Wi-Fi Localization Obfuscation: An implementation in openwifi

Lorenzo Ghiro, Marco Cominelli, Francesco Gringoli, Renato Lo Cigno

Department of Information Engineering (DII) — University of Brescia, Italy and CNIT (Consorzio Nazionale Interuniversitario per le Telecomunicazioni)

Abstract

Wi-Fi sensing as a side-effect of communications is opening new opportunities for smart services integrating communications with environmental properties, first and foremost the position of devices and people. At the same time, this technology represents an unprecedented threat to people's privacy, as personal information can be collected directly at the physical layer without any possibility to hide or protect it. Several works already discussed the possibility of safeguarding users' privacy without hampering communication performance, using signal pre-processing at the transmitter side to introduce pseudo-random (artificial) patterns in the channel response estimated at the receiver, preventing the extraction of meaningful information from the channel state, a process called *obfuscation*. One step beyond the proof-of-concept for obfuscation feasibility, is its implementation in working systems. In this work, we present the implementation of a location obfuscation technique within the openwifi project that enables fine manipulation of the radio signal at transmitter side and yields acceptable, if not good, performance, the system has been implemented for both 802.11a/g/h and 802.11n systems, including MPDU aggregation, while implementation for 802.11ac or ax is still not feasible because openwifi does not support 40 MHz channelization and beyond. This contribution discusses the implementation of the obfuscation subsystem, its performance, possible improvements, and further steps to allow authorized devices to "de-obfuscate" the signal and retrieve the sensed information.

Keywords: CSI-based Wi-Fi Localization, Smart Spaces, Privacy Protection, Location Obfuscation, FPGA Implementation, Physical Layer Security

1. Introduction and background

Wi-Fi sensing is attracting interest for many reasons: It is cheap; Wi-Fi is ubiquitous, thus it is easy to deploy; and it can be adapted to sense many different parameters. Furthermore, joint communication and sensing is one of the founding pillars of the vision for communications beyond 5G [1, 2, 3, 4] and the principles and technologies involved are not different from Wi-Fi sensing, thus studies and experiments on this subject turn out to be enablers also for future systems.

Sounding the channel to sense the environment can reveal a lot of information, useful both for communications and for ambient characterization, which can be used to enhance services. The benefit of the former are out of question, while the use and goals of the latter are still under discussion. Indeed, the survey [5] highlights how the targets of Wi-Fi sensing are most often *human beings*, their position, activity, state, even speech or mood [6], and Artificial Intelligence (AI) based sensing methods can be so sophisticated, that they can be trained to locate and recognize activities of people who do not carry Wi-Fi devices, and are thus potentially completely unaware victims of attacks.

In conclusion, joint communication and sensing fundamentally means tracking people, understanding their behavior, and collecting personal and sensitive information, threatening people's privacy and even their security. For instance, Wi-Fi sensing might be used to detect when an apartment is empty and can be robbed, or to control the whereabouts of people in public or private spaces.

This observation pushed other research groups and us to study whether there is a way to counter Wi-Fi sensing while protecting communication performance. Countering Wi-Fi sensing means concealing the information on the environment carried by the electromagnetic signals and retrieved via Channel State Information (CSI) analysis. Protecting communication performance means guaranteeing that the concealment process does not destroy the information carried by the signal. More specifically, the equalizer of a receiver should remain able to compensate for the additional channel distortion introduced by the concealment process.

With reference to Fig. 1, which presents a schematic block of integrated Wi-Fi sensing and reception, our research aims at understanding if and how it is possible to "break" the sensing chain without damaging the receiving chain. We call *obfuscation* the act of hiding non-communication-related information, distinguishing it from the more common *jamming*, whose goal is simply destroying the entire communication capability of the system.

The works presented in [7, 8, 9] are feasibility studies describing proof-of-concept architectures that can achieve obfuscation against non-authorized sensing. These studies discuss methodologies and experiments based on offline signal processing; as such, they lack an actual implementation that analyzes if and how it is possible to implement a privacy-preserving Wi-Fi system and if this system can ultimately enable legitimate sensing (e.g., gesture recognition for remote e-health systems) while preventing illegitimate one.

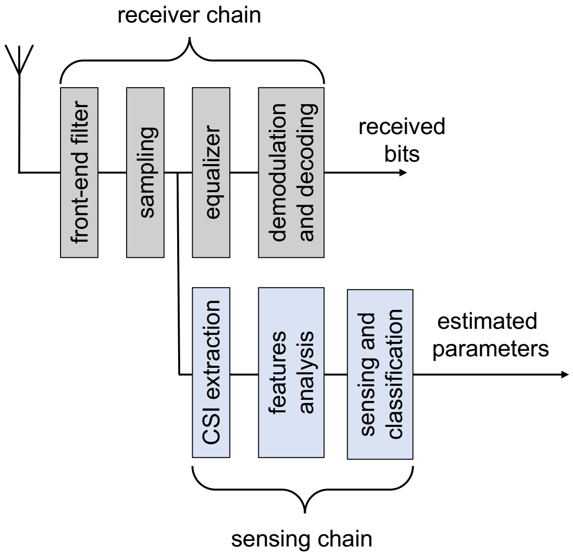

Figure 1: High-level view of the Wi-Fi sensing process.

The contribution of this paper, which extends the prototype presented in [10], is precisely in the direction of filling the gap just described. We have realized an implementation of the technique presented in [7, 11] for both 802.11g/a/h and 802.11n extending openwifi, we experiment to test the obfuscation properties and the communication performance, and we discuss the limitations imposed by the implementation framework. In the last part of the paper, we introduce directions to integrate our proposal in future standards, also discussing a possible protocol that can make the obfuscation invertible, thus allowing authorized devices to perform the desired sensing, including localization of people who do not carry any 802.11 device, e.g., for advanced e-health applications.

Focusing on the implementation, compared to [10], we have added the support for 802.11n, which implies several additional implementation details in the FPGA to address the new format of High Throughput (HT) PPDUs with respect to Legacy (802.11ag) ones. The obfuscation coefficients multiplication and filtering is now implemented in parallel for the L/HT-LTFs, SIG and DATA in the frequency domain and in the time-domain for the L/HT-STFs. Further details on how the implementation has been improved and extended are presented in Sec. 5.

2. Involved Technologies

The openwifi project[1] is an open-source implementation of a fully functional 802.11 stack for Software-Defined Radio (SDR) platforms whose simplified architecture is represented in Fig. 2. At the moment, it can be bootstrapped on a few System-on-Chip (SoC) boards manufactured by Xilinx connected to radio front-ends designed by Analog Devices (AD). Common to all the compatible SoC boards is the presence of an ARM CPU and a Zynq Field Programmable Gate Array (FPGA), connected to the AD radio via a high-speed interface. Hence, openwifi is a SoftMAC stack whose functions are split between the main ARM CPU and the FPGA. The ARM CPU runs a Linux Kernel and user-space tools: Here the openwifi 802.11 driver implements the high part of the Medium Access Control (MAC), which includes functions for preparing outgoing frames and pushing them to the FPGA via Direct Memory Access (DMA), processing incoming frames received from the FPGA, and many other tasks like management and rate control. The Zynq FPGA implements the low part of the MAC and the PHY. The part of the MAC on the FPGA executes all time critical operations like carrier sensing, frame scheduling, and ACK replying; MAC Protocol Data Unit (MPDU) aggregation and dis-aggregation of 802.11n that we discuss later is implemented in the FPGA. The PHY, instead, transforms outgoing frames into I/Q samples and delivers them to the Digital-to-Analog Converter (DAC) circuit on the radio front-end or, vice versa, it reconstructs incoming frames decoding the I/Q samples received from the Analog-to-Digital Converter (ADC).

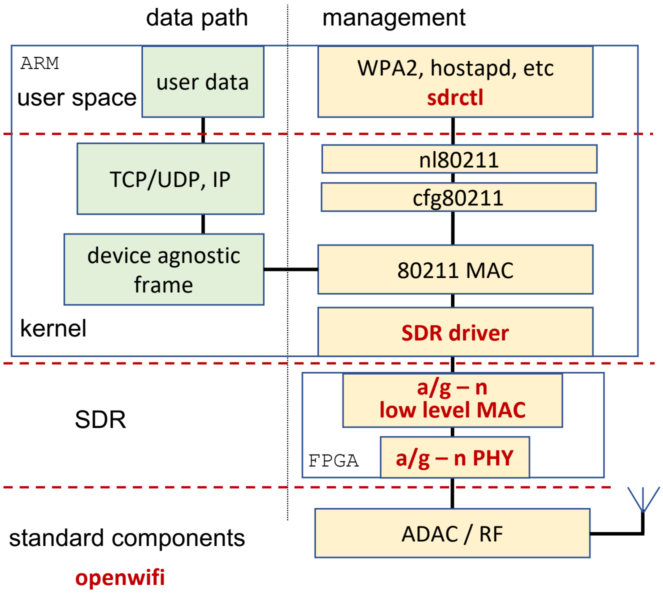

Figure 2: Simplified structure of an openwifi station, in red the components that are developed within the openwifi project.

The bitstream flashed on the FPGA is synthesized using a modified version of the Verilog project that AD releases for controlling the AD radio board from the Xilinx Zynq SoC. The original project is designed to transform the whole device into an SDR: The radio can both transmit batches of I/Q samples stored in the memory of the Linux host (running on the SoC), and store received samples in the host memory for later analysis. Our openwifi modifications add to the real-time MAC the obfuscation capabilities discussed in Sec. 4. In this paper we detail the implementation relative to 802.11n, while the implementation

[1] The openwifi project is open-source and available at: `https://github.com/open-sdr/openwifi`. The links to the codebase maintained by our research group (`https://ans.unibs.it`) are available in Appendix A at the end of this paper together with a short description of the content.

relative to 802.11a/g was presented in [10]. Modifying the PHY part of the Verilog design makes it possible to implement the obfuscation of transmitted signals as described in [7, 11], and summarized in Sec. 5. The behavior of the obfuscator can be customized from the kernel space, i.e., adding proper code to the openwifi Linux driver.

