Noise-SDR: Arbitrary Modulation of Electromagnetic Noise from Unprivileged Software and Its Impact on Emission Security

Giovanni Camurati, Aurélien Francillon
Software and System Security Group
EURECOM
Sophia-Antipolis, France
e-mail: name.surname@eurecom.fr

2022 IEEE Symposium on Security and Privacy (SP) | 978-1-6654-1316-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SP46214.2022.9833767

***Abstract*—Electronic devices generate electromagnetic noise, also known as EM leakage when the noise leaks information. Many recent research papers exploit the fact that software activity can exploit this leakage to generate radio signals. This process breaks the isolation between simple unprivileged code and the radio spectrum, letting an attacker generate physical radio signals without accessing any radio interface. Previous work has discovered many leakage sources and covert communication channels, which generally use simple modulation schemes. However, a fundamental research question has been left unexplored: to which point can attackers shape electromagnetic leakage into signals of their choice?**

The answer to this question has an important security impact that goes beyond specific attacks or platforms. Indeed, arbitrary signal modulation is a useful primitive. This would allow attackers to use advanced modulations and better exploit the channel (leakage) capacity, for example, to establish advanced communication channels, or to inject malicious signals into victim receivers. At a first analysis, arbitrary modulation seems impossible: software has limited control on the leakage and existing attacks are therefore constrained to on-off keying or frequency-shift keying.

In this paper, we demonstrate that shaping *arbitrary* signals out of electromagnetic noise is possible from unprivileged software. For this we leverage fully-digital radio techniques and call our method *Noise-SDR* because, similarly to a software-defined radio, it can transmit a generic signal synthesized in software. We demonstrate our approach with a practical implementation with DRAM accesses on *ARMv7-A*, *ARMv8-A*, *x86-64*, and *MIPS32*. We evaluate it on different types of devices, including smartphones, a laptop, a desktop, and a Linux-based IoT device. Although power, frequency and bandwidth are constrained by the properties of the leakage, we present several case studies, including transmission with advanced protocols, device tracking, and signal injection.

I. Introduction

Software often stimulates and modulates the electromagnetic emissions of the electronic device on which it is running. From a security point of view, this breaks the logic isolation between unprivileged software and the physical world. An attacker able to intentionally control this process can interact with the physical world without having any explicit access to peripherals or communication interfaces. This idea was first introduced in 1998 under the name of Soft-TEMPEST [1], [2], though the authors mention earlier records of this principle for playful applications. Soft-TEMPEST consists in displaying a specially-crafted pattern on the screen to modulate the video signal. The resulting leakage at the frequency of AM radio can be easily picked with a standard handheld device. This is useful to exfiltrate data from a compromised device, or to intentionally add noise to the leakage from the screen to prevent eavesdropping.

Over the years, a vast literature [3]–[28] in the field of Emission Security (EmSec) has studied how to exfiltrate data from air-gapped networks using software-controlled emissions. There are many different leakage sources (e.g., electromagnetic, magnetic, electrical, optical, vibrational, acoustic, thermal), but their modulation is challenging because:

> ***Challenge 1:*** *The carrier is a harmonic of a leakage whose properties (e.g., power, frequency, phase, stability) are generally not under control of unprivileged software.*[1]
>
> ***Challenge 2:*** *In general, software can only cross-modulate a binary pattern on top of the leakage (e.g., alternating memory accesses and inactivity to turn the emissions on and off).*
>
> ***Challenge 3:*** *Timer sources that control the software pattern, which modulate the leakage, are not comparable to those available to dedicated radio circuits.*

Under these conditions, only simple modulation schemes such as On-Off Keying (OOK) and Frequency Shift Keying (FSK) are easy to implement. They are used by the majority of covert channels, with simple custom protocols. A few exceptions explore more advanced modulations, including multiple OOK subcarriers using multiple threads [11], and LoRa-like Chirp Spread Spectrum (CSS) approximated with FSK [8].

However, the ability to go beyond simple modulation schemes (i.e., to achieve arbitrary modulation of electromagnetic noise) could have a significant impact on emission security. Indeed, shaping arbitrary radio signals from unprivileged software can have many applications, from establishing advanced radio links to injecting signals into other victim receivers. For this reason, in this paper, we formulate and answer the following two research questions:

> ***Question 1:*** *Is it theoretically and practically possible to generate arbitrary radio signals using the noise produced by unprivileged software?*
>
> ***Question 2:*** *What are the impact and applications of arbitrary modulation on emission security?*

[1]Using output peripherals (e.g., the speakers in [25]) is beyond our goals.

DOI 10.1109/SP46214.2022.00018

TABLE I
COMPARISON OF SOFTWARE-CONTROLLED ELECTROMAGNETIC AND MAGNETIC LEAKAGE

Name	Type	Physical layer modulation	Protocol	Applications
***Noise-SDR* (this paper)**	EM	Arbitrary (RF-PWM)	Arbitrary analog or digital protocols	Advanced software-defined radio transmissions
Soft-TEMPEST [1], [2]	EM	AM, FSK	Custom	Exfiltration (display to AM radio)
AirHopper [3], [4]	EM	FSK (A-FSK, DTMF)	Custom (raw or packet)	Exfiltration (computer screen to smartphone)
USBee [5]	EM	FSK (B-FSK)	Custom	Exfiltration (USB bus to SDR)
GSMem [6]	EM	OOK (B-ASK)	Custom	Exfiltration (computer to mobile phone)
BitJabber [7]	EM	OOK, FSK (M-FSK)	Custom	Exfiltration (computer to SDR)
EMLora [8]	EM	Approximated CSS	Custom Lora-like	Exfiltration (computer to SDR)
AIR-FI [9]	EM	OOK	Custom	Exfiltration (computer to SDR or WiFi cards that expose physical layer radio measurements)
MAGNETO [10]	M	OOK, FSK (B-FSK)	Custom	Exfiltration (computer to smartphone)
ODINI [11]	M	OOK (ASK, OOK-OFDM using multiple cores), FSK	Custom (including FEC)	Exfiltration (computer to magnetic bug)
Matyunin et al. [12]	M	OOK, FSK ('period based')	Custom	Exfiltration (laptop to smartphone)

a) Generating arbitrary radio modulation: To provide a positive answer to ***Question 1*** despite the aforementioned challenges, we take an approach to modulation that is radically different from previous work. Like previous work, we are limited to binary modulation of the leakage (e.g., 'on' and 'off' symbols) using simple patterns (e.g., 'intense memory accesses' vs. 'inactivity'). Unlike previous work, we do not map these symbols directly with the data to transmit (e.g., '1' $\rightarrow$ 'on' vs. '0' $\rightarrow$ 'off'). Instead, we add two layers of abstraction that bridge the gap with arbitrary modulation:

Abstraction 1: *Using a software-defined approach, the application data to transmit is mapped to the baseband signal of a given radio protocol.*

Abstraction 2: *Using a fully-digital radio approach, the generic multi-bit baseband signal is upconverted to intermediate frequency and approximated with a binary sequence (that can be easily modulated on the leakage in a conventional way).*

Thanks to ***Abstraction 1***, attackers can easily implement, or reuse existing, generic radio protocols, from the upper layers to the physical layer. Thanks to ***Abstraction 2***, attackers can control the amplitude, frequency, and phase of a band-pass signal at an intermediate frequency of their choice. Like in a superheterodyne transmitter, this signal is modulated on a leakage at radio frequency. We call this method *Noise-SDR*, as it uses a software-defined fully-digital approach to shape arbitrary radio signals out of noise from unprivileged software. The fully-digital method that is best suited for this context is Radio-Frequency Pulse-Width Modulation (RF-PWM) [29]–[33].

We demonstrate that *Noise-SDR* is a generic and practical technique with practical implementations on modern *ARM* smartphones, IoT devices, Laptop and Desktop computers. We start with controlling DRAM accesses as an electromagnetic leakage source, because they have been proven to be effective [7], [8]. However, any leakage source that can be controlled by software in a binary way can be used.

Although considerably more flexible than classic leakage-based transmitters, *Noise-SDR* has some hardware constraints. Indeed, the time resolution at which the leakage can be switched on and off affects the resolution, intermediate frequency and bandwidth of the radio signal (e.g, tens of kHz on *ARMv7-A* and a few MHz on *ARMv8-A*). In addition, like classic leakage-based transmitters, the frequency range is limited to the harmonics of the leakage (e.g., all the multiples of an 800 MHz DRAM clock). However, in modern devices the leakage frequencies are high and might overlap with other radio protocols [3], [6], [9]. It is natural that leakage-based transmitters have more limitations than dedicated radio hardware. However, by implementing the physical layer in software, *Noise-SDR* pushes the limits of what can be achieved with electromagnetic noise. The software implementation of the physical layer is what defines a software-defined radio as such, and all software-defined radio have limitations (e.g., frequency, bandwidth, transmission and/or reception) that depend on their cost and purpose.

b) Noise-SDR impact and security implications: Regarding ***Question 2***, *Noise-SDR* opens new opportunities to establish radio links with advanced properties and good performance. Indeed, advanced techniques that optimize transmission for different goals (e.g., distance, data rate) become readily available, making exfiltration more practical and effective. The mobile nature of smartphones leads to additional opportunities, for example, device tracking. Moreover, smartphones carry many radio transceivers, and sensors. Attackers could use *Noise-SDR* to jam, spoof, or otherwise affect one of these components. Table I compares *Noise-SDR* with existing (electro)magnetic channels.

c) Contributions: We make the following contributions:

- ***A generic approach for arbitrary noise modulation:*** We present *Noise-SDR*, a software-defined fully-digital approach to shape generic radio signals out of the noise produced by unprivileged software (though with some limitations on the available power, frequency, and bandwidth).
- ***A practical implementation:*** We demonstrate a practical implementation on several platforms and architectures (*ARMv7-A*, *ARMv8-A*, *x86-64*, and *MIPS32*), using DRAM accesses as building block to generate electromagnetic leakage.
- ***Evaluation and Security impact:*** We show the opportunities brought by *Noise-SDR*, with experiments on many devices:
  - ***Advanced transmissions:*** Performant transmissions with advanced techniques (e.g., symbol shaping, spread spectrum, multi-carriers, interleaving) and state-of-the-art protocols (e.g., AM, FM, SSTV, CW, RTTY, MFSK, PSK, multi-carrier PSK, THOR, HamDRM, LoRa,

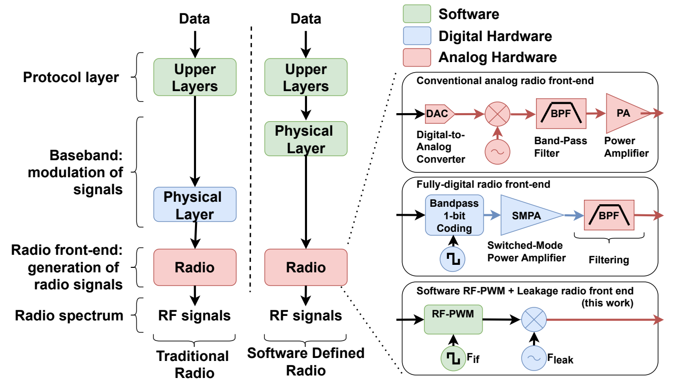


Fig. 1. Overview of different radio architectures. A conventional radio implements the physical layer in hardware, whereas an SDR implements it in software. A conventional radio front-end is made of analog/RF components, whereas a fully-digital radio uses digital hardware thanks to one-bit coding. Finally *Noise-SDR* (this paper) implements both the physical layer and one-bit coding in software, and uses the electromagnetic leakage of the underlying hardware to generate physical radio signals.

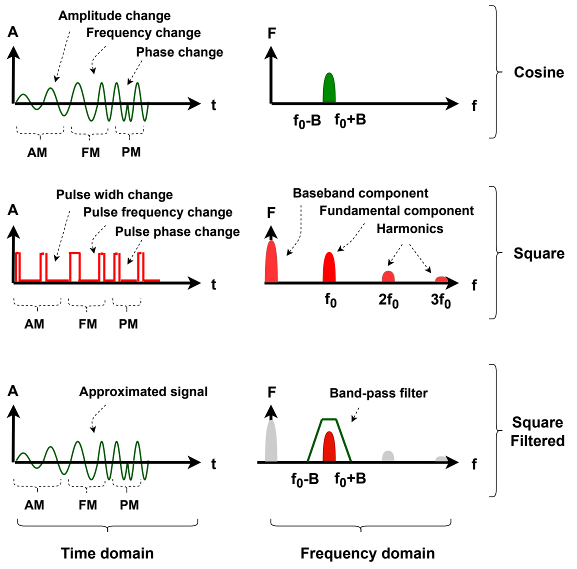


Fig. 2. Sinusoidal band-pass signal (top) vs. RF-PWM square wave signal (middle) vs. Filtered RF-PWM signal (down). For the sake of clarity, we show simple AM, FM, and PM in sequence, one after the other. Figure 3 shows a more complex example.

GLONASS C/A), each optimized for different goals.

– ***Applications beyond exfiltration:*** Case studies for possible applications, including exfiltration with different protocols, device tracking using FT4 below the noise floor (up to 5 m) with a Samsung Galaxy S5 Mini, and remote control of a Tytera UD MV 380 UHF receiver.

II. Background

A. Software-Defined Radios

A radio transmits information using electromagnetic signals. The physical layer of a protocol defines how these signals are processed to transmit and receive data. In the past, a given device implemented a fixed physical layer using dedicated hardware. Nowadays, radio hardware is often more flexible thanks to its integration with software. In particular, the Software Defined Radio (SDR) Forum[2] defines an SDR as:

> ***Software Defined Radio (SDR):*** *"Radio in which some or all of the physical layer functions are software defined"* [34].

This means that all or part of the physical layer is implemented in software. For example, baseband signals are generated in software starting from the data to send. SDRs should not be confused with Software Controlled Radios, where software simply controls the parameters of operation of the physical layer. Figure 1 compares an SDR with a traditional radio. Note that the choice of modulation has a fundamental impact on the properties of the channel (e.g., speed, distance, bandwidth, spectral efficiency, resilience to different types of noise).

The advent of SDRs has had a huge impact on security. Indeed, attackers with an SDR gain a flexible and relatively inexpensive access to the radio spectrum both in reception and transmission. SDRs are particularly useful to generate rogue signals to inject in other receivers [35], [36], or to flexibly craft packets of any protocol, including proprietary ones [37].

[2]Now *Wireless Innovation Forum*, it is an group of industrial, academic, and governmental actors interested in advancing wireless technologies including Software Defined Radios (SDRs) https://www.wirelessinnovation.org.

