BlueScream: Screaming Channels on Bluetooth Low Energy

Pierre Ayoub, Romain Cayre, Aurélien Francillon, Clémentine Maurice

BlueScream: Screaming Channels on Bluetooth Low Energy

Pierre Ayoub[†], Romain Cayre[†], Aurélien Francillon[†], and Clémentine Maurice[‡]
[†] *EURECOM*
Sophia-Antipolis, Biot, France
Email: firstname.lastname@eurecom.fr
[‡] *Univ. Lille, CNRS, Inria*
Lille, France
Email: firstname.lastname@inria.fr

***Abstract*—In recent years, a class of wireless devices has been demonstrated to be vulnerable to a new side-channel attack called Screaming Channels. This attack exploits distant electromagnetic side channels up to a few meters, when a coupling occurs between the digital activity and the radio transceiver of a system. This can happen in mixed-signal chips, where both digital and analog parts reside on the same silicon die. Until now, the Screaming Channel attack has mainly been demonstrated using custom firmware used in laboratory conditions or simple protocols – *e.g.*, Google Eddystone.**

In this paper, we evaluate an end-to-end Screaming Channel attack on a real-world firmware running on an off-the-shelf and popular Bluetooth Low Energy stack. By doing a careful analysis of Bluetooth Low Energy to find how to make the victim device leak, our results show that an attacker can manipulate the protocol such that a Screaming Channel leak happens during a radio transmission. Finally, we conducted one successful full-key recovery attack against AES using instrumented firmware and a partial-key recovery using stock firmware.

1. Introduction

Electromagnetic eavesdropping on computer applications is a 70-year-old problem in the military world, as illustrated by the TEMPEST specification from the NSA [1]. In the public domain, it has been known for at least four decades, popularized as Van Eck Phreaking through the work of Van Eck, which eavesdropped video display through their electromagnetic radiation (EMR) [2]. On the other hand, side-channel attacks are a threat to cryptosystems and have been known for at least two decades in the public domain [3]. However, physical side channels often require that the attacker is in close physical proximity with the victim target to perform his measurement, limiting their practical impact. It is especially true for electromagnetic side-channels [4] that use signals recorded through near-field (NF) probes placed at a few millimeters of the victim device.

However, the Screaming Channel attack discovered in 2018 by *Camurati et al.* [5] breaks the limitations of traditional electromagnetic (EM) side-channels. Indeed, this paper demonstrates that EM side-channels can be conducted in the far-field (FF) region, using antennas from a few centimeters to some meters, because of a coupling phenomenon between the digital and the analog part inside a mixed-signal chip. When the transceiver of the analog part is performing a radio transmission, we can observe a side-channel leakage from the digital part in the spectrum of the radio transmission. This side-channel leakage first modulates the carrier signal of the radio transmission and then is amplified and broadcasted with the carrier by the radio transceiver.

A more in-depth analysis of the Screaming Channel leakage [6] provided a better understanding of the leakage – *e.g.*, its distortions or the portability of the templates. While this analysis performs a Screaming Channel attack on Google Eddystone [7], a simple and discontinued protocol [8], there is no prior work tackling the challenges of using Screaming Channels to target complex protocols involving multiple layers.

Filling this gap is mandatory to consider Screaming Channels attacks a realistic threat. Being able to conduct an end-to-end attack at a distance from the wireless device would be critical since side-channel attacks are often not taken into account in their threat model and are difficult to mitigate.

In this paper, we tackle challenges a complex and multi-layered protocol raises by attacking Bluetooth Low Energy (BLE), a widespread protocol for secure wireless communications. More precisely, we attack the BLE key derivation mechanism – also known as the re-keying mechanism, a standard and fundamental security feature of the protocol. The impact of breaking this mechanism is critical since it would allow the recovery of the Long Term Key (LTK) independently of the pairing method used for its generation and negotiation, including LE Secure Connections.

Research question Is the Screaming Channel attack a threat for Bluetooth Low Energy, despite the limitations imposed by such complex and multi-layered protocols?

Challenges A specificity of the Screaming Channel leakage is that the transceiver of the target device must transmit data during the cryptographic operation, *i.e.*, emits radio

waves to broadcast the leak over the air. Depending on the protocol, this period can be short and may happen independently of the cryptographic activity targeted by the side-channel attack. Moreover, side-channel attacks usually involve collecting thousands of traces, while the collection rate is limited due to protocol constraints.

Contribution 1 In Section 7, we analyze how to manipulate the BLE protocol from an attacker perspective to find how to make the victim device "scream", *i.e.*, execute critical cryptographic operations during a radio transmission, and how to accurately identify this timing. In addition, we developed a framework that allowed us to collect and process numerous traces over several days.

Contribution 2 In Section 8, we evaluate the Screaming Channel attack performance against the Long Term Key (LTK) of BLE under various conditions and firmware. We demonstrate that the protocol can be exploited to fully break the AES key at the condition of reducing the radio noise of the environment during the collection process in laboratory conditions, but that this noise is problematic for an attacker in real conditions.

Contribution 3 In Section 9, we present the result of experiments using several firmware modifications, suggesting which hardware element is implicated in the Screaming Channel leakage and how it can influence its exploitation.

Our framework is published as open-source software[1] and our datasets are available as open data [9].

2. Related Work

Side-channels on Bluetooth. Concurrently to our work, *Cao et al.* [10] analyzed the session key derivation mechanism (also known as re-keying) of the BLE protocol to assess its side-channel resistance. *Cao et al.* show a conventional EMR side-channel attack, with an EM probe close to the microcontroller, at low frequency, without the radio activated. They compare the performance of traditional correlation attacks against deep learning-based attacks and do not address any of the challenges of a remote attack from the radio channel (synchronization, triggering, etc.).

Screaming Channels. Published in 2018 by *Camurati et al.* [5], the discovery of the Screaming Channel leakage describes how the leak signal propagates through the targeted mixed-signal chip and demonstrates an attack on a custom firmware, *i.e.*, constantly running cryptographic operations while transmitting modulated random packets over the air. It concentrates on how far the leak can be exploited in an ideal scenario, increasing the range of traditional side-channel attacks from dozens of centimeters to more than 10 meters. The same authors proposed a more advanced analysis of the leakage in 2020 [6]. This paper evaluates the non-linearity of the best leakage model and proposes to use new statistical tools to increase the attack performance.

Attack on Google's Eddystone. A preliminary assessment of the risk induced by the Screaming Channels attack has been conducted by *Camurati et al.* [6] by exploiting Google's Eddystone protocol. Eddystone [7] is an application-level protocol on top of BLE used for beacons, *i.e.*, small IoT objects broadcasting information (*e.g.*, a URL). First, *Camurati et al.* [6] only assessed the Screaming Channel attack using a synthetic setup with an RF coaxial cable. Second, while Eddystone uses BLE as a transmission layer, the security of the protocol is implemented at the application level using Bluetooth services, which implies that: 1) It is a non-standard and discontinued [8] protocol specific to the beacon use case. 2) The cryptographic operation is triggered using a *Characteristic Read*, an operation built over the application layer of BLE that does not involve the native security mechanisms of BLE, facilitating the exploitation. In contrast, in this paper, we aim to overcome those limitations by using a realistic setup for our evaluation and performing an end-to-end attack. Regarding the attacker receiver, we use an antenna instead of an RF coaxial cable. Regarding the attack target, we attack the key derivation mechanism of the BLE instead of the Eddystone application-level implementation.

Further extensions of Screaming Channels. Several researchers extended this preliminary work in different directions. *Wang et al.* [11] tried to increase attack performance by using a deep-learning-based approach applied to the original custom firmware, allowing to decrease the required number of traces. *Guillaume et al.* [12] elaborated a new method to take up the triggering challenge by introducing Virtual Triggering, which aims at finding leakage related to cryptographic operations contained in a trace without prior knowledge about it. This is a more difficult challenge with Screaming Channel attacks compared to traditional side channels. The same authors also explored at which frequencies the leak is detectable and exploitable [13] and concluded that the latter is present at non-harmonic frequencies and is strong enough to conduct attacks, sometimes as powerful as using harmonic frequencies. *Fanjas et al.* [14] also introduced a new real-time triggering method called Synchronization by Frequency Detection (SFD) based on *Camurati*'s triggering but implemented on an FPGA. Finally, *Danieli et al.* [15] tried to exploit the Screaming Channel leakage in other ways than performing cryptographic side-channel attacks. For instance, this paper exploits the direct readout of the Screaming Channels effect using various coupling sources (RAM, SPI, JTAG, NFS) to recover non-encrypted data.

We observed that none of these papers evaluated the Screaming Channels attack on a complex multi-layered protocol such as BLE.