3. Related Work

Preserving the leakage of (private) information from the analysis of physical layer signal characteristics is definitely not a new topic, and we can find works like [12] discussing how traffic patterns at the frame level can reveal information on the encrypted bits carried by the frames, jeopardizing all encryption efforts. CSI analysis, however, moves the physical layer attacks on privacy in a different plane, as it can actually operate as an advanced radar system, exploiting ubiquitous signals, thus the protection of communications, or obfuscation, against CSI analysis becomes of paramount importance to preserve trust in wireless communications access networks. We restrict the discussion of related work to efforts dedicated to obfuscate CSI-based sensing, and refer the reader interested in sensing and localization to the specific literature, which is now very wide and covers all aspects, from theoretical analysis called channel sounding, to localization, gesture recognition, device tracking and many other interesting applications. Seminal works can be found in [13, 14, 15, 16], while recent reviews like [5, 17, 18], already summarize the wealth of efforts dedicated to the topic.

Recently, the interest in CSI obfuscation increased because of the privacy concerns raised by the ever-growing accuracy of novel Wi-Fi sensing algorithms. Nonetheless, the topic has been tackled only by a few works from our and other research groups [7, 8, 9, 10, 11, 19, 20, 21, 22, 23, 24, 25, 26], [10] being a preliminary version of this paper.

Different authors use different names and taxonomies to define their own work and classify other works; names and terms often depend also on the authors background, so that there is not yet a widely accepted taxonomy. Without pretending to build a taxonomy we collect and comment the works cited above grouping them based on the following three broad descriptors: i) those that use fast reaction jamming; ii) those that exploit transmission side signal manipulation, and iii) those that rely on an external device, most often conceptually an Intelligent Reflective Surface (IRS), to actively modify the ambient response to Radio Frequency (RF) signals.

The works presented in [19] and [20] belong to the first category, they are based on the generation of ad-hoc signals that kill sensing frames. These works have the disadvantage of destroying communications and of being effective only against active attacks, where sensing frames can somehow be identified as malicious. Albeit interesting, we think these works go in a direction that is quite different from privacy protection, and have a different goal compared to the contribution of this paper.

The second group of papers includes [7, 8, 11, 22, 23, 25], which focus on the privacy protection (location, tracking or other information) of people against passive attacks that can be sometimes carried out even if the attacked person does not wear any 802.11 device. Passive attacks are defined as those where the attacker controls only a sensing device that works on the frames transmitted by a standard device, e.g., an Access Point (AP). The defense in most of these works is based on the manipulation of signals at the transmitter side, which must be a fixed device to guarantee stable fingerprinting, and are the foundations of the implementation presented here, while [25] is instead based on the implementation of an adversarial neural network that counters the privacy attack.

Finally [9, 21, 24, 26] belong to the third category and propose the use of an external device that changes the propagation environment mimicking additional, time-varying reflections of the system that confuse the learning and fingerprinting process of the attacker. These methods are effective also against active attacks, i.e., those where the attacker controls also one or more transmitters that inject ad-hoc traffic for the sensing process. Indeed, the implementation of these countermeasures seems, to the state of the art, much more difficult, and all works present only a proof-of-concept and not a working implementation. Only [26] actually presents results based on a binary-phase configurable IRS where the binary coefficients are changed randomly in time, mimicking the change of the environment. The system is experimented to verify the obfuscation against human motion, in an experiment conceptually similar to the Empty / Full Room we present in 6.1.

Furthermore, as of today, obfuscation techniques do not exist in real systems able to set up a Basic Service Set (BSS) because almost every implementation of the Wi-Fi standard in commercial chipsets is proprietary and cannot be modified to test and develop new functionalities. For this reason, to the best of our knowledge, works on the field of CSI obfuscation used either SDR platforms emulating the behavior of Wi-Fi chipsets, as we did in the feasibility studies mentioned in Sec. 1, or open platforms as openwifi described in Sec. 2.

The first implementation of a CSI obfuscator has been presented by the developers of openwifi [27]. The idea behind this implementation is to manipulate the transmitted signal in the time domain with a pre-filtering operation, creating a "fake" channel response that can change over time. The filter emulating the channel response is a Finite Impulse Response (FIR) filter limited to three taps and with the first tap equal to 1, which means that the most recent symbol in transmission is never altered. The limitation of this approach is that it does not enable arbitrary manipulation of the spectrum of the transmitted signal, as it mimics the behavior of a channel that introduces additional reflections (the number of taps minus 1) with delays that are exact multiples of the sample time. The approach is indeed very interesting, and exploration can be extended to delays which are not multiples of the symbol time is intriguing. The work presented in [27], a short paper, discusses its implementation and shows the effect on the CSI, but does not attempt to measure its impact on sensing or communications.

A mildly related work is also [28], where the authors goal is to identify attacks against the Wi-Fi sensing infrastructure using CSI-based analysis. The goal of the paper is clearly different from sensing obfuscation; however, they introduce and discuss techniques to monitor and protect Wi-Fi infrastructures, which is an important topic also to move further on to systems that allow legitimate Wi-Fi sensing and prevent illegitimate use.

We add to this discussion also [29], a paper that discusses and analyzes *antifragile* communications, meaning systems that can somehow anticipate or react to adversarial conditions (e.g., jamming) and exploit them at the physical layer to preserve communication quality and even improve it. The concept is very interesting, thus it would be very interesting to explore if it can be implemented in real systems. In turn, it would also be interesting to see if *antifragile* communications can help obfuscation in countering privacy attacks or, vice versa, whether they can hamper it.

Finally, we also mention [30], whose goal is finding countermeasures that can make a device fingerprinting system more robust against all possible channel impairments. This goes somehow against the scope of our work, but we think it is still relevant as it calls for further research on the robustness of techniques such as the one presented in [22] against sophisticated CSI analysis.

4. Transmitter Side CSI Distortion

The concealment of the CSI to prevent sensing can be obtained by proper pre-processing of the transmitted signals, as already introduced in Sec. 1 and 3. The initial theoretical modeling plus a proof-of-concept for testing the feasibility of this concealment process have been tackled in [7, 11, 22]: We refer the interested reader to those works for the details, reporting here only the fundamentals to make the paper self-contained.

The sensing/localization information retrieved by the CSI analyzer is embedded in the signal by the physical environment itself, generally in the form of frequency-dependent attenuation and phase rotation. However, modeling in an efficient way the actual channel response measured by the receiver remains beyond the current state-of-the-art capabilities. Nevertheless, the information is there and can be extracted thanks to Machine Learning (ML) and AI techniques to fingerprint some details of the environment, e.g., the position of a person in a room. At a later time, the extracted fingerprint can be used to classify and hence recognize the environment through a packet CSI. This means localizing the person –indeed any person– in the room, or simply to tell if a room is empty or not. The only requirement to enable this attack is that the transmitter and localization device positions are fixed; it is not important that the positions are known, but only that they are fixed, which is normally the case for any AP.

In this scenario, one possibility to prevent Wi-Fi sensing is to pre-distort the transmitted signal adding a random pattern. The pattern should conceal the information used to fingerprint the environment but, at the same time, it should not jeopardize the communication performance. Summarizing the content of [7] Sec. 3 and [11] Sec. 5, a proper pre-distortion can be obtained with the random process described by Eqs. (1) and (2), where $\mathbf{R}$ is a vector of N_{sc} uniform and independent random variables with support $(\rho_{\min}, \rho_{\max})$, N_{sc} is the number of subcarriers in the Orthogonal Frequency Division Multiplexing (OFDM) modulation, α is the memory of the Uniform-Markov process driving the pre-distortion, $\Delta_t(k)$ is the inter-frame time between frames k and $k-1$, and Θ_C is a 5-tap FIR filter to introduce correlation between adjacent frequencies as the propagation channel normally does. We use a simple moving average where min = 0.1 and max = 1.9 are bounds to guarantee that the pre-distortion multiplication never leads to unrealistic high values but, most of all, it never completely suppresses a carrier, as this would obviously hamper communications.

$$\mathcal{R}(k) = e^{-\alpha\Delta_t(k)}\mathcal{R}(k-1) + \mathbf{R} \tag{1}$$

$$A_O(k) = [1 + \mathcal{R}(k)]_{\min}^{\max} * \Theta_C \tag{2}$$

The rationale of this pre-processing is simple: the transmitted signal is distorted in such a way that the receiver's equalizer (see Fig. 1) can still compensate for the distortion, but at the same time the localization system is "fooled" by the fake channel features intentionally crafted by the transmitter. The remaining parameters are $\rho_{\min} = -0.3$, $\rho_{\max} = 0.3$, suitable values empirically found in previous works, and $\alpha = 0.2$, which means that if $\Delta_t(k) \geq 15$ s, then A_O (k) and A_O $(k\text{-}1)$ are almost completely uncorrelated, i.e., the correlation coefficient is below 5%, coherent with the fact that a person moving in a room can completely change its position within 15 seconds, thus there is no reason to maintain memory or coherence if the inter-frame time is larger. All parameters are configurable in the implementation, but we maintain the same configuration of previous works for the sake of comparison.

5. Implementation

With reference to Fig. 2, the implementation of an obfuscation layer in openwifi at the transmitter —a layer that we call P^{2}SL (*Privacy-Preserving Sub-Layer*) following the project that supported this work— requires the enhancement of both: *(i)* The Network Interface Card (NIC) driver in the Linux kernel, as detailed in Sec. 5.2, and *(ii)* The FPGA, described in Sec. 5.3.

Before delving into the details, we start with a high-level description of the modifications to openwifi. In this paper, we focus on the obfuscation of 802.11n, which requires some additional features compared to 802.11a/g that was presented in [10] and it is a completely new contribution. However, we also report some performance figures of the 802.11a/g implementation for the sake of completeness and comparison.

5.1. High-Level Design

Fig. 3 illustrates the FPGA and Linux kernel components modified to develop P^{2}SL, highlighting how our modifications focus on the NIC driver and on the PHY layer only, while the MAC remains essentially untouched. The implemented obfuscator follows the design described in Sec. 4, with an *Obfuscator* block, written in C, responsible for updating $\mathcal{R}(k)$, $\mathbf{R}$, and $A_O(k)$, every time a new frame k is pushed from the operating system to the driver for transmission.

