On the Implementation of Location Obfuscation in openwifi and Its Performance

Lorenzo Ghiro, Marco Cominelli, Francesco Gringoli, Renato Lo Cigno
DII – University of Brescia / CNIT, Italy

***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. Usually, some signal pre-processing at the transmitter side is needed to introduce pseudo-random (artificial) patterns in the channel response estimated at the receiver, preventing the extraction of meaningful information from the channel state. However, there is currently just one implementation of such techniques in a real system (openwifi), and it has never been tested for performance. 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 paper 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.**

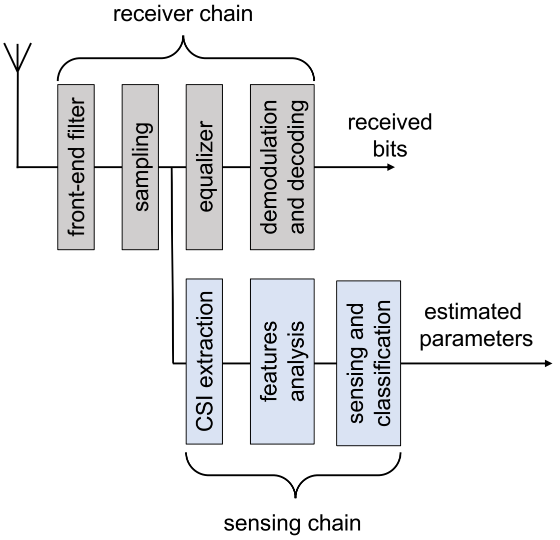

Figure 1: High-level view of the Wi-Fi sensing process.

I. 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. However, the review [1] highlights how the subjects of Wi-Fi sensing are most often *human beings*, their position, activity, state, even speech or mood [2]. This concern is not limited to Wi-Fi systems, as also for wireless communications beyond 5G it is believed that the information derived from advanced Channel State Information (CSI) analysis will be fundamental for both improving communication performance and developing innovative systems [3], [4]. 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.

This observation pushed other research groups and us to study whether there is a way to counter Wi-Fi sensing without hampering communication performance. Countering Wi-Fi sensing means concealing the information on the environment carried by the electromagnetic signals and retrieved via 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 to understand 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 [5]–[8] are feasibility studies presenting proof-of-concept architectures that can achieve obfuscation. These studies present methodologies and experiments based on offline signal processing; however, 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.

The contribution of this paper is precisely in the direction of filling the gap just described. We present an implementation of the technique presented in [6], [7] extending openwifi, and we discuss limitations imposed by the implementation framework. Next, we introduce directions for possible integration of 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.

II. Involved Technologies

The openwifi project 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

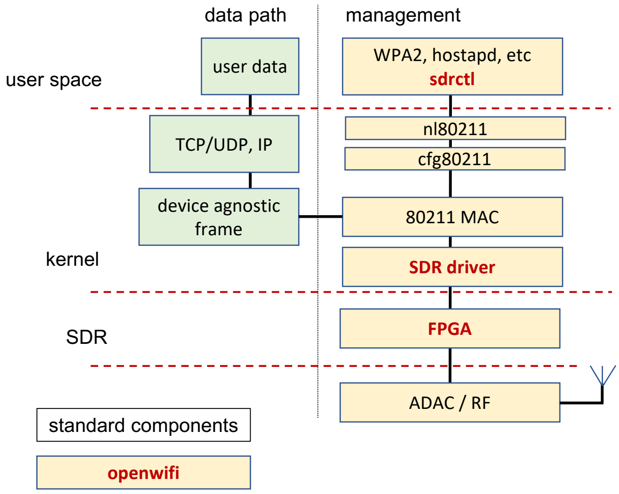

Figure 2: Simplified structure of an openwifi station, in red the components that are developed within the openwifi project.

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 MAC executes all time critical operations like carrier sensing, frame scheduling, and ACK replying; 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).