3. Background

3.1. Side-channels

Different channels. A side-channel attack is an attack against a security system using information originating from the interaction between the security system and its environment. Hence, it allows for an attacker to recover the secret

1. https://github.com/pierreay/screaming_channels_ble

key used during a cryptographic operation (CO), *e.g.*, the encryption of plaintext or signing data using a secret key. Side-channel attacks originate from cryptanalysis, popularized by *Kocher et al.*, by exploiting the time taken by the attacked cryptographic operation [3] or its power consumption [16]. Various classes of side-channels were discovered and exploited [**?**], *e.g.*: time [3], power consumption [16], acoustic waves [17], or the optical photonic emanations [18].

Electromagnetic radiation (EMR) has been identified to be a threat against cryptosystems by *Quisquater et al.* [4] with a similar impact as power consumption. Numerous algorithms have been developed to exploit this EMR measurement, the first ones being Simple EM Attack (SEMA) and Differential EM Attack (DEMA) [19] which are analogous to Simple Power Attack (SPA) and Differential Power Attack (DPA) [16]. EM side-channel have been systematically studied by *Lavaud et al.* [20], describing multiple classes depending on the emanation origin (illumination, mixing, radiation, coupling). In Screaming Channels publications [5], [6], *conventional leakage*, which is radiative EMR recorded using a near-field (NF) probe, is clearly distinguished from *Screaming Channel leakage*, which is recorded using an antenna in the far-field (FF) after the leakage has been transmitted by a radio transmitter (*e.g.*, due to mixing between the analog and the digital part of the chip) [5].

Attack steps. From a high level, a side-channel attack is conducted in two steps: 1) creating a model of the leakage, either theoretically or by collecting training measurements to build a template, 2) collecting attack measurements (called "traces") and computing correlations between predictions from the model and the actual measurements. Those collected traces are real values representing the measured physical quantity over time. Searching for the key candidate with the best correlation, the side-channel algorithm may lead to a full key recovery if the attack is successful. Often, the Pearson correlation coefficient (ρ or r) is used to compute the correlations.

AES and side-channels. AES is a block cipher, *i.e.*, it works on a block of data to encrypt plaintext into a ciphertext – or the opposite for decryption [21]. It can be used with different key sizes (*e.g.*, 128 bits), different operation modes (*e.g.*, ECB, where each block is processed separately), and different numbers of iterations called "rounds" (*e.g.*, 10 rounds for the 128-bit key size). Each round repeats all or a subpart of the following operations: 1) "AddRoundKey", which XORs each column of the internal state with a word from the key scheduling 2) "SubBytes", which apply the S-Boxes to each byte of the internal state, being the only non-linear operation 3) "ShiftRows", which cyclically shifts the last three rows in the internal state 4) "MixColumns", which operates on the internal state column-by-column. AES has been a target for side-channel attacks for two decades, *e.g.*, with timing attacks targeting the T-Table implementation [22], with power attack (DPA) against the first "AddRoundKey" operation [23], or with power attack (SPA) against the key scheduling algorithm [24].

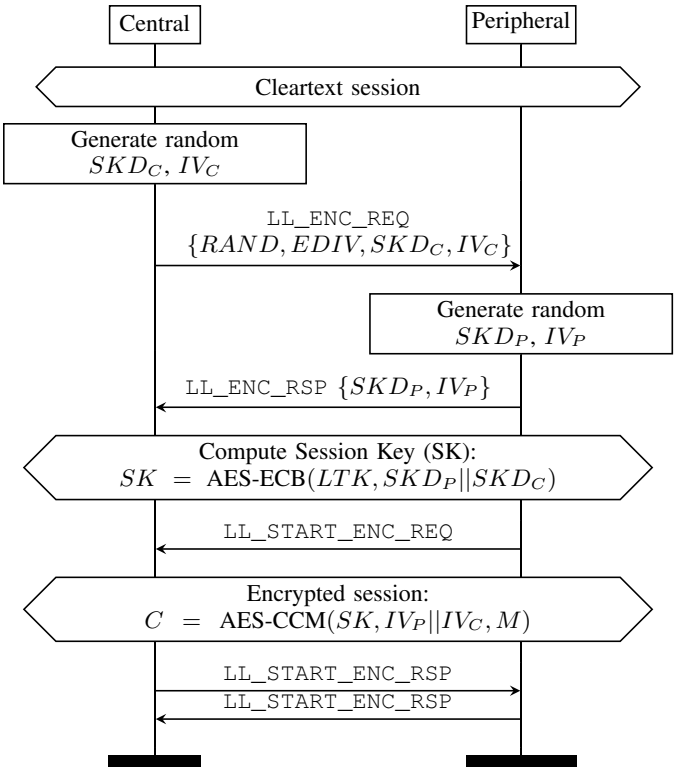

Figure 1. Session key derivation in BLE. The ciphertext C will be sent over the air to transmit the message M.

3.2. Bluetooth Low Energy

Pairing. The pairing procedure is used to exchange a set of keys (including an encryption key) that are generally stored for future use. The stored encryption key is known as the Long Term Key (LTK). The specification defines two pairing methods [25, p. 1626]:

- "LE Legacy Pairing" method will negotiate a common key (Short-Term Key (STK)) between the two devices to establish an encrypted session. This encrypted session will be used to exchange a set of keys, including the LTK.
- "LE Secure Connections" method uses the Elliptic Curve Diffie-Hellman (ECDH) algorithm to agree on a shared LTK. In this case, the LTK is directly used to establish the encrypted session to exchange the other keys securely.

In this work, we focus on a later stage of the protocol, and our approach is independent of the paring method in use.

Session key derivation. The pairing procedure is only needed the first time the two devices are used together; for each session (connection of the devices), a new session key will be generated using the Long Term Key (LTK). Following the pairing, both devices will also keep two values, `RAND` and `EDIV`, used to retrieve the LTK from the security database storing all LTK from the previous pairing. The two paired devices can now establish a secure connection with link-layer encryption by computing

a common Session Key (SK) if they share a common LTK (Figure 1 [25, p. 2957]). The SK is derived by performing an AES in ECB mode using the LTK as the key and a public random nonce as the plaintext. This random nonce, also known as the Session Key Diversifier (SKD), is composed of two concatenated random numbers respectively generated by the Central (SKD_C) and the Peripheral (SKD_P). SKD_C and SKD_P are transmitted over-the-air in plaintext using the control protocol data units (PDUs) `LL_ENC_REQ` and `LL_ENC_RSP`. After the SK computation, the two devices perform a three-way handshake to start the encrypted session by sending empty control PDUs. The Peripheral sends an unencrypted `LL_START_ENC_REQ`, the Central and the Peripheral responds with an encrypted `LL_START_ENC_RSP` and `LL_START_ENC_RSP`, respectively.

Radio transmission. A standard BLE communication [25] is divided into connection events (CEs) following the time division duplex (TDD) mechanism. During a CE, the node initiating the connection called the Central (formerly called "Master"), sends at least one packet to the second node involved in the connection, the Peripheral (formerly called "Slave"). After a short inter-frame space of 150µs, the Peripheral sends the response packet. For synchronization purposes, this simple communication pattern is repeated for each CE, whether data needs to be transmitted or not – *e.g.*, packets with an empty data field are therefore often transmitted. A field called More Data (MD) bit can be set in the protocol data unit to indicate that the CE should not end after the current PDU because a device (Central or Peripheral) has more data to transfer during the current connection event. If the MD bit is set to 0, no further transmissions will occur during the CE. Otherwise, the devices repeat the communication pattern until the MD bit is not set anymore or until the CE's time window is exhausted. Between each CE, the transmission frequency is changed according to the channel hopping algorithm. The radio of both the Central and the Peripheral are then turned off to be reconfigured for another frequency. The duration of a CE is equal to connInterval $= H * 1250$µs, with the hop interval $H \in [6; 3200]$. The next frequency, *i.e.*, the next channel, is chosen between all standard channels defined from the specification. The channel selection can be influenced by the channel map, a 5-byte value where every bit indicates if a channel is blacklisted or not. The Central transmits the initial channel map during the connection initiation but can be modified at any time during the connection, allowing a fast adaptation to changes in the radio environment (*e.g.*, interference).

Frequency hopping. A Bluetooth communication (excluding advertisement) can be located between 2.404GHz and 2.478GHz on the spectrum due to the frequency hopping (FH) algorithm. Intercepting communication from such a protocol can be challenging because consumer-grade radio equipment cannot monitor a large enough band in the spectrum. When the channel hopping sequence can be neither modified nor known *a priori* by an attacker, a possible solution is to rely on software-defined radio (SDR) flexibility and FPGA-based acceleration in order to decode a wideband region (*e.g.*, 200 MHz) in real-time [26].

4. Threat model

We consider an attacker that aims to obtain the LTK used to generate key material for secure communications between two paired devices: a victim Peripheral and a victim Central. The attacker does not need to monitor the pairing phase, and the attack is independent of the pairing method used (LE Legacy Pairing or LE Secure Connections). The attacker is able to sniff and inject packets at the physical layer. More precisely, the attacker will need to:

Sniff a BLE encryption initiation (`LL_ENC_REQ`) packet between the Central and the Peripheral, to obtain the parameters sent in the clear from the Central (addresses, `RAND` and `EDIV`),

Interrupt a previous connection. While not strictly required, this makes the attack faster by forcing the Central and Peripheral to establish a new connection and collect parameters, instead of waiting for a new connection.

