Massive Reactive Smartphone-Based Jamming using Arbitrary Waveforms and Adaptive Power Control

Matthias Schulz
Secure Mobile Networking Lab
TU Darmstadt, Germany
mschulz@seemoo.de

Francesco Gringoli
CNIT
University of Brescia, Italy
francesco.gringoli@unibs.it

Daniel Steinmetzer
Secure Mobile Networking Lab
TU Darmstadt, Germany
dsteinmetzer@seemoo.de

Michael Koch
Secure Mobile Networking Lab
TU Darmstadt, Germany
mkoch@seemoo.de

Matthias Hollick
Secure Mobile Networking Lab
TU Darmstadt, Germany
mhollick@seemoo.de

ABSTRACT

It is not commonly known that off-the-shelf smartphones can be converted into versatile jammers. To understand how those jammers work and how well they perform, we implemented a jamming firmware for the Nexus 5 smartphone. The firmware runs on the real-time processor of the Wi-Fi chip and allows to reactively jam Wi-Fi networks in the 2.4 and 5 GHz bands using arbitrary waveforms stored in IQ sample buffers. This allows us to generate a pilot-tone jammer on off-the-shelf hardware. Besides a simple reactive jammer, we implemented a new acknowledging jammer that selectively jams only targeted data streams of a node while keeping other data streams of the same node flowing. To lower the increased power consumption of this jammer, we implemented an adaptive power control algorithm. We evaluated our implementations in friendly jamming scenarios to oppress non-compliant Wi-Fi transmissions and to protect otherwise vulnerable devices in industrial setups. Our results show that we can selectively hinder Wi-Fi transmissions in the vicinity of our jamming smartphone leading to an increased throughput for other nodes or no blockage of non-targeted streams on a jammed node. Consuming less than 300 mW when operating the reactive jammer allows mobile operation for more than 29 hours. Our implementation demonstrates that jamming communications was never that simple and available for every smartphone owner, while still allowing surgical jamming precision and energy efficiency. Nevertheless, it involves the danger of abuse by malicious attackers that may take over hundreds of devices to massively jam Wi-Fi networks in wide areas.

ACM Reference format:
Matthias Schulz, Francesco Gringoli, Daniel Steinmetzer, Michael Koch, and Matthias Hollick. 2017. Massive Reactive Smartphone-Based Jamming using Arbitrary Waveforms and Adaptive Power Control. In *Proceedings of WiSec '17, Boston, MA, USA, July 18-20, 2017,* 11 pages.
DOI: 10.1145/3098243.3098253

1 INTRODUCTION

Wireless radio communication jammers have been around for decades. They are used for strategic advantages, hindering an opposing party from exchanging information, for example, in a military conflict or situations where remote trigger signals for explosive devices need to be suppressed. They are also used to protect vulnerable legacy systems from malicious communication [6, 7, 12, 19, 26, 30, 31], for example, pace makers that can be wirelessly reprogrammed without encryption and no authentication. Reactively jamming all unauthorized communication with those devices can be a life saver and protect a patient's privacy [12, 31].

Besides using jammers for friendly or public safety applications, radio jammers are also subject to abuse. Whoever owns a jammer for GSM and LTE bands can block cellular communication and in doing so also hinder victims in distress situations from using phones to make 9-1-1 calls to call for help. People tracked by the government with GPS anklets may use jammers to leave their allowed living zone but may additionally disturb other GPS applications in their vicinity. While those applications are definitely illegal in most countries of the world (see [9]), it is important to understand what malicious attackers can achieve. Existing radio communication hardware carried around by ordinary civilians can participate in attacks that we need to defend.

Smartphones, for instance, may be the most widespread radio transmission enabled devices. They are carried around by billions of people every day and are densely distributed in metropolitan areas where people constantly communicate wirelessly. A malicious attacker overtaking only a small fraction of these devices could create a network of densely distributed highly capable radio jammers that could trigger a wide spread denial of service attack against wireless communication. In addition, reactive jamming would allow to create a mesh network of cooperative distributed jamming nodes that could selectively jam any other communication while keeping an open control channel for their coordination.

In this work, we investigate the feasibility of smartphone-based wireless jammers, by implementing proof-of-concept firmwares for the Nexus 5 smartphone. The goals are always friendly-jamming applications to either hinder nodes from transmitting non-compliant Wi-Fi signals, or to protect industrial resources by using large numbers of employee smartphones to defend against attacks on legacy hardware. Our contributions are the following:

- We describe the architecture of Broadcom Wi-Fi chips required for transmitting arbitrary waveforms on any frequency supported by the chip.
- We implement a smartphone-based reactive jammer for Wi-Fi systems that can jam all receivable rates supported by Nexus 5 smartphones (e.g., 80 MHz SISO 802.11ac frames).
- We enhance the reactive jammer by sending acknowledgements to the frame transmitter to avoid retransmissions and the blockage of other non-targeted traffic.
- We further improve our jammer by using an adaptive power control algorithm to adjust the transmission power depending on the jamming success.
- We evaluate the jamming performance and the energy consumption of the different jamming approaches.

All of our experiments are designed with reproducibility in mind. Even though, our jammer would allow continuous-tone jamming, we solely focus on reactive jammers as they are required for friendly jamming applications.

The paper is structured as follows. After presenting related work in Section 2, we describe the Wi-Fi chip components and their use in jamming applications in Section 3. In Section 4, we describe the implementation of our jammers, followed by descriptions of experimental setups and evaluations in Section 5. In Section 6, discuss the results and conclude this work in Section 7.

2 RELATED WORK

Jamming comprises a wide range of attacks and methods to distort wireless communication and prevent devices from either receiving or sending valid packets. Applied methods range from simple techniques (e.g., continuous transmission of interference) to more sophisticated approaches that exploit properties of higher layer network protocols [21]. Proposed jamming attacks differ in terms of disrupting impact, implementation complexity, energy consumption, stealthiness, and anti-jamming resistance [21, 32]. The prevalent types of jamming are (1) the constant jammer that transmits noise to corrupt frames or make the receivers sense a busy channel, (2) the deceptive jammer that is similar to the constant jammer but transmits arbitrary but valid protocol frames, (3) the random jammer that pauses a random time between transmissions to increase power efficiency, and (4) the reactive jammer [29] that targets only selected frames on the fly by detecting and corrupting them. All of these jamming strategies have been shown to have a significant impact but exhibit limitations in terms of efficiency, detectability, or resistance [21].

Designing effective and efficient jammers is challenging. Constant and deceptive jamming have a high energy consumption that can be decreased by random jamming at cost of effectivity. The reactive jammer is efficient and effective but must handle strict real-time requirements: it must detect and distort a packet during transmission. A variety of research papers address efficiency aspects of jamming and show that smart jamming strategies can improve performance [22]. In [11] DeBruhl *et al.* apply game theory to develop energy-efficient jamming and anti-jamming strategies. Bayraktaroglu *et al.* [2] investigate the theoretical impacts of jamming IEEE 802.11 networks. Jamming techniques that disturb the pilot tones used for channel estimation and equalization have been shown to hinder receivers from decoding data packets [8, 15, 25]. Wilhelm *et al.* [29] demonstrate the feasibility of reactive jamming on IEEE 802.15.4 networks in Software Defined Radio (SDR) based experiments. Vo-Huu *et al.* [28] demonstrate highly efficient reactive jamming and construct a jamming pattern across multiple OFDM subcarriers in IEEE 802.11a/g/n to be amplified by de-interleaving at the receiver. Therewith, they completely block a Wi-Fi communications with a jamming power of less than 1% of the communication power in an SDR-based testbed. However, SDRs are rather expensive and cannot achieve performance results comparable to that of practical low-cost commodity hardware. This leads researchers to investigate jamming in common network devices. In [27], Vanhoef *et al.* implemented a continuous and reactive jammer on top of an open source Atheros firmware. Although having limited access to the firmware, they showed that jammers can be implemented on commodity hardware. Berger *et al.* [4, 5] investigated reactive jamming with off-the-shelf IEEE 802.11 access points. To that end, they directly modified the microcode of the Network Interface Card (NIC) facilitated by the OpenFWWF project [14] and integrated their own jamming functionality. The Nexmon project [23, 24] followed a similar approach, but aims at wireless chipsets deployed in mobile phones. Therewith, it provides the ideal foundation to investigate mobile jammers in practical networks.

