles" (in a stochastic sense, as the instantaneous power depends on reflections) the signal power at the receiver, hence improving performance.

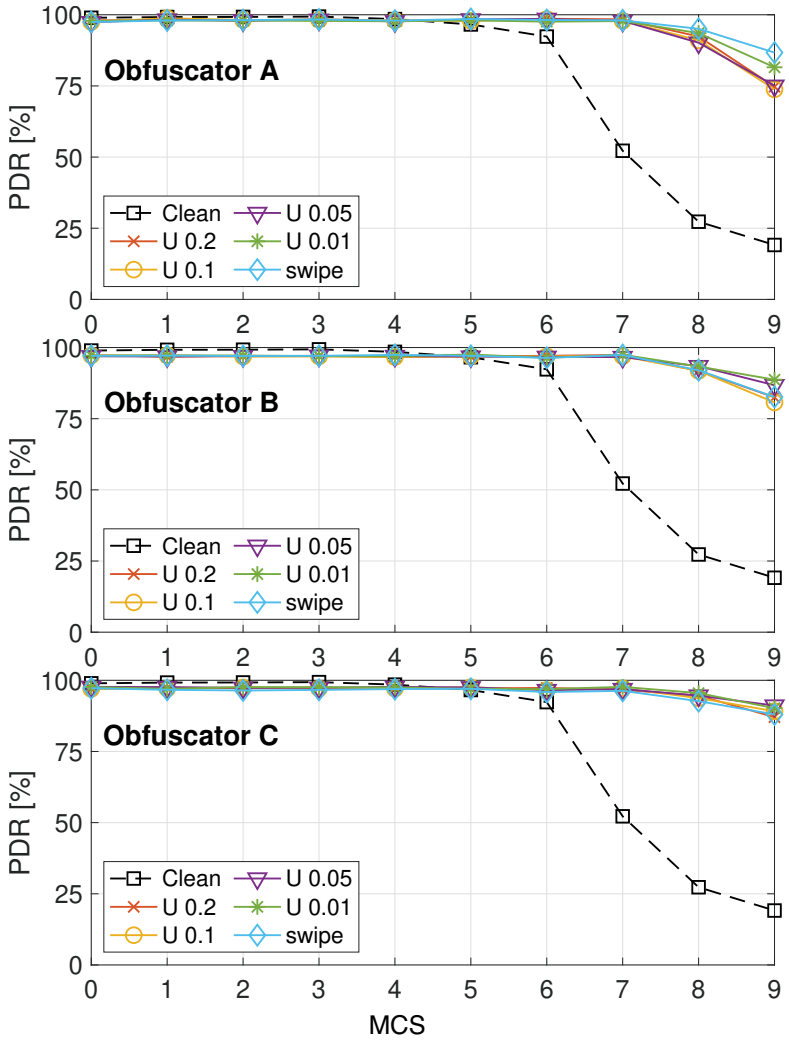


Figure 5.8: Active attack (transmitter TX_A outside the room). Average packet delivery ratio computed over all the receivers and all the positions of the obfuscator as a function of the MCS for all the obfuscation techniques tested and without the obfuscator.