Impersonate the victim Central (using the above parameters), which was previously paired with the victim Peripheral,

Initiate a connection and interact with the Peripheral victim through the BLE protocol,

Repeat session key establishment on the Peripheral side using this connection, which triggers an encryption using the LTK as a key,

Record a radio signal over the air during the encryption, at the physical layer, (*e.g.*, using an antenna connected to a software-defined radio (SDR)).

In our attack, we used profiled algorithms, where an attacker can build a template to learn the leakage model using a similar device that can be instrumented prior to the attack. Note that this is not strictly required when exploiting screaming channels because non-profiled attacks may also be performed. Once the attack is successful (LTK is recovered), the attacker can decrypt data sent over the air during BLE communications between the two victim devices – for previous and future connections [27], provided that the packet traces can be collected. According to the state of the art in BLE security, these capabilities are realistic and can be performed with affordable open-source tools [28].

5. Attack overview

Figure 2 presents a high-level overview of our attack strategy. In the first offline phase, steps may occur at different locations and times before the attack:

1) Template creation The attacker prepares a template using a different device but a similar model to the victim Peripheral [6]. Under controlled conditions, he collects a high number of traces corresponding to the session key derivation, where an AES encryption with a controlled random input (LTK, SKD) is performed by the victim Peripheral (see Figure 1).

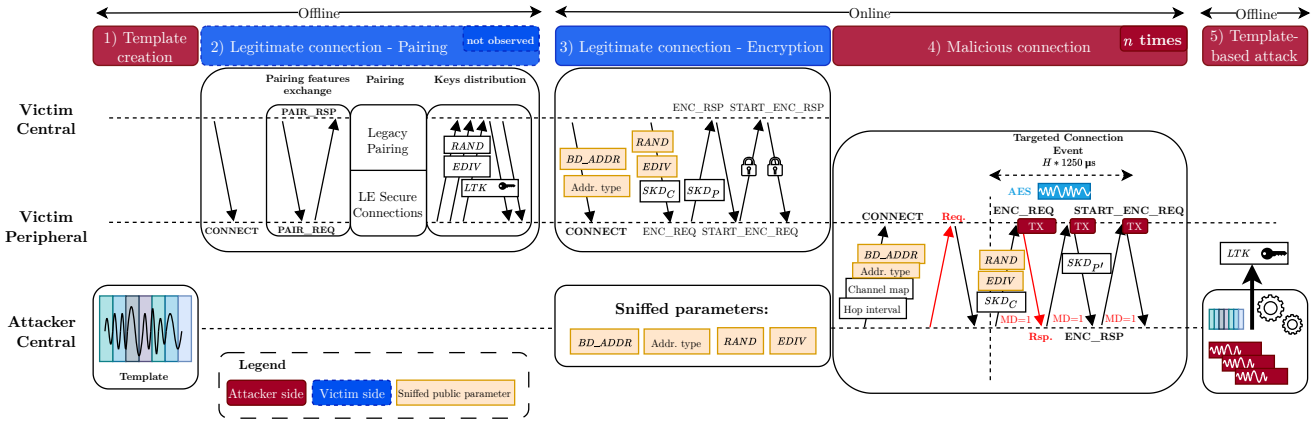

Figure 2. Attack overview.

2) Legitimate pairing The victim Peripheral and victim Central perform a pairing in secure conditions, agreeing on a shared Long Term Key (LTK) and its associated parameters (`RAND` and `EDIV`). The attacker is not present during this phase.

In the second online phase, the attacker needs to be active and in the victim's proximity during the following steps:

3) Legitimate connection sniffing After the pairing phase, the attacker passively observes one legitimate connection and encrypted session establishment between the victim Central and the victim Peripheral. From the `CONNECT` message, he collects the victim's Central Bluetooth address (`BD_ADDR`) and the address type – used later on to impersonate the victim Central. From the `LL_ENC_REQ`, he collects the `RAND` and `EDIV` associated with the LTK in use – used later on to trigger a similar session key derivation on the victim Peripheral without knowledge of the LTK.

4) Malicious connections The attacker repeatedly performs n connections by: 1) impersonating the victim Central, 2) injecting controlled channel map and hop interval, 3) starting a BLE procedure requiring a response to generate an interleaved procedure, 4) triggering the session key derivation by initiating the encrypted session establishment, 5) setting the MD bit of every transmitted PDU to 1. This protocol manipulation maximizes the victim Peripheral radio transmission duration, increasing the probability that the session key derivation occurs concurrently with the radio transmission (TX) period, therefore generating a Screaming Channel leakage. Without this protocol manipulation and using only a passive attacker device, the probability of having a Screaming Channels leakage would be weak or close to zero. Simultaneously, the attacker records the electromagnetic radiation at the correct frequency, leveraging the knowledge of the channel map in use. We present the technical details of this protocol manipulation in Section 7.

Lastly, once enough traces have been collected, the third offline phase consists of:

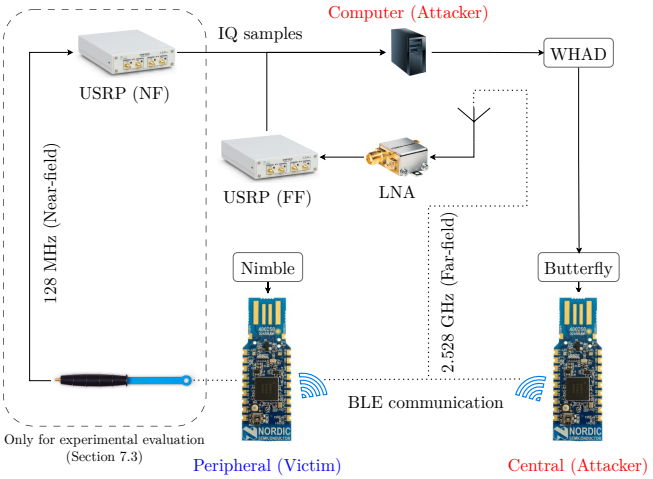

Figure 3. The experimental setup.

5) Template-based attack The attacker performs a template-based attack to recover the LTK from the collected traces using the template previously created. If needed, the attacker can perform a key enumeration to brute-force the remaining bits incorrectly recovered by the side channel. This is possible because the attacker can capture encrypted but predictable traffic, which can be used as an oracle during key enumeration. If the attack is successful, the full LTK is retrieved.

6. Experimental Setup

Figure 3 illustrates our hardware and software experimental setup in laboratory conditions (without the Central Victim described in Section 5). To characterize the impact of the protocol manipulation in Section 7, the full setup, including the USRP used for near-field (NF) measurements, was used. To evaluate the attack in a realistic scenario and its performance in Section 8, the setup excluding the USRP NF was used (*i.e.*, only including the USRP used for far-field (FF) measurements).

TABLE 1. FIRMWARE USED THROUGH THE PAPER.

Name	Code	Encryption nb.	AES inputs	Radio packets
F_{custom}	Custom C	Arbitrary	Controlled	Dummy
F_{instru}	NimBLE	Arbitrary	Controlled	Real BLE
F_{default}	NimBLE	1	Known only	Real BLE

Hardware. We use two software-defined radios capable of recording up to 56MHz of bandwidth (USRP B210 [29]). The USRP NF, only used in Section 7, is connected to a TekBox TBPS01 near-field (NF) probe [30] placed at 1cm of the target SoC and tuned at the 2nd harmonic of the CPU clock ($f_{NF} = 2 * f_{\text{clock}}$). The USRP FF is connected to an antenna placed in the far-field (FF) (typically, more than 30cm) from the target and tuned at the carrier frequency added to the previous frequency $f_{FF} = f_{\text{carrier}} + f_{NF}$. Typically, $f_{\text{clock}} = 64\text{MHz}$ for the CPU clock of the nRF52832 and $f_{\text{carrier}} = 2.420\text{GHz}$ for the Bluetooth channel 8. Depending on the experiment, we used an omnidirectional [31] or a directional [32] antenna to increase the leakage gain. A low-noise amplifier (LNA) from Mini-Circuits [33] is plugged between the Wi-Fi antenna and the SDR, maximizing the signal-to-noise ratio (SNR) of our recorded signal. The two SDRs are connected to a standard desktop computer through USB 3.0, sending raw I/Q during the recording. The attacker's dongle is a Nordic Semiconductor nRF52840 dongle, while the victim's device is a Nordic Semiconductor nRF52832 development kit (PCA10040), a well-known target for Screaming Channels analysis.

Software. The computer runs our custom Python instrumentation library built on the WHAD [34] framework, which controls the attacker's dongle. The attacker's dongle (nRF52840) is running our modified version of ButteRFly [35], a BLE firmware initially developed for the InjectaBLE attack [36] allowing to accurately inject link-layer traffic.