Besides applications in malicious abuse, jamming is also applied in the security context as “friendly jamming” or “jamming for good”. Recent work in this area utilizes jamming to either (a) block unauthorized communication or (b) secure confidential transmissions [4, 5]. To block unauthorized communication (a), reactive jamming can interfere with undesired frame transmissions and prevent them from being received at any receiver [6, 7, 12, 19, 26, 30, 31]. For example, in [12] Gollakota *et al.* protect Implantable Medical Devices (IMDs) by jamming unauthorized commands. In [30], Wilhelm *et al.* propose a firewall for IEEE 802.15.4 networks that jams according to an arbitrary rule set. Martinovic *et al.* [19] develop message authentication mechanisms for Wireless Sensor Networks (WSNs). To secure confidential transmissions (b), jamming can prevent potential eavesdroppers from decoding a signal by causing artificial interference. This effectively allows only selected receivers to receive a confidential message. For example, Wenbo *et al.* [26] control jamming signals with secret keys that only allow authenticated devices to recover transmitted signals but cause unpredictable interference to others. Anand *et al.* [1] induce interference into the null-space of a MIMO transmission and therewith jam all but the intended receiver. Gollakota *et al.* [13] propose a physical layer key exchange based on random jamming patterns that distort parts of a transmission. Kim *et al.* [17] use multiple jammers to form a physical secure area around access points in which communication remains possible. They adjust the jammers to jam everything in the outside to prevent information leakage. In [16], access points also mutually jam their transmissions to cause decoding errors at eavesdroppers. All these applications outline the valuable gain of jamming for protecting wireless networks.

Counter attacks on jamming are as vast as jamming itself. Metrics such as Packet Send Ratio (PSR), Packet Delivery Ratio (PDR), carrier sensing time, and Received Signal Strength (RSS) can indicate if jamming occurs or not [21, 32], but they cannot provide evidence alone. Xu *et al.* [32], therefore, combine these metrics with

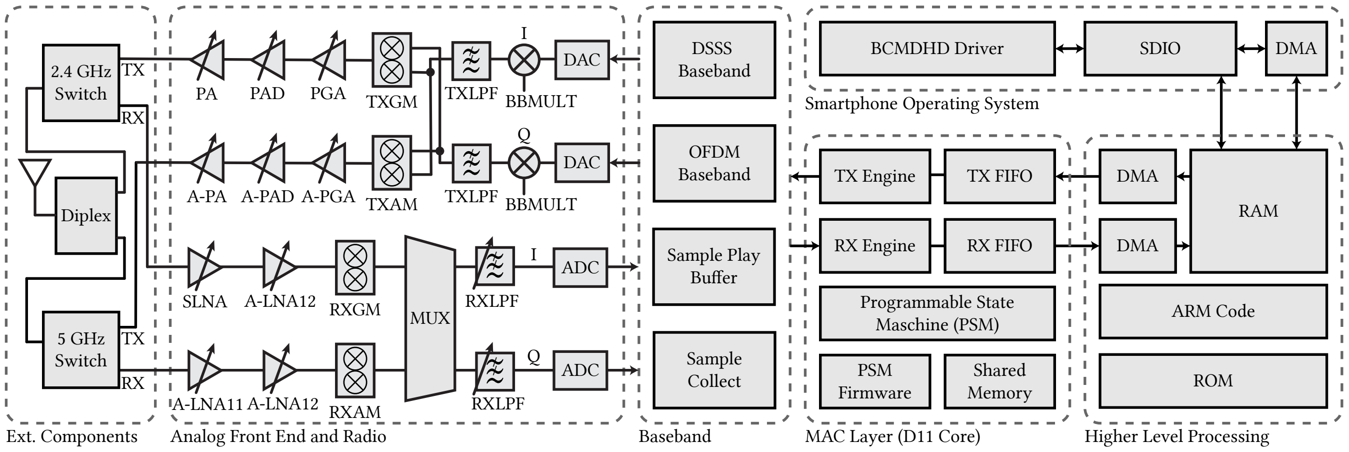


Figure 1: Architecture of a FullMAC Broadcom single-stream Wi-Fi chip, such as the BCM4339 of the Nexus 5 (based on [10]).

consistency checks on the location and signal strength to reduce miss-detection rates. A common method to overcome narrow-band jamming is the application of Direct Sequence Spread Spectrum (DSSS) and Frequency Hopping Spread Spectrum (FHSS) [20]. Lin and Noubir propose enhanced coding schemes to prevent jamming and establish jamming resilient communications [18]. In [33, 34], Yan *et al.* utilize techniques from interference cancellation and signal processing to maintain MIMO-OFDM communications under reactive jamming. They align the legitimate signal orthogonal to the jamming. Still, counter measures are costly, and launching and defending jamming attacks remains an arms-race [21].

3 BACKGROUND AND FIRST ANALYSIS

In this section, we first introduce the architecture of the BCM4339 Wi-Fi chip (see Section 3.1), which is required to understand the generation of jamming signals (see Section 3.2) and their amplification (see Section 3.3). Subsequently, we analyze the power consumption of the Wi-Fi chip in reactive jamming scenarios (see Section 3.4).

3.1 Wi-Fi Chip Architecture

For our experiments, we use the BCM4339 Wi-Fi chipset manufactured by Broadcom, which is embedded in Nexus 5 smartphones. Its architecture is illustrated in Figure 1. Similarly to all other models in the huge BCM43xx family, this chipset includes a programmable state machine (PSM) as part of the D11 core that drives all low-level and time-critical operations. As already demonstrated by previous works [4, 5], the possibility to change the PSM code makes this architecture research-friendly as it allows to completely reprogram the MAC level functionalities. During normal operations, the PSM fetches outgoing frames from the TX FIFO queue, sets up the radio and schedules the frame departure that will then be managed by the baseband circuitry whenever a transmission opportunity occurs (flow from right to left in Figure 1). The other way round, once the PSM detects that a frame is being received by the baseband decoder, it may finalize the reception and push the frame to the RX FIFO queue or drop it, for example, when it is addressed to another node. The higher level processing reported in the figure is usually implemented in the Wi-Fi driver of the operating system (SoftMAC approach), while in the Nexus 5 and similar smartphones, these operations are handled by a dedicated microcontroller (i.e., an ARM Cortex-R4) that offloads the smartphone's main CPU (FullMAC approach). To change the operation of the Wi-Fi chip, we had to patch both the ARM firmware and the D11 microcode. To this end, we used the Nexmon framework [24] that allows to write code patches in C and add a compressed version of the D11 microcode that is loaded during interface initialisation.

To implement the reactive jammer, we add a simple parser in the PSM that inspects the frame's header while the baseband is still receiving the frame data. If a jamming condition matches (e.g., the destination UDP port), we start a new transmission that interferes with the target frame at the receiver. While Berger *et al.* [4, 5] request the baseband to immediately transmit a 802.11 frame, we force the chip to transmit an arbitrary waveform that we previously stored in the sample play buffer. It contains up to 512 complex samples that are fed into the DACs for a configurable number of times. We can, hence, control the spectral shape of the jamming signal, its duration and power. With respect to [4, 5], our approach is more flexible as it allows a finer customization of the jamming parameters. For instance, we can focus the jamming energy on selected carriers like the pilots that are used at the receiver for signal equalising [8, 15, 25]. We recognise that all this is actually possible thanks to new chipset features, that were not available on the "old" 802.11g chipset where reactive Wi-Fi jamming was originally demonstrated [4, 5]. For the same reason, we are also reporting, for the first time, the performance of reactive jamming with different settings of the channel bandwidth (40 MHz and 80 MHz) introduced by the 802.11ac standard. In the next section, we discuss how we use the BCM4339 Wi-Fi chip to generate our jamming tones.