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 above. Modifying the PHY part of the Verilog design makes it possible to implement the obfuscation of transmitted signals as described in [6], [7], and in Sect. V. The behavior of the obfuscator can be customized from the kernel space, i.e., by adding proper code to the openwifi Linux driver.

III. Related Work

The interest in CSI obfuscation is high because of the privacy concerns raised by the ever-increasing accuracy of novel Wi-Fi sensing algorithms. However, the topic has been tackled only by a few works from our and other research groups [5]–[7], [9]–[13]. 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 Sect. I, or open platforms as openwifi described in Sect. II.

We limit the discussion in this paper to actual implementations of obfuscating techniques, i.e., to those works that preserve the full functioning of Wi-Fi communication. At the same time, we refer the reader interested in the topic of Wi-Fi sensing to seminal works like [14]–[17] or to recent reviews [1], [18]. For what sensing obfuscation is concerned, instead, we rely on and refer to the discussion in [6], [7].

The first implementation of a CSI obfuscator has been presented by the developers of openwifi [19]. 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 [19], 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 [20], 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.

IV. Transmitter Side CSI Distortion

The concealment of the CSI to prevent sensing can be obtained by proper pre-processing the transmitted signals, as already introduced in Sect. I and III. The initial theoretical modeling plus a proof-of-concept for testing the feasibility of this concealment process have been tackled in [5]–[7]: 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 Artificial Intelligence (AI) techniques to fingerprint some details of the environment, e.g., the position of a person in a room. Later on the extracted fingerprint can be used also to classify –and hence recognize– the environment, which means localizing the person in the room. 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 Access Point (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 sufficient for fingerprinting the environment, at the same time should not jeopardize the communication. Summarizing the content of [7] Sect. 3 and [6] Sect. V, a proper pre-distortion can be obtained with the random process described by Equations (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 $k-1$, 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; $\max = 1.9$ and $\min = 0.1$ are bounds to guarantee that the pre-distortion multiplication never leads to unrealistic high values, but most of all never completely suppress a carrier, as this would obviously hamper communications.

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

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

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 $\Delta_t(k) \geq 15\,\text{s}$, then $A_O\ (k)$ and $A_O\ (k\text{-}1)$ are almost completely uncorrelated –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.

V. 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 Sect. V-B, and ii) the FPGA, described in Sect. V-C. Before delving into the details, we start from an high-level description of the modifications to openwifi.

A. High Level Design

Fig. 3 illustrates the FPGA and Linux kernel components modified to develop the P^{2}SL, highlighting how our modifications focus on the NIC driver and on the PHY layer, while the MAC remains essentially untouched. The implemented obfuscator follows the design described in Sect. IV, 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.

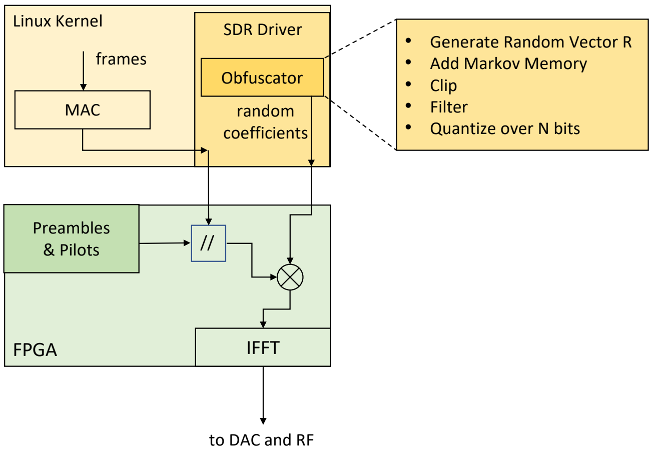

Figure 3: Overview of P^{2}SL design with kernel and FPGA modifications.