Firmware variations. During our evaluations, we used 3 firmware for the victim target depending on our needs presented in Table 1. F_{custom} is a custom firmware built upon homemade C code for experimental purposes. F_{instru} and F_{default} are based on the Peripheral example firmware from NimBLE [37], a public BLE open-source stack developed by Apache for the Mynewt RTOS [38]. This BLE stack provides AES through either: 1) a hardware implementation if the SoC implements it 2) a software implementation using the TinyCrypt [39] library otherwise. In our evaluation, we use the software AES since its leakage is stronger than that of the hardware AES, and the focus of the paper is not to assess the difficulty of attacking hardware-based implementations. Let us highlight that, to the best of our knowledge, no prior work managed to perform a successful Screaming Channels attack on a cipher in hardware. F_{instru} is an instrumented firmware built upon the NimBLE stack for evaluating the attack in favorable conditions to the attacker. F_{default} is a stock firmware built upon the NimBLE stack for evaluating the attack in realistic conditions. All radio transmissions use the GFSK modulation scheme whenever the packets are dummy or real BLE.

7. Bluetooth Low Energy Manipulation

7.1. Challenges

Three fundamental requirements of the Screaming Channel attack become challenging when using a complex protocol instead of custom firmware.

Challenge 1: Transmitting radio signals while the cryptographic operation is performed. The victim must transmit radio signals while the cryptographic operation is performed. Otherwise, the radio transmission is off, and no data is leaked over the radio. However, both the encryption and the transmission are short (order of $100\mu\text{s}$) and may happen in sequence, one after the other. Indeed, the transmission is a power-consuming operation and will be made as short as possible. Relying on the chance that both operations occur simultaneously significantly reduces the probability of success of the attack and adds a significant time overhead.

Challenge 2: Recording at the correct time. Second, the attacker has to collect data at the right time, as the victim is leaking information only for a short duration. Contrarily to a custom firmware designed to highlight the effect itself [5], on a real protocol, the attacker cannot artificially activate the radio transmission, repeat the cryptographic operation or use an artificial triggering mechanism (like a GPIO pin indicating that the encryption started).

Challenge 3: Recording at the correct frequency. Third, the attacker has to record the signal at the right frequency, as the frequency of the physical channel used by the protocol may not be constant. This is rendered difficult because 1) the leakage can be spread over numerous frequencies 2) the leakage can be impacted by interfering transmitters 3) the rapid frequency hopping of BLE.

Solutions overview. Therefore, we developed a framework to solve those challenges by leveraging the ButteRFly firmware on the attack device presented in Section 6. This attack device can control low-level parameters of the Bluetooth communication, and, while staying fully compliant with the BLE protocol, allows us to finely control the victim device behavior. Using those low-level messages, we can indirectly force the victim to increase the duration of the TX, making it transmit radio messages while performing encryption and expose a Screaming Channel leakage.

7.2. Methodology

Challenge 1: Transmitting radio signals while the cryptographic operation is performed. According to the Bluetooth specification [25], we listed a set of parameters that may influence the TX timing or increase the TX duration. For comparison purposes, we measure that the AES took approximately $250\mu\text{s}$ to fully execute on our target board – using AES-ECB from TinyCrypt provided by Apache Mynewt. However, the *SubBytes* operation – which

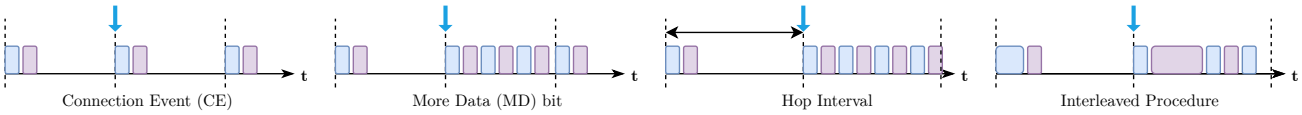

Figure 4. Illustration of BLE parameters impact.

is the side-channel target – took only 10µs to execute. We used the following parameters, illustrated in Figure 4:

Connection event (CE) Defining a precise CE at which the attacker will send the `LL_ENC_REQ` PDU allows influencing the cryptographic operation execution in time.

More Data (MD) bit We used this bit indicating that more data needs to be sent to increase the number of exchanges between the attacker Central and the victim Peripheral during a single CE, hence, without a radio reconfiguration. This bit alone only allows us to send more data, and it does not send the data in itself. We set the MD bit when sending our `LL_ENC_REQ` PDU while staying compliant with the specification.

Hop Interval Increasing this parameter allows increasing the CEs duration. With the MD bit set to 0, this parameter modifies the interval between two subsequent TX, *i.e.*, one radio reception (RX) slot and one TX slot from a Peripheral perspective. On the contrary, with the MD bit set to 1, this parameter increases the number of RX and TX cycles that can fit inside the window of one CE, which is desirable to prevent the radio reconfiguration during the cryptographic operation. We set the Hop Interval when sending the `CONNECT_IND` PDU compliant with the specification.

In the terminology used by the Bluetooth Core Specification, a Procedure is a sequence of packets, including requests and responses. We use the term "interleaved procedures" when sending multiple procedures simultaneously, resulting in interleaved request-response patterns. Having the MD bit set to 1, we sent interleaved procedures to increase the TX time during the system activity of the victim Peripheral. Before sending the request that will trigger the cryptographic operation, we sent other dummy requests that will force the Peripheral to send responses back. The choice of the dummy request can be important because some requests require larger responses than others, increasing the TX duration. The goal of this strategy is to increase the TX time of the Peripheral during its system activity, including the cryptographic operation.

Challenge 2: Recording at the correct time. The cryptographic operation occurs between two specific PDU: the `LL_ENC_REQ` PDU sent by the Central and the `LL_START_ENC_REQ` PDU sent by the Peripheral. `LL_ENC_REQ` identifies the paired devices and provides the Central random value (SKD_C), required to compute the Session Key. On the other hand, the `LL_START_ENC_REQ` needs the Session Key as the link is encrypted [25, p.2843, p. 2845]. Our attack firmware can monitor the connection, report received packets, and report parameter values like the current CE number and conditionally execute a function with low latency. We leveraged those features to send the `LL_ENC_REQ` PDU at a chosen CE, start the recording at another specified CE, and stop it when the `LL_START_ENC_REQ` PDU has been received. This guarantees the recording of potential Screaming Channel leakage during the Session Key derivation.

Challenge 3: Recording at the correct frequency. BLE uses frequency hopping, which makes it hard to record at a frequency corresponding to the leakage. The protocol allows for a Central to specify a channel map inside `ChM` field of the `CONNECT_IND` PDU [25, p. 2688] sent during the connection establishment. By setting the channel map to `0x300`, we can force the Peripheral to only use channels 8 and 9, corresponding to frequencies 2.420GHz and 2.422GHz. This technique was already used in previous work targeting Google's Eddystone protocol [6]. By recording at $f_{\text{carrier}} + 2 * f_{\text{carrier}} = 2.548\text{GHz}$ using at least 4MHz of bandwidth, we are now sure to match the Screaming Channel leakage frequency allowing us to capture both channels in the same record. However, as seen in Section 9.2, this frequency is not the only one where the leak can be observed and exploited.

7.3. Evaluation

This evaluation is done using the firmware F_{default} and using the full experimental setup, including the USRP NF from Section 6.

Connection event and Hop Interval. We empirically determined that increasing or decreasing the connection event (CE) number allowed us to modify the leakage position in time with a granularity of 80ms. Also, it was mandatory to set the hop interval greater than 12 to have a TX during the cryptographic operation.

Interleaved procedures comparison. Section 7.2 introduced the technique of "interleaved procedures" leveraging the MD bit. We compared a few procedures, detailed in Table 2. This shows that using this technique can increase the TX time by a factor of 2, increasing the probability of having a radio transmission from the Peripheral during the cryptographic operation.

Cryptographic leakage detection. The cryptographic operation leakage detection we performed is two-fold: 1) an automatic detection of the AES signal 2) a visual control of the system activity and the radio transmission. Based on an *a priori* knowledge of the AES signal inspired from previous work [5], we used an automatic detection of the latter inside a recorded signal using frequency detection and cross-correlation matching. Moreover, thanks to our framework, we were able to perform two parallel and synchronized

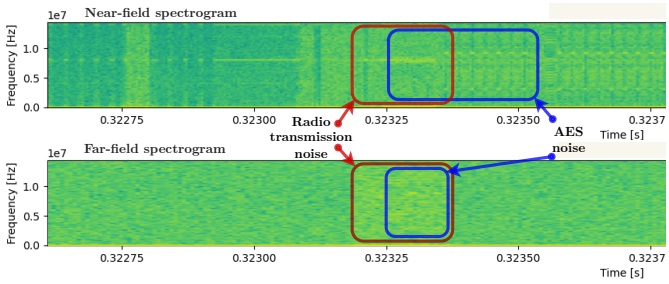

Figure 5. Leakage comparison between Conventional (upper) and Screaming Channel (bottom).