B. Fully-Digital Radios

A conventional radio transmitter is made of several analog radio-frequency components that modulate the baseband signal on a radio carrier. Unfortunately, this type of components is hard to integrate with other digital parts of the system and might not be energy efficient. For this reason, many research efforts have been spent with the goal of reducing their number to the minimum, leading to fully-digital radios:

> ***Fully-Digital Radios:*** *Radios implemented with digital components, mostly without analog blocks.*

There are many strategies to implement such radios. In general, they are based on one-bit coding to generate radio signals:

> ***One-Bit Coding:*** *A set of techniques to approximate a generic multi-bit signal with a (1-bit) square wave.*

Figure 1 compares the working principle of conventional and fully-digital radios. The conventional radio first converts the baseband signal to the analog physical domain, then it mixes it with the radio-frequency carrier, and finally amplifies it with a linear power amplifier [38]. In contrast, the fully-digital radio first generates a modulated square wave carrier using band-pass one-bit coding, then amplifies the resulting binary signal with a switching amplifier [30]. This can be made more energy efficient than the linear counterpart. The final filtering stage cancels the noise outside the band of interest.

One-bit coding is possible because the error produced by the approximation is kept outside of the frequency band of interest, where it can be easily filtered out. In this paper we focus on RF-PWM, that we describe in the following.

C. Radio-Frequency Pulse-Width Modulation (RF-PWM)

Figure 2 explains how a RF-PWM [29]–[33] square wave can approximate a generic band-pass signal used in radio communications. The generic band-pass signal, shown in the top plot, is

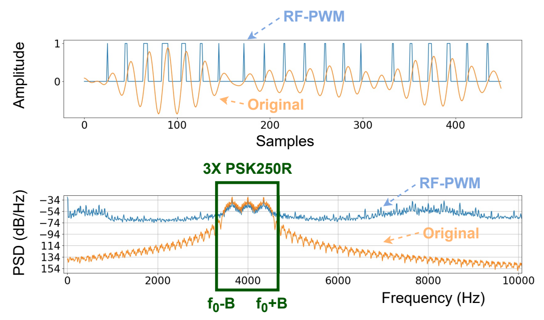


Fig. 3. Example of RF-PWM in the time (top) and frequency (bottom) domain. In the band of interest, the Power Spectral Density (PSD) of RF-PWM signal matches the original sinusoidal signal. The modulation is 3X PSK250R (three PSK subcarriers).

a sinusoidal wave at frequency f_0, with instantaneous changes in amplitude $a(t)$ and phase $\theta(t)$. In the frequency domain, it occupies a bandwidth $2B$ from f_0-B to f_0+B. It can be written as:

$$x(t)=a(t)\cos(2\pi f_0 t+\theta(t)) \tag{1}$$

The RF-PWM signal, shown in the middle plot, is a generic square wave with amplitude 1 or 0, whose pulses show instantaneous changes in width, frequency, and phase. The square wave has fundamental frequency f_0, phase $\theta(t)$, and duty-cycle $\delta(t)$ (ratio between pulse width and pulse period) set to $\frac{asin(a(t))}{\pi}$. Such square wave can be decomposed into an infinite sum of components as follows:

$$\begin{aligned} baseband &= \delta(t) \\ fundamental &= \frac{2}{\pi} a(t) cos(2\pi f_0 t+\theta(t)) \\ harmonics &= \sum_{k=2}^{k=+\infty} \frac{2}{k\pi} sin(k\pi\delta(t)) cos(2\pi k f_0 t+k\theta(t)) \end{aligned} \tag{2}$$

It is clear that the fundamental component of the RF-PWM square wave has the same form as the generic band-pass signal that we want to approximate. If the fundamental frequency is sufficiently larger than the bandwidth, the baseband component and the harmonics are well separated from the fundamental and can be ignored. Indeed, a simple band-pass filter can select the fundamental component, producing the desired sinusoidal band-pass signal, shown in the lower plot. Figure 3 shows a practical example using a pass-band signal with three orthogonal subcarriers, each modulated in phase (3X PSK250R), clearly visible around the fundamental frequency at 4 kHz. In the frequency band of interest the RF-PWM is clearly a good approximation of the original sinusoidal wave, while the baseband and harmonics of the square wave can be disregarded as out-of-band noise. In summary, we can define RF-PWM as follows:

> ***RF-PWM:*** *A band-pass one-bit coding technique that represents a generic band-pass signal as the fundamental component of a square wave, ignoring the other components as out-of-band noise.*

The RF-PWM technique should not be confused with (baseband) Pulse Width Modulation (PWM). The PWM method consists in controlling the amplitude of the baseband component by changing the duty cycle, ignoring the fundamental and the harmonics.

RF-PWM advantage: In RF-PWM the frequency of the square wave is the same as the frequency of the desired signal (i.e., $f_{rfpwm}=f_0$). This is easier to implement than the high-frequency square wave required by other methods such as PWM and Delta-Sigma ($\Delta\Sigma$) (e.g., $f_{pwm} \gg f_0$). For example, RF-PWM requires a lower sampling rate.

Importance of time resolution: In any one-bit coding scheme, the higher the time resolution at which the square wave is defined, the higher the accuracy at which the amplitude, phase, and frequency of the desired signal are represented. More details and a numerical example for RF-PWM are given in Appendix B.

D. *Software-Controlled (Electromagnetic) Emissions*

As we have seen in Section I, an extensive literature in Emission Security (e.g., [3]–[28]) has discovered a wide range of emissions that can be controlled from unprivileged software. We could generalize them as the following attack primitive:

> ***Software-Controlled Leakage:*** *An unintended physical leakage from an electronic device, with two properties:*
> - ***Carrier(s):*** *The physical leakage is made of one or more carriers (e.g., the harmonics of a high-speed clock) that propagate from the device and can be received by an attacker.*
> - ***Software Modulation:*** *Unprivileged software can, through a certain physical effect, modulate the carriers (generally in a very simple way).*

Electromagnetic carriers: Our approach is general to any type of emission, but we focus on electromagnetic leakages. In general, high-speed clocks (e.g., CPU clock, DRAM clock) and data lines (e.g., HDMI, USB, Ethernet) are strong sources of leakage. In general, they are digital signals that present several sinusoidal harmonics at the multiples of the fundamental frequency. An in-depth modeling of digital signals, their electromagnetic emissions, and their interference with other radio systems is given in [39]. To reduce interference, some systems (e.g., DRAM used in desktop computers) use Spread Spectrum Clock (SSC). In this case, the clock frequency follows a chirp, reducing the peak of the emissions as energy is spread over a larger frequency range. SSC emissions can be still effectively used as carriers using two possible methods: (i) despreading the clock at reception [7], (ii) observing that a SSC clock can be modeled as a finite set of frequency components each acting as a normal subcarrier [8]. Besides unintended emissions, other intentional radio signals might act as carriers [40]–[43].

Minimal unprivileged modulation: In general, unprivileged code has a very minimal control on the underlying leakage. In the simplest case, the software can alternate one operation that triggers strong emissions with a period of inactivity, leading to a binary modulation of the amplitude of the carrier. A typical example are intense accesses to DRAM on *x86-64/AMD* machines [6]–[8], [44]. While this method is likely portable to other architectures, we are not aware of any covert channel designed for devices such as smartphones with the *ARM* architecture. Similarly, an HDMI video signal can be modulated with a pattern of pixels of two different colors [1], [45]. An extensive analysis of how different instructions modulate leakage carriers was conducted in several studies (e.g, [46]–[50]). These extensive

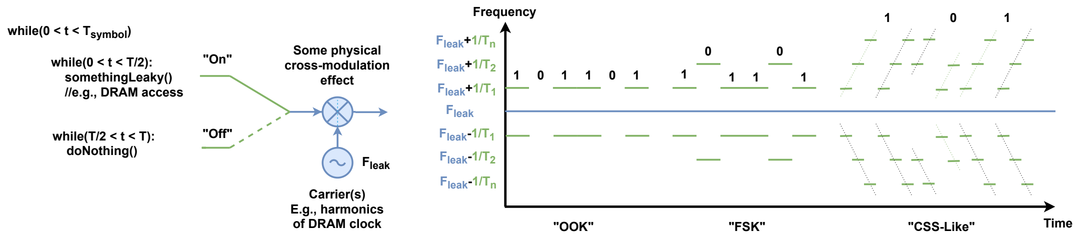


Fig. 4. Generalizing the results of previous work, we model software as a simple OOK modulator that cross-modulates a square wave on an existing leakage. Controlling the timings of the square wave leads to OOK, FSK, and CSS-Like modulation.

experimental analyses show how software-modulated leakages propagate at a very large distance from a large and diverse set of devices.

A simple model: Figure 4 shows a simple generalization of how previous work achieves simple modulation of a leakage carrier. By alternating a leaky operation and inactivity with period T, software cross-modulates a square wave on the leakage F_l, producing a component at $F_l \pm 1/T$ (and harmonics). With this primitive, software can use: (i) OOK modulation (e.g., [6]) by turning the component on or off for the duration of a symbol T_{symbol}, (ii) FSK (e.g., [10]) by using two different periods T_1 and T_2 to encode ones and zeros, (iii) CSS-Like (e.g., [8]) by approximating a chirp with multiple values of T and encoding ones and zeros in the phase of the chirp. Overall, we can assume that an attacker is able to cross-modulate a square wave on top of an existing leakage. In other words, the attacker has access to a binary amplitude modulator.

III. The *Noise-SDR* Approach

In this paper, we address the problem of crafting arbitrary signals from the noise produced by unprivileged software (**Question 1**). We propose *Noise-SDR*, a software-defined fully-digital approach that achieves arbitrary modulation, solving the challenges presented in Section I. *Noise-SDR* is shown in Figure 5, which should be compared with the generalization of previous work shown in Figure 4. Both previous work and *Noise-SDR* have a Radio Frequency (RF) stage that uses a special software pattern (e.g., intense memory accesses) to cross-modulate a square wave on an electromagnetic leakage. To achieve arbitrary modulation, *Noise-SDR* adds two levels of abstraction (Stages 1-2) to conventional leakage-based transmitters (Stage 3), with the following architecture:

- **(Stage 1) Software-Defined Baseband (BB):** The generic baseband signal of a given protocol is generated in software. Many existing libraries can be leveraged for this scope.
- **(Stage 2) Software Fully-Digital Intermediate Frequency (IF):** The generic baseband signal is embedded in the fundamental component of a square wave using RF-PWM, a pass-band one-bit coding technique used by fully-digital radios. Also this step is entirely performed in software.
- **(Stage 3) Electromagnetic RF Leakage:** The RF-PWM square wave is modulated on an electromagnetic leakage (e.g., harmonics of the DRAM clock) using a special software pattern (e.g., intense memory accesses followed by inactivity).

This architecture solves all the main challenges presented in Section I:

- **Solution to challenge 1:** In general, software does not control the frequency and phase of the underlying leakage source. For example, it does not control the clock of the DRAM. To solve this problem we rely on the IF stage. The choice of the intermediate frequency F_{IF} gives us some freedom in the choice of the carrier $f_c = F_{leak} + F_{IF}$. In addition, we gain control on the phase $\theta(t)$ of the output signal, which lets us implement frequency or phase modulation.
- **Solution to challenge 2:** In general, software is only able to cross-modulate a square wave on the leakage (e.g., by alternating intense activity with inactivity). Instead, we want to generate an arbitrary sinusoidal band-pass signal modulated in amplitude, frequency, and phase. To solve this problem, we leverage band-pass one-bit coding. In particular, we use RF-PWM, which represents such signal as the fundamental component of a square wave, ignoring the harmonics as out-of-band noise.
- **Solution to challenge 3:** In general, timer sources available to software have lower accuracy and resolution than those available to dedicated radio hardware. In addition, one-bit coding techniques generally require oversampling and a good time resolution. To address this problem, we chose to use RF-PWM. Compared to other techniques (e.g., PWM, $\Delta\Sigma$) it works with a square wave at the frequency of the signal that we want to generate. This lowers the requirements in terms of accuracy and resolution. For example, it has weaker requirements on the sampling rate.

With this architecture, *Noise-SDR* achieves arbitrary modulation. Figure 5 shows a few non-exhaustive examples. They include analog and digital protocols, using single or multiple subcarriers, amplitude frequency or phase modulation, or even spread spectrum (based on chirps or direct spreading).

Threat model: *Noise-SDR* is a general approach not tied to a specific threat model, but it has some minimum requirements:

- **Leakage with minimal control:** The target device should have a leakage source that software can cross-modulate with a square wave (i.e., apply binary changes to the amplitude like in OOK). As we have seen in Section I and Section II this is a reasonable assumption as there are many well known leakage sources of this type.
- **Timer source:** Software should have access to a relatively accurate time source. This is often a reasonable assumption (e.g., in native code on Linux, Windows, and Android). When this is not the case, an extensive literature in the field of micro-architectural attacks has shown how to implement

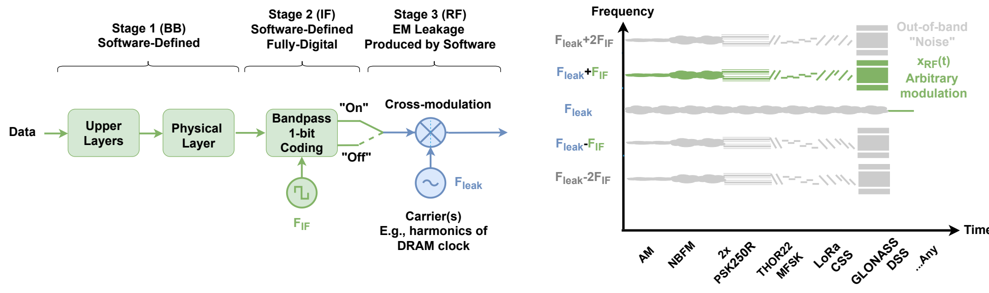


Fig. 5. Overview of *Noise-SDR* (compare with previous work in Figure 4). With its software-defined fully-digital approach, *Noise-SDR* achieves arbitrary modulation (control of the frequency F_{IF}, amplitude $a(t)$, and phase $\theta(t)$, of an arbitrary signal $a(t)\cos(2\pi(F_{leak}+F_{IF})t+\theta(t))$ at carrier frequency $F_{leak}+F_{IF}$). The spectrogram depicts a few (non-exhaustive) examples. See the evaluation for actual measurements (e.g., Figure 8, Figure 13).

accurate timers in other ways (e.g., [51]–[55]). For example, the simplest method is to use a spinning counter.