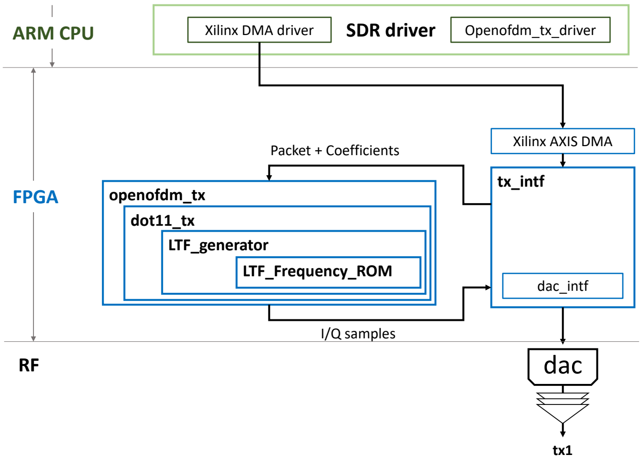

Figure 4: Detailed structure of the openwifi as modified by P^{2}SL with focus on the transmitter side of the FPGA.

In our implementation the $\mathbf{R}$ size N_{sc} is equal to 64, reflecting the fact that openwifi currently supports only 20 MHz channel bandwidth with 64 OFDM subcarriers. We used the Fixed Point Math library for C [21] 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 digits, and retain a good accuracy when doing operations with 5 decimal digits. Space constraints and the choice of not implementing an Arithmetic Logic Unit (ALU) in the FPGA imply that further approximations need to be introduced and controlled at the PHY layer.

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 MAC to PHY.

With reference to Fig. 4, our modifications to the openwifi on the FPGA side can be summarized in three main key

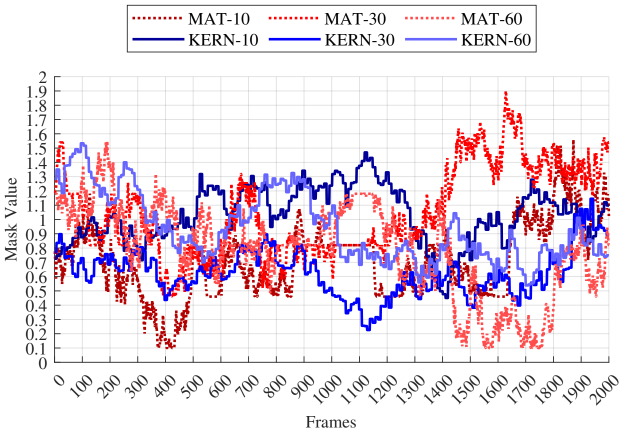

Figure 5: Realizations of Equations (1) and (2) for 2000 frames equally spaced by 1 ms: With Matlab (red) and implemented in the kernel driver (blue).

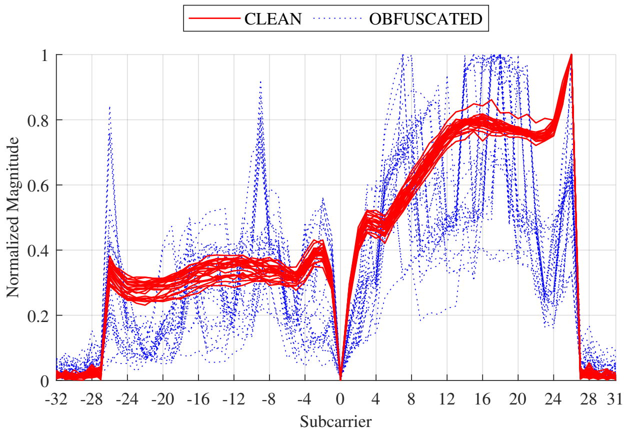

Figure 6: Normalized Magnitude of subcarriers measured at a receiver for short burst of frames with obfuscation off (red) and on (dashed blue)

points:

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

Fig. 5 to 7 discuss the impact of implementation impairments before describing the implementation details.

Fig. 5 compares three realizations of the Equations (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 (Equation (2)) significantly modify the behavior.

Sect. VI 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 frames which are much less energetic (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.