TABLE 2. SYSTEM ACTIVITY LEAKAGE DURATION BASED ON PROCEDURE INTERLEAVING METHOD.

Method	Leak duration
None	108µs
ATT_Read_Request	225µs
ATT_Read_Multiple_Request	234µs
ATT_Find_Information_Request	206µs

recordings. Figure 5 shows a comparison of two recordings. The upper one at $2 * f_{\text{clock}}$Hz using a near-field (NF) probe is recording at the 2nd harmonic of the CPU's clock, where a conventional side-channel can be recorded. The bottom one at $f_{\text{carrier}} + 2 * f_{\text{clock}}$Hz using a far-field (FF) antenna is recording at the 2nd harmonic of the CPU's clock added to the frequency of BLE radio carrier for Bluetooth channel number 8. On the NF recording, the full AES computation can always be identified (upper blue rectangle) since we have all the electromagnetic radiation from the inherent system activity. For both the NF and the FF recordings, we can identify when the TX is happening (red rectangles). On the FF recording, only a partial AES computation can be identified (bottom blue rectangle) because the TX duration was not long enough – still, the exploitable part for a side-channel attack has already leaked. The first goal was to identify the instant the AES computation is happening using the NF recording while completing a BLE connection procedure. The second goal was to manipulate BLE parameters to force this AES computation to occur during the radio transmission. To complete those goals, we leveraged both the automatic AES detection and the visual inspection – where the second goal is achieved when the red and blue boxes are overlapped similarly to Figure 5.

8. Screaming Channel Attack on BLE

We now detail the Screaming Channel leakage exploitation during a BLE communication, using our framework described in Section 7.

8.1. Metrics

We are using the following metrics to evaluate our attack and compare it to previous work:

Partial Guessing Entropy (PGE) [40]. The PGE defines the rank (*i.e.*, the index) of the correct subkey among a list of all possible subkeys classified from the most probable to the least probable according to the side-channel output. For a single subkey, it hence estimates how many guesses are needed to find the correct subkey.

Key Rank [41]. The key rank defines the rank (*i.e.*, the index) of the correct key among a list of all possible keys classified from the most probable to the least probable. Estimating how many guesses are needed to find the correct key is representative of the complexity of the key recovering. The key rank enumeration is a key brute force leveraging knowledge of the side-channel output. We are using the Histogram-based Enumeration Library (HEL) from *Poussier et al.* [42], using both key rank estimation and key enumeration. Performing a key enumeration is an offline procedure that happens after the side-channel attack.

8.2. Side-channel attack on AES during SK derivation

Since a Screaming Channel attack can be seen as a conventional EM side-channel coupling to a radio transmitter, we start by considering a conventional side-channel attack on BLE. We target the step after the first S-Box (`AddRoundKey` and `SubBytes`) inside the 1^{st} AES execution, where the key is the LTK and the plaintext is SKD. SKD is known but partially controlled, because $SKD = SKD_P||SKD_C$ with SKD_P chosen by the Peripheral. As mentioned in Section 2, Cao et al. [10] analyzed the feasibility of BLE only for the conventional case (*i.e.*, near-field (NF)), concurrently to our work. Our work exploits the same side-channel vulnerability over the Screaming Channel (*i.e.*, far-field (FF)), significantly increasing its impact.

8.3. Profiled Correlation Attack

We used the Profiled Correlation Attack used by *Camurati et al.* [6]. The training (*i.e.*, profiling) datasets are collected using a training device similar to the test device but entirely controlled by the attacker, while the attack datasets are collected on the victim device. Previous works on Screaming Channels [5], [6] use time diversity by averaging numerous traces and re-executing the encryption with the same parameters. This approach drastically reduces the noise for both training and attack traces, improving the profile's efficiency. In our case, this is possible using firmware F_{instru} since we control the AES inputs. However, this approach is not practicable with firmware F_{default}, where half of the input (SKD_P) is controlled by the victim and will change for each attack trace according to the BLE protocol. The impact on the attack performance of this averaging technique is discussed in Section 8.6.

In Tables 3 and 4, for "Profile" and "Attack" columns, $x\text{k} * y$ means $x * 10^3$ traces containing y AES operations. When a key enumeration value is provided, we could perform a full key recovery – illustrated by "✅". We use

TABLE 3. ATTACK RESULTS USING F_{INSTRU}.

#	Env.	Dist. (cm)	Profile (#)	Attack (#)	PGE median	Key rank	Key enum.
A_1	Office	100	64k $*$ 100	12k $*$ 300	3	2^{58}	
A_2 ✓	Office	10	64k $*$ 300	2k $*$ 300	1	2^{27}	12s
A_3 ✓	Office	120	16k $*$ 300	10k $*$ 300	1	2^{33}	45min
A_4 ✓	Office	120	16k $*$ 300	16k $*$ 1	0	2^{27}	12s
A_5	Office	120	16k $*$ 1	10k $*$ 300	2	2^{54}	
A_6	Office	120	16k $*$ 1	16k $*$ 1	4	2^{76}	

TABLE 4. ATTACK RESULTS USING F_{DEFAULT}.

#	Env.	Dist. (cm)	Profile (#)	Attack (#)	PGE median	Key rank	Key enum.
A_7 ✓	Anechoic	10	16k $*$ 1	16k $*$ 1	1	2^{34}	1.5h
A_8	Office	120	30k $*$ 1	20k $*$ 1	5	2^{60}	
A_9	Office	120	65k $*$ 1	40k $*$ 1	7	2^{60}	

the icons "" and "" for the omnidirectional and the directional antennas, respectively (see Section 6).

8.4. Evaluation on firmware F_{instru}

Table 3 summarizes the conditions and the results of our attacks using firmware F_{instru} and only the USRP FF from Section 6. This instrumented firmware allows using the averaging technique mentioned in Section 8.3, denoted by $x\text{k} * y$ with y the number of averaged AES ($y > 1$). A_1 attempts to attack at 1 meter inside an office environment, reducing the key rank to 2^{58}. The leak is exploitable, but this attack used the omnidirectional antenna from Section 6 with low gain, resulting in noisy traces. To reduce the noise, we first performed the A_2 attack at a smaller distance using 10 centimeters. In these conditions, we successfully recover the full key in only 12 seconds by lowering the key rank to 2^{27}. To confirm the attack feasibility at a higher distance, we used the directional antenna with higher gain for attacks A_3 to A_6. With a profile based on averaging training traces in A_3 and A_4, we systematically perform a full key recovery. However, when using non-averaged traces for the profile in A_5 and non-averaged traces at all in A_6, the performance is rapidly decreasing with a key rank down to 2^{54} and 2^{76}, respectively.

Averaging attack traces is not a realistic requirement for performing our attack in real-life conditions since the attacker cannot control the plaintext at every cryptographic operation. However, these experiments show that noise reduction (through averaging traces or another method) is a critical requirement for a full key recovery.

8.5. Evaluation on firmware F_{default}

Table 4 summarizes the conditions and the results of our attacks using firmware F_{default} and only the USRP FF from Section 6. This corresponds to the most challenging conditions for an attacker since no instrumentation can be used, making the averaging technique impossible. A_7 is an attack inside an anechoic box with an omnidirectional antenna, isolating the setup from the environmental noise, leading to a key rank of 2^{34} and a full key recovery in less than 2 hours. A_8 and A_9 are two attempts to reproduce A_7 performance at a higher distance (120 cm), using the directional antenna. In A_9, we chained two LNAs, but it did not improve the result. Figure 6 shows the key rank over the number of traces for A_7 and A_9. A_7 convergence speed is lower than A_9 because we use less training traces for the profile, however, A_7 converges to a lower key rank than A_9 because the training traces were less noisy.

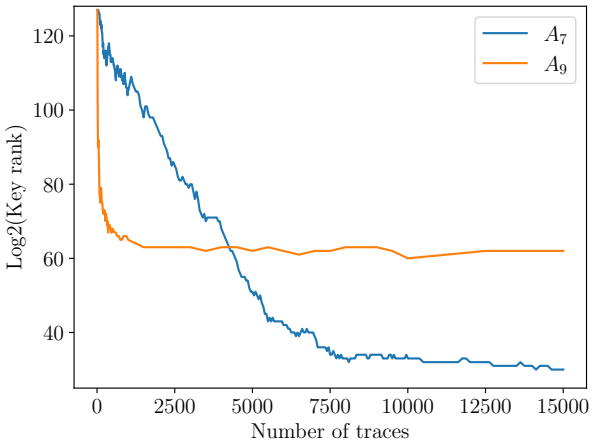

Figure 6. Key rank over number of traces for A_7 and A_9.

While we were able to conduct a full key recovery inside an anechoic environment, we observed a strong impact of the noise in an office environment, preventing a full key recovery.