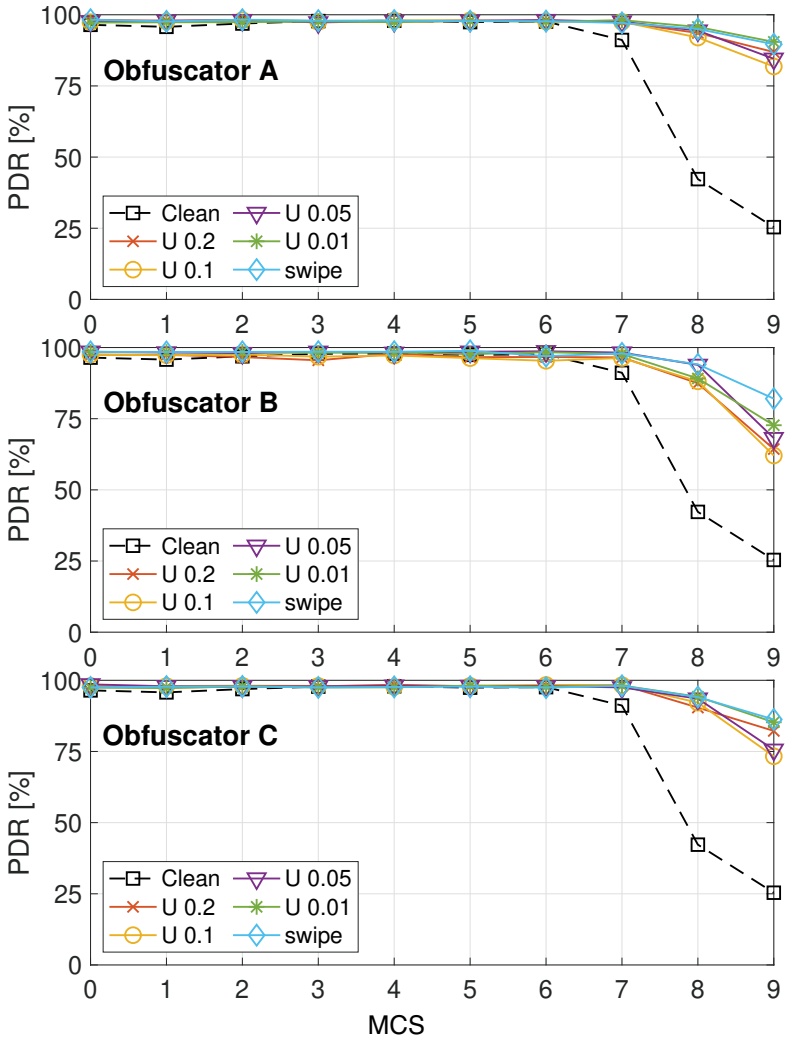


Figure 5.9: Passive attack (transmitter TX_P inside the room). Average packet delivery ratio computed over all the receivers and all the positions of the obfuscator as a function of the MCS for all the obfuscation techniques tested and without the obfuscator.

Conclusions

In this work, we have addressed several concerns about the privacy of Bluetooth and Wi-Fi users. As we have seen, modern computers and their remarkable computational capabilities empower novel techniques to capture and analyze wireless signals, fostering attacks that have long been considered infeasible or impractical.

In Chapter 2, we revealed that Classic Bluetooth is inadequate for ensuring anonymity and location privacy. To this end, we developed a full-band Bluetooth sniffer to quickly uncover the full address of Classic Bluetooth devices and re-identify target connections over many days. In our implementation, we used two affordable SDRs and a workstation with an Intel Core i7 CPU, proving that even attackers with limited resources can easily monitor all the Bluetooth devices in a specific area. One possible application would be scanning for specific Bluetooth connections, e.g., between a smartphone and a car infotainment system, to derive the traffic patterns of commuters at a road junction or infer the traveling habits of specific users. In Chapter 3, we obtained similar results for Bluetooth Low Energy and went even further by proving that it is possible to implement a SDR-based system that monitors all BLE traffic in real-time. A general-purpose GPU stands at the system's core, performing all the time-consuming demodulation and decoding operations in parallel on all the BLE channels. From a technical standpoint, it is difficult to fit the high latency of GPU applications into the tightly time-constrained radio operation. However, we showed that we could conceive some applications in which the two domains (parallel computing and wireless communications) can constructively work together to enable efficient, real-time analysis of wireless signals. Moreover, as the available computing power keeps increasing, we showed that Bluetooth packet analysis is no longer confined to single-channel sniffing. The results in Chapter 2 and Chapter 3 considerably lower the barrier for mounting passive attacks against protocols that use frequency-hopping techniques, calling for a critical investigation of the privacy features offered by Bluetooth.

The other part of this work focused on Wi-Fi technology instead. In the introductory chapter, we discussed how an attacker could potentially use Wi-Fi communications to localize unaware targets, even if they do not carry any Wi-Fi device. Even if we find Wi-Fi sensing applications compelling from a technological standpoint, we firmly believe that new technologies cannot come at the price of reducing people's privacy. Therefore, in Chapter 4, we proposed several techniques to "obfuscate" the CSI of Wi-Fi signals and effectively prevent unauthorized device-free localization applications. At the same

time, the proposed obfuscation techniques do not impair Wi-Fi communications, which is a significant shortcoming of classic jamming techniques. To the best of our knowledge, the idea of applying a pseudo-random pre-distortion to the transmit signal is novel in the scenario of privacy-preserving countermeasures against Wi-Fi sensing. Therefore, albeit promising, further validation of this technique must be obtained in different environments and for other classes of Wi-Fi sensing attacks. On a broader setting—not addressed in this dissertation—one could even imagine deploying networks that permit Wi-Fi sensing operations only for some authorized nodes. For example, cryptographic techniques can be used in such a scenario to communicate the exact CSI obfuscation sequence to some authorized CSI extractor. By exactly reversing the obfuscation scheme, it would be interesting to show that Wi-Fi sensing capabilities are restored.