- **Software execution:** *Noise-SDR* requires to execute code on the target, but without any privilege, permission, or access to any peripheral. Such code is normally assumed unable to transmit radio signals, but with *Noise-SDR* it can establish advanced radio communications. Previous work on exfiltration with unprivileged code has similar requirements [6]–[8]. In Section IX we will mention a possible relaxation of those requirements.

IV. Implementation

A. Fldigi-Noise-SDR

Architecture: Fldigi [56] is a popular SDR tool that supports a wide range of amateur radio protocols, each optimized for different applications (e.g., distance, speed, error correction, robustness to a certain type of noise). In the Android version [57], a Java GUI application wraps the native code. We integrated Fldigi with our code for the RF-PWM and leakage stages, resulting in a standalone tool that can run on *ARMv7-A*, *ARMv8-A*, *x86-64*, and *MIPS32*. The tool is based on intense memory accesses to DRAM that modulate the emissions of the DRAM clock, and it does not require privileges on Linux, Windows, and Android. In our tool, Fldigi is used as a library of modems to modulate the input data using the desired protocol. Using this modular architecture, adding modulations and protocols is straightforward (e.g., custom or ported from GNURadio).

Supported protocols: Fldigi-*Noise-SDR* supports the protocols available with Fldigi for Android (e.g., CW [58], RTTY [59], MFSK [60], PSK [61], THOR [62], Olivia [63]). A comparison of their characteristics and performance is given in [64]. We also added a simple LoRa-like CSS protocol similar to [8] and a real LoRa implementation based on [37]. The supported modes use several different modulation schemes, including On-Off Keying (OOK), Binary Frequency Shift Keying (BFSK), M-ary Frequency Shift Keying (MFSK), Offset Incremental Frequency Shift Keying (IFK+), Phase Shift Keying (PSK), Orthogonal Frequency Division Multiplexing (OFDM), and Chirp Spread Spectrum (CSS). Many use advanced methods to work in challenging conditions, including Forward Error Correction (FEC) and interleaving. Being implemented in software, the physical layer has many configuration options (e.g., bandwidth, bit rate, coding rate, symbol shaping). We will provide more details in Section V.

Interface: For simplicity, we interface with the tool with a command-line interface. For example, the command

```
> ./fldigi-noise
    -sdr -i secret.txt -m MODE_3X_PSK250R -c 4000
```

modulates the content of the text file using three orthogonal PSK subcarriers, generates the corresponding RF-PWM square wave at an intermediate frequency of 4 kHz, and finally turns it into a physical radio signal using a leakage. The same functions called by the CLI tool could be easily exposed to other code, including a Java application on Android.

RF-PWM: The algorithm to generate an RF-PWM square wave is illustrated in Figure 6. The first step consists in generating a modulated IF sinusoidal carrier in a conventional way, in this case using Fldigi modems (see Appendix A for a simple explanation). The second step consists in identifying all the periods of the sinusoidal wave by looking at the zero crossings. They will correspond to the periods of the RF-PWM square wave. The amplitude of each period can be identified as well (e.g., as the maximum over the period). The last step consists in computing the pulse width for each period, by simply applying a pre-distortion to the value of the amplitude (see Equation 2). The resulting $T_{high,i}$ and T_i that characterize the square wave are streamed to the next stage (Stage 3 that controls the leakage). Being implemented in software, this algorithm operates at discrete time with a sampling frequency of F_s. In our context a sampling rate of $F_s = 80$kHz is adapted to most protocols and scenarios, but it can be easily changed. The listing of our C++ implementation (Listing 1) is available in Appendix B.

Leakage: Our tool modulates the emissions of DRAM clock (and its harmonics) using memory accesses. To generate the RF-PWM square wave, we repeat intense memory accesses during a pulse $T_{high,i}$, followed by inactivity till the end of the period T_i (time is measured at the ns resolution with *clock_gettime*). On x86-64/AMD, accessing DRAM is a proven method to generate strong leakage [6], [7], [44] even in presence of SSC [8]. We extend this approach to *ARMv7-A* and *ARMv8-A* smartphones using similar yet different techniques. Like previous work on *x86-64* [7], [65], we take inspiration from Rowhammer attacks [66], [67] that address the same problem of bypassing the cache for direct access to DRAM. We also support *MIPS32* used on a WiFi-enabled Linux Module. In all cases the code does not require any privilege to run.

- ***x86-64*:** On *x86-64*, DRAM can be written directly using

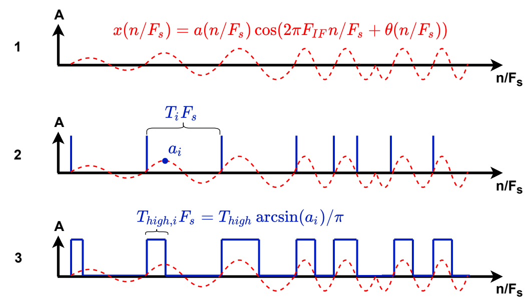

A
1
$x(n/F_s) = a(n/F_s)\cos(2\pi F_{IF} n/F_s + \theta(n/F_s))$
n/Fs
A
2
$T_i F_s$
a_i
n/Fs
A
3
$T_{high,i} F_s = T_{high} \arcsin(a_i)/\pi$
n/Fs

Fig. 6. Visualization of the algorithm to generate the RF-PWM signal. Sinusoidal wave modulated in amplitude/frequency/phase (top). Duration and amplitude of each period identified at the zero-crossings (middle). Corresponding RF-PWM square wave (bottom), whose fundamental component matches the original sinusoidal wave.

stream instructions. Like [8], [44], we use *_mm_stream_si128* to trigger a leakage during $T_{high,i}$.

- ***ARMv7-A*:** On *ARMv7-A*, unprivileged direct access to DRAM is possible in Android with the ION allocator [66], or from the GPU [55]. We empirically observed that simply allocating and freeing a small chunk of memory with ION produces a strong leakage. We use this to generate a leakage during $T_{high,i}$.
- ***ARMv8-A*:** On *ARMv8-A*, unprivileged direct access to DRAM is possible with non-temporal and cache maintenance instructions [67]. In particular, the *DC CIVAC* instruction cleans and invalidates an address in data cache, so that a following load accesses DRAM. We use this method produce strong leakages during $T_{high,i}$.
- ***MIPS32*:** On *MIPS32*, we alternate counter increments during $T_{high,i}$ with sleeps, similarly to [44].

Code listings for these four architectures are shown in Appendix B.

B. Other Implementations Of Noise-SDR

Offline *Noise-SDR*: The leakage stage of *Noise-SDR* for a given application can run alone on the target device, reading the timings of the square wave from a precomputed file. Similarly the RF-PWM stage can run alone to produce an RF-PWM square wave from a generic baseband signal generated with another SDR tool (e.g., *GNURadio* [68], *Qsstv* [69], *FLDigi* [56], *WSJT-X* [70]). GNURadio is a generic framework (also available for Android [71]) for signal processing, which allows defining custom C++ and Python blocks. The RF-PWM and leakage stages could be easily added to signal processing flows this way. We demonstrate the offline method for protocols such as FT4 [72], GLONASS C/A [73], HamDRM [74], SSTV [75], AM [76], and NBFM [77] (but they could be easily integrated into Fldigi-*Noise-SDR*, too).

Other leakage sources: *Noise-SDR* is not limited to using the leakage produced by memory accesses to DRAM. We have conducted preliminary experiments using other methods and sources to generate the leakage. For example, (i) math operations in JavaScript [44], and (ii) screens displaying pixels of different color [1], [3], [45], [78], [79]. However, they are outside the scope of this paper. Indeed, for sake of clarity we focus on one leakage source (memory accesses on several architectures) to show the novel idea of arbitrary modulation.

V. Experimental Evaluation

A. Preliminary Considerations

A whole design space: *Noise-SDR* is a generic way to establish a communication link, optimized for the desired properties. With *Noise-SDR*, the attacker gains the ability to explore a whole design space for transmissions, while, previous work generally provides fixed design points. For example, Table II shows a non-exhaustive list of digital and analog protocols that can be used with *Noise-SDR*.

A theoretical point of view on the design space: Given a noisy channel, the Shannon-Hartley theorem [80] about channel capacity expresses the trade-offs between the data rate, the bit error rate, the bandwidth occupied by the signal, and the signal-to-noise ratio over that bandwidth. Given a certain Signal-to-Noise Ratio (SNR), bandwidth, and error rate, the maximum achievable data rate has a limit. Existing protocols take design choices in this space [81]. With *Noise-SDR*, attackers can easily explore this design space, too.

Evaluation: Given a specific leakage source on a specific device, we compare the performance of different protocols and modulation techniques. This is meaningful because the same device has the same properties, such as, power, clock stability, and available bandwidth. Different leakage sources/devices will have different properties. Therefore, not all protocols are suitable for all devices, and the same modulation does not have the same performance on all devices. One of the advantages of *Noise-SDR* is the ability to flexibly choose the best fit. Extensive experimental analyses of leakage are outside the scope of this paper and available in literature [46]–[50].

Experimental setup at reception: We use an Ettus Research USRP B210 SDR [82] peripheral, connected to a laptop. One of the advantages of *Noise-SDR* is that we do not need to design a custom signal processing block to implement the receiver (unless we use a custom protocol). We can simply use popular SDR tools, such as, Gqrx [83] to control the SDR, and FLDigi [56], Qsstv [69], WSJT-X [70], and gnss-sdr [84]. There exist many open-source implementations for many protocols based on GNURadio [68], for example, for LoRa [37]. The ability to leverage existing high-quality receivers let us quickly experiment with many protocols, without any engineering effort. We always use a standard monopole antenna. Only for Direct Sequence Spread Spectrum (DSSS) we use a NAE-HPROBE-15 antenna [85] and an additional TEXBOX TBWA2 amplifier [86].

Experimental setup at transmission: We run *Noise-SDR* on the target devices in Table III in a realistic home environment. We avoid any coupling between transmission and reception. For the general measurements, we run the code in foreground, and we disable other communication interfaces (airplane mode on mobile devices).

Metrics: In amateur radio speed is given in words (6 characters) per minute (wpm). For analog protocols we report the audio quality and rate. For the other protocols we report the amount of time required to transmit an amount of data. For the SNR, we report the values computed by the reception tools (when available). The SNR is defined over a certain noise bandwidth (usually 2.5 kHz in amateur radio). The percentage of correctly received words ('% COPY') is a good metric for comparing protocols at reception [64].

B. Establishing Advanced Channels

Baseline: Previous was generally based on OOK [6], FSK [10] (both without symbol shaping), or, recently, approximated Lora-like

TABLE II
SOME DIGITAL AND ANALOG PROTOCOLS

Name	Modulation	Bandwidth	Reference
Simple CW	OOK	>50 Hz to >200 Hz	[58], [64]
Simple RTTY	BFSK	>270 Hz to >370 Hz	[59], [64]
LoRa-like	CSS	Custom 8000 Hz	[8], [37]
CW	Shaped OOK	50 Hz to 200 Hz	[58], [64]
RTTY	Shaped BFSK	270 Hz to 370 Hz	[59], [64]
LoRa	CSS	Custom 8000 Hz	[37], [87]
MFSK	8FSK-32FSK	154 Hz to 1920 Hz	[60], [64]
THOR	IFK+	173 Hz to 1800 Hz	[62], [64]
PSK	BPSK	31 Hz to 1800 Hz	[61], [64]
Multi-carrier PSK	PSK OFDM	650 Hz to 3600 Hz	[64]
FT4	4-GFSK	90 Hz	[72]
HamDRM	QAM OFDM	2.4 kHz	[74]
GLONASS C/A	DSSS	0.511 MHz	[73]
AM	Analog AM	10 kHz	[76]
NBFM	Analog FM	12.5 kHz	[77]
SSTV	Analog FM	2.5 kHz	[75]

CSS [8]. For this reason, we chose the following baseline:

- **Simple CW (OOK):** CW [58] uses Morse code encoding and then turns the carrier on and off to represent marks and spaces.
- **Simple RTTY (FSK):** RTTY [59] uses two tones to send ones and zeros.
- **Simple LoRa-like (CSS):** We implemented simple Lora-like CSS by removing advanced techniques like FEC and interleaving from the full LoRa implementation used by *Noise-SDR*.

Table IV (IV.1-IV.5) shows the results of the baseline. For each device/protocol, we report the maximum distance for reliable transmission. CW20 (Table IV.1) has good performance regarding distance (thanks to its very low bandwidth) but it is very slow.

Reducing Inter Symbol Interference (ISI) with shaping: Sharp transitions between symbols occupy a large bandwidth and lead to ISI, reducing performance. This form of noise can be removed by shaping the spectrum of the symbols with appropriate filters. This is commonly implemented Fldigi and other tools that work with real SDRs. However, previous work cannot use them, because they generally result in a signal with concurrent amplitude and frequency changes at multi-bit resolution. Instead, *Noise-SDR* is an SDR and is able to generate these signals. All Fldigi modes with symbol shaping (e.g., PSK) are also available in Fldigi-*Noise-SDR*.

Using phase modulation: For the first time, we show PSK with a leakage. PSK can achieve a high spectral efficiency (the number of bits transmitted per second per hertz of occupied bandwidth) [64]. In addition, PSK is often used to modulate the subcarriers of a high-speed OFDM transmission. All PSK modes in Fldigi-*Noise-SDR* use symbol shaping for optimal performance. See Table IV (IV.12-IV.20) for examples of (multi-carrier) PSK.

DSSS for resilience, secrecy, and multiple-access: For the first time, this paper shows DSSS with a leakage. DSSS is a spread spectrum technique based on multiplying a PSK signal with a pseudorandom spreading code. Spreading makes the communication more resistant to narrow-band noise. The autocorrelation properties of the code make detection and tracking possible below the noise floor. Using a secret code (or cryptographically secure) code makes the transmission hard to detect or jam. The low cross-correlation between codes makes it possible for two transmitters to use the same frequency (Code Division Multiple Access (CDMA)). DSSS is possible thanks to: (i) the phase modulation of the leakage, (ii) the reuse of Global Navigation Satellite System (GNSS) protocols. Existing SDR receivers [84] which implement code detection and tracking can precisely determine the frequency and phase of the codes, working even with unstable clocks or moving targets. We experimented with the GLONASS C/A code (Table IV.18). We also implemented a 'slowed down' mode of GLONASS (including code and data) that runs 10 times slower (Table IV.19). Similarly, we experimented the transmission of 2 GPS C/A codes simultaneously, 100 times slower (Table IV.20). The problem of synchronization (solved by GNSS receivers) is one of the reasons why previous work [8] avoided DSSS in favor of CSS. Both CSS and DSSS are transparently supported by *Noise-SDR*'s SDR approach.