8.6. Impact of noise on the profile

Note that each attack cannot be improved by simply collecting more traces because they reached their "convergence point", *i.e.*, the attack performance does not increase while increasing the number of traces. This phenomenon is strongly tied by the profile quality. To build a profile, we first compute the Pearson Correlation Coefficients (PCCs) (ρ) over the traces to test for statistical differences depending on the AES inputs, as shown in Figure 10 (see Appendix A). The samples with a significant coefficient correspond to the `SubBytes` operation of AES and are used as point of interests (POIs) to create the profile. To do so, we estimate the mean value and the standard deviation (σ) for each possible classes, *i.e.*, the 256 possible values of $p \oplus k$ for each subkey. We empirically observed on our profiles that the main difference between a profile leading to a full key recovery (A_4) and a profile which does not (A_6 and A_9) is the value of the standard deviation. Table 5 illustrates with three representative examples the comparison between the means of both the PCCs and the standard deviation. All traces were

TABLE 5. MEANS OF PCC ($\overline{\rho}$) AND STD. DEVIATION ($\overline{\sigma}$).

Attack	$\overline{\rho}$	$\overline{\sigma}$	Key rank	Full key recovery
A_4	0.5	0.15	2^{27}	✓
A_6	0.05	1.5	2^{76}	✗
A_9	0.275	0.8	2^{60}	✗

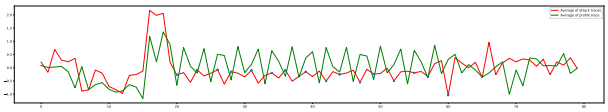

Figure 7. Profile reuse attempt with firmware F_{default}.

normalized using z-score normalization [43], [44] before computing the values. Analogous to the floor noise in radio communication, having a high standard deviation in our profile create a statistical floor noise, *i.e.*, despite using more traces, the distinguisher cannot accurately estimate the most probable subkey above a specific level.

9. Leakage Characterization

9.1. Profile reuse

In profiled side-channel attacks, it is common to create a profile using a controlled device in a lab and then use this profile to attack another instance of the same device in the field [45], [46], [47]. This assumes that the training device's leakage closely matches that of the target device. However, in our case, minor changes in the hardware setup or firmware of the target device significantly impact the leakage. We faced this issue despite the use of normalization techniques, *e.g.*, z-score normalization, to improve profile portability [43], [44]. The possible root causes of those differences are, among others, static code layout, dynamic state of registers, and impulse response of the hardware reception system. In particular, we observed that: 1) training traces recorded in the anechoic box at 10 cm (A_7) are different from attack traces recorded at more than 1 meter (A_{7-8}), 2) training traces recorded with the firmware F_{instru} ($A_{8-9-10-11}$) are different from the attack traces recorded with firmware F_{default} (A_{6-7-12}). Consequently, profile reuse is complicated in attacks using either complex firmware or hardware setups. Figure 7 illustrates a profile reuse attempt using firmware F_{default} for both training and attack traces, but in different conditions. The profile has been created inside an anechoic environment using a small antenna. The attack traces were recorded inside an office environment using a directional antenna and an LNA. We observe that the profile mean trace (green) and the attack mean trace (red) are closely similar but do not exactly match.

9.2. Leaking frequencies

Understanding at which frequency a leakage may be found and exploited is not trivial. *Li et al.* [48] formalized

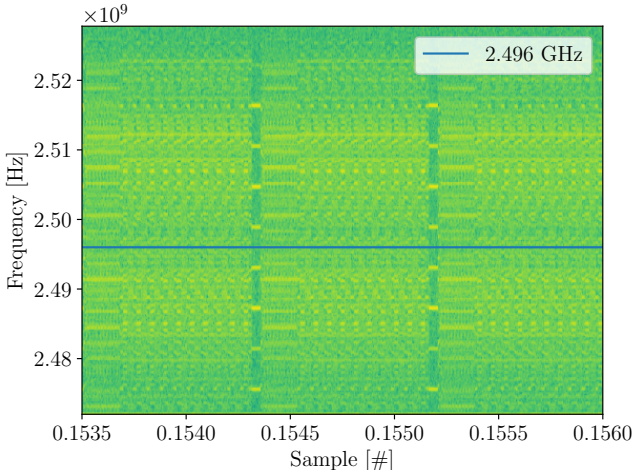

Figure 8. Third-order intermodulation product at 2.496 GHz between the 32 MHz CPU sub-clock and the 2.4 GHz carrier.

and simulated how a CMOS transistor can act as an amplitude modulator, resulting in electromagnetic radiations (EMRs) at the harmonics of a clock signal modulated in amplitude by a data signal. *Camurati et al.* [5] described the Screaming Channel leakage as a substrate coupling between the amplitude-modulated signal from the digital part and the analog part of the SoC. While only harmonic frequencies were used for this initial attack, *Guillaume et al.* [13] shows that many frequencies can be exploited to successfully recover a key without investigating the reason for the presence of the leak at those frequencies.

Therefore, we analyzed the spectrum in the frequency domain using 56MHz of bandwidth with a USRP B210 [29], monitoring the leakage generated by firmware F_{custom} (see Table 1) running AES encryption and continuous radio transmissions concurrently. We identified that the leakage was present at predictable frequencies that were not the harmonics of the CPU clock itself but the harmonics of derived clocks. The nRF52832 provides a main CPU clock ("`HCLK64M`") running at 64MHz, but also derived clocks "`PCLK32/16/1M`" provided to various peripherals, running at respectively 32, 16, and 1 MHz. Figure 8 shows the signal generated by the AES leak at 2.496GHz for a carrier transmitting at 2.4GHz. The two sidebands correspond to the AES modulating the `PCLK32M` clock signal in amplitude. This signal has been recorded at $3 * 32\text{MHz} + 2.4\text{GHz}$ and corresponds to a third-order intermodulation product between the `PCLK32M` clock of the nRF52832 and the radio transceiver carrier. The number of derived clocks inside the SoC, and the number of harmonics and the bandwidth of the leaked signal, explains the observations from *Guillaume et al.* [13] reporting successful attacks at "non-harmonics" frequencies, which was only considering the presence of the 64MHz clock and its harmonics. In summary, we postulate that the "non-harmonics" frequencies are, in fact, due to the harmonics of multiple internal clocks.

9.3. Hardware component impacting the leakage

We explored the hypothesis that the memory controller (flash) may be the root cause of the leak. We observed that the nRF52832 has an Instruction Cache (I-Cache) [49], allowing the CPU to fetch the firmware code to execute instructions directly from the cache instead of soliciting the flash memory (Figure 12, in Appendix A). To evaluate the impact of this cache, we record two datasets of F_{custom} firmware running AES encryptions and radio transmissions simultaneously, with and without the I-Cache enabled. We observe that the recorded traces are intermittent (Figure 11, in Appendix A). This suggests that the leakage is present when the instruction is fetched from the flash controller (*i.e.*, a cache miss) and absent when fetched from the cache (*i.e.*, a cache hit). We observe that the I-Cache is disabled on NimBLE by default, potentially for reliability reasons (for constant-time execution). Even with the I-Cache enabled, we were still able to correlate with the AES inputs in our experiments. These observations suggest that the code base itself, the compiler, and the linker decisions may affect the leak exploitability if the cache is enabled, providing an interesting direction for future work.

10. Discussion

Protocol manipulation. Using protocol manipulation from the attacker's side only, we demonstrated that it is possible to force the victim device (with unmodified real-world firmware) to transmit while the target cryptographic activity occurs. While we leveraged BLE specific techniques, we identified a set of generic requirements for the Screaming Channel to happen. First, increasing the radio transmission (TX) duration by manipulating physical layer parameters largely increases the Screaming leakage probability. Such a low-level influence can be generated directly by injecting specific values at the physical layer or indirectly by interfering with the channel itself or triggering target retransmission mechanisms. Second, generating traffic, especially requests expecting a response from the victim device, may also lead to an increased TX duration. Third, taking a fine-grained control of the timing of operations, like the control of CEs using hop interval in BLE, is also helpful for the attacker. Moreover, our BLE manipulations are not exhaustive. Indeed, manipulating other parameters could have an impact on the TX duration, like injecting a bigger maximum transmission unit (MTU) through its dedicated control PDU. We demonstrated this for BLE; however, for other protocols, a case-by-case analysis is required, and we expect this to be more challenging for some protocols.

Attack complexity. Our work shows that performing Screaming Channels in realistic conditions is challenging and that the exploitability can be impacted by many factors. Moreover, building an experimental setup and collecting datasets require significant engineering work, motivating us to release our framework as open-source software and our collected traces as an open dataset to facilitate the reproducibility of our work and encourage the community to explore related topics. We identified several research directions that could significantly improve the attack performance. First, the difficulty of reusing profiles complicates the attack. Understanding the root causes of the leakage differences or significantly improving the profiling and normalization algorithms could help to reuse profiles across devices. Second, a detailed evaluation of the impact of the instruction cache could be relevant in specific scenarios. Third, evaluating the feasibility of attacking a hardware implementation of AES with Screaming Channel remains an open challenge. Finally, our experiments show that the radio setup and environment significantly impact the attack performance, indicating a need for optimization.