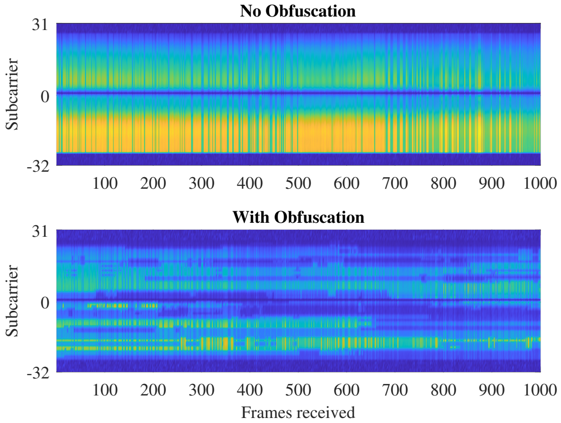

Figure 7: Magnitude of the CSI collected from 1000 frames, with brighter tending to yellow indicating a larger magnitude.

B. Kernel Driver

As already anticipated, the code we have introduced in the openwifi driver is mainly responsible for the generation of the $N_{\rm sc}$-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 Equation (1)) despite the lack of a kernel clock, relying instead on jiffies [22];

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

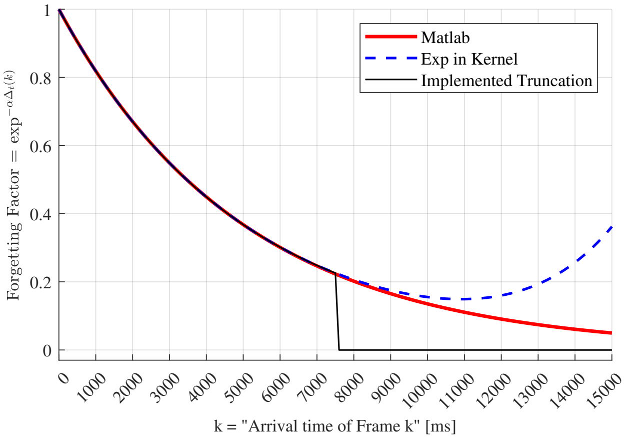

Figure 8: Approximation of $\exp^{-\alpha\Delta_t(k)}$ for $\Delta_t \in [0, 10000]$ ms computed in kernel with a 6-th Taylor-McLaurin approximation versus the Matlab computation; the precision decays after 10 s.

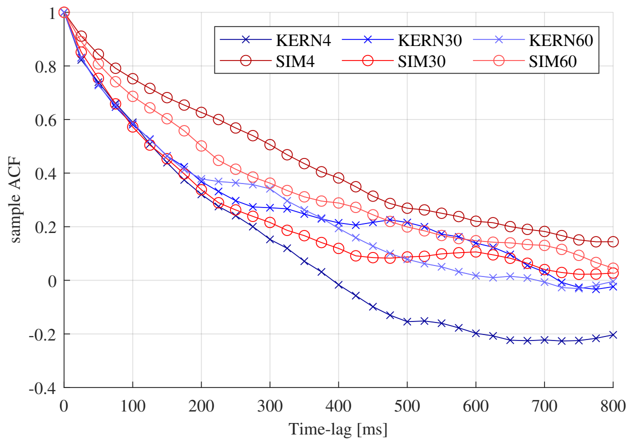

Figure 9: Comparison of the ACF computed on the realizations of the Markovian processes described by Equation (2) for 3 subcarriers. The red curves (with circles) represent the ACF obtained with Matlab simulations, while the blue (with x) ones have been generated by the kernel code.

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 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 our implementation is slightly smaller than the theoretic one, indicating a modestly weaker memory. This can be justified by the truncation of the exponential explained with Fig. 8.