3.2 Signal Generation

As described above, the sample play buffer is limited to 512 samples that can be played back in a loop. To avoid discontinuities at the end of a buffer, we only generate tones on frequencies whose periods completely fit into the buffer. To this end, we implemented an inverse discrete Fourier transform (IDFT) on the ARM processor by summing up the outputs of a Coordinate Rotation Digital Computer (CORDIC) function. While an inverse fast Fourier transform (IFFT) is optimized for sizes that equal powers of two, the IDFT works with arbitrary numbers of samples to operate on. This allows us to generate cyclic buffer contents with arbitrary subcarrier spacing. For example, to generate tones with a spacing of 1 MHz on a channel with 80 MHz bandwidth that is sampled at 160 MSps, we use a 160 samples IDFT and can define an amplitude and a phase for each subcarrier. In the jamming scenario, the phase of the tones is of minor importance, but it helps to equally distribute the transmit power between the in-phase (I) and quadrature (Q) components of the complex samples, which reduces the peak-to-average-power ratio (PAPR[1]) of a signal. The latter is important, as energy efficient jamming requires to send tones with high average powers. Peaks in the baseband signal would require to reduce the signal's amplitudes to the dynamic range of the digital-to-analog converters (DACs), which automatically reduces the power in the transmitted signal.

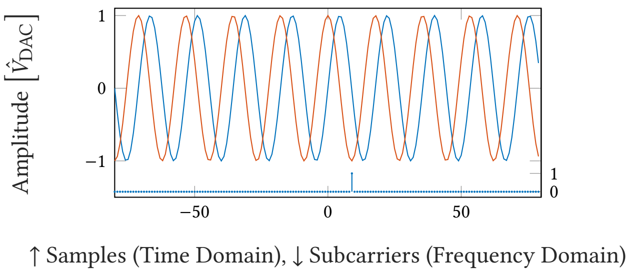
Figure 2 shows a 160-sample window of cyclically repeating time- and frequency-domain samples. An amplitude of one corresponds to the unknown peak voltage $\hat{V}_{\mathrm{DAC}}$ generated by the DACs. For Figure 2a we created a single tone by only setting one subcarrier leading to minimal PAPR of 0 dB and maximum power of 0 dBr (measured relatively to the output power of the DACs, i.e., before any amplification of the analog signals). Figure 2b shows the same for a signal with 80 active subcarriers. Due to the high PAPR of 19.03 dB the average power is only -19.03 dBr resulting in even less power per subcarrier. As a conclusion, we aim for jamming signals with few but high power subcarriers, similar to those used for jamming pilot tones.

3.3 Signal Amplification

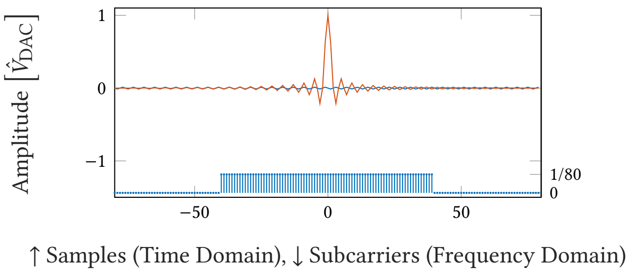
After signal generation, the DACs create an analog waveform that needs amplification. The first step is a multiplication at the baseband multiplier (BBMULT) to increase the analog amplitude before up-conversion in the mixers. Then a chain of three adjustable amplifiers is used to amplify the radio frequency signal: (1) power amplifier (PA), (2) power amplifier driver (PAD), and (3) programmable gain amplifier (PGA). To simplify gain selections, the firmware stores amplifier settings in a table addressable by an index. On the Nexus 5, the gain is mainly modified by setting the PGA value and slightly by setting BBMULT, while all other gains are set to maximum. Figure 3 shows gain settings for channel 7 (2.4 GHz band) and 106 (5 GHz band). The lower the index, the higher the gain.

To measure the output power for a given index and channel setting, we read the transmit signal strength indicator (TSSI) after sending a 4 MHz test tone with -9.03 dBr power measured at the DACs (TSSI only shows a tendency of the output power at the antenna port—to get values in dBm, a conversion is required). The result is illustrated in Figure 4. We observe, that the TSSI saturates for high gains (index lower 20), especially in the 5 GHz band. The difference in TSSI for 20 and 40 MHz bandwidth is small, while 80 MHz bandwidth requires significantly higher gains to achieve similar TSSIs as transmissions on lower bandwidths. For adjusting transmission powers, we simply switch between power indices.

[1]We calculate the PAPR in dB of a signal s according to $\mathrm{PAPR}_{\mathrm{dB}}(s) = 20\log_{10}\frac{\max(|s|)}{\mathrm{rms}(s)}$.



(a) Signal generated out of one subcarrier with normalized time-domain amplitudes leads to PAPR of 0 dB and average Power of 0 dBr.



(b) Signal generated out of 80 subcarriers with normalized time-domain amplitudes leads to PAPR of 19.03 dB and average Power of -19.03 dBr.

Figure 2: Peak-to-average-power ratio (PAPR) and average power analysis of single- and multi-tone signals. Each plot shows IQ time-domain samples at the top and the selected frequency-domain subcarriers at the bottom.

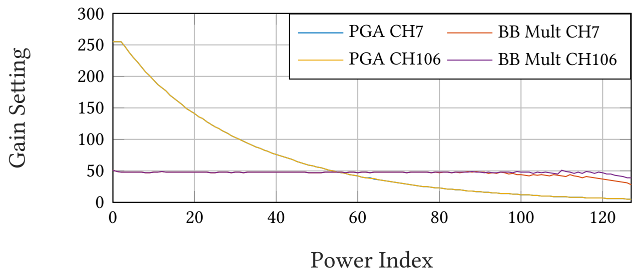


Figure 3: Programmable gain amplifier (PGA) and baseband multiplication (BBMULT) settings by index.

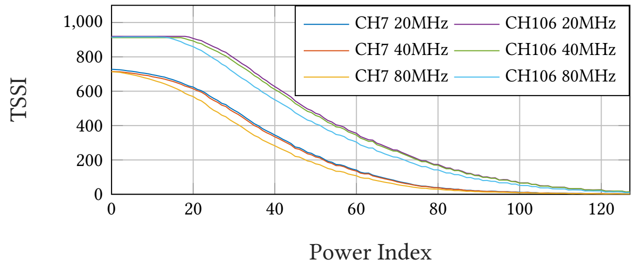


Figure 4: Transmit signal strength indicator (TSSI) measured in two bands, by measuring the power of a 4 MHz test tone. The power indices point to gain settings in a table.

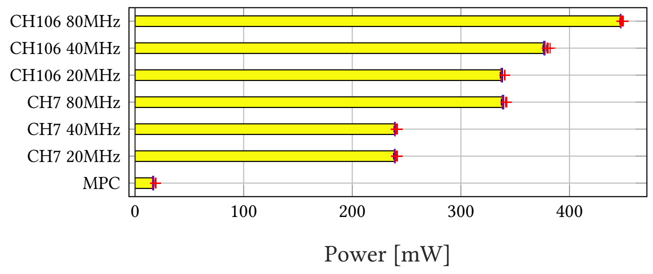


Figure 5: Power consumption for different channel specifications vs. power consumption with minimum power consumption (MPC) enabled.

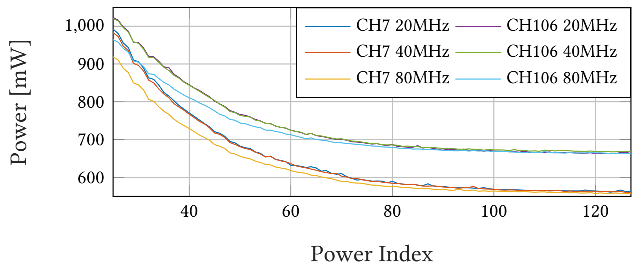


Figure 6: Power consumed by the smartphone for continuously transmitting the 4 MHz TSSI test tone at different power indices (minus the power consumption for operating the Wi-Fi chip with MPC disabled).