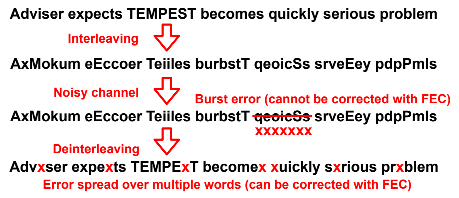

Adviser expects TEMPEST becomes quickly serious problem
Interleaving
AxMokum eEccoer Teiiles burbstT qeoicSs srveEey pdpPmls
Noisy channel
Burst error (cannot be corrected with FEC)
AxMokum eEccoer Teiiles burbstT ~~qeoicSs~~ srveEey pdpPmls
xxxxxxx
Deinterleaving
Advxser expexts TEMPExT becomex xuickly sxrious prxblem
Error spread over multiple words (can be corrected with FEC)

Fig. 7. Interleaving: shuffling bits over time spreads burst errors over multiple words, where FEC can correct them, improving resilience to fading/interference.

Improving noise resilience with FEC and interleaving: The various sources of noise that affect a radio channel might introduce errors in the received data, even if the modulation scheme is itself robust. Several techniques exist to counter this problem. FEC [88] adds m redundant bits of information every n bits of data (coding rate $n/(n+m)$). On the one hand this reduces the efficiency of the transmission, on the other hand it makes the receiver able to detect and correct up to a certain number of bit errors. In other words, a low error rate can be achieved with a lower SNR, at the price of a lower effective data rate due to redundancy. Sometimes, errors are localized at a certain moment in time (e.g., interfering signal, fading condition) leading to many errors for a single word, so that FEC is not enough to correct them. Interleaving [88] spreads the bits of a single word of data over multiple words over time. If a burst error occurs on a full word after interleaving, it will result in smaller errors on many words after de-interleaving. Such smaller errors spread over multiple words are more likely to be detected and corrected by FEC. Figure 7 informally summarizes this concept. Previous work on exfiltration uses simple custom protocols without these techniques (to the best of our knowledge only [11] uses FEC). Instead, Fldigi-*Noise-SDR* supports a large number of protocols that use FEC and interleaving to achieve high robustness in challenging environments (THOR, PSKR (Robust), real LoRa, to cite a few). With *Noise-SDR* supporting existing advanced protocols is transparent and available to any attacker, because *Noise-SDR* behaves like an SDR. Instead, adding FEC and interleaving to previous work at transmission and reception would add significant development efforts and advanced knowledge in radio communications and signal processing. The advantage of FEC and interleaving (larger distance at the price of lower effective rate) is particularly evident in Table IV when comparing the same mode with and without them (e.g., IV.6 vs. IV.5, IV.13 vs. IV.12, IV.16 vs.

TABLE III
DEVICES FOR PROTOCOL COMPARISON

	Device	Type	Arch.	OS Family	DRAM	F_{leak}	$(F_{IF}+B)_{max}$	SSC	Harmonics n
A	HP ENVY	Laptop	*x86-64*	Ubuntu	DDR3	800 MHz	15.062 kHz	yes	1
B	PC	Desktop	*x86-64*	Windows	DDR3	800 MHz	35.062 kHz	yes	1
C	Samsung Galaxy S5 Mini	Phone	*ARMv7-A*	Android	n.a.	400 MHz	15.062 kHz	no	1-11, 13-19, 26
D	Innos D6000	Phone	*ARMv8-A*	Android	LPDDR3	800 MHz	1.130 MHz	no	1-4
E	8Devices Carambola2	IoT	MIPS	OpenWRT	DDR2	400 MHz	35.062 kHz	no	1-6

TABLE IV
ADVANCED CHANNELS SPEED-DISTANCE TRADE-OFF FOR 100% COPY

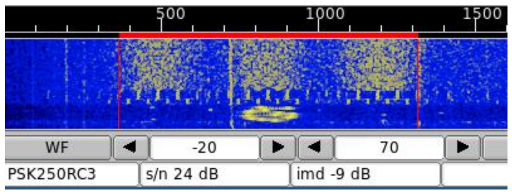
	Protocol	Speed	A (cm)	B (cm)	C (cm)	D (cm)	E (cm)
IV.1	Simple CW20	20 wpm	-	200	2	-	300
IV.2	Simple CW100	100 wpm	-	2	-	-	60
IV.3	Simple RTTY50	66 wpm	-	1	3	0	30
IV.4	Simple RTTY75	100 wpm	-	0	2	-	25
IV.5	LoRa-like 8 kHz, SF=8	16 bytes, 1.128 s	-	75	8	0	210
IV.6	LoRa 8 kHz, SF=8	16 bytes, 1.928 s	-	120	9	3	300
IV.7	MFSK32	120 wpm	0	20	15	1	300
IV.8	MFSK128	480 wpm	-	9	8	0	84
IV.9	THOR4	14 wpm	8	250	110	10	>500
IV.10	THOR16	58 wpm	0	105	65	4	>500
IV.11	THOR100	352 wpm	-	30	5	2	65
IV.12	PSK125	200 wpm	0	100	4	0	40
IV.13	PSK125R	110 wpm	0	250	15	1	75
IV.14	3xPSK250R	660 wpm	-	2	1	-	50
IV.15	2xPSK500	3200 wpm	-	-	0 (Unreliable)	-	1 (Unreliable)
IV.16	2xPSK500R	1760 wpm	-	-	1	-	10
IV.17	HamDRM A QAM4	1140x960RGB, 45 s	-	-	0 (Needs multiple runs)	-	5
IV.18	GLONASS C/A	511 chips per 1 ms	-	-	-	0	-
IV.19	GLONASS /10	511 chips per 10 ms; 5 bps	-	-	-	0	-
IV.20	GPS C/A /100 (2 codes)	1023 chips per 100 ms	-	-	-	0	-
IV.21	FT4	77 bits, 4.48 s	0	100	500 (If detected, see Figure 12)	1	500
IV.22	AM	16-bit 44.1 kHz audio	-	4	5	0	50
IV.23	NBFM	16-bit 44.1 kHz audio	-	10	10	0	>400
IV.24	SSTV Martin1	320x256RGB, 114 s	-	2	5	0	30

IV.15), and also in the distance achieved by FT4 (Table IV.21).

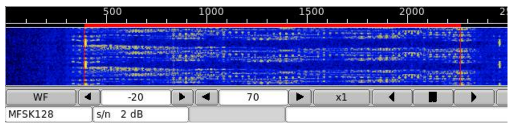
Using IFK+ for resilience to multi-path: The Offset Incremental Frequency Shift Keying used by THOR [62] is robust because symbols are encoded in the frequency difference with the previous symbol instead of a fixed value. In addition, incremental keying makes it particularly robust to Inter Symbol Interference (ISI) due to multi-path reception [89], typical of realistic indoor environments with walls and obstacles. Results are shown in Table IV.9-IV.11.

Using OFDM for higher data rates: Orthogonal Frequency Division Multiplexing consists in modulating multiple orthogonal subcarriers to increase the data rate. Usually, each carrier is modulated in phase or Quadrature Amplitude Modulation (QAM) (phase and amplitude). Previous work has shown elementary examples of transmission with multiple OOK-modulated magnetic-field carriers using multiple CPU cores [11]. Using multiple threads and cores is necessary because a single thread cannot generate a signal more complex than a single OOK or FSK carrier using previously known techniques, while an OFDM signal with multiple subcarriers shows both amplitude and frequency variations at multiple levels. Instead, *Noise-SDR* can generate a generic signal, including PSK-OFDM, using a single thread and performant existing protocols. They can achieve high data rates, as shown in Table IV (IV.14-IV.17). Multiple cores and threads can be used to further transmit more data at other frequencies, with the same or different protocols. For example, a smartphone could run one instance of Fldigi-*Noise-SDR* transmitting with 3X PSK250R on one core, and another instance transmitting with PSK31 on another core. See Figure 8a for an example of multiple PSK subcarriers with a single thread, and Figure 8c for independent processes transmitting with three different protocols concurrently.

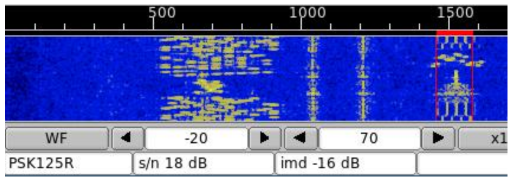
Using Reed Solomon Identifier (RSID) for automatic detection and tuning: An attacker might not know the exact protocol and frequency at which the victim device is transmitting. They could change over time depending on the load, or the same device could transmit at multiple frequencies with different protocols. Small differences between devices could lead to slightly different frequency offsets from the nominal value. For this reason, it is very useful to prepend an easy-to-detect unique signal before starting the transmission. To this purpose, Fldigi-*Noise-SDR* can use RSIDs [90]. These sequences can be easily decoded at very low SNR (−16 dB) and uniquely identify the protocols, parameters, and frequency of the transmission (with a precision of 2.7 Hz). The Fldigi receiver is able to scan for one or more of the supported protocols over a frequency band, and automatically start demodulating and decoding once a transmission is detected. For most protocols, the RSID is easier to decode than the data. From a security perspective, this is useful to detect at least the presence of a transmitter even when the signal is not good enough to decode the data. Note that attackers interested in being more stealth could use custom codes. Nevertheless, the presence



(a) One thread: 3X PSK250R (3 BPSK carriers, 660 wpm).



(b) One thread: MFSK128 (16-FSK 480 wpm).



(c) Concurrent processes: THOR22 (18-IFK+ 78 wpm), RTTY45 (BFSK 60 wpm), PSK125R (BPSK 110 wpm).

Fig. 8. Non-exhaustive examples of advanced communications with Fldigi-*Noise-SDR* at 5 cm from the device E in an office, at 400 MHz. All modes but RTTY45 use FEC and interleaving and are preceded by an RSID.

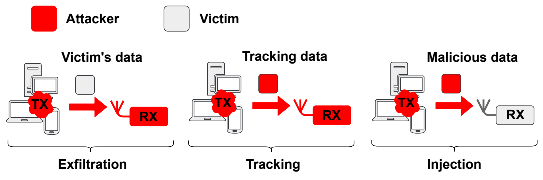


Fig. 9. Some security applications of *Noise-SDR*: (i) data exfiltration from a victim device, (ii) tracking of a victim device with a fixed beacon signal, (iii) injection of malicious data from a victim device to another victim receiver.

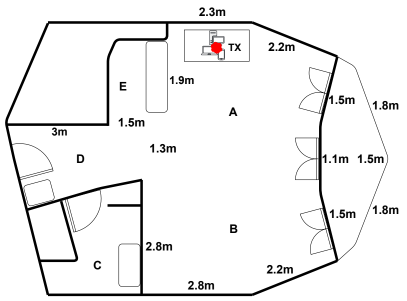


Fig. 10. Anonymized and simplified scheme of the home environment. The target is placed on a desk as in a typical workstation. Letters denote different areas. The presence of walls, appliances, and other objects results in a realistic scenario.

of other legitimate amateur radio transmissions could conceal the attack, too. Figure 8 shows several modes preceded by their RSID. The red lines show that Fldigi was able to perfectly identify and track the carrier frequency and start decoding, without any manual effort.

The advantages of analog protocols: Digital protocols offer many advantages, but also analog protocols are useful in some applications. Transmitting audio with AM and Narrow-Band Frequency Modulation (NBFM) is particularly useful during the analysis of a device and the implementation or tuning of new techniques. The audio signals offer a quick human-understandable feedback and tolerate huge distortion and noise. The SSTV protocol transmits color images with NBFM. In a practical security scenario, analog protocols could be useful to exfiltrate audio and images. Results are shown in Table IV (IV.22-IV.24).

Summary: We tested many analog and digital protocols with different properties (Table II) on many devices (Table III) with different architectures. This shows how attackers can use *Noise-SDR* to establish advanced communication channels, covering a large design space without any effort. Reading Table IV by row shows the improvements made possible by *Noise-SDR* for all devices, whereas reading it by column highlights the differences among devices. Figure 8 shows some example spectrograms.

VI. Security Impact

Noise-SDR uses electromagnetic leakage to achieve advanced radio transmission, using arbitrary modulation and state-of-the-art protocols. This requires executing code without any privilege, permission, or access to output peripherals, which would normally be unable to transmit radio signals. Despite some limitations on the power, bandwidth, and frequency of the leakage, *Noise-SDR* has many security applications. In the following, we describe several case studies for exfiltration, tracking, and injection, summarized in Figure 9.

A. Exfiltration

Threat model: An attacker compromises a victim device, and then runs *Noise-SDR*'s code to exfiltrate sensitive data to a receiver nearby. The receiver is also controlled by the attacker. In the worst case for the attacker, the victim is air-gapped, that is, disconnected from the network. Although compromising air-gapped networks is possible (e.g., Stuxnet [91]), one might assume that the absence of available transceivers prevents exfiltration of data even when an attacker can execute code. However, using a physical leakage for transmission breaks this assumption [3]–[28].

Impact of *Noise-SDR*: Compared to previous work, *Noise-SDR* offers several advantages for exfiltration. For example, (i) use of previously-unavailable advanced techniques, modulations, and protocols, that offer great performance gains, (ii) ability to choose a convenient trade-off in a huge design space according to the attackers needs and goals, (iii) flexible software-defined physical layer that can be dynamically adapted to the conditions, (iv) software-defined approach that enables the reuse of optimized tools for existing protocols at transmission and reception, (v) little/no previous knowledge in radio communications required.

Example, practical exfiltration: We show two practical setups that can be used on the field, shown in Figure 11 (left). The Tytera MD UV 380 [92] is used to receive Very High Frequency (VHF) or Ultra High Frequency (UHF) radio signals. After down-conversion, the resulting audio signal is sent to Fldigi for Android [57] running on a smartphone. Fldigi can then decode any of the protocols supported by Fldigi-*Noise-SDR*. A similar approach could be replicated with other reception tools and radio peripherals. For example, the Tytera handheld radio can be replaced

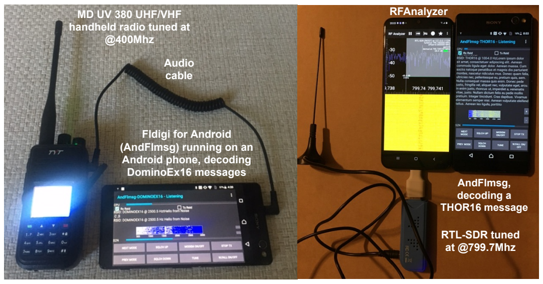

MD UV 380 UHF/VHF handheld radio tuned at @400Mhz
Audio cable
Fldigi for Android (AndFlmsg) running on an Android phone, decoding DominoEx16 messages
RFAnalyzer
AndFlmsg, decoding a THOR16 message
RTL-SDR tuned at @799.7Mhz

Fig. 11. Two portable receivers: (left) based on an handheld radio, (right) based on an SDR dongle. Fldigi is used for decoding: (left) DominoEx16 at 1.5 m from device E, (right) THOR16 at 3 m from device B.

by a ∼25$ RTL-SDR dongle [93] controlled by the RFAnalyzer application [94], as shown in Figure 11 (right). In these examples, the excellent properties of DominoEX16 and THOR16 make it possible to correctly receive from device E (1.5 m, 115 wpm) and device B (3 m, 58 wpm) using extremely simple and popular equipment, and without any engineering effort. Also note that this is a case in which using a relatively small bandwidth is very convenient both to work at low power and to use simple receivers (e.g., compared to large-bandwidth SDRs like the USRP210). At shorter distances we can use faster protocols, as shown in Table IV.