We presented another approach to CSI obfuscation in Chapter 5, assuming that the attacker can also control Wi-Fi transmissions and must not rely on existing traffic only. Of course, under such assumptions, it is impossible to pre-distort the Wi-Fi signal to obfuscate salient features in the CSI and prevent device-free localization. The obfuscation is instead generated using a device emulating a programmable signal relay or a smart surface, thus paving the road toward the concept of smart spaces. This approach has been sometimes already considered in the literature, but it has never been applied to the problem of preventing device-free localization. Again, we have compared the obfuscation performance of several different CSI obfuscation algorithms and verified that they prevent localization without impairing legitimate Wi-Fi communications.

Finally, we want to stress that all the experiments described in this work have been validated empirically with prototypes we designed and implemented to address the different problems. We believe the proposed solutions can be widely adopted in principle and even find their way into future versions of wireless communication standards to protect users' privacy effectively. We notice that some considerations can be readily applied without any modification to the two standards we considered. For example, to avoid BLE device tracking over many days, Central and Peripherals might agree to drop their connection periodically and restart a new connection with a new, random, Access Address. On the other hand, for the application of Wi-Fi sensing countermeasures, Wi-Fi chipset manufacturers might provide users with the option of choosing whether they prefer maximum performance (no CSI obfuscation) or increased privacy (with some limitation on expected performance). Anyway, the main goal of this work is not to propose definitive solutions but to investigate yet-unexplored threats to security and privacy in wireless systems and raise awareness about them.

Bibliography

[1] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool Release: Gathering 802.11n Traces with Channel State Information," *SIGCOMM Computer Comm. Review*, vol. 41, no. 1, p. 53,

[2] E. C. Jimenez, P. K. Nakarmi, M. Naslund, and K. Norrman, "Subscription Identifier Privacy in 5G Systems," in *2017 International Conference on Selected Topics in Mobile and Wireless Networking, MoWNeT 2017*, 2017.

[3] Institute of Electrical and Electronics Engineers (IEEE), "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz," *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012)*, 2013. DOI: 10.1109/IEEESTD.2013.6687187.

[4] Bluetooth SIG, *Bluetooth Core Specification 5.3*, Jul. 2021.

[5] "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021. DOI: 10.1109/IEEESTD.2021.9363693.

[6] Bluetooth SIG. "2022 Bluetooth Market update." (2022), [Online]. Available: https://www.bluetooth.com/2022-market-update/ (visited on 09/20/2022).

[7] "IEEE Registration Authority FAQ." (2022), [Online]. Available: https://standards.ieee.org/faqs/regauth/ (visited on 09/2022).

[8] R. H. Barker, "Group synchronizing of binary digital sequences," *Communication Theory*, pp. 273–287, 1953.

[9] L. Charles Lee, *Error-Control Block Codes for Communications Engineers*. Artech House, 2000.

[10] H. Zhu, F. Xiao, L. Sun, R. Wang, and P. Yang, "R-TTWD: Robust Device-Free Through-The-Wall Detection of Moving Human With WiFi," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1090–1103, 2017. DOI: 10.1109/JSAC.2017.2679578.

[11] Y. Zheng *et al.*, "Zero-Effort Cross-Domain Gesture Recognition with Wi-Fi," in *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2019, pp. 313–325.

[12] F. Meneghello, D. Garlisi, N. Dal Fabbro, I. Tinnirello, and M. Rossi, "Sharp: Environment and person independent activity recognition with commodity ieee 802.11 access points," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2022. DOI: 10.1109/TMC.2022.3185681.

[13] M. Abbas, M. Elhamshary, H. Rizk, M. Torki, and M. Youssef, "WiDeep: WiFi-based Accurate and Robust Indoor Localization System using Deep Learning," in *Int. Conf. on Pervasive Computing and Communications (PerCom)*, IEEE, 2019.

[14] E. Khorov, I. Levitsky, and I. F. Akyildiz, "Current Status and Directions of IEEE 802.11be, the Future Wi-Fi 7," *IEEE Access*, vol. 8, pp. 88 664–88 688, May 2020.