3.4 Power Consumption

Especially on mobile devices, power consumption is of importance and influences how long a jammer can operate on battery power. Hence, we attached a Monsoon Power Monitor to the battery ports of a Nexus 5 smartphone to measure the instantaneous power consumption with 5 kSps. Using this setup, we first measured the phones idle power consumption with turned off display and LTE modem, but active minimum power consumption (MPC) mode of the Wi-Fi chip. MPC allows the ARM controller in the Wi-Fi chip to handle interrupts, while the radio, physical layer and D11 cores are on standby—neither receiving nor transmitting. In this mode, the phone consumes 16 mW. Turning MPC off enables the lower layer cores in the Wi-Fi chip and activates the reception of Wi-Fi frames. As illustrated in Figure 5, this mode constantly consumes between 239 and 447 mW without transmitting anything. The power consumption increases with the channel bandwidth and in the 5 GHz Wi-Fi band. To collect the measurement results, we run experiments for 60 seconds, split the middle 40 seconds into 200 millisecond intervals over which we calculated the median after averaging to get an estimate of the power consumption. We use the median to reduce outliers due to non-deterministic tasks run on the CPU of the smartphone.

Afterwards, we measured the additional power required for continuously transmitting a 4 MHz TSSI test tone as illustrated in Figure 6. Even using the smallest gains (largest power index), requires at least 556 mW in the 2.4 GHz band and 662 mW in the 5 GHz band. The difference is due to operating the analog front-end at higher frequencies which increases power consumption. To save power when operating the jammer, we focus our work on developing reactive jammers that only transmit short signals when required to destroy a frame. The power for operating the receiver is nevertheless continuously consumed.

4 IMPLEMENTATION

In this section, we present the three jammers that we implemented. As continuous jammers are too destructive for friendly-jamming scenarios, we only built reactive jammers. The first type—called Reactive Jammer (see Section 4.1)—transmits pilot tones whenever a jamming condition matches. The second type, the Acknowledging Jammer (see Section 4.2), is a novel extension to reactive jammers and sends back acknowledgements to the transmitter of a jammed frame. This avoids blocking non-targeted streams queued for transmission, as well as fallback to more robust modulation and coding schemes (MCS)[2]. To reduce power consumption of our second jammer, we introduce the Adaptive Power Control Jammer (see Section 4.3) as third type. It adjusts its transmission power according to an observed jamming success indication.

4.1 Reactive Jammer

A sophisticated jamming approach is a reactive jammer that only jams if a communication is detected and a jamming condition matches. For this work, we implemented a jammer similar to the one presented by Berger *et al.* in [4, 5]. However, instead of using the SoftMAC Broadcom chips of wireless routers, we implemented our Reactive Jammer in the FullMAC Wi-Fi chip of the Nexus 5 smartphone. Besides an increased mobility, it also supports 802.11ac to analyze more frame types and allows the transmission of arbitrary waveforms read from IQ buffers.

To implement the reactive jammer, we extracted and analyzed the microcode of the programmable state machine running in the D11 core. Parts of the code are very similar to the annotated open firmware OpenFWWF [14] for older Broadcom chips (BCM4306 and BCM4318) which helped with the analysis. However, we still had to understand how the new 802.11ac physical layer registers work and their effect on the transmission of arbitrary waveforms. To generate the latter, we created an IDFT function in the ARM processor to fill the sample play buffer with a repeatable waveform of our choice. Starting the transmission is, however, done by the state machine of the D11 core. It handles every incoming frame, while it is still being received. As an exemplary jamming condition, we check all unencrypted incoming frames with UDP payload and match the UDP port to a value of our choice. If it matches, we trigger the tone transmission, which loops over the sample buffer for a predefined number of iterations. That lets us control the length of the jamming tone. The power of the transmission is set by deactivating hardware power control and directly writing to tables (memory) of the physical layer that store the gain values of the amplifiers (see Section 3.3). We evaluate our Reactive Jammer in Section 5.2.

[2]In this work, we use the abbreviation MCS also for legacy 802.11a/b/g rates. In the context of 802.11n/ac, we refer to the MCS index.

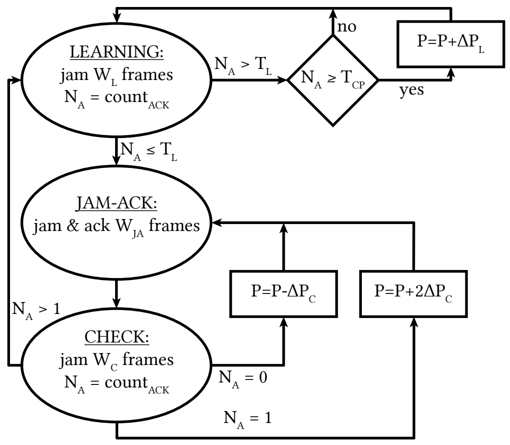


Figure 7: State machine of the Adaptive Power Control Jammer.

4.2 Acknowledging Jammer

During our pre-evaluation of Wi-Fi jamming scenarios, we realized that nodes sending multiple parallel data streams throttle all of their transmission, if only one stream gets jammed. This is due to the fact, that the MAC layer in the Wi-Fi chip does not differentiate between data streams, which could be UDP transmissions on different ports. In case the jamming goal is to block only selected network services, while other should continue to work, an extension to our Reactive Jammer is required. Nodes avoid retransmissions and fallbacks to more robust MCS settings, if they receive acknowledgements for their transmissions. Due to jamming the acknowledgement of the intended receiver is missing, hence, we decided to fake it with our Acknowledging Jammer. Thereto, we need to schedule a new acknowledgement in the transmit engine of the D11 core. Even though the jamming signal is transmitted over the frame that is currently being received, the state machine of the D11 core anyways reacts at the end of a reception to evaluate the correctness of an incoming frame. We use this event to schedule the transmission of the fake acknowledgement with the correct timing and present its evaluation in Section 5.5.

4.3 Adaptive Power Control Jammer

The jammers presented above transmit at a fixed power that we configure before the experiments. We usually choose a high power to ensure successful jamming results. In many scenarios, however, this value exceeds the actual required minimum power to destroy frames at the receivers. This choice is not only energy inefficient but may also disturb neighbouring communications. To address these two issues, we added an adaptive power control algorithm to the microcode of the D11 core. It implements the state machine illustrated in Figure 7. In the initial state "LEARNING", we jam every target frame in a window of W_L received frames and count how many acknowledgement frames N_A come from the receiver. If N_A exceeds the targeted minimum T_L, a new learning window starts. If N_A is also larger than T_{CP}, the transmission power is increased by ΔP_L. If N_A is below or equal to the targeted minimum T_L, we

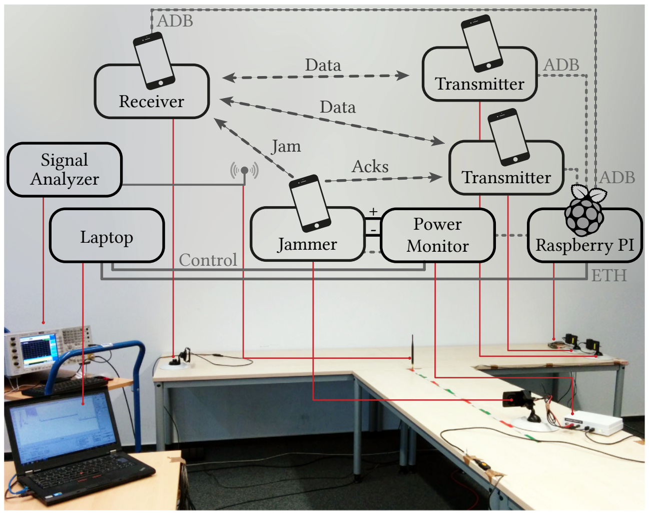


Figure 8: Experimental setup in our office building. The two transmitters, the receiver and the jammer are placed in the corners of an equilateral triangle.

enter the "JAM-ACK" state. In this state, we jam and acknowledge a window of W_{JA} frames. If the power determined before is sufficient, we keep jamming successfully while sending correctly forged acknowledgements to the transmitter. As a consequence, the transmitter keeps a high datarate and a very short contention window that avoids throttling additional data streams. At the end of each "JAM-ACK" window, we enter the "CHECK" state. It tests whether a new learning phase is required by jamming a window of W_C frames without acknowledging them. Instead, we again count how many acknowledgements N_A come back. In case, N_A is greater than one, a new learning phase is started. Otherwise, we reenter the "JAM-ACK" state and either reduce the jamming power by ΔP_C if no acknowledgements were received (jamming was perfect), or we increase the jamming power by $2 \cdot \Delta P_C$ if exactly one acknowledgement was received (meaning we are directly at the edge of the minimum required jamming power). This state machine is by far not optimal and we did not consider convergence properties of the algorithm. Instead, we only intent it to be a proof-of-concept for demonstrating that it works in a practical setup. We evaluate its performance in Section 5.6.