The size N_{sc} of the $\mathbf{R}$ vector is fixed to 64 for efficiency, reflecting the fact that openwifi currently supports only 20 MHz channel bandwidth with 64 OFDM subcarriers. The number of

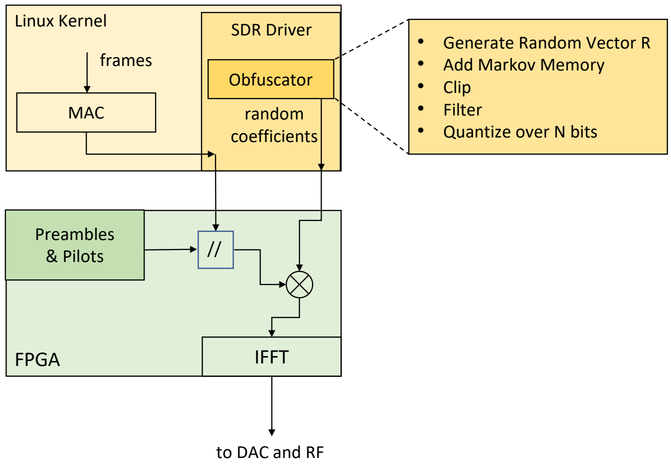

Figure 3: Overview of P^{2}SL design with kernel and FPGA modifications.

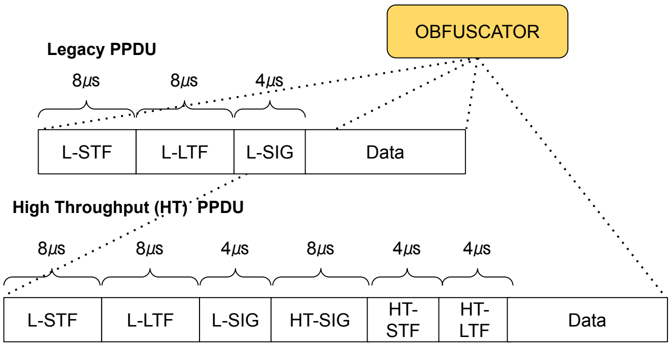

Figure 4: The obfuscation must be applied to all the fields of a PLCP Protocol Data Unit (PPDU), including all headers. This requires a different manipulation of the fields composing the preamble of Legacy and HT PLCP Protocol Data Units (PPDUs).

these subcarriers devoted to the guard bands changes between 802.11a/g and 802.11n, with other differences related to the modulation & coding and to the symbol guard time.

As depicted in Fig. 4, the structure of the PLCP Protocol Data Unit (PPDU) differs significantly between 802.11a/g (Legacy PPDU) and 802.11n (HT PPDU for the single stream case). Moreover, when MPDU aggregation is enabled, the obfuscation must be applied to the entire PPDU, which implies different manipulation in the FPGA. When MPDU aggregation is enabled, then the 'Data' field of the PPDU contains a sequence of MPDU, each composed of the MAC header and the IP packet as payload. The aggregation is performed inside the FPGA, which reads the data through a DMA independently from the driver. Indeed, the driver writes the MPDUs in a First In First Out (FIFO) buffer, and the FPGA reads them at its own rate and disposition. Obfuscation coefficients are instead written by the driver in a single-position register, so they get overwritten every time a new MPDU is enqueued. Obfuscation coefficients are thus coherent only with the time of the transmission, not with the time of MPDU formation. In earnest, we cannot guarantee this implementation ensures the Markovian properties of the output process; however, the clip and filter operation of Eq. (2) already distort the Markov-Uniform process of Eq. (1), and these are necessary approximations for a real implementation.

For the driver implementation, we use the Fixed Point Math library for C [31] to implement the clipping and filtering operation within the kernel, which notoriously does not support floating point algebra. The coefficients of $A_O(k)$ are thus 4-Bytes (32-bits) Fixed Point variables, where the first 14 bits and the remaining $32 - 14 = 18$ bits represent the signed integer part and the decimal part of each number, respectively. This means we can manipulate numbers with up to 4 integer decimal digits and retain good accuracy when doing operations with 5 decimal digits. Further approximations of the theoretical obfuscation mask are introduced in the FPGA at the PHY layer and need to be carefully controlled to avoid disrupting communications capabilities, as discussed in Sec. 5.3.

We can devise different architectures for this scope, and we can even imagine to implement these operations directly in hardware. Unfortunately, such implementation would require a lot of space in the FPGA, and also the implementation of a whole Arithmetic and Logic Unit (ALU) for the generation of the obfuscation process. In practice this would probably mean re-implementing the entire openwifi project, most probably also requiring more powerful and expensive FPGAs, which goes beyond the scope of this paper.

In summary, we have customized the openwifi driver to:

1. Update the $A_O(k)$ mask for every frame to be transmitted;
2. Quantize the N_{sc} multipliers $\in A_O(k)$;
3. Write the N_{sc} quantized multipliers in dedicated FPGA registers for each frame transferred from the non real-time MAC to the real-time MAC and Physical Layer Convergence Procedure (PLCP).

Again with reference to Fig. 2, the modifications to the openwifi on the FPGA side can be summarized in three main key points:

1. Introduction of new FPGA registers writable from the driver to receive the vector of N_{sc} quantized multipliers;
2. Generation of Preambles and Pilots in the frequency domain to apply the obfuscation also on them with a different implementation for Legacy and HT PPDU;
3. Multiplication of all the symbols in every frame with the pre-distortion N_{sc} multipliers.

With the support of Figs. 5 to 7, we discuss the impact of the major implementation impairments before describing the implementation details.

Fig. 5 compares three realizations of the Eqs. (1) and (2) generated with Matlab (red) with those generated by the implementation in the kernel driver. These are the processes that multiply the carriers, we selected the carriers 10, 30 and 60. The quantization of the kernel implementation is evident and it leads to piecewise constant values for a few frames, while the Matlab implementation has a behavior which is more in-line with the intuition of a Uniform-Markov process, albeit clipping and filtering (Eq. (2)) significantly modify the behavior.

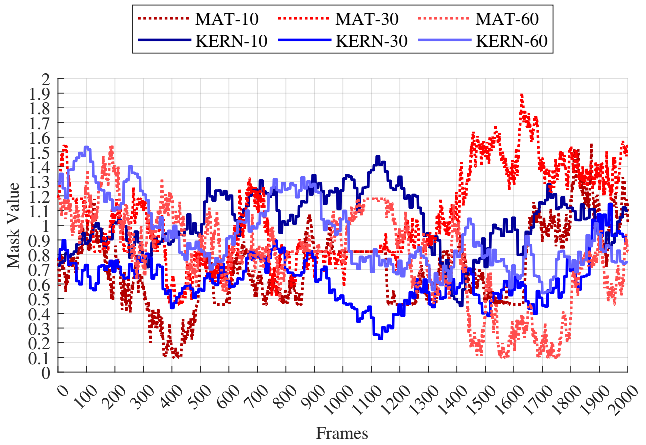

Figure 5: Three realizations of Eqs. (1) and (2) for 2000 frames equally spaced by 1 ms with Matlab (MAT- in red) and implemented in the kernel driver (KERN- in blue). The realizations refer to subcarriers 10, 30, and 60 respectively (out of the 64 total subcarriers).

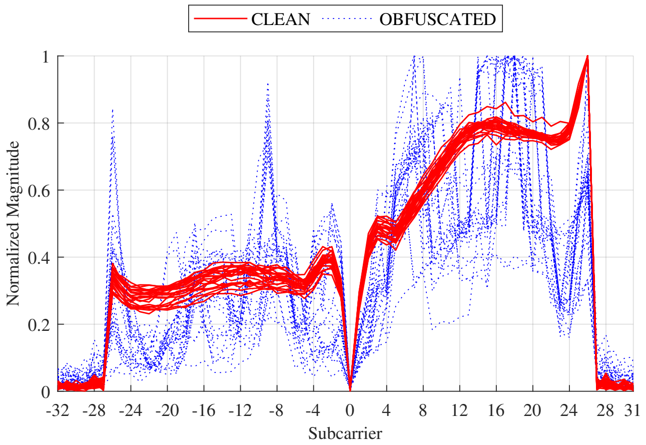

Figure 6: Normalized Magnitude of subcarriers measured at a receiver for short burst of frames with obfuscation off (red) and on (dashed blue)

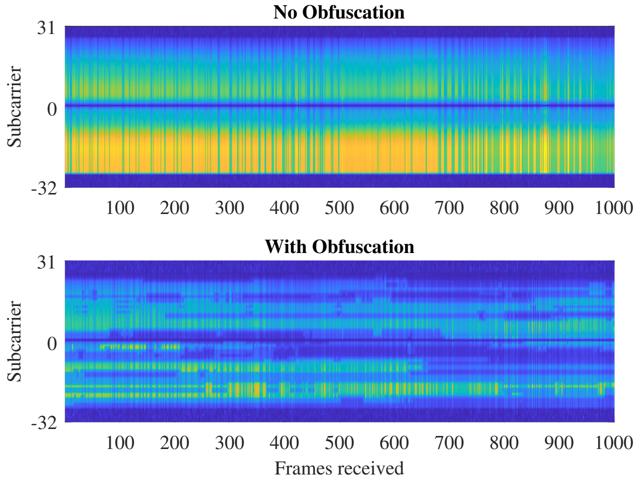

Figure 7: Magnitude of the CSI collected from 1000 frames, with brighter tending to yellow indicating a larger magnitude.

Sec. 6 analyzes if and how these approximations impact performance, but we think that quantization and the short-time piecewise constant behavior do not affect the pre-distortion process significantly, as the key feature of pre-distortion is the continuous change *of the overall* signal amplitude. This is shown in Fig. 6, which compares the CSI at a receiver for a short burst of frames with (in blue) and without (in red) the obfuscator. Without the obfuscator, the amplitude is remarkably constant, thus allowing fingerprinting; with the obfuscator, the behavior is clearly random.

Also Fig. 7 qualitatively shows the effect of obfuscation with a heatmap of the CSI amplitude for 1000 frames. Without obfuscation, the channel response remains fairly constant, shown by the blue and yellow bands remaining constant over time; when the obfuscation is activated, the pattern is blurred as intended. Two effects emerge from this picture. First, the obfuscated frames are (on average) less energetic, and this is because the FPGA implementation forced us to apply only attenuations, which should be compensated by the Automatic Gain Control (AGC) in the DAC board, but they are evidently not, or at least not completely. Second, in both the obfuscated and the clean realizations it seems there are much less energetic frames (the blue-greenish vertical lines). We do not have a *certain* explanation for this, but it is most probably due to the AGC behavior at the receiver (a standard, off-the-shelf ASUS AP), and it seems to have little influence on functionality and performance.