[15] K. Chetty, G. Smith, and K. Woodbridge, "Through-the-Wall Sensing of Personnel Using Passive Bistatic WiFi Radar at Standoff Distances," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 50, no. 4, pp. 1218–1226, 2012.

[16] F. Adib and D. Katabi, "See through walls with WiFi!" In *Conf. of the Special Interest Group on Data Communication (SIGCOMM)*, Hong Kong, Aug. 2013: ACM, 2013, pp. 75–86.

[17] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: Indoor Localization via Channel Response," *ACM Comput. Surv.*, vol. 46, no. 2, 25:1–25:32, Dec. 2013.

[18] K. Wu, J. Xiao, Y. Yi, D. Chen, X. Luo, and L. Ni, "CSI-Based Indoor Localization," *Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1300–1309, Jul. 2013.

[19] M. Widmaier, M. Arnold, S. Dorner, S. Cammerer, and S. ten Brink, "Towards Practical Indoor Positioning Based on Massive MIMO Systems," in *90th IEEE Vehicular Technology Conf. (VTC2019-Fall)*, Honolulu, HI, USA, Nov. 2019, pp. 1–6.

[20] S. D. Bast, A. P. Guevara, and S. Pollin, "CSI-based Positioning in Massive MIMO systems using Convolutional Neural Networks," in *91st IEEE Vehicular Technology Conf. (VTC2020-Spring)*, Antwerp, Belgium, May 2020, pp. 1–5.

[21] S. Shi, S. Sigg, L. Chen, and Y. Ji, "Accurate Location Tracking From CSI-Based Passive Device-Free Probabilistic Fingerprinting," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 6, pp. 5217–5230, Jun. 2018.

[22] Cerar, Gregor and Švigelj, Aleš and Mohorčič, Mihael and Fortuna, Carolina and Javornik, Tomaž, "Improving CSI-based Massive MIMO Indoor Positioning using Convolutional Neural Network," in *Joint European Conf. on Networks and Communications 6G Summit (EuCNC/6G Summit)*, Porto, Portugal, Jun. 2021, pp. 276–281.

[23] X. Wang, X. Wang, and S. Mao, "Indoor Fingerprinting With Bimodal CSI Tensors: A Deep Residual Sharing Learning Approach," *IEEE Internet of Things Jou.*, vol. 8, no. 6, pp. 4498–4513, Mar. 2021.

[24] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based Fingerprinting for Indoor Localization: A Deep Learning Approach," *Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[25] G.-S. Wu and P.-H. Tseng, "A Deep Neural Network-Based Indoor Positioning Method using Channel State Information," in *Int. Conf. on Computing, Networking and Communications (ICNC)*, IEEE, 2018, pp. 290–294.

[26] T. F. Sanam and H. Godrich, "An Improved CSI Based Device Free Indoor Localization Using Machine Learning Based Classification Approach," in *26th European Signal Processing Conf. (EUSIPCO)*, IEEE, 2018, pp. 2390–2394.

[27] C. Cai, L. Deng, M. Zheng, and S. Li, "PILC: Passive Indoor Localization Based on Convolutional Neural Networks," in *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*, Wuhan, China: IEEE, Mar. 2018.

[28] Schmidt, Erik and Inupakutika, Devasena and Mundlamuri, Rahul and Akopian, David, "Sdr-fi: Deep-learning-based indoor positioning via software-defined radio," *IEEE Access*, vol. 7, pp. 145 784–145 797, Oct. 2019.

[29] A. Foliadis, M. H. C. Garcia, R. A. Stirling-Gallacher, and R. S. Thomä, "CSI-Based Localization with CNNs Exploiting Phase Information," in *IEEE Wireless Communications and Networking Conf. (WCNC)*, Nanjing, China, Mar. 2021, pp. 1–6.

[30] Z. Zhou, J. Yu, Z. Yang, and W. Gong, "MobiFi: Fast Deep-Learning based Localization using Mobile Wifi," in *IEEE Global Communications Conf. (GLOBECOM)*, Taipei, Taiwan, Dec. 2020, pp. 1–6.

[31] R. Prasad, *OFDM for Wireless Communications Systems*. London, UK: Artech House, 2004.