5 EXPERIMENTAL EVALUATION

In this section, we describe the experimental setup and the experiments performed for the three different jammers directly followed by experiment evaluations and discussions. We intentionally chose a simple setup to ease reproducibility and provide all required source codes[3] to repeat the experiments used in our evaluation.

[3]The source code to rebuild the reactive jamming firmware is available under https://nexmon.org/wisec2017_nexmon_jammer. To avoid easy abuse, the jammer only targets frames with destination MAC address "NEXMON" and source MAC address "JAMMER".

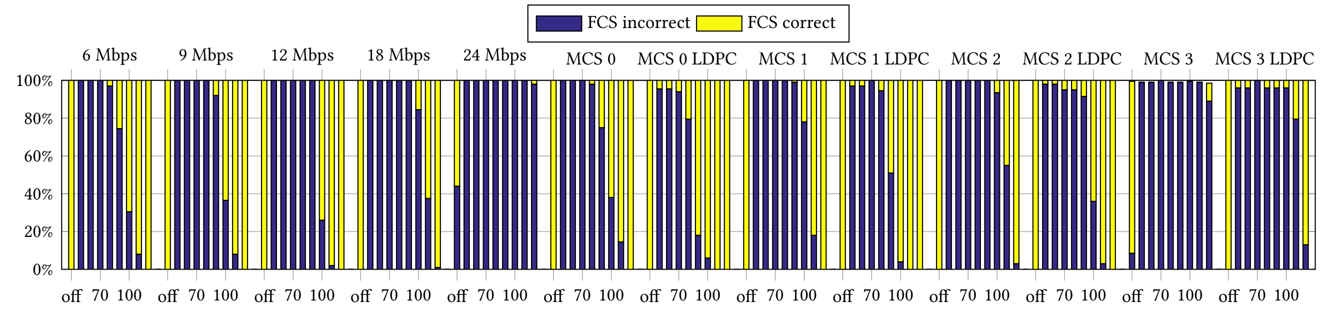


Figure 9: Jamming frames on channel 116 in the 5 GHz band with 20 MHz bandwidth and different rates. The bars indicate correct frame receptions, when performing experiments with no jamming (off) and jamming at power index 50 (high power) to 120 (low power). The jamming-tone length is fixed to 128 us and the frame length to 1540 bytes.

5.1 Experimental Setup

In Figure 8, we illustrate our experimental setup. It consists of one or two Nexus 5 smartphones used as frame transmitters and one Nexus 5 smartphone as frame receiver. The transmitters are used to inject Wi-Fi frames with UDP payload at a fixed rate directly from the Wi-Fi firmware to avoid any jitter and additional queuing introduced by the operating system. Nevertheless, the injection complies to the 802.11 medium access control algorithms (i.e., CSMA). To achieve this, we used the Nexmon framework [24] to create a set of patches for each of our experiments. The receiving node also uses Nexmon to activate monitor mode and capture the received frames using tcpdump. We make sure that frames with incorrect frame check sequence (FCS) are also captured to be able to evaluate how many frames were corrupted either by the jammer (if the headers up to the UDP header are still correct) or by other effects that also destroy the frame headers. In general, we receive all frames, as long as the PLCP headers are correctly received.

As jammer, we also used a Nexus 5 smartphone with modified firmware for our jamming experiments. To evaluate the power consumption during jamming, we attach the Monsoon Power Monitor[4] to the battery ports of the jamming Nexus 5 and use a Laptop running Windows to capture the energy traces. All smartphones are controlled using the Android debugging bridge (ADB) over USB. We use a Raspberry Pi 3 as the controlling node to coordinate the experiments. After the measurement, USB is passed through the Power Monitor to the Raspberry Pi. For all experiments, the two transmitters, the receiver and the jammer are mounted using car mount holders attached to plates and placed in the corners of an equilateral triangle with side length of 2.8 meters. For increased side lengths, the received signal powers of both the data frame and the jamming signal equally decrease with distance (assuming line of sight behaviour), hence, the jamming-to-signal ratios stay constant. Additionally, for massive jamming scenarios, we can assume high densities of smartphones usable for jamming, so that distances of only a few meters between nodes are likely.

Even though not required to perform the experiments, we used a spectrum analyzer with 160 MHz real-time bandwidth and abilities to capture in the 2.4 and 5 GHz Wi-Fi bands to debug during development and to verify that the jammer works correctly. In the next subsection, we describe the experiments and evaluation using our Reactive Jammer.

[4]Monsoon Power Monitor: http://msoon.github.io/powermonitor/

5.2 Evaluating our Reactive Jammer

Our main intention in the evaluation of our Reactive Jammer is to verify the reliability and effectiveness of jamming with respect to the percentage of jammed frames at the receiver. As the Nexus 5 does not have a port for external antennas (that is matched to 50 Ohms used in measurement equipment), we focus our experiments solely on over-the-air jamming. This allows a realistic evaluation of the jamming performance in an office room and also incorporates antenna effects.

For successful jamming, two requirements need to be fulfilled. First, the jammer needs to correctly receive and decode the frames up to the part included in the jamming condition. Secondly, the jamming signal needs to have sufficient power at the receiving node to interfere with the jammed frame. In a pre-evaluation of our jamming setup, we realized that direct communication between two Nexus 5 is prone to reception errors for high MCS settings, even in the small setup we used for our experiments.

To evaluate the jamming performance, we sent frames without retransmission at different MCS settings from the transmitting smartphone next to the wall to the receiver at the other end of the long table (see Figure 8). In the first round, the jammer was deactivated to evaluate how many frames can be correctly received. In the other rounds, we started jamming at power index 50 (high power) and increased it in steps of 10 to 120 (low power). As jamming signals, we used pilot tones of 20 MHz bandwidth transmissions at subcarriers $\{\pm 7, \pm 21\}$. The results are illustrated in Figure 9. Each bar represents one round and shows the number of correctly received frames (FCS correct) and damaged frames (FCS incorrect). Starting from the legacy rate of 24 Mbps, the number of incorrectly received frames was already more than 40 percent. Increasing the MCS to 36 Mbps or MCS 4 for high-throughput rates, resulted only in erroneous frame receptions.

Starting the jammer at power index 50 allows to corrupt all frames with deactivated low-density parity-check (LDPC) and even with the latter, close to 100 percent of the frames are jammed. Reducing the jamming power by setting higher power indices leads to the expected effect of a lower number of corrupted frames. 24 Mbps and MCS 3 transmissions are still very vulnerable to jamming even at these low transmission powers. Only LDPC can decrease the impact of jamming effects by providing better error-correction abilities. In the next section, we present how our jammer can be used in a practical friendly-jamming scenario.

5.3 Reactively Jamming Non-Compliant 802.11ac Transmission

One application of friendly jamming is to counter the attack of malicious nodes that might setup rogue access points or communicate in a destructive way in a network environment. For example, a misbehaving pair of Wi-Fi devices might use non-compliant 80 MHz wide channels in the 2.4 GHz band that interfere with other communications in this shared band. Forcing the rogue nodes to switch to compliant transmissions can be achieved by jamming their communications so that their information exchange stops. Such a jammer, however, also needs the ability to receive Wi-Fi transmissions on illegal channel setups.

To evaluate this scenario, we deactivated the validation of channel specifications in the Wi-Fi firmware to be able to use 80 MHz wide channels in the 2.4 GHz band. Setting the carrier frequency to channel 9 (2452 MHz), covers the band from 2412 to 2492 MHz and, thus, almost the whole 2.4 GHz band. Then we evaluated, whether our jammer can receive and jam 20in80, 40in80 and 80in80 MHz transmissions using VHT MCS 0. 20in80 MHz means transmitting at the lowest of the four 20 MHz channels covered by the 80 MHz channel. 40in80 MHz means using the lower 40 MHz sideband, while the transceiver is still tuned to Wi-Fi channel 9. As jamming signals we used pilot tones at subcarriers {-117, -103, -98, -75} for 20in80 MHz, {-117, -89, -75, -53, -39, -11} for 40in80 MHz and {±103, ±75, ±39, ±11} for 80in80 MHz transmissions. We adjusted BBMULT to achieve an equal power of all tones in all bands and set the tone transmission lengths to 128 us for 20in80 MHz, 64 us for 40in80 MHz and 32 us for 80in80 MHz transmissions.