Example, flexible exfiltration: To show the flexibility of *Noise-SDR*, we demonstrate the following practical scenario and speed/distance trade-off. We consider device E on a desk in the home environment of Figure 10. We use Fldigi-*Noise-SDR* to exfiltrate a private DSA key (771 bytes) at 400 MHz. This application does not tolerate errors at reception. An attacker that can get close to the device (e.g., 1 cm to 50 cm on the same desk), but only for a short period of time, can use 3X PSK250R. At 660 wpm the exfiltration takes only a few seconds. However, an attacker might be limited to covertly listen in another separate room, but with more time available. In this case, THOR22 shows excellent performance thanks to its IFK+ that is resistant to multi-path reception. We were able to correctly receive the key (sent line by line) in some minutes from area C (i.e., at more than 5 m, enclosed by a 15 cm-think wall). We noticed that, while other protocols like MFSK and PSKR work in these conditions too, THOR22 is superior in ease of use (e.g., tuning) and reliability.

B. Tracking Beacons

Threat model: In a setting similar to exfiltration, the attacker runs *Noise-SDR*'s code to transmit a fixed beacon signal that can be used to identify and track the victim. Also in this case the attacker controls the receiver. This scenario is particularly relevant for mobile and IoT devices. For example, let us consider an Android application without any special privilege or permission. Users would install this application believing that their privacy will be preserved, because there are no permissions which would allow, for example, to transmit tracking signals (and airplane mode is additionally available). With Noise-SDR, transmitting tracking signals becomes possible and effective without requiring any privilege or permission to access peripherals, breaking this assumption.

Impact of *Noise-SDR*: Since tracking requires only a fixed message, attackers can use an offline implementation of *Noise-SDR*

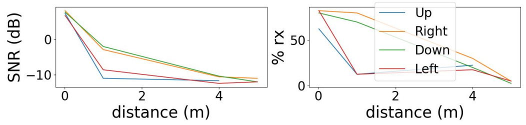


Fig. 12. SNR and percentage of received FT4 beacons at different distances and from different directions from device C at 1.2 GHz in an office.

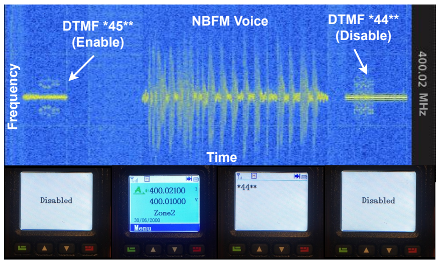


Fig. 13. Injection from device E to a receiver on channel 400.02 MHz (UHF band) at 1 m. *Noise-SDR* remotely controls a TYT MD-UV 380, using DTMF signaling [95] to enable and disable the receiver, and NBFM audio to transmit a voice message.

where the RF-PWM stage transmits a precomputed wave. This way, a complex protocol (e.g., very resilient using symbol shaping, FEC and interleaving) can be implemented with a very tiny piece of code, easy to conceal and with excellent transmission performance. In addition, DSSS could be used to achieve secrecy (secret or even cryptographic codes, transmission below the noise floor) and multiple access (with orthogonal codes like GPS).

Example, RSID-based beaconing: RSIDs have excellent reception properties even below the noise floor. In addition, they occupy a small bandwidth and there exist many codes that can be assigned to different devices for concurrent tracking. For example, we place devices C and E side by side on a desk in the home environment of Figure 10. We assign them the RSID of PSK250R at $f_{IF} = 8$kHz and THOR22 at $f_{IF} = 9$kHz, respectively. Both devices transmit their RSID continuously, with Fldigi-*Noise-SDR* running in background. At 2.5 m at the 1.4 GHz harmonic with a USRP B210 we can clearly distinguish both codes. Device E has a much stronger leakage than device C (also see Table IV), and tracking is possible all over areas A, B, C, D, and E (i.e., more than 5 m in many directions and in another closed room behind a 15 cm wall). The same is possible with the MD-UV 380 portable setup using the 400 MHz harmonic.

Example, FT4-based beaconing: To improve tracking for device C despite the lower transmission power, we use FT4 [72]. With its small bandwidth, high coding rate, and transmission synchronized with Universal Coordinated Time (UTC) time, FT4 is a perfect protocol to achieve large distance at very low power. With a USRP B210 and the WSJT-X [70] SDR receiver, reception is possible up to 5 m, as shown in Figure 12 for multiple directions (note that the experiment was carried in a realistic indoor environment and free-space loss is not the only factor that affects reception). Table IV reports more results for other devices.

C. Injection

Threat model: The attacker uses *Noise-SDR* on one victim (compromised beforehand or remotely) to inject malicious signals into another victim receiver (on the same platform or nearby). This is useful when placing a real transmitter close to the victim receiver is impractical. Leveraging a victim device for transmission can also be less expensive and more stealthy than using radio equipment.

Impact of *Noise-SDR*: Previous work transmits using simple custom protocols. In some cases [3], [6], [9] reception is possible with standard devices, provided that the attacker controls them and has access to low-level processing (e.g., raw WiFi data [9]). Instead, thanks to arbitrary modulation and its software-defined approach, *Noise-SDR* can generate valid signals of existing real-world protocols. For this reason, we believe it could start a novel line of research on signal injection and spoofing from unprivileged software.

Example, controlling a handheld UHF radio: The TYT MD-UV 380 [92] is a very popular handheld two-way radio operating in the UHF and VHF bands, supporting both digital and analog transmissions. Like many types of radio equipment, it supports remote control functions (e.g., enable/disable) through DTMF signaling [95]. We assume that it has been programmed to listen on Channel 1 at 400.020 MHz NBFM, and that it can be enabled and disabled using the DTMF codes *45** and *44** respectively. Note that once disabled the radio can be enabled only when it receives the *45** code, or by reprogramming it (which requires physical access). We use *Noise-SDR* running on a Carambola2 (device E) to transmit the DTMF codes and a voice message at the correct frequency ($f_c = F_{leak} + F_{IF} = 400\text{MHz} + 19.2\text{kHz}$). We can successfully enable the radio, transmit a voice message, and disable the radio, as shown in Figure 13. In the home environment of Figure 10, the voice transmitted by device E on the desk can be heard clearly all over most of areas A, B, D (i.e., more than 4 m). DTMF is less robust, but we successfully disabled the radio at 4 m in area B by repeatedly sending the off sequence for around 1 min. Among our devices, only the device E can transmit in the frequency range of the UHF radio, but this range is reasonable for other IoT devices with similar hardware. The choice of the intermediate frequency (and thus of the channel frequency) is limited by the bandwidth of device E (see Table III). More complex radio equipment (e.g., repeaters, sirens) could offer even more functions through DTMF. Compromising some digital equipment nearby could be then used to control the radios. Also, a repeater could further broadcast a message at higher power reaching more devices at larger distance. *Noise-SDR* is not limited to a specific protocol, and the same principle could be applied to other types of control and data signals. Some directions for future research on GNSS are mentioned in Section IX.

VII. Countermeasures

Low-accuracy timers: The lower the time resolution, the stricter the constraints on the generated signal, to the point where RF-PWM is not practical. Countermeasures that mitigate timing side channels by preventing accurate timing measurements (e.g., [51], [96], [97]) could be applied to *Noise-SDR*, too. However, the arms race with attackers (e.g., [51]–[55]) shows that this solution is not definitive, because other timing sources are available (e.g., calibrated counters).

Countermeasures against Rowhammer: Those countermeasures against Rowhammer that focus on detecting or preventing fast DRAM accesses could also be applied to *Noise-SDR*, but they are not a definitive solution. The detection of electromagnetic leakages [65] is not practical on mobile devices. Some of the approaches proposed for *ARMv8-A* [67] are not likely to be detected by observing cache misses (e.g., [98]) and cannot be easily forbidden in unprivileged code. Moreover, gadgets in system calls could be exploited from unprivileged code [67] (though not fast enough for flipping bits, they could be sufficient for *Noise-SDR* modulation). Finally, offensive research on Rowhammer is active and keeps finding several ways to access DRAM quickly (e.g, [55], [67], [99], [100]).

Careful design: A careful RF design reduces leakages and coupling between components, possibly at a higher cost. Shielding reduces the emissions from a device, but it can be at least partially bypassed [8], [11]. SSC reduces the peak power of the DRAM emissions, but software can still modulate the sub-harmonics [8].

VIII. Related Work

Fully-digital radios: Several fully-digital radios have been proposed in literature, based on one-bit coding as explained in Section II. Alternatively, a Direct-to-RF Converter (DRFC) synthesizes a radio signal directly with a high-speed multi-bit digital-to-analog converter (DAC). We refer the reader to specialized literature [29], [30], [101]. These techniques work with real hardware [30]–[33], whereas *Noise-SDR* uses software-induced leakages only.

Rowhammer: Code running Rowhammer attacks on *x86-64* machines can be detected through its electromagnetic leakages [65], and it can be used for covert channels [7]. Rowhammer attacks against smartphones with *ARM* architecture have been shown in [66] using DMA. Other approaches include exploiting the GPU [55] or ArmV8 cache maintenance instructions [67] for fast DRAM access. However, Rowhammer attacks aim at flipping bits in DRAM, and [7] uses a custom *FSK* protocol on *x86-64*, not generic signals on *ARM*.

Soft-TEMPEST: We have covered Soft-TEMPEST [1], [2] and covert channels [3]–[28] in Section I. *Noise-SDR* goes beyond previous modulation schemes, introducing arbitrary signal modulation.

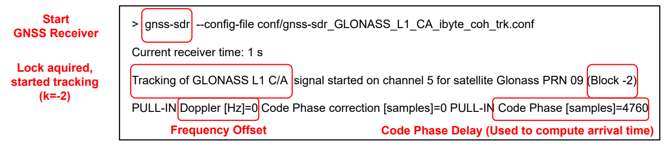
Simple radios: Makers have built radio transmitters with very few and simple components, described in numerous blog posts and repositories. For example: [44], [45] use Soft-TEMPEST leakages to send simple tunes; [102], [103] implement *RTTY* transmission with the Local Oscillator (LO) leakage of a SDR receiver; [104]–[110] use a cable connected to a microcontroller output pin to implement *OOK*, *FM*, *AFSK*, *NTSC*, *PAL*, *AM*; examples of clock signal *FSK* and delay line *FKS* simple radios are shown in [111]; [112] switches Ethernet speed to transmit CW signals; [113] uses VGA to transmit *DVB-T*, *PAL*, and *NTSC*; [114] uses VGA for *FM*; [115] uses the VGA DACs to feed an RF modulator; [116] uses the DAC of a reverse-engineered USB-to-VGA converter as DRFC SDR able to transmit generic signals including *GSM* and *GPS*; [117] is a direct sampling receiver using the analog-to-digital converter (ADC) of a microcontroller; [118] is a fully-digital FPGA-based BLE receiver. These works use signal processing tricks (e.g., undersampling) to deal with limited resources, and they were a source of inspiration, even if theoretical explanations are often informal or lacking. Naturally, those that use output pins or multi-bit VGA DACs are those that achieve the best results, but these outputs are not available to unprivileged software on smartphones and computers. Note that [105], [106] are fundamentally different from *Noise-SDR*. They first convert baseband audio into a $\Delta\Sigma$ signal, and then they transmit it with a simple fixed BFSK using an output pin. While this method achieves analog FM audio transmission, it cannot generate amplitude and phase modulated radio signals (only BFSK). Instead, *Noise-SDR* uses RF-PWM to generate arbitrarily modulated radio signals (starting from an OOK square wave generated with memory accesses). To the best our knowledge, none of these works achieves arbitrary modulation on smartphones or laptops using purely software leakages.

Nexmon SDR: Some Broadcom *WiFi* chips can transmit a short arbitrary signal in the *WiFi* band. Using a firmware patch, Nexmon [119], [120] turns them into an SDR, to build *WiFi* covert channels [121], or to jam *WiFi* networks [122].

IX. Discussion and Future Work

Limitations: *Noise-SDR* achieves arbitrary modulation, but it also has some limitations. First, the achievable power, frequency, and bandwidth are inherently limited by the leakage and the ability of the attacker to control it. Nevertheless, leakage often appears at several harmonics covering a large spectrum from a few MHz to several GHz. Moreover, *Noise-SDR* controls the intermediate frequency of the RF-PWM carrier, letting the attacker chose an offset (of up to hundreds of kHz on *ARMv8-A*) from the carrier frequency. For example, this is enough to align the carrier frequency to that of a GLONASS satellite in the band around an harmonic of the DRAM clock at 1.6 GHz. The second limitation is the bandwidth, constrained by the execution time of the instructions that produce the leakage. In our implementation we can reach tens of kHz on *ARMv7-A*, *x86-64* and *MIPS32*, and a few MHz on *ARMv8-A*. In our experiments on *x86-64* the limit is actually set by the harmonics of the SSC clock. However, this bandwidth is sufficient for a large number of useful protocols. Moreover, a large bandwidth is not always desirable. Using a very small bandwidth is a strategy used by many protocols to achieve large distance at low power [81] (e.g., FT4 [72] uses only 90 Hz). Finally, another limitation is the resolution and stability of time measurements. Software timers are not as accurate as dedicated radio hardware, but their performance (e.g., ns resolution of *clock_gettime*) is sufficient for many protocols. Moreover, state-of-the-art receivers have excellent algorithms to track clock drifts.