5.2. *Kernel Driver*

As already anticipated, the code we have introduced in the openwifi driver is mainly responsible for the generation of the N_{sc}-long vectors of obfuscation coefficients and their transfer to the FPGA. These two main features have been implemented as the following sequence of steps:

Step 1 Compute the interframe time ($\Delta_t(k)$ of Eq. (1)) despite the lack of a kernel clock, relying instead on jiffies [32];

Step 2 The computation of the $A_O(k)$ coefficients including the evaluation of the exp function without support for floating point algebra;

Step 3 Quantization of $A_O(k)$ coefficients;

Step 4 Ovveride of the FPGA dedicated registers with updated $A_O(k)$ values.

The memory of the Markov process was originally designed to fall below a meaningful value in about 15 s. However, in the kernel implementation it loses precision after 10 s because of the simple 6-th order Taylor-McLaurin expansion we used to approximate *exp* (see Fig. 8). Thus we decided to truncate the exponential at $\Delta_t = 7.5$ s and simply de-correlate frames with a larger inter-arrival time rather than using higher order or more complex approximations that would have slow down too much the obfuscation process.

Fig. 9 compares the sample Autocorrelation Function (ACF) exhibited by some realizations of the $A_O(k)$ random process generated by our driver with similar ones computed with Matlab. The

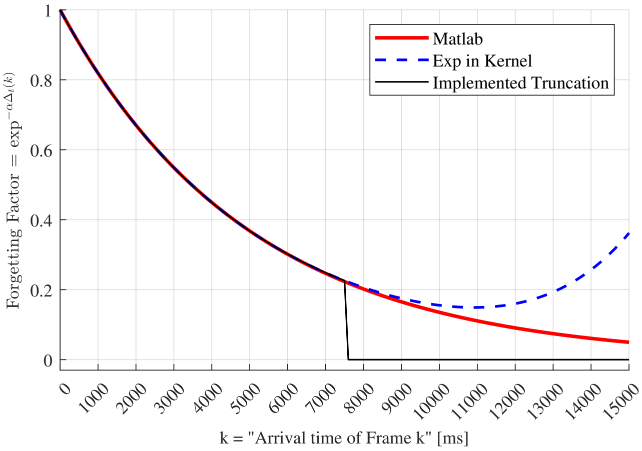

Figure 8: Approximation of $\exp^{-\alpha\Delta_t(k)}$ for $\Delta_t \in [0, 10000]$ ms computed in kernel with a 6-th Taylor-McLaurin approximation versus the Matlab computation; the precision decays after 10 s.

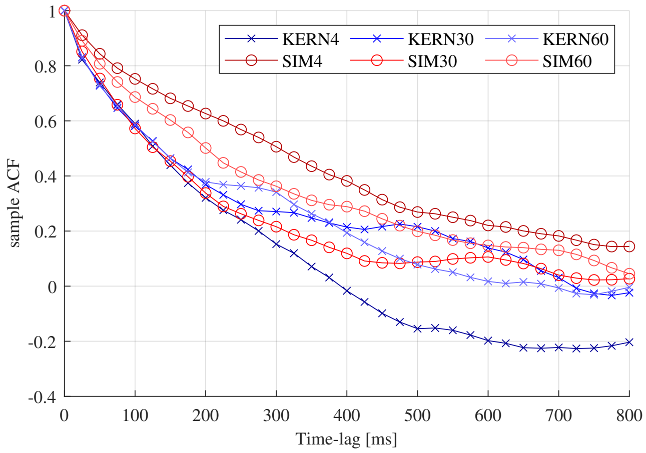

Figure 9: Comparison of the ACF computed on the realizations of the Markovian processes described by Eq. (2) for subcarriers 4, 30 and 60. The red curves (with circles) represent the ACF obtained with Matlab simulations (SIM4/30/60), while the blue (with x) ones have been generated by the kernel code (KERN4/30/60).

kernel curves (in blue) are obtained querying the development board every millisecond, synchronized with frame generation to obtain proper values of the ACF. The difference between the Matlab and the kernel curves suggests that the ACF achieved with the implementation decays slightly faster than the theoretic one, indicating a modestly weaker memory. This can be justified by the truncation of the exponential explained with Fig. 8. The long negative tail of KERN60 may look surprising, but it is perfectly coherent with a Uniform-Markov process and does indicated the existence of memory with that time lag. Furthermore, the initial behavior of KERN curves replicates the behavior of SIM60, indicating that they can well belong to the same processes population.

5.3. FPGA Design

The FPGA side of openwifi, written in Verilog [33], has been modified to implement the obfuscation of the IQ samples of each frame. The implementation presented in [10] could be applied only to Legacy PPDU and furthermore did not obfuscate the initial pilots (L-STF) generated directly in the time domain. The implementation reported here extends also to HT PPDU and obfuscate all preambles (L-STF and L-LTF), thus making the 802.11n frames generated in openwifi fully obfuscated, including the aggregated ones. The main FPGA modifications can be summarized as follows:

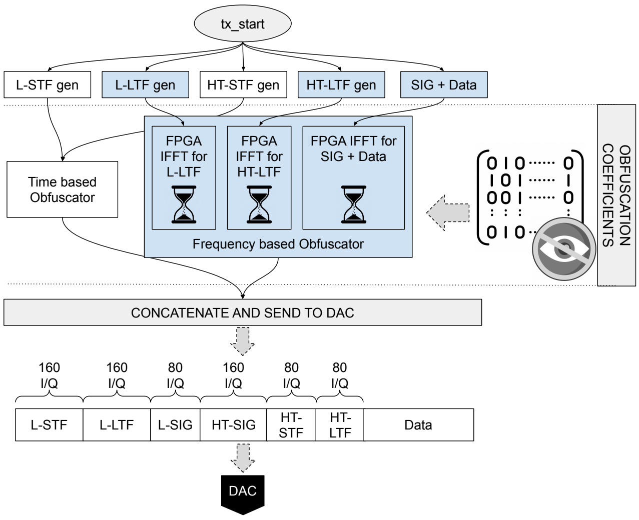

Figure 10: Workflow of the HT PPDU obfuscation within the FPGA.

1. *Addition & wiring of FPGA registers.* A set of FPGA registers, configurable at runtime by the driver, are allocated and pinned, in cascade, to all modules involved in a frame transmission, starting from the Xilinx Axis DMA down to the internal blocks of the openofdm_tx module, which is where IQ samples are actually manipulated. From an hardware point of view this operation coincide with the wiring of different input/output pins. The matrix drawn in Fig. 10 represents the obfuscation coefficients made available to the FPGA through these registers.

   In particular, these registers can store up to 128 bits, which implies that each of the $N_{sc} = 64$ obfuscation coefficients must be quantized by the driver over 2 bits only, we call these coefficients oc_s, and they are properly assigned by the kernel driver based on $A_O(k)$ thresholds. Extending the registers capacity is easy, and its implementation would require only a new FPGA synthesis; however, the manipulation of IQ samples with more accurately quantized coefficients would remarkably increase the complexity of algebraic operations.

   As already mentioned, the structure of the FPGA manipulation allows only attenuation of IQ samples to avoid subsequent overflows. This means that we actually reduce the transmission power of frames, which is already very low due to the limited capacity of the AD Analog-Digital-Analog Converter (ADAC) front end. In a real system we expect that an AGC amplifier would compensate this, leading to transmit frames with the same average power of non-manipulated frames, though this aspect may require additional research and fine-tuned manipulation that accounts also for OFDM Peak-to-Average power properties.

2. *Obfuscation of Preambles.* In the original openwifi design, the preamble samples—constant for each frame—were stored as time domain samples in a dedicated ROM memory and then pre-pended to the frame after the Inverse Fast Fourier Transform (IFFT) block on the FPGA. This pre-tabling and on-the-fly operations reduced the requirements of the FPGA in terms of both memory and computing resources, but it is partially not compatible with our obfuscation scheme, which requires to store the preambles in the frequency domain, so that they can be multiplied by the same $A_O(k)$ coefficients as all the other symbols of the frame k. In light of this consideration, the abstract obfuscator block visible in Fig. 4 has been concretely implemented in the FPGA by defining the workflow depicted in Fig. 10. There we report a top-down time flow, with actions that appear at the same height that are performed in parallel:

   - Whenever a new `tx_start` event is fired, the FPGA reacts by activating 5 parallel State Machines. These machines are responsible, respectively, for the generation of the obfuscated L-STF, L-LTF, HT-STF, HT-LTF and SIG+DATA fields that compose an 802.11n frame.

   - We categorize the 5 machines in 2 groups: i) the *Time-based* and ii) the *Frequency-based* ones. The definition of the LTF, SIG and DATA fields require the obfuscation to be applied in the frequency domain and are depicted in blue used to indicate *frequency-based obfuscation*. Each of them embeds a dedicated FPGA IFFT block to provide the I/Q samples to the subsequent concatenator block.

   - The STF fields (both the Legacy and HT ones), are obfuscated more efficiently in the time domain, without requiring time-consuming operations involving the computation of an IFFT. We sum the twelve 16-samples tones that define the STF and that we precomputed in a dedicated memory, each one pre-distorted using the coefficient from the mask corresponding to its own frequency; this can be done for these fields thanks to the small number of possible combinations. This is why we have chosen to represent the L-STF and HT-STF FPGA obfuscating machines differently, highlighting their different architecture.

   - All obfuscated I/Q samples are at the end concatenated and sent to the DAC for transmission.

3. *Signal pre-distortion.* The I/Q samples should be multiplied by the $A_O(k)$ coefficients before the IFFT. Unfortunately, the space availability and speed of operation of the FPGA forced us to approximate the multiplication as a simple right-shifts operation, resulting in attenuation only. The right-shift operation is chosen for each subcarrier according to the quantization thresholds described by Eq. (3):

$$\begin{cases} \text{if } (oc_s = 00) & \Rightarrow IQ_{obf} = IQ_{in} \gg 3 \\ \text{else if } (oc_s = 01) & \Rightarrow IQ_{obf} = IQ_{in} \gg 2 \\ \text{else if } (oc_s = 10) & \Rightarrow IQ_{obf} = IQ_{in} \gg 1 \\ \text{else } (oc_s = 11) & \Rightarrow IQ_{obf} = IQ_{in} \end{cases} \tag{3}$$

where:

- oc_s is the quantized obfuscation coefficient belonging to $A_O(k)$ associated to subcarrier s;
- IQ_{in} is the I/Q sample before obfuscation;
- IQ_{obf} is the same I/Q sample after the desired right-shift operation has been applied, thus, the obfuscated version of the I/Q sample.