We illustrate our results in Figure 10. Similar to Figure 9, the first bar indicates experiments without jamming. We observe, that transmissions are reliably received. During the experiments, the 2.4 GHz band in our office building was unused except for some beacon transmissions on channel 1. Starting the jammer at power index 50 (high power, second bar), all frames are corrupted at the receiver. Using lower transmission powers, the number of jamming successes reduce. Most vulnerable are 80 MHz transmissions given our chosen jamming pattern.

5.4 Multi-Node Jamming Analysis

In the previous experiments, we focused on single link communications with only one active link and deactivated retransmissions. In this section, we extend our scenario by using two nodes sending frames in parallel on different UDP ports (3939 and 4040). Their MCS settings were fixed to 24 Mbps (OFDM) with five retransmissions for non-acknowledged frames. Starting from the third retransmission, we used a fallback rate of 1 Mbps (DSSS). All experiments were performed on channel 13 in the 2.4 GHz band. Additionally, we saturated the transmit queues in the Wi-Fi chips to send as many frames as possible.

At the receiving node, we again captured frames using tcpdump in monitor mode including frames with bad FCS. In Figure 11 we illustrate each experiment with one bar that represents the overall achieved UDP payload bit rate (including retransmitted frames). The bars are split to represent the bit rate dedicated to frames on port 3939 or 4040 additionally distinguishing between correct and incorrect frame check sums (FCSs). The first two bars show that transmissions with only one active node reach 18.2 Mbps. If two nodes are active at the same time, the throughput is split evenly and sums up to 18.6 Mbps. By activating our reactive jammer, the throughput of the jammed node vanishes, while the throughput of the second node increases to 17.5 Mbps which is almost the rate a single transmitter achieves. In the next section, we change the setup to transmitting both streams with only one node.

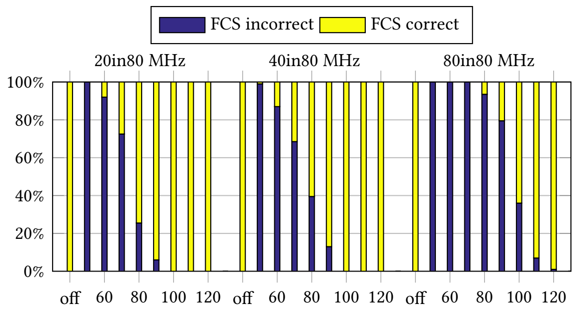


Figure 10: Jamming rogue 802.11ac transmissions on channel 9 in the 2.4 GHz band using 80 MHz bandwidth and sending 20, 40 and 80 MHz frames at VHT MCS 0. The frame length is fixed to 1540 bytes and the jamming-tone length to 128, 64 and 32 us for 20, 40 and 80 MHz bandwidth. The bars represent experiments with no jamming (off) and jamming at power index 50 (high power) to 120 (low power).

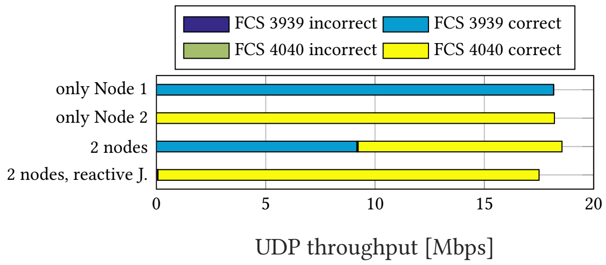


Figure 11: UDP throughputs in Mbps with either one or two active nodes using different UDP ports (Node 1: 3939, Node 2: 4040). Without jamming, the throughput splits evenly. Using the reactive jammer the throughput of Node 2 increases, while the throughput of Node 1 drops to a negligible rate of only incorrectly received frames. The jamming-tone length is fixed to 460.8 us to cover the complete remaining bytes of the 1540 bytes frame transmitted at 24 Mbps.

5.5 Stream-Selective Jamming

Instead of using reactive jammers to hinder a node from communicating completely, one may intent to specifically jam the communication of a certain service distinguishable by port numbers,

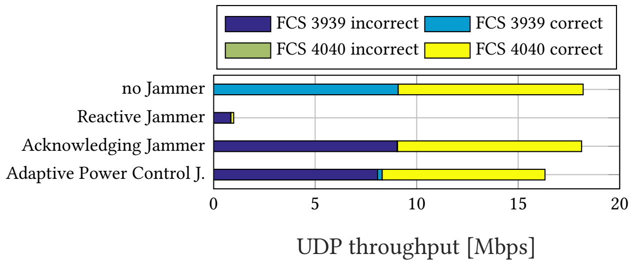


Figure 12: UDP throughputs with two streams sent by one node using UDP ports 3939 and 4040. Without jamming, the throughput splits evenly, using the Reactive Jammer kills the whole throughput, while the Acknowledging and Adaptive Power Control Jammers still allow communication. The jamming-tone length is fixed to 460.8 us to cover the complete remaining bytes of the 1540 bytes frame transmitted at 24 Mbps.

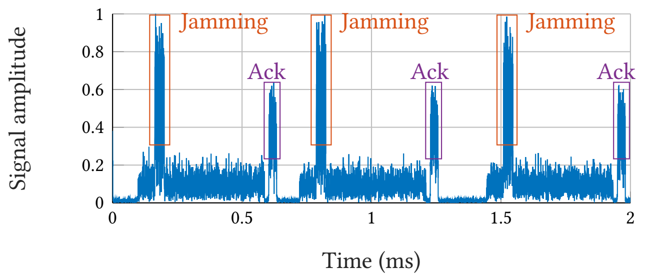


Figure 13: Operation of the Acknowledging Jammer. After jamming a frame, the jammer sends an acknowledgement to the transmitter indicating correct frame reception at the destination node. It takes roughly 20 us between receiving the UDP port number and sending the jamming signal.

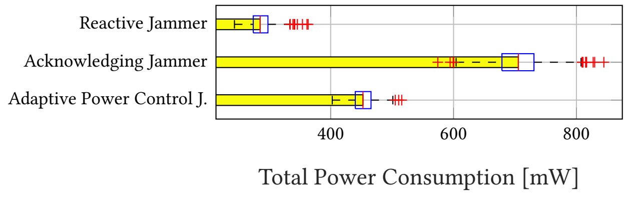


Figure 14: Total power consumption of the jammer, measured when jamming one transmitting node with two UDP streams (compare Figure 12).

while other services should still be available. As a usage scenario, we assume an industrial mesh network with legacy nodes that cannot be updated. The nodes offer multiple services of which one is vulnerable to remote code execution attacks. To protect the nodes while still being able to operate the non-vulnerable services, we employ our friendly reactive jammer. To evaluate this setup, we send two UDP streams on ports 3939 and 4040 from only one node. The results in Figure 12 show that the throughputs split evenly between the two streams, if no jammer is active. Using the reactive jammer, however, the whole throughput drops as the transmitter's MAC layer applies its backoff algorithm to all Wi-Fi frames—not differentiating between different upper layer streams.

To overcome this problem and allow communication on ports that are not jammed, we developed the Acknowledging Jammer. Its operation is illustrated in Figure 13. Whenever a frame is received and the jamming condition matches, our jammer reactively jams it and also transmits an acknowledgement to the frames transmitter. Assuming correct reception of the frame, the transmitter can continue transmitting frames. The results are illustrated in Figure 12. Using the acknowledging jammer, frames to the jammed port 3939 are corrupted, while frames to 4040 are still correctly received and the throughput between jammed and non-jammed frames splits evenly. In the next section, we continue with a power consumption analysis for the presented jamming approaches.