C. FPGA Design

The FPGA side of openwifi, written in Verilog [23], has been modified to implement the obfuscation of the IQ samples of each frame, as schematically illustrated in Fig. 10. Though conceptually simple, this requires to modify the state machine that implements the 802.11 Transmission Chain, and to design an approximated multiplication stage. The implementation of a complete ALU running in parallel on $2 \times N_{\text{sc}}$ IQ samples is a considerable effort and will not be feasible in small FPGAs, thus is left for future work. The main FPGA modifications can be summarized as follows:

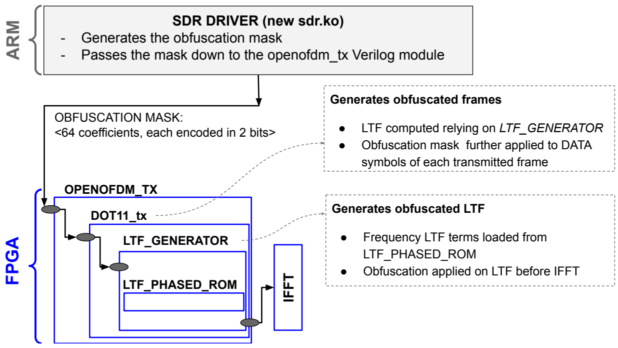
ARM
SDR DRIVER (new sdr.ko)
- Generates the obfuscation mask
- Passes the mask down to the openofdm_tx Verilog module
OBFUSCATION MASK:
<64 coefficients, each encoded in 2 bits>
Generates obfuscated frames
• LTF computed relying on LTF_GENERATOR
• Obfuscation mask further applied to DATA symbols of each transmitted frame
FPGA
OPENOFDM_TX
DOT11_tx
LTF_GENERATOR
LTF_PHASED_ROM
IFFT
Generates obfuscated LTF
• Frequency LTF terms loaded from LTF_PHASED_ROM
• Obfuscation applied on LTF before IFFT

Figure 10: Overview FPGA modifications.

1) *Addition & wiring of FPGA registers.* With reference to Fig. 4, a set of FPGA registers configurable at runtime by the driver have been allocated and pinned, in cascade, to all modules involved in a frame transmission, starting from the XILINX AXIS DMA downto the internal blocks of the OPENOFDM_TX module, where IQ samples are actually manipulated. From an hardware point of view this operation coincide with the wiring of different input/output pins. The obfuscation coefficients are thus made available to the FPGA through these registers. In particular, these registers can store up to 128 bits, which implies that each of the $N_{\text{sc}} = 64$ 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.
2) *Obfuscation of Preambles.* In the original openwifi design, the preamble samples—constant for each 802.11g frame—were stored in the time domain in a dedicated ROM memory, and then pre-pended to the frame after the 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 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. We therefore need to change the sequence of operations defined by openwifi for transmitting a frame. To this end, we introduce a new (Verilog-defined) Finite State Machine (FSM) obfuscating the Long Training Field (LTF) in the preamble of 802.11g frames, and we integrate this FSM within the pipelined workflow previously defined in the openwifi FPGA. This way, we ensure that preambles are obfuscated before being pre-pended to the IQ flow sent to the Inverse Fast Fourier Transform (IFFT) module for modulation and transmission.
3) *Signal pre-distortion.* The IQ 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. This should be a minor impairments since the Analog-Digital-Analog Converter (ADAC) board normalizes the power output. In particular, the right-shift operation to be applied is chosen for each subcarrier according to the quantization thresholds described by Equation (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 IQ sample before obfuscation;
- IQ_{obf} is the same IQ sample after the desired right-shift operation has been applied, thus, the obfuscated version of the IQ sample.

According to this implementation of the obfuscation technique in FPGA, the amplitude of each subcarrier is attenuated (or left unchanged) when the obfuscation is turned on.

VI. Performance

The goals of the experimental performance evaluation are two:

1) Quantify the effectiveness of the implemented obfuscation in preventing localization (Sect. VI-A);
2) Determine if and how much the obfuscation deteriorates communication performance (Sect. VI-B).