Open directions: *Noise-SDR* opens novel opportunities beyond data exfiltration. First, *Noise-SDR* could facilitate the security evaluation of complex systems regarding unintended emissions. The analyst could define a number of protocols, each representative of a given threat model, and then transmit beacons using *Noise-SDR*. A frequency scan would reveal whether the beacons can be retrieved from a given distance and direction, and at which frequency. This would also be useful to evaluate countermeasures like shielding. Second, *Noise-SDR* could leverage noise from high-speed digital components to inject malicious radio packets into victim receivers on the same platform. Indeed, their clock frequency often overlaps and interferes with radio protocols [123]–[126]. In particular, DRAM frequency or its harmonics sometimes overlap with GLONASS at 1.6 GHz (Table III). Future research could try to inject fake satellites, to produce erroneous location results, or to exploit software vulnerabilities (similar to [127]). As a preliminary result, Figure 14 shows the reception of a GLONASS C/A code transmitted by an Innos D6000 at 800.875 MHz (the same is possible with a Galaxy A30S using the harmonic at 1794 MHz). To



```
> gnss-sdr --config-file conf/gnss-sdr_GLONASS_L1_CA_ibyte_coh_trk.conf
Current receiver time: 1 s
Tracking of GLONASS L1 C/A signal started on channel 5 for satellite Glonass PRN 09 (Block -2)
PULL-IN Doppler [Hz]=0 Code Phase correction [samples]=0 PULL-IN Code Phase [samples]=4760
```

Fig. 14. Reception with *USPR B210* of a *GLONASS C/A Code* sent at 800.875 MHz with an Innos D6000. Using $F_{IF} = 875$kHz corresponds to satellite $k = -2$. Future research could try to inject the 1.6 GHz harmonic directly in the GNSS receiver.

ensure that the lock at reception is valid, we: (i) did not overfit the lock/tracking parameters of the default configuration of the gnss-sdr receiver, (ii) observed that the lock occurs only when transmission is on, (iii) observed that the lock always corresponds to the expected satellite for different choices of the tuning frequency. Third, *Noise-SDR* could be used to modulate radio carriers intentionally emitted by the circuit leveraging effects like Screaming Channels [42], [43], and building a Second-Order Soft-TEMPEST [40], [41]. For example, we were already able to leverage the carrier emitted by the Near Field Communication (NFC) reader of a Nokia 3.1 to transmit data with PSK31. Fourth, phase modulation and DSSS open new directions for research on localization (e.g., Doppler of moving targets) and/or undetectable secret transmissions (below the noise floor and with cryptographically secure spreading codes [128]). Fifth, *Noise-SDR* is a general approach that could be extended to other types of physical leakage. Finally, leakage control could be achieved without explicit code execution. For example, a website could download and display an image to control the screen's leakage, or memory accesses could be triggered with GPU-accelerated rendering in the browser [55], or by sending WiFi packets [129].

X. Conclusion and Replicability

We have presented *Noise-SDR*, a software-defined fully-digital approach to obtain a high degree of control on electromagnetic leakage compared to previous work. Despite some limitations on frequency and bandwidth, *Noise-SDR* achieves arbitrary modulation in amplitude, frequency, and phase. Like an SDR, *Noise-SDR* modulates a generic baseband signal generated in software on an electromagnetic leakage acting as carrier. This way, theoretical and practical knowledge in radio communications (e.g., advanced techniques, high-performance protocols, state-of-the-art tools) becomes readily available to the attacker. *Noise-SDR* has a fundamental impact on emission security for two main reasons. First, attackers can establish advanced communication channels that were previously thought unavailable to unprivileged software. Second, attackers can leverage SDR tools to use existing high-performance radio protocols without any knowledge or effort, or to design complex custom solutions. To ensure the replicability of our results, *Noise-SDR* is open source and available at: https://github.com/eurecom-s3/noise-sdr.

Acknowledgments

We would like to thank Andrea Possemato, Giulia Clerici, Matteo Guarrera, Elie Bursztein, Jean-Michel Picod, the anonymous reviewers and the shepherd for their feedback, help or interesting discussions. This work was partially supported by a Google Faculty Research Award.

References

[1] M. G. Kuhn and R. J. Anderson, "Soft Tempest: Hidden data transmission using electromagnetic emanations," in *Information Hiding*, D. Aucsmith, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 124–142.

[2] R. Anderson and M. G. Kuhn, "Soft Tempest - an opportunity for NATO," 1999.

[3] M. Guri, G. Kedma, A. Kachlon, and Y. Elovici, "AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies," in *9th International Conference on Malicious and Unwanted Software: The Americas MALWARE 2014, Fajardo, PR, USA, October 28-30, 2014*. IEEE Computer Society, 2014, pp. 58–67.

[4] M. Guri, M. Monitz, and Y. Elovici, "Bridging the air gap between isolated networks and mobile phones in a practical cyber-attack," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 4, May 2017.

[5] ——, "USBee: Air-gap covert-channel via electromagnetic emission from USB," in *14th Annual Conference on Privacy, Security and Trust, PST 2016, Auckland, New Zealand, December 12-14, 2016*. IEEE, 2016, pp. 264–268.

[6] M. Guri, A. Kachlon, O. Hasson, G. Kedma, Y. Mirsky, and Y. Elovici, "GSMem: Data exfiltration from air-gapped computers over GSM frequencies," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, J. Jung and T. Holz, Eds. USENIX Association, 2015, pp. 849–864.

[7] Z. Zhan, Z. Zhang, and X. Koutsoukos, "Bitjabber: The world's fastest electromagnetic covert channel," in *2020 IEEE International Test Conference (ITC)*. IEEE, 2020.

[8] C. Shen, T. Liu, J. Huang, and R. Tan, "When LoRa meets EMR: Electromagnetic covert channels can be super resilient," in *2021 2021 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2021, pp. 529–542.

[9] M. Guri, "AIR-FI: Generating covert wi-fi signals from air-gapped computers," *CoRR*, vol. abs/2012.06884, 2020.

[10] M. Guri, A. Daidakulov, and Y. Elovici, "MAGNETO: Covert channel between air-gapped systems and nearby smartphones via cpu-generated magnetic fields," *CoRR*, vol. abs/1802.02317, 2018.

[11] M. Guri, B. Zadov, and Y. Elovici, "ODINI: Escaping sensitive data from faraday-caged, air-gapped computers via magnetic fields," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1190–1203, 2020.

[12] N. Matyunin, J. Szefer, S. Biedermann, and S. Katzenbeisser, "Covert channels using mobile device's magnetic field sensors," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016, pp. 525–532.

[13] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici, "PowerHammer: Exfiltrating data from air-gapped computers through power lines," *IEEE Trans. Information Forensics and Security*, vol. 15, pp. 1879–1890, 2020.

[14] Z. Shao, M. A. Islam, and S. Ren, "Your noise, my signal: Exploiting switching noise for stealthy data exfiltration from desktop computers," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 1, pp. 07:1–07:39, 2020.

[15] V. Sepetnitsky, M. Guri, and Y. Elovici, "Exfiltration of information from air-gapped machines using monitor's LED indicator," in *IEEE Joint Intelligence and Security Informatics Conference, JISIC 2014, The Hague, The Netherlands, 24-26 September, 2014*. IEEE, 2014, pp. 264–267.

[16] J. Loughry and D. A. Umphress, "Information leakage from optical emanations," *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 3, pp. 262–289, 2002.

[17] M. Guri, B. Zadov, and Y. Elovici, "LED-it-GO: Leaking (A lot of) data from air-gapped computers via the (small) hard drive LED," in *Detection of Intrusions and Malware, and Vulnerability Assessment - 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings*, ser. Lecture Notes in Computer Science, M. Polychronakis and M. Meier, Eds., vol. 10327. Springer, 2017, pp. 161–184.

[18] M. Guri, B. Zadov, A. Daidakulov, and Y. Elovici, "xLED: Covert data exfiltration from air-gapped networks via switch and router leds," in *16th Annual Conference on Privacy, Security and Trust, PST 2018, Belfast, Northern Ireland, Uk, August 28-30, 2018*, K. McLaughlin, A. A. Ghorbani, S. Sezer, R. Lu, L. Chen, R. H. Deng, P. Miller, S. Marsh, and J. R. C. Nurse, Eds. IEEE Computer Society, 2018, pp. 1–12.

[19] M. Guri and D. Bykhovsky, "aIR-Jumper: Covert air-gap exfiltration/infiltration via security cameras & infrared (IR)," *Comput. Secur.*, vol. 82, pp. 15–29, 2019.

[20] R. Hasan, N. Saxena, T. Halevi, S. Zawoad, and D. Rinehart, "Sensing-enabled channels for hard-to-detect command and control of mobile devices," in *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, K. Chen, Q. Xie, W. Qiu, N. Li, and W. Tzeng, Eds. ACM, 2013, pp. 469–480.

[21] V. Subramanian, A. S. Uluagac, H. Cam, and R. A. Beyah, "Examining the characteristics and implications of sensor side channels," in *Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9-13, 2013*. IEEE, 2013, pp. 2205–2210.

[22] M. Guri, "AiR-ViBeR: Exfiltrating data from air-gapped computers via covert surface vibrations," *CoRR*, vol. abs/2004.06195, 2020.

[23] M. Guri, Y. A. Solewicz, and Y. Elovici, "MOSQUITO: Covert ultrasonic transmissions between two air-gapped computers using speaker-to-speaker communication," in *IEEE Conference on Dependable and Secure Computing, DSC 2018, Kaohsiung, Taiwan, December 10-13, 2018*. IEEE, 2018, pp. 1–8.

[24] M. Guri, Y. A. Solewicz, A. Daidakulov, and Y. Elovici, "Acoustic data exfiltration from speakerless air-gapped computers via covert hard-drive noise ('DiskFiltration')," in *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, ser. Lecture Notes in Computer Science, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds., vol. 10493. Springer, 2017, pp. 98–115.

[25] B. Carrara and C. Adams, "On acoustic covert channels between air-gapped systems," in *Foundations and Practice of Security - 7th International Symposium, FPS 2014, Montreal, QC, Canada, November 3-5, 2014. Revised Selected Papers*, ser. Lecture Notes in Computer Science, F. Cuppens, J. García-Alfaro, A. N. Zincir-Heywood, and P. W. L. Fong, Eds., vol. 8930. Springer, 2014, pp. 3–16.

[26] M. Guri, Y. A. Solewicz, A. Daidakulov, and Y. Elovici, "Fansmitter: Acoustic data exfiltration from (speakerless) air-gapped computers," *CoRR*, vol. abs/1606.05915, 2016.

[27] M. Guri, "POWER-SUPPLaY: Leaking data from air-gapped systems by turning the power-supplies into speakers," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 516, 2020.

[28] M. Guri, M. Monitz, Y. Mirski, and Y. Elovici, "BitWhisper: Covert signaling channel between air-gapped computers using thermal manipulations," in *2015 IEEE 28th Computer Security Foundations Symposium*, 2015, pp. 276–289.

[29] F. Raab, "Radio frequency pulsewidth modulation," *IEEE Transactions on Communications*, vol. 21, no. 8, pp. 958–966, August 1973.

[30] P. A. Nuyts, P. Reynaert, and W. Dehaene, *Continuous-time digital front-ends for multistandard wireless transmission*. Springer, 2014.

[31] S. Kulkarni, I. Kazi, D. Seebacher, P. Singerl, F. Dielacher, W. Dehaene, and P. Reynaert, "Multi-standard wideband OFDM RF-PWM transmitter in 40nm CMOS," in *ESSCIRC Conference 2015 - 41st European Solid-State Circuits Conference (ESSCIRC)*, 2015, pp. 88–91.

[32] J. S. Walling, H. Lakdawala, Y. Palaskas, A. Ravi, O. Degani, K. Soumyanath, and D. J. Allstot, "A Class-E PA with pulse-width and pulse-position modulation in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 6, pp. 1668–1678, 2009.

[33] M. Grozing, J. Digel, T. Veigel, R. Bieg, J. Zhang, S. Brandl, M. Schmidt, C. Haslach, D. Markert, and W. Templ, "A RF pulse-width and pulse-position modulator IC in 28 nm FDSOI CMOS," *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–4, 2018.

[34] S. D. R. Forum, "SDRF Cognitive Radio Definitions Working Document SDRF-06-R-0011-V1.0.0," 2007. [Online]. Available: http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf

[35] J. Pohl and A. Noack, "Universal radio hacker: A suite for analyzing and attacking stateful wireless protocols," in *12th USENIX Workshop on Offensive Technologies, WOOT 2018, Baltimore, MD, USA, August 13-14, 2018*, C. Rossow and Y. Younan, Eds. USENIX Association, 2018.

[36] A. Dubey, D. Vohra, K. Vachhani, and A. Rao, "Demonstration of vulnerabilities in GSM security with USRP B200 and open-source penetration tools," in *2016 22nd Asia-Pacific Conference on Communications (APCC)*, 2016, pp. 496–501.

[37] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming, and A. Burg, "An open-source LoRa physical layer prototype on GNU Radio," 2020.

[38] A. Behzad, *Wireless LAN Radios: System Definition to Transistor Design (IEEE Press Series on Microelectronic Systems)*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2008.

[39] K. Slattery and H. Skinner, *Platform interference in wireless systems: Models, measurement, and mitigation*. Newnes, 2011.

[40] E. Cottais, J. L. Esteves, and C. Kasmi, "Second order soft-TEMPEST in RF front-ends: Design and detection of polyglot modulations," *2018 International Symposium on Electromagnetic Compatibility (EMC EUROPE)*, pp. 166–171, 2018.

[41] J. L. Esteves, E. Cottais, and C. Kasmi, "Second order soft Tempest: From internal cascaded electromagnetic interactions to long haul covert channels," *2019 URSI Asia-Pacific Radio Science Conference (AP-RASC)*, pp. 1–3, 2019.

[42] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, “Screaming channels: When electromagnetic side channels meet radio transceivers,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM, 2018, pp. 163–177.

[43] G. Camurati, A. Francillon, and F.-X. Standaert, “Understanding screaming channels: From a detailed analysis to improved attacks,” *IACR Transactions on Cryptographic Hardware and Embedded Systems (CHES 2020)*, vol. 2020, no. 3, pp. 358–401, June 2020.

[44] W. Entriken, “System bus radio,” 2013. [Online]. Available: https://github.com/fulldecent/system-bus-radio

[45] E. Thiele, “Tempest for Eliza,” 2001. [Online]. Available: http://www.erikyyy.de/tempest/

[46] R. Callan, A. Zajić, and M. Prvulovic, “A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events,” in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014, pp. 242–254.

[47] A. Zajić and M. Prvulovic, “Experimental demonstration of electromagnetic information leakage from modern processor-memory systems,” *IEEE Transactions on Electromagnetic Compatibility*, vol. 56, no. 4, pp. 885–893, 2014.

