Breaking Mobile Firmware Encryption through Near-Field Side-Channel Analysis

Aurélien Vasselle, Philippe Maurine, Maxime Cozzi

Breaking Mobile Firmware Encryption through Near-Field Side-Channel Analysis

Aurélien Vasselle
eShard, LIRMM
Pessac, France

Philippe Maurine
Maxime Cozzi
LIRMM
Montpellier, France

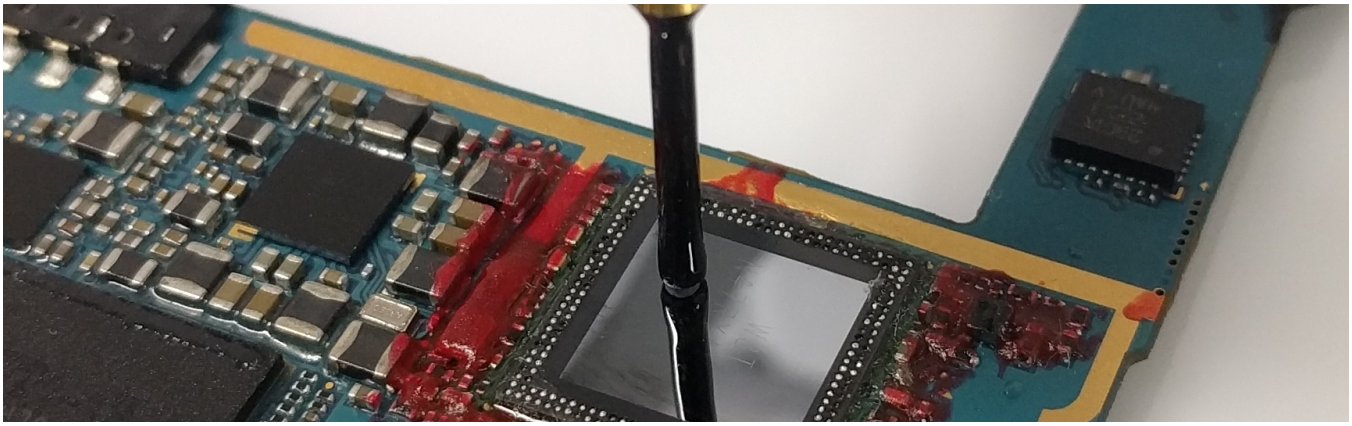


Figure 1: Near-field acquisition setup

ABSTRACT

Physical attacks constitute a significant threat for any cryptosystem. Among them, Side-Channel Analysis (SCA) is a common practice to stress the security of embedded devices like smartcards or secure controllers. Nowadays, it has become more than relevant on mobile and connected devices requiring a high security level. Yet, their applicability to smartphones is not obvious, as the architecture of modern System-on-Chips (SoC) is becoming ever more complex.

This paper describes how a secret AES key was retrieved from the hardware cryptoprocessor of a smartphone. It is part of an attack scenario targeting the bootloader decryption. The focus is on practical realization and the challenges it brings. In particular, catching meaningful signals emitted by the cryptoprocessor embedded in the main System-on-Chip can be troublesome. Indeed, the Package-on-Package technology makes access to the die problematic and prevents straightforward near-field electromagnetic measurements. The described scenario can apply to any device whose chain-of-trust relies on firmware encryption, such as many smartphones or Internet-of-Things nodes.

ASHES'19, November 15, 2019, London, United Kingdom

CCS CONCEPTS

• **Security and privacy → Side-channel analysis and countermeasures**.

KEYWORDS

Side-channel Attack, Cryptography, Firmware, Secure Boot, Smartphone, Mobile Phone, Hardware Security, AES-CBC

1 INTRODUCTION

1.1 Secure Boot

Secure Boot refers to the verification of authenticity of any sensitive code executed by a device. As there are many entities involved in recent software architectures, a chain-of-trust is established from the device reset, up to trusted services running in a Trusted Execution Environment (TEE). An overview of the different stages is given in Fig. 2. Each developer can extend his trust to a new one by cryptographically assessing the authenticity of software before execution.