6. Performance

The goals of the experimental performance evaluation are two:

1. Quantify the effectiveness of the implemented obfuscation in preventing localization (Sec. 6.1);
2. Determine if and how much the obfuscation deteriorates communication performance (Sec. 6.2).

As already mentioned, we have focused the implementation description on 802.11n / HT PPDU, as this is entirely novel and was never presented. Nonetheless, we report some performance figures that are also related to 802.11a/g and Legacy PPDU for the sake of completeness and comparison.

6.1. Localization Obfuscation

To address our first goal we use IPERF sessions transmitting packets with the obfuscation on or off. During each session a person stands in one of the eight positions indicated by $\mathbf{P}_i$ in Fig. 11.

In general, to interpret our experimental results the reader should always make reference to Fig. 11, which shows the schematic layout of the Wi-Fi devices used to run experiments. Four localizing devices labeled as LN capture traffic, and the CSI extracted from the captured traffic feeds the Convolutional Neural Network (CNN) described in [7, 11] that learns the sensed environment and classifies the person position in the room.

In all the experiments the transmitter is a Xilinx ZC706-G development board running the modified version of openwifi and operating as AP, the Rx device acts as a STAtion (STA) associated to the AP. The receiver can be any 802.11 capable device, as a normal PC or a smartphone, but in the present case it is an Asus RT-AC86U device like the localizing devices L1–L4. The CSIs are extracted using the methodology and software described in [34].

An attacker must train the system before attempting a localization attack, thus we run each IPERF session twice to build both a training and a testing dataset. We capture both datasets during the same day, leaving between each data collection session a reasonable time gap of at least 10 minutes to make localization

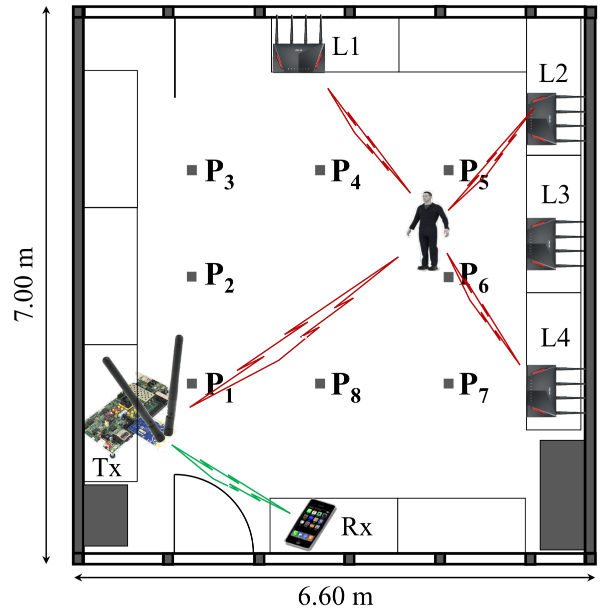

Figure 11: Placement of Transmitter (TX), Receiver (RX) and Localizing devices in our Lab at the University of Brescia.

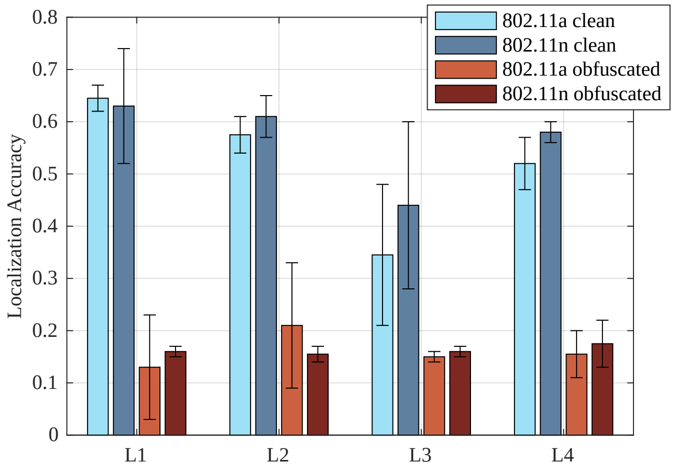

Figure 12: Bar chart comparing, for each localizing device L1-L4, the mean localization accuracy when captured traffic is CLEAN (i.e., not obfuscated, blue bars) and when it is instead OBFUSCATED (orange bars). Whiskers indicates the maximum and minimum accuracy achieved while attempting localization attacks.

results credible. The classification task can output 8 values (corresponding to the 8 positions of Fig. 11), so the accuracy of the localization compares with a random guess that yields a baseline value of $1/8 = 12,5\%$.

Fig. 12 shows the localization accuracy for all 4 localizing devices when the obfuscation is turned on or off. Compared to the theoretical results in [7, 11] the CLEAN (i.e., not obfuscated) localization is less accurate. We think that this is due to the use of a 20 MHz system as allowed by openwifi instead of a 80 MHz 802.11ac system, which stresses how it is important to study and design anti-sensing systems as sensing will become extremely accurate with the evolution of technology. We observe instead that moving from 802.11a/g (results also in [10]) to 802.11n does not change localization capabilities. This is expected as the CSI extraction is fundamentally based only on the initial preambles and nothing else. Indeed, also the HT-LTF preamble contributes to the CSI and it does include the coefficients of 4 additional subcarriers, but apparently this small additional information is not enough to improve the classification capabilities of the CNN. Rather, more subcarriers as allowed by 40 MHz, 80 MHz, or 160 MHz, would mean a greater amount of ambient information embedded in OFDM signals. With larger bandwidth we expect sensing and localization to become more effective and accurate.

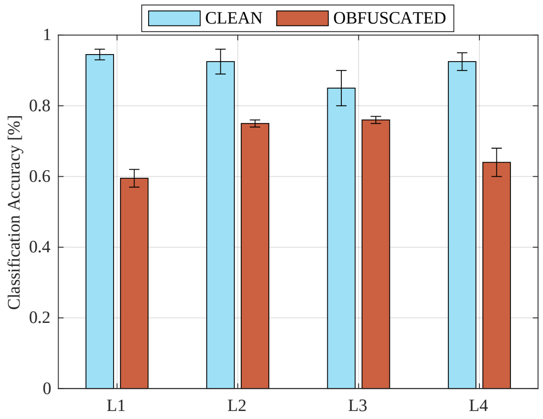

Figure 13: Bar chart comparing, for each localizing device L1-L4, the mean accuracy in identifying an empty of full (with people) room.

The OBFUSCATED results are instead just above the random choice, indicating that the obfuscation process is very effective and makes a localization attack fundamentally useless. Furthermore, as already noted in previous works, the position of the localization device has a significant influence on the localization performance, but this cannot be predicted, and this dependence is stronger for the CLEAN results than for the OBFUSCATED ones.

In another, novel experiment, we have modified the CNN that performs the localization to return only two values: Room Empty; Room Full. Training of this CNN is done with a long term analysis (tens of minutes) of frames transmitted with the room empty at a low frame rate of a few frames per minute, and similarly with a person doing normal activities (standing, moving, sitting, . . .) inside the room. Testing of the localization is performed with a similar procedure. We did not attempt to design a novel CNN from scratch, as this activity goes beyond the scope of this paper, but we are aware that different methods, e.g., accounting for time variability, can be better suited to discriminate the two conditions in this specific case. Our goal remains testing the obfuscation implementation and check if it is somehow effective or not.

Fig. 13 reports the bar chart of the average accuracy obtained from the same four localizing devices L1–L4, with whiskers indicating minimum and maximum. The experiment has been run only for 802.11n and clearly in this case a random guess returns 0.5. Somehow surprisingly even with CLEAN transmissions the accuracy is not as high as one may expect thinking that this task is easier than spotting the precise position of a person. Indeed, the original CNN was designed to classify specific patterns that arise with a person standing still in a place, and it was only modified, not re-designed, for this experiment, so from the learning algorithm point of view, the task may not be simpler at all. For instance a person moving, standing, sitting does not give rise to a single specific pattern, but to multiple, varying patterns, which may in the end confuse the classifier. For what the goal of this paper is concerned, however, the fact that with the obfuscator on the accuracy reduces consistently for all localizing device is enough to state that obfuscation works efficiently even if, at least for positions L2 and L3 the result may still be good enough for some form of attack, and one can think that, using fusion techniques similar to those studied in [8, 11] the attack can be effective. Thus, also in this case additional research in obfuscation techniques may be needed before the methodology and technology is mature for a standardization process.

6.2. Communication Performance

To address our second experimental goal we use again iperf over UDP turning on/off the obfuscation and varying the channel PHY bitrate *cr* and the UDP payload length *pl* to measure if and how much the obfuscation hampers the communication performance. Finally, we analyze if the aggregation mechanism is affected by the obfuscation, since in this case the architecture of our implementation does not allow matching accurately the attenuation pattern with the frame transmission scheme. In particular, we fix PHY rate on the radio interface and measure the Packet Delivery Rate (PDR) at the receiver, and we do this for four different payload lengths: $pl \in PL = [200, 500, 1000, 1470]$ bytes.

For this experiment L1–L4 devices are not used, with the focus only on the communication performance between Tx and Rx. The communication performance metric is the PDR computed by the receiver as:

$$\mathrm{PDR}(cr, pl) = \frac{\#\ \text{received packets}}{\max(\text{seqno}) - \min(\text{seqno})} \qquad (4)$$

where the denominator of Eq. (4) is not the number of transmitted packets per iperf session, but the difference between the maximum and minimum sequence number intercepted by the receiver device to avoid confusing losses on the channel with frames that are not transmitted for whatever reason or with iperf synchronization problems. For each element of the experiment space $(cr, pl) \in CR \times PL$ we performed at least 10 iperf sessions, with more repetitions for high values of *cr* to gain more statistical confidence.