[48] R. L. Callan, A. G. Zajić, and M. Prvulovic, “FASE: Finding amplitude-modulated side-channel emanations,” in *Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, OR, USA, June 13-17, 2015*, D. T. Marr and D. H. Albonesi, Eds. ACM, 2015, pp. 592–603.

[49] M. Prvulovic, A. Zajić, R. L. Callan, and C. J. Wang, “A method for finding frequency-modulated and amplitude-modulated electromagnetic emanations in computer systems,” *IEEE Transactions on Electromagnetic Compatibility*, vol. 59, no. 1, pp. 34–42, 2017.

[50] S. Sangodoyin, F. Werner, B. B. Yilmaz, C. L. Cheng, E. M. Ugurlu, N. Sehatbakhsh, M. Prvulovic, and A. Zajić, “Side-channel propagation measurements and modeling for hardware security in iot devices,” *IEEE Transactions on Antennas and Propagation*, pp. 1–1, 2020.

[51] D. Kohlbrenner and H. Shacham, “Trusted browsers for uncertain times,” in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, T. Holz and S. Savage, Eds. USENIX Association, 2016, pp. 463–480.

[52] M. Schwarz, C. Maurice, D. Gruss, and S. Mangard, “Fantastic timers and where to find them: High-resolution microarchitectural attacks in javascript,” in *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, ser. Lecture Notes in Computer Science, A. Kiayias, Ed., vol. 10322. Springer, 2017, pp. 247–267.

[53] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, “ARMageddon: Cache attacks on mobile devices,” in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, T. Holz and S. Savage, Eds. USENIX Association, 2016, pp. 549–564.

[54] B. Gras, K. Razavi, E. Bosman, H. Bos, and C. Giuffrida, “ASLR on the line: Practical cache attacks on the MMU,” in *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.

[55] P. Frigo, C. Giuffrida, H. Bos, and K. Razavi, “Grand pwning unit: Accelerating microarchitectural attacks with the GPU,” in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 195–210.

[56] D. Freese, “FLDigi.” [Online]. Available: http://www.w1hkj.com/

[57] ——, “AndFlmsg.” [Online]. Available: http://www.w1hkj.com/files/AndFlmsg/

[58] Sigidwiki, “Morse Code (CW).” [Online]. Available: https://www.sigidwiki.com/wiki/Morse_Code_(CW)

[59] ——, “RTTY50.” [Online]. Available: https://www.sigidwiki.com/wiki/Radio_Teletype_(RTTY)

[60] ——, “MFSK.” [Online]. Available: https://www.sigidwiki.com/wiki/Multi_Frequency_Shift_Keying_(MFSK)

[61] ——, “PSK.” [Online]. Available: https://www.sigidwiki.com/wiki/Phase_Shift_Keying_(PSK)

[62] ——, “THOR.” [Online]. Available: https://www.sigidwiki.com/wiki/THOR

[63] ——, “OLIVIA.” [Online]. Available: https://www.sigidwiki.com/wiki/Olivia

[64] D. Freese, “FLDigi Modes.” [Online]. Available: http://www.w1hkj.com/FldigiHelp-3.21/Modes/

[65] Z. Zhang, Z. Zhan, D. Balasubramanian, B. Li, P. Volgyesi, and X. Koutsoukos, “Leveraging EM side-channel information to detect rowhammer attacks,” in *2020 IEEE Symposium on Security and Privacy (S&P'20)*, 2020, pp. 862–879.

[66] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, “Drammer: Deterministic rowhammer attacks on mobile platforms,” in *CCS*, Oct. 2016, pwnie Award for Best Privilege Escalation Bug, Android Security Reward, CSAW Best Paper Award, DCSR Paper Award.

[67] Z. Zhang, Z. Zhan, D. Balasubramanian, X. D. Koutsoukos, and G. Karsai, “Triggering rowhammer hardware faults on ARM: A revisit,” in *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security, ASHES@CCS 2018, Toronto, ON, Canada, October 19, 2018*, C. Chang, U. Rührmair, D. Holcomb, and J. Guajardo, Eds. ACM, 2018, pp. 24–33.

[68] “GNU Radio.” [Online]. Available: https://www.gnuradio.org/

[69] “Qsstv.” [Online]. Available: http://users.telenet.be/on4qz/

[70] J. Taylor, “WSPR instructions.” [Online]. Available: https://physics.princeton.edu/pulsar/K1JT/WSPR_Instructions.TXT

[71] B. Bloessl, “GNU Radio Android Toolchain.” [Online]. Available: https://github.com/bastibl/gnuradio-android

[72] J. Taylor, “FT4.” [Online]. Available: https://physics.princeton.edu/pulsar/k1jt/FT4_Protocol.pdf

[73] G. C. S. I. Center, “GLONASS interface control document,” 1998. [Online]. Available: https://www.unavco.org/help/glossary/docs/ICD_GLONASS_4.0_(1998)_en.pdf

[74] Sigidwiki, “HamDRM.” [Online]. Available: https://www.sigidwiki.com/wiki/WinDRM

[75] ——, “SSTV.” [Online]. Available: https://www.sigidwiki.com/wiki/SSTV

[76] ——, “AM.” [Online]. Available: https://www.sigidwiki.com/wiki/Amplitude_Modulation_(AM)

[77] ——, “NBFM.” [Online]. Available: https://www.sigidwiki.com/wiki/NFM_Voice

[78] Y. Hayashi, N. Homma, M. Miura, T. Aoki, and H. Sone, “A threat for tablet pcs in public space: Remote visualization of screen images using EM emanation,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, G. Ahn, M. Yung, and N. Li, Eds. ACM, 2014, pp. 954–965.

[79] Z. Liu, N. Samwel, L. Weissbart, Z. Zhao, D. Lauret, L. Batina, and M. A. Larson, “Screen gleaning: A screen reading TEMPEST attack on mobile devices exploiting an electromagnetic side channel,” *CoRR*, vol. abs/2011.09877, 2020.

[80] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, no. 4, pp. 623–656, 1948.

[81] Y. Roth, J.-B. Doré, L. Ros, and V. Berg, “The physical layer of low power wide area networks: Strategies, information theory's limit and existing solutions,” in *Advances in Signal Processing: Reviews, Vol. 1, Book Series*, Aug. 2018.

[82] Ettus Research, “USRP B210.” [Online]. Available: http://www.ettus.com/all-products/UB210-KIT/

[83] A. Csete, “Gqrx SDR.” [Online]. Available: https://gqrx.dk/

[84] “gnss-sdr.” [Online]. Available: https://gnss-sdr.org/

[85] NewAE Technology Inc., “NAE-HPROBE-15.” [Online]. Available: https://www.newae.com/products-1/NAE-HPROBE-15

[86] TEKBOX, “TBWA2.” [Online]. Available: https://www.tekbox.com/product/tbwa2-wideband-rf-amplifiers/

[87] G. Ferré and A. Giremus, “LoRa physical layer principle and performance analysis,” in *25th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2018, Bordeaux, France, December 9-12, 2018*. IEEE, 2018, pp. 65–68.

[88] A. F. Molisch, *Wireless Communications*, 2nd ed. Wiley Publishing, 2011.

[89] Sigidwiki, “DominoEX.” [Online]. Available: https://www.sigidwiki.com/wiki/DominoEX

[90] P. Lindecker, “Technical aspects of RS ID and Call ID and use,” 2010. [Online]. Available: https://tapr.org/wp-content/uploads/DCC2010-RS-ID-Call-ID-F6CTE.pdf

[91] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” *IEEE Secur. Priv.*, vol. 9, no. 3, pp. 49–51, 2011. [Online]. Available: https://doi.org/10.1109/MSP.2011.67

[92] Tyetera, “TYT MD-UV 380.” [Online]. Available: https://www.tyt888.com/?mod=product_show&id=127

[93] RTL-SDR, “RTL-SDR.” [Online]. Available: https://www.rtl-sdr.com/about-rtl-sdr/

[94] D. Mantz, “RFAnalyzer,” 2014. [Online]. Available: https://github.com/demantz/RFAnalyzer

[95] Sigidwiki, “DTMF.” [Online]. Available: https://www.sigidwiki.com/wiki/Dual_Tone_Multi_Frequency_(DTMF)

[96] Y. Cao, Z. Chen, S. Li, and S. Wu, “Deterministic browser,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 163–178.

[97] M. Schwarz, M. Lipp, and D. Gruss, "Javascript zero: Real javascript and zero side-channel attacks." in *NDSS*, 2018.

[98] Z. B. Aweke, S. F. Yitbarek, R. Qiao, R. Das, M. Hicks, Y. Oren, and T. Austin, "ANVIL: Software-based protection against next-generation rowhammer attacks," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 743–755.

[99] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer.js: A remote software-induced fault attack in javascript," in *Detection of Intrusions and Malware, and Vulnerability Assessment - 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings*, ser. Lecture Notes in Computer Science, J. Caballero, U. Zurutuza, and R. J. Rodríguez, Eds., vol. 9721. Springer, 2016, pp. 300–321.

[100] M. Lipp, M. T. Aga, M. Schwarz, D. Gruss, C. Maurice, L. Raab, and L. Lamster, "Nethammer: Inducing rowhammer faults through network requests," *CoRR*, vol. abs/1805.04956, 2018.

[101] F. Chierchie, G. J. González, J. I. Morales, E. E. Paolini, J. Cousseau, and P. S. Mandolesi, "Baseband model for uniformly sampled RF-PWM," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–9, 2020.

[102] RTL-SDR.com, "RTL-SDR." [Online]. Available: https://www.rtl-sdr.com/update-rtl-sdr-transmitting-1270-mhz/

[103] "RTL TRX." [Online]. Available: https://github.com/tejeez/rtl-trx

[104] A. Cui, "Funtenna," 2015. [Online]. Available: https://github.com/funtenna

[105] O. Mattos and O. Weigl, "Turning the Raspberry Pi into an FM transmitter." [Online]. Available: http://icrobotics.co.uk/wiki/index.php/Turning_the_Raspberry_Pi_Into_an_FM_Transmitter

[106] "FM transmitter RPi3." [Online]. Available: https://github.com/PNPtutorials/FM_Transmitter_RPi3

[107] "Rasberry pirate radio FM transmitter." [Online]. Available: https://www.rtl-sdr.com/raspberry-pirate-radio-fm-transmitter/

[108] CNLhor, "Esp8266 analog broadcast television interface," 2016. [Online]. Available: https://github.com/cnlohr/channel3

[109] T. Yapo, "Experimental software-defined radio using a serial port," 2018. [Online]. Available: https://github.com/tedyapo/serial-port-sdr

[110] M. Seet, "PAL-Streamer," 2020. [Online]. Available: https://hackaday.io/project/171977-pal-streamer

[111] D. Spill, "Ridiculous radios," 2018. [Online]. Available: https://hackaday.com/wp-content/uploads/2018/12/Ridiculous-Radios-Supercon-2018.pdf

[112] J. Lipkowski, "Etherify - bringing the ether back to ethernet," 2020. [Online]. Available: https://lipkowski.com/etherify/

[113] F. Bellard, "Analog and digital TV (DVB-T) signal generation," 2005. [Online]. Available: https://bellard.org/dvbt/

[114] B. Kania, "VGASIGFM radio transmitter using a VGA graphics card," 2009. [Online]. Available: https://bk.gnarf.org/creativity/vgasig/vgasig.pdf

[115] P. Rudolph, "VGAtoIQBaseband," 2013. [Online]. Available: https://wiki.das-labor.org/w/VGAtoBaseband

[116] S. Markgraf, "Osmo-fl2k for using USB 3.0 VGA adapters as SDR transmitters," 2018. [Online]. Available: https://osmocom.org/projects/osmo-fl2k/wiki

[117] L. Cruz, "PiccoloSDR (WIP)," 2021. [Online]. Available: https://github.com/luigifcruz/pico-stuff/tree/main/apps/piccolosdr

[118] B. Newhouse, "One-bit BT," 2021. [Online]. Available: https://github.com/newhouseb/onebitbt

[119] M. Schulz, D. Wegemer, and M. Hollick. (2017) Nexmon: The C-based firmware patching framework. [Online]. Available: https://nexmon.org

[120] M. Schulz, "Teaching your wireless card new tricks: Smartphone performance and security enhancements through wi-fi firmware modifications," Ph.D. dissertation, Darmstadt University of Technology, Germany, 2018.

[121] M. Schulz, J. Link, F. Gringoli, and M. Hollick, "Shadow Wi-Fi: Teaching smartphones to transmit raw signals and to extract channel state information to implement practical covert channels over wi-fi," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2018, Munich, Germany, June 10-15, 2018*, J. Ott, F. Dressler, S. Saroiu, and P. Dutta, Eds. ACM, 2018, pp. 256–268.

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

[123] H. Lin, C. Lu, H. Tsai, and T. Kung, "The analysis of EMI noise coupling mechanism for GPS reception performance degradation from SSD/USB module," in *2014 International Symposium on Electromagnetic Compatibility, Tokyo*, 2014, pp. 350–353.

[124] H.-N. Lin, "Analysis of platform noise effect on performance of wireless communication devices," 2012.

[125] A. C. Scogna, H. Shim, J. Yu, C. Oh, S. Cheon, N. Oh, and D. Kim, "RFI and receiver sensitivity analysis in mobile electronic devices," in *DesignCon*, vol. 7, 2017, pp. 1–6.

[126] S. Moon, S. Kim, D. Kim, D. Yi, S. Lee, and J. Shin, "Analysis and estimation on EMI effects in AP-DRAM interface for a mobile platform," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, 2016, pp. 1329–1334.

[127] T. Nighswander, B. M. Ledvina, J. Diamond, R. Brumley, and D. Brumley, "GPS software attacks," in *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, T. Yu, G. Danezis, and V. D. Gligor, Eds. ACM, 2012, pp. 450–461.

[128] R. E. Navas, F. Cuppens, N. Boulahia-Cuppens, L. Toutain, and G. Z. Papadopoulos, "Physical resilience to insider attacks in IoT networks: Independent cryptographically secure sequences for DSSS anti-jamming," *Comput. Networks*, vol. 187, p. 107751, 2021.

[129] C. Shen and J. Huang, "Earfisher: Detecting wireless eavesdroppers by stimulating and sensing memory EMR," in *18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2021, April 12-14, 2021*, J. Mickens and R. Teixeira, Eds. USENIX Association, 2021, pp. 873–886.