A. 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 a capital **P** 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 with capital L capture traffic, and the CSI extracted from the captured traffic feeds the Convolutional Neural Network (CNN) described in [6], [7] 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 [24].

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 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 how the localization accuracy drops, for all 4 localizing devices, when the obfuscation is active. Compared to the theoretical results in [6], [7] the CLEAN (i.e., not obfuscated) localization is less accurate. We think that this is due to the use of a 20 MHz 802.11g system as allowed by openwifi instead of a 80 MHz 802.11ac system, which stress how it is important to study and design anti-sensing systems as sensing will become extremely accurate as technology evolves. The OBFUSCATED results are just above the random choice, indicating that the obfuscation process can still be improved. 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.

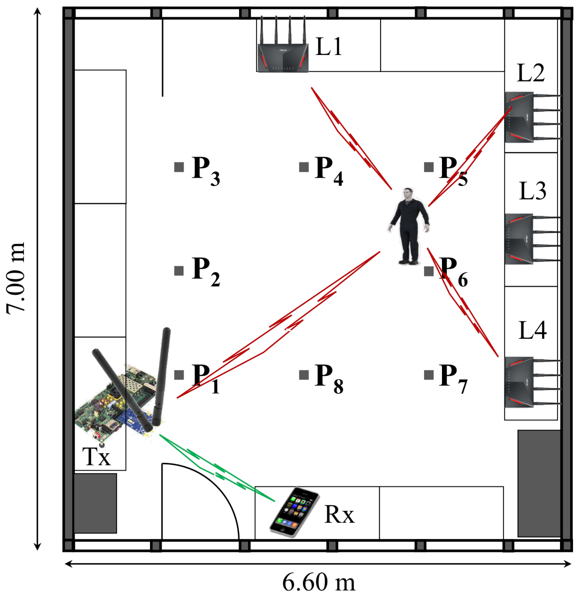

Figure 11: Placement of Transmitter (TX), Receiver (RX) and Localizing devices in our Lab at the University of Brescia.

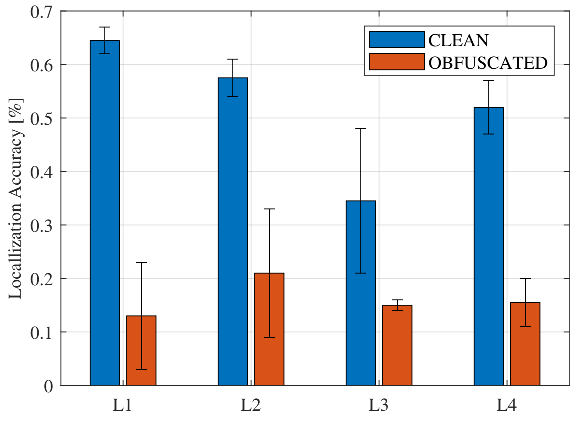

Figure 12: Bar chart comparing, for each localizing device L1-L4, the mean localization accuracy when captured traffic is CLEAN (i.e., not obfuscated, blue bars) and when it is instead OBFUSCATED (orange bars). Whiskers indicates the maximum and minimum accuracy achieved while attempting localization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

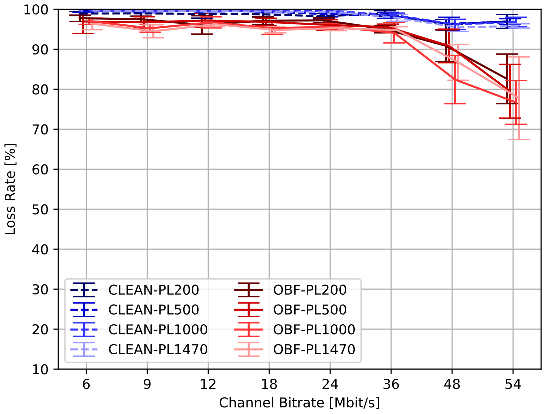

Figure 13: PDR as a function of the channel bitrate with obfuscation off (blue), and on (red). Different curves refer to different frame lengths (IPERF UDP payload indicated). For each experiment the curves indicate the mean PDR, while the bars report the confidence interval with 95% confidence level.