Fig. 14 compares the mean PDR achieved when turning the obfuscation on (blue curves) and off (red curves). The top plot refers to HT PPDUs (802.11n) and the bottom plot to Legacy PPDUs (802.11a/g). The PDR without obfuscation is in line with similar measures, very close to 100% with a small degradation for high PHY transmission rates. For 80211n we use a single spatial stream and 800 ns inter-symbol spacing. Turning on the obfuscation results in a slight degradation of the performance, which is clearly increasing with the PHY data rate and it is larger and more evident for 802.11n, which has more fragile modulation schemes. It is clear and probably obvious, as we already observed with the Matlab+SDR proof-of-concept on 802.11ac, that the highest PHY rates are fragile. What is instead very interesting to note here, is that overall the performance is more than acceptable, even if the implementation on openwifi introduces many approximations and indeed forces architectural choices that makes the transmitted signals very different, we can colloquially say “ugly”, compared to the standard, non obfuscated signals, but also compared to the signals generated with Matlab and high-end SDR systems. This observation—even if one considers that the loss of performance is consistent—clearly indicates that more research and efforts in finding good obfuscation methodologies will lead in the future to perfectly performing systems.

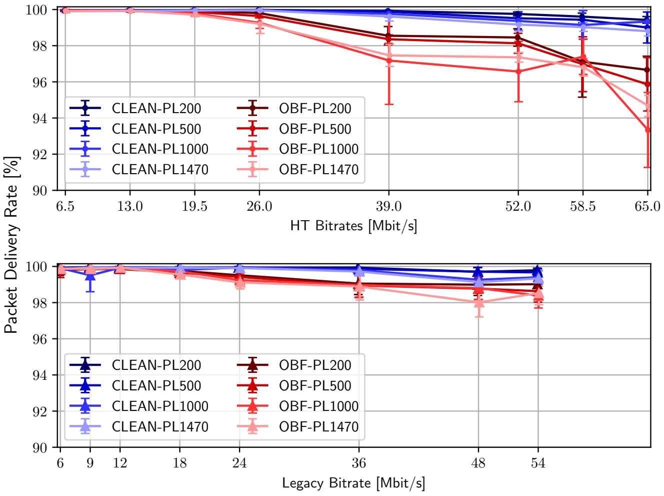

Figure 14: PDR as a function of the channel bitrate with obfuscation off (CLEAN- in blue), and on (OBF- in red) as a function of the PHY Bitrate. Different curves refer to different frame lengths (iperf UDP payload indicated in the legend as PLnn in bytes). For each experiment the curves indicate the mean PDR, while the bars report the confidence interval with 95% confidence level. The top plot refers to 802.11n and the bottom plot to 802.11a/g.

We now turn our attention to packet aggregation, or more precisely, MPDUs aggregation into PPDUs. This is one of the main innovations in 802.11n and more recent 802.11 versions, and it normally guarantees higher throughput, thanks to a much more efficient use of the channel time compared to the transmission of a single MPDU at a time, which implies wasting much more time in channel contention and transmission of PHY preambles. Aggregation is a very complex task, and it is rarely studied, as it is difficult to analyze or measure it without having access to the chip that implements it.

In openwifi aggregation is done in the FPGA, and this, as we mentioned, create a mis-alignment with the obfuscation mask generation process, which runs in the driver and is based on the generation time of every single MPDU. To try to mitigate this problem, we have crafted a method that freezes the obfuscation coefficients during the transmission phase, even if they are regenerated in the meantime, as changing the obfuscation mask within the same PPDU would obviously prevent the equalizer of

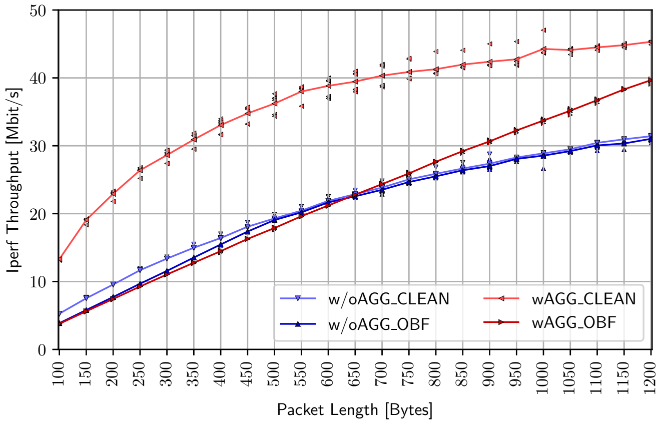

Figure 15: Throughput obtained with and without aggregation, with and without obfuscation as a function of the frame payload length. The curves interpolate the average values over all the experiments for each packet length. Single experiment values of throughput are drawn as scattered marks for each packet length, and they are remarkably close to the average.

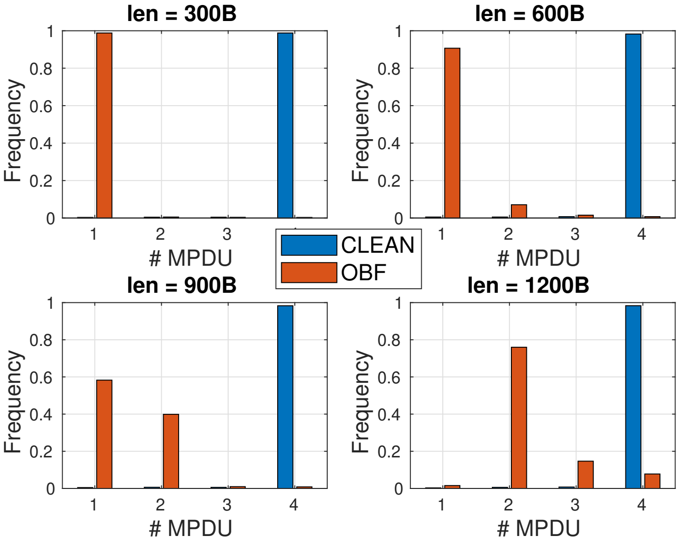

Figure 16: Histogram of the relative frequency of the number of aggregated MPDUs per PPDU with and without obfuscation.

the receiving device to work properly. Moreover, the obfuscation coefficients used for an aggregated PPDU are those of the most recent MPDU generated, even if this MPDU is actually not transmitted in the PPDU. This may look wrong, but it is indeed the best way we have to approximate a Markovian memory with the memory of the implemented system, because it guarantees that the autocorrelation between the PPDU in transmission and the previous PPDU is the closest possible (with the constraints of the implementation) to the one 'designed' with Eq. (1).

We run these experiments connecting the transmitter and the receiver directly with a cable, as their complexity makes the interpretation of results on a real wireless channel with interference impossible.

Fig. 15 reports the throughput of a saturated IPERF session with and without aggregation and with and without obfuscation as the size of the MPDUs increase from 100 to 1200 bytes. The maximum aggregation level possible with openwifi is 4 MPDUs per PPDU. Beyond a size of 1200 bytes the aggregation mechanism of openwifi does not work properly and we stopped the experiments. Aggregation is still an experimental feature of openwifi, and we could not dig into the reason of this issue; furthermore, it is not possible to fix the Modulation and Coding Scheme (MCS) (as we did to measure the PDR) if the aggregation is active, thus there is some uncertainty as far the used MCS is concerned. It is clear that without obfuscation the gain with aggregation is significant across all MPDU sizes. With the obfuscation on, instead, the throughput gain seems marginal and increases slightly as the MPDU size increases, which is not what one would expect. Furthermore, while the throughput without aggregation remains the same with or without obfuscation in this case there is a clear penalty to pay for improved privacy.

To further investigate this matter, we measured the fraction of PPDU that aggregates 1, 2, 3, or 4 MPDU and we report the empirical probability density function (pdf) in Fig. 16 for MPDU sizes of 300, 600, 900 and 1200 bytes. This further analysis immediately explain the result, showing that with the obfuscation on the aggregation algorithms on the FPGA tends to transmit non-aggregated PPDU, and this especially when the MPDU size is small. We think that this is due to some timing glitch introduced by the obfuscation procedure, whereby the aggregation algorithm does not "see" that there are MPDU in the buffer that can be aggregated. Only when the MPDU size is large these timing impairments have a smaller impact and allow some level of aggregation and an increase in throughput. Furthermore, we observed that with the obfuscation on the MCS oscillates between MCS=7 and MCS=6, due to the increased frame losses, and this phenomenon may also influence aggregation, also triggering frame re-transmissions. The interaction between obfuscation and aggregation, and indeed in general between obfuscation principles and implementation details and impairments, still need research and scrutiny, delving deeper in the implications of pre-distortion and channel manipulation in general.

7. A Protocol to Negotiate Obfuscation

So far, we have seen that anti-sensing techniques can be implemented in standard devices preserving acceptable communication performance as shown in Sec. 6.2. However, a solution that fully preserves unhampered communications and allows sensing at selected and legitimate receivers is preferable to the blind obfuscation we presented. This section is devoted to discuss how such a solution can be implemented in the 802.11 standard, and different feasible flavors of it, from the simple solution of de-obfusction only at the receiver, to more sophisticated techniques enabling multi-point sensing at legitimate devices. First of all, let us recall that the CSI-based sensing and localization we try to counter with this work is fundamentally different from positioning techniques proposed in 802.11az[2] that are based on

[2] See the 802.11az Task Group page for further details (`https://standards.ieee.org/ieee/802.11az/7226/`).

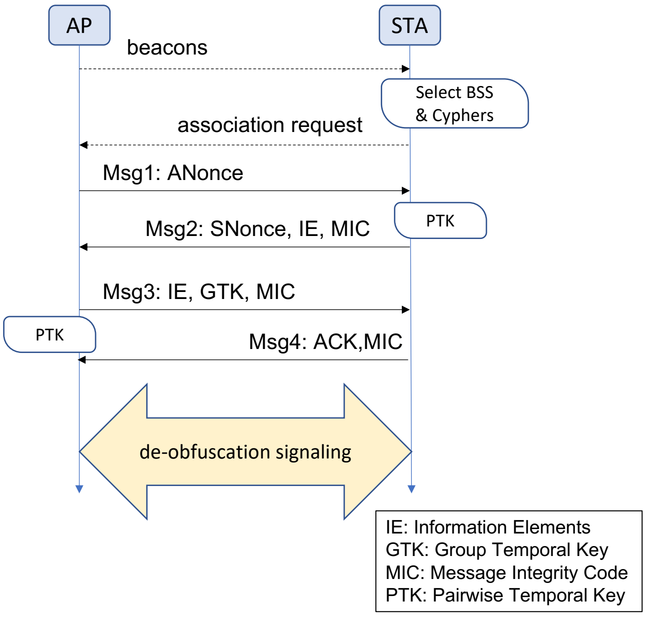

Figure 17: 802.11i Four Way Handshake (4WHS) showing also beacons, association and the position of additional signaling messages for de-obfuscation.