Impact. Our attack demonstrates that Screaming Channels may be a threat against realistic firmware and off-the-shelf Bluetooth Low Energy stacks in the future. The attack still requires performance improvements before claiming that it is a fully realistic threat. Reducing the noise, which has a critical impact on attack performance, seems to be the predominant challenge. It underlines the need for a more robust radio setup, better statistical pre-processing algorithms, or signal diversity such as frequency diversity or space diversity, which are promising directions for future work.

11. Countermeasures

We propose several countermeasures at different levels.

Cryptographic countermeasures. 1) Masking [50], [51], where the secret is divided into multiple "shares" and mixed with internal random values during cryptographic computations, such that an attacker cannot gain information about intermediate values. 2) Hiding [52] involves controlling execution time and power consumption during cryptographic operations to appear random or constant, preventing attackers from gaining information through side-channel measurements. 3) Re-keying [53] involves frequently changing the key, preventing an attacker from collecting sufficient traces before the key is updated.

Physical countermeasures. Shielding [54] or decoupling consists in attenuating the physical leakage propagation. A hardware designer can insert a shield between the analog and the digital part of the SoC to reduce the Screaming Channel leakage. Alternatively, if the cryptographic operation is implemented in hardware, the AES S-boxes themselves can be shielded and isolated. However, shielding is inherently complicated in a fully integrated radio system that needs to transmit data over the air.

Protocol countermeasures. Protocol countermeasures against side channels include limiting the number of connections in a period of time [10] so that an attacker cannot collect enough traces in a reasonable amount of time. Limiting the number of failed encrypted session establishments in a given period or per pairing or adding a waiting time after each failed session establishment also seems effective. These countermeasures are suited for a protocol specification. To counteract Screaming Channels in particular, it is essential to ensure that the cryptographic operation (CO) happens at

random times or during the second slot of the connection event when the radio is disabled. These countermeasures are suited to be ensured at the firmware level.

Countermeasures limitations. Each of those countermeasures needs to be carefully considered, as they may have negative side effects. The cryptographic countermeasures increase the implementation complexity and introduce a performance overhead. The physical countermeasures are quite complex and not well studied, and they may increase the cost significantly. The protocol countermeasures seem to be the most straightforward to apply and effective against this specific attack. While we need to ensure that no corner case could lead to a denial of service for legitimate users, we recommend specifying them in the protocol specification.

12. Conclusion

In this paper, we showed how an attacker could manipulate a set of parameters of the BLE protocol to make a victim device "scream", *i.e.*, execute critical cryptographic operations during a radio transmission while staying compliant with the protocol specification. Leveraging these mechanisms allowed us to conduct a successful end-to-end Screaming Channel attack on a victim Peripheral device in an environment isolated from noise, leading to a full key recovery of the Long Term Key, used to establish a secure session between the two devices. However, assessing the attack in a realistic environment only leads to a partial key recovery due to its increased radio noise. Future work to improve profile reuse or reception diversity is a promising direction, as it may change the outcome of an attack from a partial key recovery to a full key recovery. Regarding the improvement potential, our results led toward a threat that should be considered for the future of IoT communications.

Acknowledgments

This research work was supported by the French National Agency for Research (ANR), in part by the project MobiS5 (ANR-18-CE39-0019) and in part the France 2030 project PEPR REV (ANR-22-PECY-0009). The authors thank Karel Král from Google and Rachida Saroui from École Normale Supérieure (ENS) for the discussions and collaborations on preliminary experiments on the topic. Finally, we are highly grateful for the meaningful comments from the reviewers.

References

[1] N. S. Agency, "TEMPEST: A Signal Problem," NSA, Tech. Rep., 1972.

[2] W. Van Eck, "Electromagnetic radiation from video display units: An eavesdropping risk?" *North-Holland Computers & Security*, pp. 269–286, 1985.

[3] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '96. Berlin, Heidelberg: Springer-Verlag, 1996, p. 104–113.

[4] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security*, ser. E-SMART '01. Berlin, Heidelberg: Springer-Verlag, 2001, p. 200–210.

[5] 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*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 163–177. [Online]. Available: https://doi.org/10.1145/3243734.3243802

[6] 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, Jun. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8594

[7] Google. (2015) Eddystone. [Online]. Available: https://github.com/google/eddystone

[8] A. D. Blog. (2018) Discontinuing support for android nearby notifications. [Online]. Available: https://android-developers.googleblog.com/2018/10/discontinuing-support-for-android.html

[9] P. Ayoub, "Screaming channels on bluetooth low energy," Aug. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.13384278

[10] P. Cao, C. Zhang, X.-J. Lu, H.-N. Lu, and D.-W. Gu, "Side-channel analysis for the re-keying protocol of bluetooth low energy," *Journal of Computer Science and Technology*, vol. 38, no. 5, p. 1132–1148, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1007/s11390-022-1229-3

[11] R. Wang, H. Wang, and E. Dubrova, "Far field em side-channel attack on aes using deep learning," *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*, Nov 2020. [Online]. Available: http://dx.doi.org/10.1145/3411504.3421214

[12] J. Guillaume, M. Pelcat, A. Nafkha, and R. Salvador, "Virtual triggering: a technique to segment cryptographic processes in side-channel traces," in *2022 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, Nov. 2022. [Online]. Available: http://dx.doi.org/10.1109/SiPS55645.2022.9919238

[13] ——, "Attacking at non-harmonic frequencies in screaming-channel attacks," in *Smart Card Research and Advanced Applications*, S. Bhasin and T. Roche, Eds. Cham: Springer Nature Switzerland, 2024, pp. 87–106.

[14] C. Fanjas, C. Gaine, D. Aboulkassimi, S. Pontié, and O. Potin, "Combined fault injection and real-time side-channel analysis for android secure-boot bypassing," in *Smart Card Research and Advanced Applications - 21st International Conference, CARDIS 2022, Birmingham, UK, November 7-9, 2022, Revised Selected Papers*, 2022, pp. 25–44. [Online]. Available: https://doi.org/10.1007/978-3-031-25319-5_2

[15] E. Danieli, M. Goldzweig, M. Avital, and I. Levi, "Revealing the secrets of radio embedded systems: Extraction of raw information via rf," *IEEE Transactions on Information Forensics and Security*, vol. 19, p. 2066–2081, 2024. [Online]. Available: http://dx.doi.org/10.1109/TIFS.2023.3345131

[16] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[17] D. Genkin, A. Shamir, and E. Tromer, "Acoustic cryptanalysis," *J. Cryptology*, vol. 30, pp. 392–443, 2017.

[18] J. Ferrigno and M. Hlavác, "When AES blinks: Introducing optical side channel," *IET Inf. Secur.*, vol. 2, no. 3, pp. 94–98, 2008. [Online]. Available: https://doi.org/10.1049/iet-ifs:20080038

[19] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side-channel(s)," in *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '02. Berlin, Heidelberg: Springer-Verlag, 2002, p. 29–45.

[20] C. Lavaud, R. Gerzaguet, M. Gautier, O. Berder, E. Nogues, and S. Molton, "Whispering devices: A survey on how side-channels lead to compromised information," *Journal of Hardware and Systems Security*, vol. 5, pp. 143 – 168, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:233685396

[21] N. I. of Standards and T. (NIST), *Advanced Encryption Standard (AES)*, Nov. 2001. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/197/final

[22] D. J. Bernstein, "Cache-timing attacks on AES," 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:2217245

[23] S. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an ASIC AES implementation," in *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, vol. 2, 2004, pp. 546–552 Vol.2.

[24] S. Mangard, "A simple power-analysis (SPA) attack on implementations of the AES key expansion," in *Information Security and Cryptology — ICISC 2002*, P. J. Lee and C. H. Lim, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 343–358.

[25] B. W. Group, *Bluetooth Core Specification*. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-5-3/

[26] C. Lavaud, "Reconfigurable Systems for the Interception of Compromising Sporadic Signals," Theses, Université de Rennes 1, Jan. 2022.