APPENDIX A
BASICS OF RADIO COMMUNICATIONS

Complex notation: A generic signal $x(t)$ can be conveniently represented as the projection on the real axis of a complex vector with *amplitude* $a(t)$ and *angle* $\theta(t)$ (the angle can be expressed in terms of *frequency* f and *phase* $\phi(t)$):

$$\begin{aligned} x(t) = \mathcal{R}e\{a(t)e^{i(2\pi ft+\phi(t))}\} = a(t)\cos(2\pi ft+\phi(t)) \\ \theta(t) = 2\pi ft+\phi(t) \end{aligned} \tag{3}$$

Frequency domain and radio signals: A *baseband signal* $x_{bb}(t)$ with amplitude $a_{bb}(t)$ and angle $\theta_{bb}(t)$ carries the information that has to be transmitted. In the frequency domain it occupies the range $-B$ to B, where the *bandwidth* B is its maximum frequency component. This signal is shifted (*up-converted*) to a *carrier frequency* f_c, resulting into a *modulated* carrier:

$$x_{rf}(t) = a_{bb}(t)\cos(2\pi f_c t+\theta_{bb}(t)) \tag{4}$$

In the frequency domain, this signal occupies the range $f_c - B$ to $f_c + B$. In this case, the term bandwidth is often used to indicate the occupied spectrum $2B$.

Modulation: Any type of analog or digital amplitude and/or angle (frequency or phase) modulation can be implemented with a proper choice of the baseband signal. See Table V for a brief list. The modulated carrier x_{rf}(t) can be rewritten as the sum of its *in-phase (I)* and *quadrature (Q)* components:

$$\begin{aligned} x_{rf}(t) = I(t)\cos(2\pi f_c t) + Q(t)\cos(2\pi f_c t+\frac{\pi}{2}) \\ I(t) = a_{bb}(t)\cos(\theta_{bb}(t)),\ Q(t) = a_{bb}(t)\sin(\theta_{bb}(t)) \end{aligned} \tag{5}$$

Implementing a *quadrature modulator* (Equation 5) is often more convenient than implementing a *polar modulator* (Equation 4).

Superheterodyne: In this two-stage architecture, the baseband signal is first modulated on an intermediate frequency f_{IF}, and then further up-converted to the radio-frequency carrier f_c.

Discrete time and quantization: Digital software and hardware work with a discrete-time digital representation of signals, that is, a sequence of samples $x_s[i] = x(t = i/F_s)$, where F_s is the sampling rate. Each sample $x_s[i]$ is represented with a digital number that can only take a finite number of values (quantization levels).

Modems: In short, a modem takes the input bits to transmit, and generates the samples $x_s[i] = a_{bb}[i/F_s] * cos(2\pi f_{IF} i/F_s +$

TABLE V
MODULATION TYPES

Acronym	Name	Short Description
AM	Amplitude Modulation	Analog amplitude
FM	Frequency Modulation	Analog frequency
OOK	On Off Keying	Digital amplitude (on/off)
ASK	Amplitude Shift Keying	Digital amplitude
FSK	Frequency Shift Keying	Digital frequency
B-FSK	Binary FSK	Digital frequency (two values)
M-FSK	M-ary FSK	Digital frequency (M values)
IFK	Incremental FSK	Digital frequency
DTMF	Dual Tone Multi Frequency	FSK
GFSK	Gaussian FSK	Gaussian-filtered FSK
A-FSK	Audio FSK	FSK in the audio bandwidth
PSK	Phase Shift Keying	Digital phase
B-PSK	Binary PSK	Digital phase (two values)
QAM	Quadrature Amplitude Modulation	Digital phase (four values)
DSSS	Direct Sequence Spread Spectrum	Code to spread the spectrum
CSS	Chirp Spread Spectrum	Chirp to spread the spectrum
OFDM	Orthogonal Frequency Division Multiplexing	Orthogonal subcarriers
USB	Upper Side Band	Filter lower half band

$\theta_{bb}[i/F_s]$) of the intermediate frequency wave, that is then sent to the radio-frequency stage. $a[i] = a_{bb}[i/F_s]$ and $\theta[i] = \theta_{bb}[i/F_s]$ are chosen based on the bits to transmit and the specifications of the protocols. The simplest example is OOK with symbol duration T_{sym}. In this case, a bit '1' consists of the samples $x[i] = cos(2\pi f_{IF} i/F_s)$ for $i = [0, T_{sym}F_s]$, whereas a '0' is $x[i] = 0$ for $i = [0, T_{sym}F_s]$. Similar exaplanations exist for other modulations, and their C++ implementation is visible in Fldigi [56] and GNURadio [68] source code.

APPENDIX B
IMPLEMENTATION DETAILS

RF-PWM: Conversion from a sinusoidal IF carrier (modulated in amplitude/frequency/phase) to an RF-PWM square wave is straightforward following Equation 2. Figure 6 visualizes the process. Listing 1 shows our C++ implementation used in Fldigi-*Noise-SDR*.

Leakage generation: *Noise-SDR* requires a source of leakage (Section IV). Inspired by previous work on exfiltration [1], [3], [8], [44], [45] and Rowhammer [66], [67], we have implemented *Noise-SDR* on *x86-64*, *ARMv7-A*, *ARMv8-A*, and *MIPS32*. The code sequence (*leakyOperation*) used to trigger the leakage emission during the high periods of the RF-PWM square wave is shown in Listing 2, Listing 3, Listing 4, and Listing 5, respectively.

Leakage control: To generate the desired RF-PWM square wave, *Noise-SDR* simply alternates an operation with strong leakage (during the high period) with inactivity (during the low period). The timings of the square wave (high and low periods) are stored in a static array (filled by the RF-PWM stage) for fast access. The code that controls the square wave is shown in Listing 6. Timing measurements (*get_ns()*) that control the pulses are based on *clock_getttime*. The fundamental difference between this code and that provided in previous work is that duty-cycle, frequency, and phase of the pulses are controlled in a generic way following an RF-PWM square wave, to modulate an arbitrary sinusoidal signal on the first harmonic.

Time resolution, quantization levels, and bandwidth: *Noise-SDR* is implemented in software and therefore it works at discrete

```
while (i
   < outputBufferIndex && outputBuffer[i++] < 0) {
 }
 while (i < outputBufferIndex && len < EDGESIZE) {
  uint64_t j = 0;
  double a = outputBuffer[i];
  while (i + j < outputBufferIndex
      and outputBuffer[i + j] >= 0) {
   j++;
   if (outputBuffer[i + j] > a)
    a = outputBuffer[i + j];
  }
  while (i + j < outputBufferIndex
      and outputBuffer[i + j] < 0) {
   j++;
  }
  if (len < EDGESIZE - 1) {
   edges[len++] = (asin(a) /
        M_PI) * (uint64_t)(1e9 * j / samplerate);
   edges[
       len++] = (uint64_t)(1e9 * j / samplerate);
  }
  i += j;
 }
}
```

Listing 1. From a sinusoidal IF carrier (*outputBuffer*) modulated in amplitude/frequency/phase to the corresponding RF-PWM square wave timings (*edges*).

```
void stream(void) {
 _mm_stream_si128(&reg, reg_one);
 _mm_stream_si128(&reg, reg_zero);
}
```

Listing 2. *leakyOperation* for *x86-64* native code (inspired from [8], [44]).

```
static inline void ion_leak(void) {
 ion_user_handle_t ion_handle;
 ion_alloc(ion_fd
     , len, 0, (0x1 << chipset), 0, &ion_handle);
 ion_free(ion_fd, ion_handle);
}
```

Listing 3. *leakyOperation* for *ARMv7-A*/ (or *ARMv8-A*) native code (inspired from [66]).

```
__attribute__((naked)) \
void hammer_civac(uint64_t *addr) {
 __asm volatile("LDR X9, [X0]");
 __asm volatile("DC CIVAC, X0");
 __asm volatile("DSB 0xB");
 __asm volatile("RET");
}
```

Listing 4. *leakyOperation* for *ARMv8-A* native code (inspired from [67]).

time. In RF-PWM, the time resolution $T_{res} = 1/F_{res}$ at which the square wave is generated affects the resolution of amplitude, frequency, and phase of the first harmonic at f_0 (F_{IF} in *Noise-SDR*). The quantization levels (the possible discrete values) are:

$$\begin{aligned} f_0 &= \frac{F_{res}}{q}, q \geq 2 \\ \theta_k &= 2k\pi f_0 \frac{q}{F_{res}}, q \in \left[-\left\lfloor \frac{F_{res}}{2kf_0} \right\rfloor, \left\lfloor \frac{F_{res}}{2kf_0} \right\rfloor \right) \\ a_k &= \sin(k\pi q \frac{f_0}{F_{res}}), q \in \left[0, \frac{1}{2k} \frac{F_{res}}{f_0} \right) \end{aligned} \tag{6}$$

```
cnt++; // Followed by sleep during the low period
```

Listing 5. *leakyOperation* for *MIPS32* native code (inspired from [44]).

```
    int x = 0;
    uint64_t mid, reset;
    uint64_t start = get_ns();
    while (x < len) {
      mid = start + edges[x++];
      reset = start + edges[x++];

      // High period
      while (get_ns() < mid) {
// basic operation
#ifdef LEAKY_OP_CIVAC
        hammer_civac(address);
#elif LEAKY_OP_ION
        ion_leak();
#elif LEAKY_OP_STREAM
        stream();
#elif LEAKY_OP_CNT
        cnt++;
#endif
      }
      // Low period
#ifdef LEAKY_OP_CNT
      clock_sleep_trap(clock_port
          , CLOCK_MONOTONIC, reset / NSEC_PER_SEC
          , reset % NSEC_PER_SEC, &remain);
#else
      while (get_ns() < reset) {};
#endif
      // Reset the starting point
      start = reset;
    }
```

Listing 6. RF-PWM square wave control.

Note that the maximum achievable f_0 frequency is $F_{res}/2$.

A numerical example: Let us consider a 3X PSK250R signal that occupies a total bandwidth $2B = 950$Hz. We chose sampling frequency $F_s = 80$kHz. Therefore, our device should have a time resolution $T_{res} = 1/F_{res}$ in the order of tens of µs, at least. From now on, we can consider $F_{res} = F_s = 80$kHz. To avoid any overlap with the baseband component, $f_0 = F_{IF}$ should be bigger than 950 Hz. Because of the sampling theorem, $F_{IF} + B$ should not exceed $F_s/2$, so F_{IF} cannot exceed 39.525 kHz. We chose, for example, $F_{IF} = 5$kHz. In this case, there are $80\text{kHz}/5\text{kHz} = 16$ samples per period. So there are 8 possible values of the duty-cycle between 0 and 0.5 (in RF-PWM the duty-cycle cannot be higher than 0.5 because of the pre-distortion $\delta(t) = \frac{asin(a(t))}{\pi}$). This corresponds to having a resolution of $log2(8) = 3$ bits on the amplitude. Since there are 16 possible values for the phase of the pulse, the phase resolution is 4 bits.

Advantage over PWM: For simplicity, let us focus on the unmodulated intermediate carrier $x(t)$ at $F_{IF} = 5$kHz. As depicted in Figure 15, PWM approximates $x(t)$ with the baseband component of a square wave at $f_{pwm} >> 2f_{IF}$. RF-PWM approximates $x(t)$ with the fundamental component of a square wave at $f_{RF-PWM} = F_{IF}$, exploiting the fact that $x(t)$ is a band-pass signal, too. Consequently, PWM requires a higher time resolution to generate a square wave at higher frequency. This would become quickly impractical using leakage. An in-depth comparison of the two techniques is outside the scope of this paper, and can be found in [30].

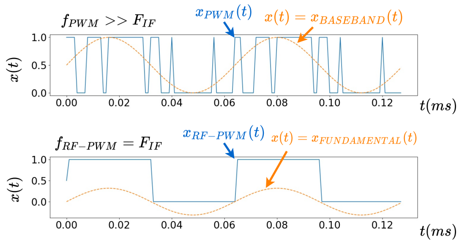


Fig. 15. Comparison between PWM (top) and RF-PWM (bottom).

TABLE VI
TESTING THE PRESENCE OF CONTROLLABLE LEAKAGE ON SMARTPHONES

Model	*ARM*	DRAM	*f* (MHz)	Harmonics n
Innos D6000	V8-A	LPDDR3	400	$1-4$
Nokia 3.1	V8-A	LPDDR3	13.56	7 (NFC)
Samsung Galaxy A30S	V8-A	LPDDR4	1794	1
Samsung S7 Exynos	V8-A	LPDDR4	1794	1
Samsung Galaxy S5 Mini	V7-A	n.a.	200	1-11, 13-19,26
Samsung M31	V8-A	LPDDR4	1794	1 (rare)
Samsung Galaxy J7	V8-A	LPDDR3	-	-
Samsung Galaxy Young	V7-A	n.a.	-	-
Sony Xperia C5	V8-A	LPPDR4	400	1-11
Sony Xperia X	V8-A	LPDDR3	-	-
Motorola Moto E6S	V7-A	LPDDR3	400	1,2
Google Nexus 5	V7-A	LPDDR2	200	$1-5$,8,12 16,20,24
Google Pixel XL	V8-A	LPDDR4	-	-
Google Pixel 2	V8-A	LPDDR4	-	-
Wiko Fever	V8-A	LPDDR3	-	-
Huawei P8 Lite	V8-A	LPDDR3	-	-
Huawei P10	V8-A	LPDDR4	-	-
Huawei P8 SE	V8-A	LPDDR3	-	-
OnePlus 7 Pro PE	V8-A	LPDDR4	-	-

APPENDIX C
ADDITIONAL EXPERIMENTAL DATA

Controllable leakage on additional devices: We studied the existance of controllable leakage on many smartphones. In particular, we tested the existence of leakage that can be modulated with simple tunes and chirps by unprivileged software, which is a necessary requirement for *Noise-SDR* and other Soft-TEMPEST techniques. As shown in Table VI, in many devices one or more harmonics of an electromagnetic leakage can be controlled by performing memory accesses. The Samsung Galaxy A30S is noticeable because it is able to transmit a GLONASS C/A code at 800 MHz as we explained in Section IX. In addition, we also analyzed a laptop (Dell Inspiron 14R 5437). It has a DDR3 DRAM with SSC at 800 MHz. We tested NBFM, PSK31, RTTY45, MFSK128, and Olivia on one SSC subharmonic at 800 MHz, with the antenna in proximity to the device.

Modulating an intentional radio carrier: The Nokia 3.1 in Table VI is particularly interesting. Since we did not observe any leakage from DRAM, we might think that exfiltration is not possible. However, we noticed that DRAM accesses modulate the radio carrier intentionally emitted at relatively high power by the NFC reader when close to a tag. This is probably due to some coupling effect on the smartphone platform that hosts both the DRAM and the NFC reader. As a result, we were able to use *Noise-SDR* to transmit, for example, with PSK31.