Time of Flight (ToF) and Angle-of-Arrival (AoA) and require the active cooperation of the receiver: They focus on the localization of a cooperating device, and not to the sensing and tracking of people who may even not carry a device. Rather, this discussion can be related to 802.11bf ubiquitous Wi-Fi sensing.[3]

First of all, consider the Four Way Handshake (4WHS) of the Wi-Fi Protected Access (WPA) negotiation as defined in the standard [36] and schematically reported in Fig. 17, which establishes a cryptographically secure communication channel between an AP and a STA. This channel is then used to transmit user data, but it can also be used to transfer signaling and management information that we represent in the figure with the bi-directional block arrow at the end of the 4WHS. Even though the authentication procedure was slightly modified with the introduction of the Simultaneous Authentication of Equals (SAE) procedure [37], our considerations still apply as they only concern the 4WHS that anyways follows the SAE step.

In the following two subsections we describe the requirements of two different de-obfuscation procedures: The first one to only improve the communication performance, and the second one to also enable multi-point sensing. Recall that transmissions useful for sensing are only those of the AP, while STAs transmissions are useless because STAs can move. We consider that beacons are in any case obfuscated, but since they are always transmitted at the lowest possible transmission rate, STAs will be able to decode them with high probability as shown in Fig. 14. Clearly, these protocols ensure privacy just as long as the AP is trusted: If the attacker controls the infrastructure, then different solutions should be sought.

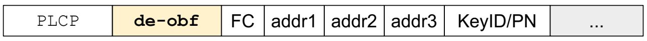

Figure 18: Modified 802.11 frame (simplified) for the iterative approach including the position of the `DE-OBF` field, encrypted with the same cypher used for the frame but with a dedicated ephemeral key derived during 4WHS.

7.1. Improving Communications Performance

Fig. 14 shows that the communication performance can suffer at high transmission speeds. This result confirms the conclusions we draw in early works (specifically [7] and [11]) and is due to the high sensitivity of higher-order Quadrature Amplitude Modulation (QAM) modulations to signal distortion. Albeit we cannot exclude that theoretic work on the obfuscation function can improve the situation, a simple and practical solution is letting the legitimate receiver remove the obfuscation artifacts. In practice the receiver should know the pre-distortion mask in Eq. (2) that is applied to each frame for obfuscating its CSI: this allows to remove the obfuscation from all OFDM symbols before they are decoded.

The first possibility is an *iterative decoding approach*: Including the obfuscation mask in every frame as an additional field, let's call it `DE-OBF`, between the standard 802.11 PLCP header and the data part, as shown in Fig. 18. The advantage of this approach is its robustness (each frame remains strictly a datagram completely independent from others) and the lack of any synchronization requirement. The overhead is $N_{\mathrm{sc}} \times N_{\mathrm{bo}}$ bits, where here N_{sc} is the number of carriers actually used (i.e., pilots excluded) and N_{bo} is the number of bits to represent the multiplication factor (2 in our implementation). Assuming to improve the multiplication granularity to 8 bits, the overhead ranges from 52 bytes for simple 20 MHz 802.11a/g systems to 484 bytes for an advanced 160 MHz 802.11ac system.[4] These fields should be transmitted at basic-rate since they need to be decoded correctly with high probability and a short Cyclic Redundancy Code (CRC) code may be needed to protect them. They must also be encrypted, possibly using the same cipher as the data part, but using a dedicated ephemeral key derived during the 4WHS. The key disadvantage of this approach, apart from the overhead, is the need for iterative decoding: first the receiver needs to decode the frame until `DE-OBF` without inverting the obfuscation function, just as we do in this implementation, next, using the knowledge of the `DE-OBF` field, the complete decoding of the frame is carried out.

A second possibility is a *time-based protocol*, i.e., the AP and STA share a common pseudo-random generator, synchronize (seed) it appropriately, and then extract a new value from it every T_{obf} s. This choice departs slightly from the implementation we presented, as it entails a time-based evolution of the process in Eq. (1) and not a frame-based one. As there is always

[3] The 802.11bf PAR was approved in Sept. 2020 and has already released some draft documents (`https://standards.ieee.org/ieee/802.11bf/10365/`). For a recent survey of the Task Group activities see [35].

[4] Note we have considered a single stream 802.11ac transmission. For more complex encodings the overheads scale with the number of spatial streams.

the possibility of de-synchronization of AP and STA that have to be re-synced as discuss hereinafter, we assume that the devices time-counting is good enough for standard operation. A frame-based evolution is difficult to conceive because of frame losses and re-transmissions, which would lead to continuous de-synchronization of AP and STA.

The initial synchronization can be obtained at the end of the 4WHS (see Fig. 17) with two different approaches:

1. Derive the seed from the ephemeral keys;
2. Explicitly transmit the seed in a management frame that, being protected by the encrypted channel, will be secure.

The second option introduces a new frame, which may be a drawback in the standardization procedure, but has the advantage that re-synchronization during normal operation comes almost for free. For instance we can imagine that, after a "long" silence period, the AP (recall that only the AP→STA traffic is used for sensing) re-starts by transmitting this management frame with a new seed, which does not need obfuscation as a single frame does never allow any meaningful sensing. Similarly, the AP can re-seed the STA if too many re-transmissions happened, assuming that repeated losses may be caused by the loss of the obfuscation mask sync.

With the first option, instead, there is the need for the receiver to communicate the loss of sync to the transmitter. This implies the need of a further function to understand that the sync is lost. Both functions require some form of signaling communication between the AP and the STA, thus the advantage of not introducing a novel management frame is partially lost.

The advantage of the time-based approach is an almost zero overhead on the channel but, most of all, it does not require iterative decoding. On the other hand, the STA has to continuously update the obfuscation mask in its NIC even when the mask is not used in order to ensure the alignment of the sequences.

We can also imagine mixed solutions between the iterative decoding approach and the time-based protocol. For instance, in the time-based protocol, the AP can send the obfuscation mask not as a header field, but as user-data. This way the receiver does not need to do iterative decoding, but decodes the frame based on its own obfuscation mask, which is then compared with the transmitter one, and if they do not match a re-sync is needed. To reduce the overhead the AP can send this information only every N_{obf} frames or send only some values of the mask in every frame (e.g., the amplification factor of a fraction of the subcarriers), or a mix of the two.

The selection of the most appropriate solution can be done only with a formal design of the protocol and also after appropriate experiments that measure the achievable performance, but this is outside the scope of this paper.

Per STA obfuscation and de-obfuscation may look an extreme choice, and quite resource consuming. If the BSS is fully trusted one may think of using the same masking pattern for all the STAs to simplify the system. This solution, however, works properly only with the iterative approach. Conversely, with the time-based protocol with proactive re-sync by the AP, long silence period leads to loss sync problems calling for aggressive transmissions of management re-seeding frames that should be STA-dependent, as the WPA encryption is different for every station. Furthermore, the fact that the the pattern is the same for all stations may be an advantage for an attacker who wants to invert the obfuscation function. Thus we do not suggest to take this direction.

7.2. Enabling Multi-Point Sensing

Wi-Fi sensing is not just an annoying threat to privacy. It can indeed be a useful function to provide innovative services such as multi-point sensing, as proposed in [8, 11, 38], which promises to achieve much better performance than single-point sensing. The de-obfuscation protocol described above can help maintain high communication performance, but in general prevents sensing unless it is done at the STA that receives the information flow. This, however, will hardly work, as the CSI-based sensing techniques we are considering are based on fingerprinting of the environment, hence require that also the sensing device(s) are in a fixed position.

At this early stage of the analysis, it seems excessively difficult to design a de-obfuscation protocol that allows exploiting the standard transmissions from an AP when they are obfuscated with per-STA pre-processing. Rather, it would be simpler to exploit a specific sensing channel, which in some sense is similar to an active attack as we considered in [9, 24]. Clearly, if sensing is legitimate, we cannot talk about an 'attack,' but obfuscation is welcome in any case to prevent non-legitimate use of this sensing channel.

We define a sensing channel as a low frame-rate flow. Beacons themselves can be used for CSI-based sensing, which is the reason why we assume that a complete privacy-preserving architecture contemplates also the obfuscation of beacons, but additional frames can be used to improve sensing capabilities or precision. Ideally, the problem can be solved using one (or more) common secure channel(s) to distribute the obfuscation parameters of beacons and additional frames devoted specifically to sensing.

It is well known that Wi-Fi traditionally has problems with multicast [39], thus we only sketch here some possible paths to explore in the future, without the claim to present a full, detailed design. A possible solution is to implement a separate obfuscation pattern for the sensing channel and exploit a group key (also known as shared key or multicast keys) to distribute the channel sensing parameters to all STAs in the BSS. Group keys are generated at the end of the 4WHS together with session keys, and can be used for several purposes, whose discussion is not due here. Using a group key and some additional signaling similar to what we discussed in Sec. 7.1, we can obtain the de-obfuscation of the sensing channel. This can be done for a subset of fixed STAs devoted to sensing, or can be done for all STAs also improving the reception of beacons. Attackers would not be able to access it because they are not allowed to enter the BSS. An obvious exception are public BSSs, but this discussion goes beyond our scope.

8. Conclusions and Future Work

Wi-Fi sensing is a novel technology that promises a huge leap in communication services under Wi-Fi coverage. Indeed, the notion of joint communication and sensing is one of the pillars of networking beyond 5G, making the scope of channel sounding and sensing go beyond Wi-Fi. At the same time these technologies pose unprecedented threats to privacy and security, as the information leak happens at the physical layer so it cannot be countered with cryptographic tools.