[32] Y. Xie, Z. Li, and M. Li, "Precise power delay profiling with commodity wifi," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15, Paris, France: ACM, 2015, pp. 53–64, ISBN: 978-1-4503-3619-2. DOI: 10.1145/2789168.2790124. [Online]. Available: http://doi.acm.org/10.1145/2789168.2790124.

[33] F. Gringoli, M. Schulz, J. Link, and M. Hollick, "Free your CSI: A channel state information extraction platform for modern Wi-Fi chipsets," in *13th Int. Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH '19)*, Los Cabos, Mexico: ACM, 2019, pp. 21–28.

[34] M. Schulz, F. Gringoli, J. Link, and M. Hollick, "Shadow Wi-Fi: Teaching smartphones to transmit raw signals and to extract channel state information to implement practical covert channels over Wi-Fi," in *Int. Conf. on Mobile Systems, Applications, and Services (MobiSys'18)*, Munich, Germany, June 2018: ACM, 2018, pp. 256–268.

[35] X. Guo, N. Ansari, F. Hu, Y. Shao, N. Elikplim, and L. Li, "A Survey on Fusion-Based Indoor Positioning," *Comm. Surveys & Tutorials*, vol. 22, no. 1, pp. 566–593, Mar. 2020.

[36] X. Wang, L. Gao, and S. Mao, "CSI Phase Fingerprinting for Indoor Localization with a Deep Learning Approach," *Internet of Things Journal*, vol. 3, no. 6, pp. 1113–1123, Dec. 2016.

[37] F. Kosterhon, "Device-Free Indoor Localization: A User-Privacy Perspective," M.S. thesis, Technische Universität Darmstadt, Secure Mobile Networking Lab, Department of Computer Science, Apr. 2020. DOI: `https://www.doi.org/10.13140/RG.2.2.25468.56965`.

[38] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, 2014. arXiv: `1412.6980 [cs.LG]`.

[39] S. Hay and R. Harle, "Bluetooth tracking without discoverability," in *Proc. International Symposium on Location and Context Awareness (LoCA)*, Tokyo, Japan, May 2009.

[40] M. S. Bargh and R. de Groote, "Indoor localization based on response rate of bluetooth inquiries," in *Proc. ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments*, ser. MELT '08, San Francisco, California, USA, 2008, pp. 49–54.

[41] V. Kostakos, "Using bluetooth to capture passenger trips on public transport buses," *Personal and Ubiquitous Computing*, pp. 1–13, 2008.

[42] D. Spill and A. Bittau, "Bluesniff: Eve meets alice and bluetooth.," in *USENIX WOOT*, 2007.

[43] *Project ubertooth*, 2019. [Online]. Available: `http://ubertooth.sourceforge.net/` (visited on 06/2019).

[44] M. Chernyshev, C. Valli, and M. Johnstone, "Revisiting Urban War Nibbling: Mobile Passive Discovery of Classic Bluetooth Devices Using Ubertooth One," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1625–1636, Jul. 2017.

[45] W. Albazrqaoe, J. Huang, and G. Xing, "Practical bluetooth traffic sniffing: Systems and privacy implications," in *Proc. ACM MobiSys*, 2016.

[46] S. Sarkar, J. Liu, and E. Jovanov, "A robust algorithm for sniffing ble long-lived connections in real-time," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9014318.

[47] "Micro:bit Educational Foundation." (2022), [Online]. Available: https://microbit.org/ (visited on 11/2022).

[48] "Bluefruit LE Sniffer." (2022), [Online]. Available: https://www.adafruit.com/product/2269 (visited on 11/2022).

[49] "BtleJack: a new Bluetooth Low Energy swiss-army knife." (2022), [Online]. Available: https://github.com/virtualabs/btlejack (visited on 11/2022).

[50] Ellisys. "Bluetooth analyzers comparison chart." (2019), (visited on 09/2019).

[51] Frontline. "Sodera Wide Band Bluetooth Protocol Analyzer." (2019), [Online]. Available: http://www.fte.com/products/sodera.aspx (visited on 09/2019).

[52] M. Ryan, "Bluetooth: With low energy comes low security," in *USENIX Workshop on Offensive Technologies*, Washington, D.C., 2013.

[53] K. Fawaz, K.-H. Kim, and K. G. Shin, "Protecting privacy of BLE device users," in *USENIX Security*, Aug. 2016.