5.6 Power Consumption Analysis

To benefit from a smartphone's mobility, users rely on a low power consumption to maximize the time on battery power. Hence, we did a power consumption analysis in parallel to the experiments described in Section 5.5. We illustrate the results for operating the jammer in our three implemented modes in Figure 14. As expected, the reactive jammer has the lowest power consumption of only 285 mW (resp. 30.7 hours runtime[5]), while 238 mW are allotted to operating the receiver with turned off MPC (see Figure 5). The low power consumption can be explained by the low number of frames that had to be jammed due to the increasing backoff at the transmitter for non-acknowledged frames. The Acknowledging Jammer consumes 705 mW (resp. 12.4 hours runtime), which is 2.5 times more power than operating the Reactive Jammer. The reasons for the higher consumption are twofold. First, the jammer needs to transmit an acknowledgement in addition to jamming. Secondly, the jammer needs to jam more frames as frames are transmitted at the highest rate without backoff delays.

The best option to reduce the power consumption would be to reduce the transmission power. This, however, may lead to jamming misses, if the transmitter reduces its MCS settings to transmit more robust frames that are harder to jam with low jamming powers. To avoid this problem, we developed the Adaptive Power Control Jammer, described in Section 4.3. It adjusts its jamming power according to the measured jamming success rate. In Figure 12, we observe that the Adaptive Power Control Jammer also leads to equally split throughputs for the two UDP streams. The total throughput is reduced to 16.3 Mbps and 2.8 percent of the targeted frames are not jammed. The correct frame receptions are due to the power adjustments. Whenever we reduce the power as a result of the "CHECK" state, a couple of frames may not be jammed until we increase the power again in the next iteration of the "CHECK" state. Nevertheless, the power consumption in our example setting reduces to 453 mW (resp. 19.3 hours runtime), which is 64 percent of the power consumed by the Acknowledging Jammer. As there is still room for improving the power adaptation algorithm, a user may optimize the jammer for either high jamming accuracy or low power consumption.

[5]Runtime calculation is based on standard LG BL-T9 batteries for the Nexus 5 with a typical energy of 8.74 Wh, respectively a capacity of 2.3 Ah.

6 DISCUSSION

For this work, we implemented and evaluated three Wi-Fi jammers on Nexus 5 smartphones by writing patches for the firmware of the Wi-Fi chip. The Reactive Jammer, jams every incoming frame using a waveform covering every pilot of a Wi-Fi OFDM symbol, whenever a jamming condition matches. The Acknowledging Jammer is an extension of the Reactive Jammer, that transmits an acknowledgement back to the transmitter of the jammed frame. Therewith, it avoids retransmissions and allows the transmitter to continue sending frames without blocking the transmission of frames that do not match the jamming condition. To reduce the power consumption of the Acknowledging Jammer, we further enhanced it resulting in the Adaptive Power Control Jammer that aims at reducing its jamming power to a minimum required to destroy the targeted frames at the receiver.

Our evaluation in our test setup demonstrated the effectiveness of jamming in both the 2.4 and 5 GHz bands. We successfully jammed frames with a success rate of up to 100 percent, depending on the jamming power and the used MCS index. Both legacy 802.11a and high-throughput 802.11n transmissions were vulnerable to our jammer. 802.11n was only harder to jam than 802.11a, when LDPC was enabled as forward error correction scheme. Regarding 802.11ac transmissions, we considered a rogue communication on an 80 MHz channel in the 2.4 GHz band which is not standard compliant. We successfully demonstrated that our jammer can detect those transmissions and react on them to hinder the data exchange.

During our experiments, we also came across a number of limitations of using the Nexus 5 as a reactive jammer. First of all, this smartphone has difficulties to correctly receive frames at high MCS indices, if they are transmitted by another Nexus 5 smartphone. A correct reception is, however, required to check jamming conditions based on the frame's payload. Without proper reception, jamming conditions are limited to the information provided in the PLCP header that is always transmitted at a robust rate. Additionally, the Nexus 5 has only one Wi-Fi antenna limiting its reception abilities to single-stream transmissions. Nevertheless, the Nexus 5 is sufficient as a platform to evaluate our three jammers in the field and it is possible to port our code to other more advanced Broadcom Wi-Fi chips as they are all based on the same architecture.

While a simple reactive jammer is sufficient to achieve jamming goals that hinder all transmissions from a node, it is not the optimal choice if jamming goals require to only jam one application or service, but allow any other Wi-Fi communication of the same node. As the MAC layer does not differentiate which packet streams are jammed and which are allowed to pass, the whole throughput of a node is throttled by jamming one out of many streams. To overcome this limitation, we introduce our Acknowledging Jammer, which—to the best of our knowledge—was first proposed in this work. By sending acknowledgements back to the transmitter, we trick it into believing that all frames arrived. Hence, retransmissions are avoided and streams that should not be jammed can continue to flow at the same rate as if no jammer was used. Acknowledging frames is, however, only useful in the case when a node should still be able to communicate. For completely stopping a communicating node, simple reactive jamming is more efficient as the backoff at the transmitter hinders the jammed node from congesting the channel with frames that are getting jammed anyways. Additionally, the Acknowledging Jammer consumes more power than the Reactive Jammer.

To reduce the power consumption, we proposed the Adaptive Power Control Jammer that we implemented as a proof of concept based on a simple state machine in the firmware of the Wi-Fi chip. It is sufficient to demonstrate that adaptively adjusting the jamming power based on the received acknowledgements of a receiver is working to reduce the power consumption at the jammer, while still jamming a high percentage of the targeted frames. The optimization of this state machine is, however, out of the scope of this work.

Overall, our work proves that sophisticated Wi-Fi jammers can be implemented in off-the-shelf smartphones. This allows to implement and massively distribute reactive jammers, as almost every one of us carries such a device around in daily life. Besides the friendly jamming applications proposed in this work to either block non-compliant devices or protect industrial networks containing otherwise vulnerable devices, omni-present jammers could also be used to perform wide-spread malicious attacks on our wireless infrastructure. As firmwares are generally proprietary, users need to trust the firmware's developers that their hardware is not affected by malware that might transform one's phone into a remotely controlled jamming device. Only open firmwares and open specifications of radio devices would allow an end user to verify by himself that its firmware exhibits only benign behaviour. Knowing about the abilities of smartphone-based jammers at least allows to discuss possible solutions and countermeasures to avoid massive attacks. These become more likely since the discovery of remote-code-injection vulnerabilities in Wi-Fi firmwares [3] that allow to remotely transform regular phones into jammers. Especially on old phones, these holes might never be fixed by manufacturers and due to the lack of open-source firmwares also not by the community.

7 CONCLUSION

In this work, we proved that one can easily transform off-the-shelf smartphones into efficient, mobile jamming devices. We demonstrated that small modifications to the firmware running in the Wi-Fi chip allow us to quickly react on frame receptions to transmit jamming signals in time to destroy the frame during its transmission. We are, thereby, not limited to predefined waveforms, but can design our own signals as IQ samples that are directly injected into the baseband. This opens the possibility to use such cheap devices in place of more complex software-defined radio platforms. This flexibility clearly poses a serious threat. If a malicious attacker manages to inject a modified firmware into a large number of devices, he could launch tremendous, distributed attacks against networks in the 2.4 and 5 GHz bands. We, instead, used the flexibility for friendly jamming applications and presented innovative jamming techniques that involve the transmission of a forged, matching acknowledgement to cheat the transmitter into believing that no transmission was actually jammed. Together with a proof-of-concept prototype that automatically determines the optimal transmission power, we presented the first 802.11ac compliant and energy efficient personal jamming platform.

8 ACKNOWLEDGMENTS

This work has been funded by the German Research Foundation (DFG) in the Collaborative Research Center (SFB) 1053 "MAKI – Multi-Mechanism-Adaptation for the Future Internet", by LOEWE NICER, LOEWE CASED, and by BMBF/HMWK CRISP. The work of F. Gringoli was partly supported by the European Commission (EC) in the framework of the H2020-ICT-2014-1 project WiSHFUL (Grant agreement no. 645274).

REFERENCES

[1] Narendra Anand, Sung-Ju Lee, and Edward W. Knightly. 2012. Strobe: Actively securing wireless communications using zero-forcing beamforming. In *IEEE International Conference on Computer Communications (INFOCOM) 2012*. IEEE, 720–728.

[2] Emrah Bayraktaroglu, Christopher King, Xin Liu, Guevara Noubir, Rajmohan Rajaraman, and Bishal Thapa. On the Performance of IEEE 802.11 under Jamming. In *IEEE Conference on Computer Communications (INFOCOM) 2008*. IEEE, 1265–1273.