This paper has presented the implementation in openwifi of an anti-sensing obfuscation technique working for both 802.11a/g and 802.11n versions of the standard; preliminary proof-of-concept works have shown that these techniques work well also for 802.11ac, so we have no reason to imagine that implementations in such systems—that cannot be done for technological reasons in openwifi—will not work. The measures of its performance compared with those obtained in previous works with Matlab+SDR emulation show that despite the limitations imposed by a real implementation the proposed obfuscation still works. Furthermore, possible protocols to allow legitimate inversion of the obfuscation function have been discussed. This work therefore shows that the idea of signal obfuscation is fully implementable without hampering any functionality of Wi-Fi. Moreover, it indicates how it can be standardized to achieve a win-win solution able to maintain high communication performance with all the advantages of sensing while fully protecting the users.

Acknowledgement

This work has been partially funded at the University of Brescia by GÉANT Educational Activities and Services Agreement ref. SER-21-142, Project “Design and Implementation of an 802.11 Privacy Preserving Sub-Layer (DI-P^{2}SL).”

Furthermore, the work has been enabled in its early stage by the donation of two Xilinx ZC706 evaluation board by the Xilinx University Program (`https://www.xilinx.com/support/university.html`) that we used for the FPGA implementation.

References

[1] W. Saad, M. Bennis, M. Chen, A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems, IEEE Network 34 (3) (2020) 134–142.

[2] H. Tataria, M. Shafi, A. F. Molisch, M. Dohler, H. Sjöland, F. Tufvesson, 6G Wireless Systems: Vision, Requirements, Challenges, Insights, and Opportunities, Proceedings of the IEEE 109 (7) (2021) 1166–1199.

[3] H. Wymeersch, A. Pärssinen, T. E. Abrudan, A. Wolfgang, K. Haneda, M. Sarajlic, M. E. Leinonen, M. F. Keskin, H. Chen, S. Lindberg, P. Kyösti, T. Svensson, X. Yang, 6G Radio Requirements to Support Integrated Communication, Localization, and Sensing, in: 2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), 2022, pp. 463–469.

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

[5] I. Nirmal, A. Khamis, M. Hassan, W. Hu, X. Zhu, Deep Learning for Radio-Based Human Sensing: Recent Advances and Future Directions, IEEE Comm. Surveys & Tutorials 23 (2) (2021) 995–1019.

[6] Y. Gu, Y. Wang, T. Liu, Y. Ji, Z. Liu, P. Li, X. Wang, X. An, F. Ren, EmoSense: Computational intelligence driven emotion sensing via wireless channel data, IEEE Trans. Emerg. Top. Comput. Intell. 4 (3) (2020) 216–226.

[7] M. Cominelli, F. Kosterhon, F. Gringoli, R. Lo Cigno, A. Asadi, IEEE 802.11 CSI randomization to preserve location privacy: An empirical evaluation in different scenarios, Elsevier Computer Networks 191 (22) (2021) 107970.

[8] M. Cominelli, F. Gringoli, R. Lo Cigno, On the properties of device-free multi-point CSI localization and its obfuscation, Elsevier Computer Communications 189 (2022) 67–78.

[9] M. Cominelli, F. Gringoli, R. Lo Cigno, AntiSense: Standard-compliant CSI obfuscation against unauthorized Wi-Fi sensing, Elsevier Computer Communications 185 (2022) 92–103.

[10] L. Ghiro, M. Cominelli, F. Gringoli, R. Lo Cigno, On the Implementation of Location Obfuscation in openwifi and Its Performance, in: 20th IEEE Mediterranean Communication and Computer Networking Conference (MedComNet 2022), 2022, pp. 64–73.

[11] M. Cominelli, F. Gringoli, R. Lo Cigno, Passive Device-Free Multi-Point CSI Localization and Its Obfuscation with Randomized Filtering, in: 19th IEEE Mediterranean Communication and Computer Networking Conference (MedComNet), 2021, pp. 1–8.

[12] H. Rahbari, M. Krunz, Secrecy beyond encryption: obfuscating transmission signatures in wireless communications, IEEE Communications Magazine 53 (12) (2015) 54–60.

[13] K. Chetty, G. Smith, K. Woodbridge, Through-the-Wall Sensing of Personnel Using Passive Bistatic WiFi Radar at Standoff Distances, IEEE Trans. on Geoscience and Remote Sensing 50 (4) (2012) 1218–1226.

[14] F. Adib, D. Katabi, See through walls with WiFi!, in: ACM Int. Conf. of the Special Interest Group on Data Communication (SIGCOMM), Hong Kong, 2013, pp. 75–86.

[15] Z. Yang, Z. Zhou, Y. Liu, From RSSI to CSI: Indoor Localization via Channel Response, ACM Computing Surveys 46 (25) (2013) 1–32.

[16] K. Wu, J. Xiao, Y. Yi, D. Chen, X. Luo, L. Ni, CSI-Based Indoor Localization, IEEE Trans. Parallel Distrib. Syst. 24 (7) (2013) 1300–1309.

[17] R. C. Shit, S. Sharma, D. Puthal, P. James, B. Pradhan, A. van Moorsel, A. Y. Zomaya, R. Ranjan, Ubiquitous localization (UbiLoc): A survey and taxonomy on device free localization for smart world, IEEE Comm. Surveys & Tutorials 21 (4) (2019) 3532–3564.

[18] Ma, Y. and Zhou, G. and S. Wang, S., WiFi Sensing with Channel State Information: A Survey, ACM Computing Surveys 52 (46) (2019) 1–36.

[19] D. Nguyen, C. Sahin, B. Shishkin, N. Kandasamy, 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, 2014, p. 15–22.

[20] M. Schulz, F. Gringoli, D. Steinmetzer, M. Koch, 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), 2017, p. 111–121.

[21] Y. Qiao, O. Zhang, W. Zhou, K. Srinivasan, A. Arora, PhyCloak: Obfuscating Sensing from Communication Signals, in: 13th USENIX Conf. on Networked Systems Design and Implementation (NSDI'16), Santa Clara, CA, USA, 2016, p. 685–699.

[22] M. Cominelli, F. Kosterhon, F. Gringoli, R. Lo Cigno, A. Asadi, An Experimental Study of CSI Management to Preserve Location Privacy, in: 14th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization (WiNTECH), London, UK, 2020, pp. 1–8.

[23] L. F. Abanto-Leon, A. Bäuml, G. Sim, M. Hollick, A. Asadi, Stay Connected, Leave no Trace: Enhancing Security and Privacy in WiFi via Obfuscating Radiometric Fingerprints, Proc. ACM Meas. Anal. Comput. Syst. 4 (44) (2020) 1–31.

[24] M. Cominelli, F. Gringoli, R. Lo Cigno, Non Intrusive Wi-Fi CSI Obfuscation Against Active Localization Attacks, in: 16th IFIP/IEEE Conf. on Wireless On demand Network Systems and Services (WONS), 2021, pp. 87–94.

[25] Y. Zhou, H. Chen, C. Huang, Q. Zhang, Wiadv: Practical and robust adversarial attack against wifi-based gesture recognition system, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 6 (2) (2022) 92:1–92:17.

[26] P. Staat, S. Mulzer, S. Roth, V. Moonsamy, M. Heinrichs, R. Kronberger, A. Sezgin, C. Paar, IRShield: A countermeasure against adversarial physical-layer wireless sensing, in: 2022 IEEE Symposium on Security and Privacy (SP), 2022, pp. 1705–1721.

[27] X. Jiao, M. Mehari, W. Liu, M. Aslam, I. Moerman, Openwifi CSI Fuzzer for Authorized Sensing and Covert Channels, in: 14th ACM Conf. on

Security and Privacy in Wireless and Mobile Networks, 2021, p. 377–379.

[28] P. Y. Chan, A. I.-C. Lai, P.-Y. Wu, R.-B. Wu, Physical Tampering Detection Using Single COTS Wi-Fi Endpoint, Sensors 21 (16) (2021) 1–15.

[29] M. Lichtman, M. T. Vondal, T. C. Clancy, J. H. Reed, Antifragile communications, IEEE Systems Journal 12 (1) (2018) 659–670.

[30] W. Yan, T. Voigt, C. Rohner, RRF: A Robust Radiometric Fingerprint System that Embraces Wireless Channel Diversity, in: Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks, ACM, 2022, pp. 85–97.

[31] I. Voras, Fixed point math library for c, `https://sourceforge.net/projects/fixedptc`, copyright (c) 2010-2012. Ivan Voras <ivoras-freebsd.org> Released under the BSDL. (2020).

[32] J. Corbet, A. Rubini, G. Kroah-Hartman, Linux device drivers, 3rd Edition, O'Reilly Media, 2005.

[33] D. Thomas, P. Moorby, The Verilog® hardware description language, 5th Edition, Springer Science & Business Media, 2008.

[34] F. Gringoli, M. Schulz, J. Link, M. Hollick, Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets, in: 13th ACM Int. Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH '19), Los Cabos, Mexico, 2019, pp. 21–28.

[35] F. Restuccia, IEEE 802.11bf: Toward Ubiquitous Wi-Fi Sensing (3 2021). `arXiv:2103.14918`.

[36] Wireless lan medium access control (mac) and physical layer (phy) specifications, amendment 6: Medium Access Control (MAC) Security Enhancements (2004).

[37] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (2013).

[38] E. Gönültaş, E. Lei, J. Langerman, H. Huang, C. Studer, CSI-Based Multi-Antenna and Multi-Point Indoor Positioning Using Probability Fusion, IEEE Trans. on Wireless Communications (2021 (Early Access)).

[39] C. E. Perkins, M. McBride, D. Stanley, W. Kumari, J.-C. Zúñiga, Multicast Considerations over IEEE 802 Wireless Media, RFC 9119, IETF (10 2021).

Appendix A. Code and development details

The development described and tested in this work is a fork of the original openwifi project, which we thoroughly documented and made publicly available through github. The obfuscation mechanism, as described, requires modifications both to the Linux driver and to the FPGA design. The fork related to the Linux driver and software is available at `https://github.com/ansresearch/openwifi/tree/csiobfuscation-ans`; the fork related to the FPGA hardware design is available at `https://github.com/ansresearch/openwifi-hw/tree/csiobfuscation-ans`. Modification to the FPGA can probably be optimized and further studies on obfuscation techniques and design for implementation optimization are welcome.

The authors are willing to discuss and cooperate for furthering the project. Additional information related to the DI-P^{2}SL project is available at `https://ans.unibs.it/projects/di-p2sl/`.