[27] D. Antonioli, "Bluffs: Bluetooth forward and future secrecy attacks and defenses," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 636–650. [Online]. Available: https://doi.org/10.1145/3576915.3623066

[28] J. Wu, R. Wu, D. Xu, D. Tian, and A. Bianchi, "Sok: The long journey of exploiting and defending the legacy of king harald bluetooth," in *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 23–23. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00023

[29] E. Research, "USRP B210 SDR kit - dual channel transceiver (70 MHz - 6GHz)." [Online]. Available: https://www.ettus.com/all-products/ub210-kit/

[30] TekBox, "Tbps01 probe," 2024. [Online]. Available: https://www.tekbox.com/product/tekbox-tbps01-emc-near-field-probes/

[31] GoTronic, "2,4 ghz antenna," 2024. [Online]. Available: https://www.gotronic.fr/art-antenne-2-4-ghz-a24-hasm450-31786.htm

[32] TP-Link, "Tl-ant2424b," 2024. [Online]. Available: https://www.tp-link.com/fr/home-networking/antenna/tl-ant2424b/

[33] Mini-Circuits, "Zx60-272ln-s+," 2024. [Online]. Available: https://www.minicircuits.com/pdfs/ZX60-272LN-S+.pdf

[34] D. Cauquil and R. Cayre, "One for all and all for whad: Wireless shenanigans made easy !" DEF CON 2024, DEF CON Security Conference, 8-11 August 2024, Las Vegas, NV, USA, 2024. [Online]. Available: https://defcon.org/html/defcon-32/dc-32-speakers.html

[35] R. Cayre, "Butterfly," 2024. [Online]. Available: https://github.com/RCayre/injectable-firmware

[36] R. Cayre, F. Galtier, G. Auriol, V. Nicomette, M. Kaâniche, and G. Marconato, "InjectaBLE: Injecting malicious traffic into established Bluetooth Low Energy connections," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2021)*, Taipei (virtual), Taiwan, Jun. 2021. [Online]. Available: https://laas.hal.science/hal-03193297

[37] A. S. Foundation, "Nimble," 2024. [Online]. Available: https://github.com/apache/mynewt-nimble

[38] ——, "Mynewt," 2024. [Online]. Available: https://mynewt.apache.org/

[39] Intel, "Tinycrypt," 2024. [Online]. Available: https://github.com/intel/tinycrypt

[40] H. Pahlevanzadeh, J. Dofe, and Q. Yu, "Assessing cpa resistance of aes with different fault tolerance mechanisms," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016, pp. 661–666. [Online]. Available: https://ieeexplore.ieee.org/document/7428087

[41] N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert, "Security evaluations beyond computing power," in *Advances in Cryptology – EUROCRYPT 2013*, T. Johansson and P. Q. Nguyen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 126–141. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-38348-9_8

[42] R. Poussier, F.-X. Standaert, and V. Grosso, "Simple key enumeration (and rank estimation) using histograms: An integrated approach," in *Cryptographic Hardware and Embedded Systems – CHES 2016*, B. Gierlichs and A. Y. Poschmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 61–81. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-53140-2_4

[43] M. A. Elaabid and S. Guilley, "Portability of Templates," *Journal of Cryptographic Engineering*, vol. 2, no. 1, pp. 63–74, May 2012.

[44] D. P. Montminy, R. O. Baldwin, M. A. Temple, and E. D. Laspe, "Improving cross-device attacks using zero-mean unit-variance normalization," *Journal of Cryptographic Engineering*, vol. 3, pp. 99–110, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:18343838

[45] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Workshop on Cryptographic Hardware and Embedded Systems*, 2002. [Online]. Available: https://api.semanticscholar.org/CorpusID:9694193

[46] T.-H. Le, C. Canovas, and J. Clédière, "An overview of side channel analysis attacks," in *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '08. Association for Computing Machinery, 2008, p. 33–43. [Online]. Available: https://doi.org/10.1145/1368310.1368319

[47] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, vol. 4, no. 2, 2020.

[48] H. Li, A. T. Markettos, and S. Moore, "Security evaluation against electromagnetic analysis at design time," in *Proceedings of the 7th International Conference on Cryptographic Hardware and Embedded Systems*, ser. CHES'05. Springer-Verlag, 2005, p. 280–292. [Online]. Available: https://doi.org/10.1007/11545262_21

[49] N. Semiconductor, *nRF52832 Product Specification*, 2021.

[50] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, nov 1979. [Online]. Available: https://doi.org/10.1145/359168.359176

[51] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *Advances in Cryptology - EUROCRYPT 2013*, ser. Lecture Notes in Computer Science, vol. 7881. Springer, 2013, pp. 142–159.

[52] J. Lee and D.-G. Han, "Security analysis on dummy based side-channel countermeasures—case study: Aes with dummy and shuffling," *Applied Soft Computing*, vol. 93, p. 106352, 2020.

[53] M. Medwed, F.-X. Standaert, J. Großschädl, and F. Regazzoni, "Fresh re-keying: Security against side-channel and fault attacks for low-cost devices," in *Progress in Cryptology – AFRICACRYPT 2010*, D. J. Bernstein and T. Lange, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 279–296.

[54] M. Wang, V. V. Iyer, S. Xie, G. Li, S. K. Mathew, R. Kumar, M. Orshansky, A. E. Yilmaz, and J. P. Kulkarni, "Physical design strategies for mitigating fine-grained electromagnetic side-channel attacks," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*, 2021, pp. 1–2.

Appendix

AES leak with F_{default} firmware

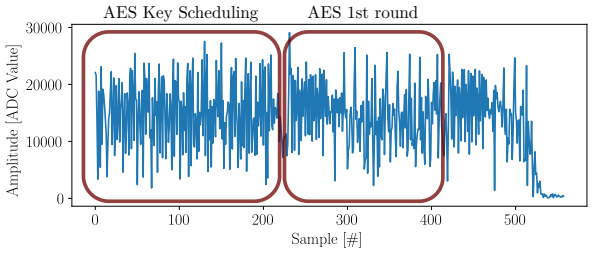

Figure 9. AES leakage in amplitude at $f_{\text{carrier}} + 2 * f_{\text{clock}}$Hz during BLE communication.

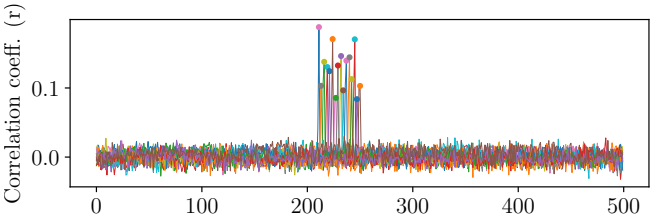

Figure 10. Correlation on amplitude for A_7 during a radio transmission with GFSK at $f_{\text{carrier}} + 2 * f_{\text{clock}}$Hz.

Instruction Cache Experiment

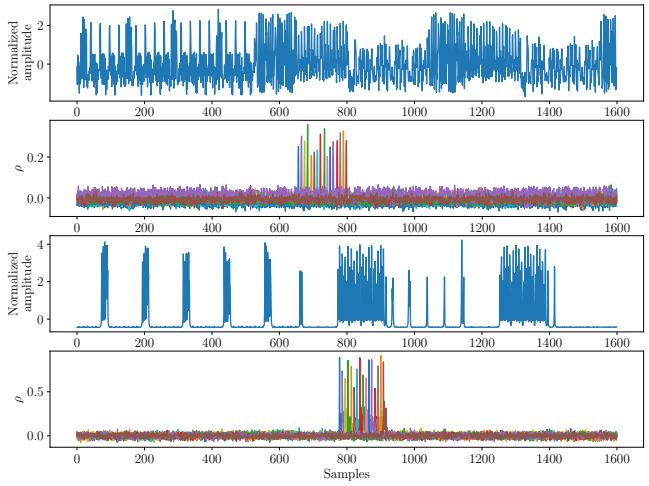

Figure 11. Correlations with instruction cache disabled (top) and enabled (bottom).

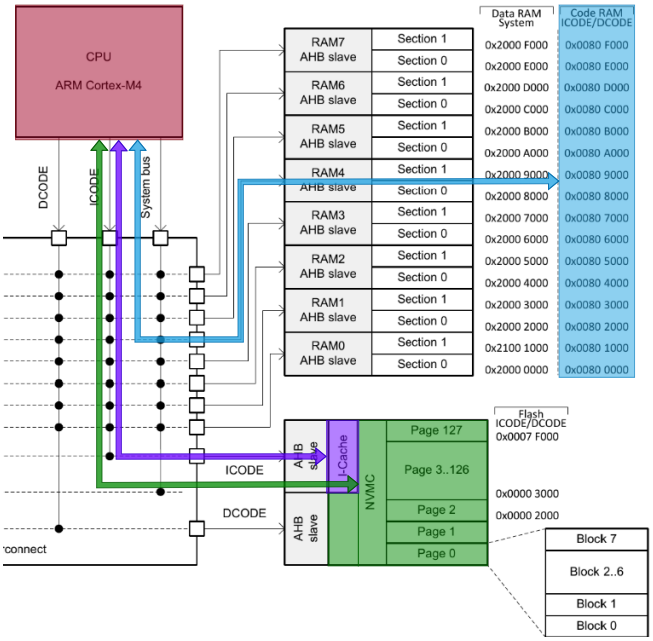

Figure 12. The CPU (red) can fetch instructions from either the RAM (blue), the Instruction Cache (purple) or the Flash (green). [49, p. 24]