B. Communications

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 verify for what channel/traffic conditions the activation of the obfuscation hampers the communication performance. In particular, we imposed the Tx radio interface to use the 802.11g supported Data Rates, namely: $cr \in CR = [6, 9, 12, 18, 24, 36, 48, 54]$ Mbit/s, and $pl \in PL = [200, 500, 1000, 1470]$ 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 Packet Delivery Rate (PDR) computed by the receiver as:

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

where the denominator of Equation (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 of with IPERF synchronization problems. If max(seqno) − min(seqno) is too small, the experiment is discarded. 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. 13 compares the mean PDR achieved when turning the obfuscation off (blue) and on (red). The PDR without obfuscation is in line with similar measures, very close to 100% with a small degradation for 48 and 54 Mbit/s. Obfuscation degrades performance only slightly up to 36 Mbit/s, while for higher rated the degradation is larger, similarly with what we observed in previous works with the Matlab implementation and 802.11ac. The conclusion is that, whatever the Wi-Fi technology adopted, the highest PHY rates are fragile and it is difficult to find a pre-distortion algorithm that preserves the communication un-hampered.

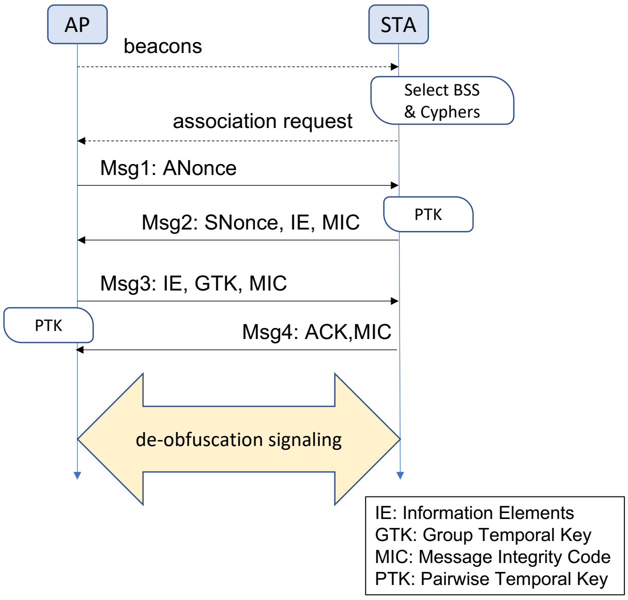

Figure 14: 802.11i Four Way Handshake (4WHS) showing also beacons, association and the position of additional signaling messages for de-obfuscation.

VII. 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 Sect. VI-B. 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 possible different 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[1] that are based on 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.[2]

First of all, consider the 4WHS of the Wi-Fi Protected Access (WPA) negotiation as defined in the standard [26] and schematically reported in Fig. 14, 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 [27], our considerations still apply as they only concern the 4WHS that anyways follows the SAE step.

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

[2] 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 [25].

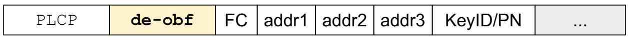

Figure 15: Modified 802.11 frame (simplified) for the iterative approach including the position of the DE-OBF field, encrypted with the same cypher used for the frame but with a dedicated ephemeral key derived during 4WHS.

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. 13. 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.

A. Improving Communications Performance

Fig. 13 shows that the communication performance can suffer at high transmission speeds. This result confirms the conclusions we draw in early works (specifically [7] and [6]) 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 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. 15. 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 $N_{\text{sc}} \times N_{\text{bo}}$ 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.[3] 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. And they must 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, seed it (synchronize 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 Equation (1) and not a frame-based one. As there is always 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. 14) 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,

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

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.

B. 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 [6], [28], 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 [8], [13]. 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 [29], 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 Sect. VII-A, 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.