[54] F. Gringoli, N. Ali, F. Guerrini, and P. Patras, "A Flexible Framework for Debugging IoT Wireless Applications," in *2018 Workshop on Metrology for Industry 4.0 and IoT*, Apr. 2018, pp. 230–235. DOI: 10.1109/METROI4.2018.8428337.

[55] Y. Ma, G. Zhou, and S. Wang, "Wifi sensing with channel state information: A survey," *ACM Comput. Surv.*, vol. 52, no. 3, Jun. 2019, ISSN: 0360-0300. DOI: 10.1145/3310194. [Online]. Available: https://doi.org/10.1145/3310194.

[56] Y. Qiao, O. Zhang, W. Zhou, K. Srinivasan, and A. Arora, "PhyCloak: Obfuscating Sensing from Communication Signals," in *13th Conf. on Networked Systems Design and Implementation*, USENIX Association, 2016, pp. 685–699.

[57] A. Welkie, L. Shangguan, J. Gummeson, W. Hu, and K. Jamieson, "Programmable Radio Environments for Smart Spaces," in *16th ACM Workshop on Hot Topics in Networks*, Palo Alto, CA, USA, Nov. 2017, pp. 36–42.

[58] C. Liaskos, A. Tsioliaridou, S. Nie, A. Pitsillides, S. Ioannidis, and I. F. Akyildiz, "On the Network-Layer Modeling and Configuration of Programmable Wireless Environments," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1696–1713, Jul. 2019.

[59] M. Schulz, F. Gringoli, D. Steinmetzer, M. Koch, and M. Hollick, "Massive Reactive Smartphone-Based Jamming Using Arbitrary Waveforms and Adaptive Power Control," in *10th ACM Conf. on Security and Privacy in Wireless and Mobile Networks (WiSec)*, Boston, MS, USA, 2017, pp. 111–121.

[60] D. Nguyen, C. Sahin, B. Shishkin, N. Kandasamy, and K. R. Dandekar, "A Real-Time and Protocol-Aware Reactive Jamming Framework Built on Software-Defined Radios," in *ACM Workshop on Software Radio Implementation Forum*, Chicago, Illinois, USA, 2014, pp. 15–22.

[61] X. Jiao, M. Mehari, W. Liu, M. Aslam, and I. Moerman, "Openwifi csi fuzzer for authorized sensing and covert channels," in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '21, Abu Dhabi, United Arab Emirates: Association for Computing Machinery, 2021, pp. 377–379, ISBN: 9781450383493. DOI: 10.1145/3448300.3468255. [Online]. Available: https://doi.org/10.1145/3448300.3468255.

[62] Abanto-Leon, Luis F. and Bäuml, Andreas and Sim, Gek Hong (Allyson) and Hollick, Matthias and Asadi, Arash, "Stay Connected, Leave no Trace: Enhancing Security and Privacy in WiFi via Obfuscating Radiometric Fingerprints," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, 44:1–44:31, 3, Article 44 Dec. 2020.

[63] gr-bluetooth, *Bluetooth for gnu radio*, 2019. [Online]. Available: http://gr-bluetooth.sourceforge.net/ (visited on 06/2019).

[64] A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems (2nd Edition)*. Pearson, 1996, ISBN: 978-0138147570.

[65] F. J. Harris, C. Dick, and M. Rice, "Digital receivers and transmitters using polyphase filter banks for wireless communications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, no. 4, pp. 1395–1412, Apr. 2003. DOI: 10.1109/TMTT.2003.809176.

[66] Mordor Intelligence, *Smart Wearable Market - Growth, Trends, and Forecast (2019 - 2024)*. 2019.

[67] T. Aura, "Cryptographically generated addresses (cga)," in *International Conference on Information Security*, Springer, 2003, pp. 29–43.

[68] S. C. Kim, W. L. Plishker, and S. S. Bhattacharyya, "An efficient gpu implementation of an arbitrary resampling polyphase channelizer," in *2013 Conference on Design and Architectures for Signal and Image Processing*, Oct. 2013, pp. 231–238.

[69] M. Schulz, D. Wegemer, and M. Hollick. "Nexmon: The C-based Firmware Patching Framework." (May 2017), [Online]. Available: https://nexmon.org.

[70] M. Di Renzo, M. Debbah, D. Phan-Huy, and et al., "Smart radio environments empowered by reconfigurable AI meta-surfaces: an idea whose time has come," *J Wireless Com Network*, vol. 129, 2019.