[3] Gal Beniamini. 2017. Over The Air: Exploiting Broadcom's Wi-Fi Stack (Part 1). (2017). https://googleprojectzero.blogspot.de/2017/04/over-air-exploiting-broadcoms-wi-fi_4.html

[4] Daniel S. Berger, Francesco Gringoli, Nicolò Facchi, and Ivan Martinovic. 2014. Gaining insight on friendly jamming in a real-world IEEE 802.11 network. In *ACM Conference on Security and Privacy in Wireless & Mobile Networks (WiSec) 2014*. Oxford, United Kingdom.

[5] Daniel S. Berger, Francesco Gringoli, Nicolò Facchi, Ivan Martinovic, and Jens B. Schmitt. 2016. Friendly Jamming on Access Points: Analysis and Real-World Measurements. *IEEE Transactions on Wireless Communications* 15, 9 (2016), 6189–6202.

[6] James Brown, Ibrahim Ethem Bagci, Alex King, and Utz Roedig. 2013. Defend your home!: jamming unsolicited messages in the smart home. In *ACM Workshop on Hot Topics on Wireless Network Security and Privacy (HotWiSec) 2013*. ACM, New York, New York, USA, 1–6.

[7] Yifeng Cai, Kunjie Xu, Yijun Mo, Bang Wang, and Mu Zhou. 2013. Improving WLAN throughput via reactive jamming in the presence of hidden terminals. In *IEEE Wireless Communications and Networking Conference (WCNC) 2013*. IEEE, 1085–1090.

[8] T Charles Clancy. 2011. Efficient OFDM denial: Pilot jamming and pilot nulling. In *IEEE International Conference on Communications (ICC) 2011*. IEEE, 1–5.

[9] Federal Communications Commission. 2017. Jammer Enforcement. (2017). https://www.fcc.gov/general/jammer-enforcement

[10] CYPRESS 16. *Single-Chip 5G WiFi IEEE 802.11ac MAC/Baseband/Radio with Integrated Bluetooth 4.1 and FM Receiver*. CYPRESS. Document No. 002-14784 Rev. *G.

[11] Bruce DeBruhl, Christian Kroer, Anupam Datta, Tuomas Sandholm, and Patrick Tague. 2014. Power napping with loud neighbors - optimal energy-constrained jamming and anti-jamming. *ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec) 2013* (2014), 117–128.

[12] Shyamnath Gollakota, Haitham Hassanieh, Benjamin Ransford, Dina Katabi, and Kevin Fu. 2011. They can hear your heartbeats: non-invasive security for implantable medical devices. In *ACM Conf. of the Special Interest Group on Data Communication (SIGCOMM) 2011*. Toronto, Canada.

[13] Shyamnath Gollakota and Dina Katabi. 2011. Physical layer wireless security made fast and channel independent. In *IEEE International Conference on Computer Communications (INFOCOM) 2011*. IEEE, 1125–1133.

[14] Francesco Gringoli and Lorenzo Nava. 2009. OpenFWWF: Open FirmWare for WiFi networks. (2009). http://netweb.ing.unibs.it/~openfwwf/

[15] Myeongsu Han, Takki Yu, Jihyung Kim, Kyungchul Kwak, and Sungeun Lee. 2008. OFDM channel estimation with jammed pilot detector under narrow-band jamming. *IEEE Transactions on Vehicular Technology* 57, 3 (2008), 1934–1939.

[16] Morten Lisborg Jorgensen, Boyan Radkov Yanakiev, Gunvor Elisabeth Kirkelund, Petar Popovski, Hiroyuki Yomo, and Torben Larsen. 2007. Shout to Secure: Physical-Layer Wireless Security with Known Interference. In *IEEE Global Telecommunications Conference (GLOBECOM) 2007*. IEEE, 33–38.

[17] Yu Seung Kim, Patrick Tague, Heejo Lee, and Hyogon Kim. 2012. Carving secure wi-fi zones with defensive jamming. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS) 2012*. ACM, New York, New York, USA, 53–54.

[18] Guolong Lin and Guevara Noubir. 2005. On link layer denial of service in data wireless LANs: Research Articles. *Wireless Communications & Mobile Computing* 5, 3 (May 2005), 273–284.

[19] Ivan Martinovic, Paul Pichota, and Jens B. Schmitt. 2009. Jamming for good: a fresh approach to authentic communication in WSNs. In *ACM Conference on Wireless Network Security (WiSec) 2009*. ACM, New York, USA, 161–168.

[20] Aristides Mpitziopoulos, Damianos Gavalas, Grammati Pantziou, and Charalampos Konstantopoulos. 2007. Defending Wireless Sensor Networks from Jamming Attacks. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) 2007*. IEEE, 1–5.

[21] Konstantinos Pelechrinis, Marios Iliofotou, and Srikanth V. Krishnamurthy. 2011. Denial of Service Attacks in Wireless Networks: The Case of Jammers. *IEEE Communications Surveys & Tutorials* 13, 2 (2011), 245–257.

[22] Alejandro Proano and Loukas Lazos. 2010. Selective Jamming Attacks in Wireless Networks. In *IEEE International Conference on Communications (ICC) 2010*. IEEE, 1–6.

[23] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. 2016. DEMO: Using NexMon, the C-based WiFi firmware modification framework. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec) 2016*. ACM, Darmstadt, Germany, 213–215.

[24] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. 2017. Nexmon: The C-based Firmware Patching Framework. (2017). https://nexmon.org

[25] Chowdhury Shahriar, Shabnam Sodagari, Robert McGwier, and T Charles Clancy. 2013. Performance impact of asynchronous off-tone jamming attacks against OFDM. In *IEEE International Conference on Communications (ICC) 2013*. IEEE, 2177–2182.

[26] Wenbo Shen, Peng Ning, Xiaofan He, and Huaiyu Dai. 2013. Ally Friendly Jamming: How to Jam Your Enemy and Maintain Your Own Wireless Connectivity at the Same Time. In *IEEE Symp. on Security and Privacy (S&P) 2013*. IEEE, 174–188.

[27] Mathy Vanhoef and Frank Piessens. 2014. Advanced Wi-Fi attacks using commodity hardware. In *Annual Computer Security Applications Conference (ACSAC) 2014*. ACM, New York, New York, USA, 256–265.

[28] Triet D. Vo-Huu, Guevara Noubir, and Tien D. Vo-Huu. 2016. Interleaving Jamming in Wi-Fi Networks. *ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec) 2016* (2016), 31–42.

[29] Matthias Wilhelm, Ivan Martinovic, Jens B. Schmitt, and Vincent Lenders. 2011. Short paper: reactive jamming in wireless networks: how realistic is the threat?. In *ACM Conference on Wireless Network Security (WiSec) 2011*. ACM Request Permissions, New York, New York, USA, 47.

[30] Matthias Wilhelm, Ivan Martinovic, Jens B. Schmitt, and Vincent Lenders. 2011. WiFire: a firewall for wireless networks.. In *ACM Conf. of the Special Interest Group on Data Communication (SIGCOMM) 2011*. ACM Press, New York, New York, USA, 456–457.

[31] Fengyuan Xu, Zhengrui Qin, Chiu C Tan, Baosheng Wang, and Qun Li. IMDGuard: Securing implantable medical devices with the external wearable guardian. In *IEEE Conference on Computer Communications (INFOCOM) 2011*. IEEE, 1862–1870.

[32] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. 2005. The feasibility of launching and detecting jamming attacks in wireless networks. In *ACM International Symposium on Mobile ad hoc Networking and Computing (MobiHoc) 2005*. ACM Request Permissions, New York, USA, 46–57.

[33] Qiben Yan, Huacheng Zeng, Tingting Jiang, Ming Li, Wenjing Lou, and Y T Hou. 2014. MIMO-based jamming resilient communication in wireless networks. In *IEEE International Conference on Computer Communications (INFOCOM) 2014*. IEEE, 2697–2706.

[34] Qiben Yan, Huacheng Zeng, Tingting Jiang, Ming Li, Wenjing Lou, and Y. Thomas Hou. 2016. Jamming resilient communication using MIMO interference cancellation. *IEEE Transactions on Information Forensics and Security* 11, 7 (July 2016).