VIII. 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 and measured its performance in comparison with those obtained in previous works with Matlab+SDR emulation, showing 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] I. Nirmal, A. Khamis, M. Hassan, W. Hu, and X. Zhu, "Deep Learning for Radio-Based Human Sensing: Recent Advances and Future Directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 995–1019, Feb. 2021.

[2] Y. Gu, Y. Wang, T. Liu, Y. Ji, Z. Liu, P. Li, X. Wang, X. An, and F. Ren, "EmoSense: Computational Intelligence Driven Emotion Sensing via Wireless Channel Data," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 4, no. 3, pp. 216–226, Jun. 2020.

[3] H. Tataria, M. Shafi, A. F. Molisch, M. Dohler, H. Sjöland, and F. Tufvesson, "6G Wireless Systems: Vision, Requirements, Challenges, Insights, and Opportunities," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1166–1199, Mar. 2021.

[4] W. Saad, M. Bennis, and M. Chen, "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[5] M. Cominelli, F. Kosterhon, F. Gringoli, R. Lo Cigno, and 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, Sep. 2020, pp. 1–8.

[6] M. Cominelli, F. Gringoli, and 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)*, Ibiza, Spain, Jun. 2021, pp. 1–8.

[7] 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," *Elsevier Computer Networks*, vol. 191, no. 22, p. 107 970, May 2021.

[8] M. Cominelli, F. Gringoli, and R. Lo Cigno, "AntiSense: Standard-compliant CSI obfuscation against unauthorized Wi-Fi sensing," *Elsevier Computer Communications*, vol. 185, pp. 92–103, Mar. 2022.

[9] 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.

[10] 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.

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

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

[13] M. Cominelli, F. Gringoli, and 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)*, Klosters, Switzerland, Mar. 2021, pp. 87–94.

[14] 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, Apr. 2012.

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

[16] Z. Yang, Z. Zhou, and Y. Liu, “From RSSI to CSI: Indoor Localization via Channel Response,” *ACM Computing Surveys*, vol. 46, no. 25, pp. 1–32, Dec. 2013.

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

[18] Ma, Y. and Zhou, G. and S. Wang, S., “WiFi Sensing with Channel State Information: A Survey,” *ACM Computing Surveys*, vol. 52, no. 46, pp. 1–36, Jun. 2019.

[19] X. Jiao, M. Mehari, W. Liu, M. Aslam, and I. Moerman, “Openwifi CSI Fuzzer for Authorized Sensing and Covert Channels,” in *14th ACM Conf. on Security and Privacy in Wireless and Mobile Networks*, Abu Dhabi, ARE, Jun. 2021, pp. 377–379.

[20] P. Y. Chan, A. I.-C. Lai, P.-Y. Wu, and R.-B. Wu, “Physical Tampering Detection Using Single COTS Wi-Fi Endpoint,” *Sensors*, vol. 21, no. 16, Aug. 2021.

[21] 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.

[22] J. Corbet, A. Rubini, and G. Kroah-Hartman, *Linux device drivers*, 3rd ed. O’Reilly Media, 2005.

[23] D. Thomas and P. Moorby, *The Verilog® hardware description language*, 5th ed. Springer Science & Business Media, 2008.

[24] 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 ACM Int. Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH ’19)*, Los Cabos, Mexico, Oct. 2019, pp. 21–28.

[25] F. Restuccia, *IEEE 802.11bf: Toward Ubiquitous Wi-Fi Sensing*, Mar. 2021. arXiv: 2103.14918.

[26] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Amendment 6: Medium Access Control (MAC) Security Enhancements, IEEE Std. 802.11i, 2004.

[27] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std. 802.11ac, 2013.

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

[29] C. E. Perkins, M. McBride, D. Stanley, W. Kumari, and J.-C. Zúñiga, “Multicast Considerations over IEEE 802 Wireless Media,” RFC 9119, Oct. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9119.