This operation can be performed thanks to asymmetric cryptography algorithms such as RSA or ECC, or sometimes with authenticated encryption modes such as Galois/Counter Mode (GCM) or Synthetic Initialization Vector (SIV). One major advantage of the former is that disclosure of the private key only depends on

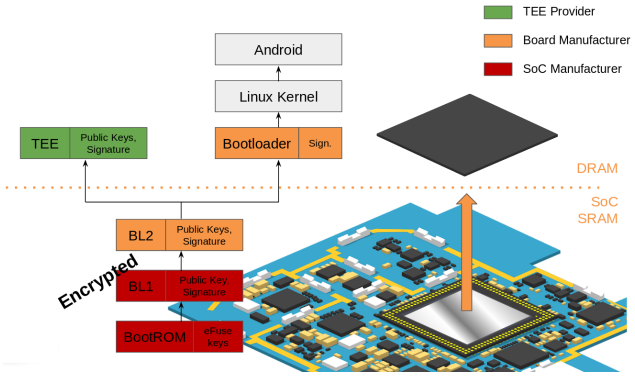


Figure 2: Secure Boot sequence overview

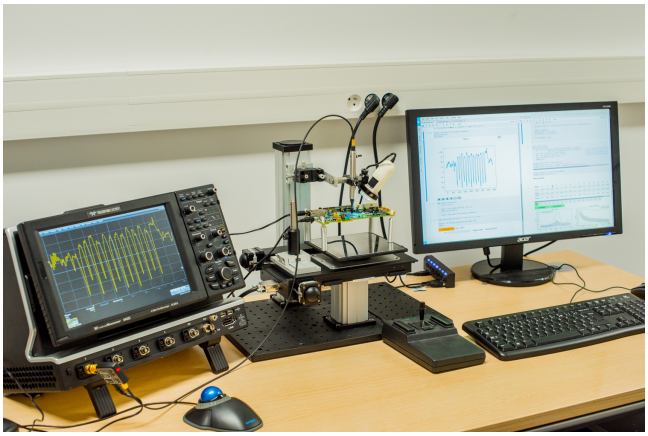


Figure 3: Side-channel Acquisition Bench

the security of its certificate authority; whereas symmetrical keys are necessarily manipulated by the devices in the field, which simplifies physical access to the secret, and eventually its extraction. Therefore, most secure boot implementations are based on hybrid solutions to guarantee a strong asymmetrical authenticity, and maintain confidentiality with the fast decryption capabilities of a block cipher.

Generally speaking, a secure boot implementation does not require firmware confidentiality to maintain its authenticity. Kerckhoffs's principle recalls that a cryptosystem should be secure even if everything about the system, except the key, is known [17]. In that case, the private key that is at the root of trust should remain the only weak point.

Yet, it is really common to see device manufacturers encrypt their firmware. Despite casting suspicion around undisclosed software and backdoors, this decision might be beneficial. Indeed, without this layer of secrecy, an attacker can for example study the patches applied between firmware updates in order to find flaws in the boot sequence. To that extent, firmware encryption allows manufacturers to keep non-patched devices safe from widespread attacks after public release of vulnerabilities. In fact, encryption is one of the strongest barrier against firmware reverse-engineering and most of the time prevents vulnerability exposure.

In the context of such black-box system, the attack surface is limited, and only few paths remain. This paper explores the use of side-channel analysis to disclose the firmware encryption key and expose unseen vulnerabilities.

1.2 Side-Channel Analysis

Side-channel attacks aim to retrieve secret information from observation of a device processing it. This is particularly useful to disclose cryptographic keys by analyzing the intermediate data processed during an algorithm. Indeed, intermediate data often provide information related to a small part of the secret key. Using statistical tools [6], the key can be fully retrieved by pieces using a divide-and-conquer approach.

A common way to get valuable measurements of the intermediate data is to exploit physical quantities (e.g. electromagnetic (EM) field [1], power consumption [21], heat [16], ...) collected when a physical device executes the cryptographic operation. As detailed in Section 3, near-field EM emanations turn out to be relevant for complex SoCs. It increases the chances to get valuable information as it captures local activity, and prevents observations from being disturbed by surrounding noise sources.

Complex devices also have the tendency to leak sensitive information through their micro-architectural behavior. Such flaws were recently exploited with memory addressing [28], or cache [24] and branch prediction [20] mechanisms.

To be successful, it is necessary to collect a significant number of traces. The order of magnitude lies in the range from a dozen to millions of traces. This highly depends on the quality of measurement and the countermeasures implemented in the cryptography code. Indeed, specific protections can be implemented to avoid the secret key from being exposed. They usually impede the attack by significantly raising the quantity of measurement required.

1.3 Related Work

Several EM side-channel attacks targeted the computation of asymmetric cryptography algorithms executed on smartphones. In particular, the OpenSSL library is frequently hardened thanks to the work of various security researchers.

In [12], ECDSA key extraction was performed without opening the mobile. The signal analysis was able to reveal the sequence of operation, which is key-dependent, using cheap audio equipment. This analysis demonstrates that fast devices such as smartphones can still leak sensitive information at very low frequency.

The authors of [2] were also able to attack a recent implementation of RSA, which integrates some protections against SCA.

Symmetric algorithms, on the contrary, are much faster and simpler to compute. The literature shows EM attacks on the BeagleBone Black development board [3, 25]. In these studies, custom AES implementations are shown vulnerable against SCA. The acquisitions were performed mainly over external decoupling capacitors.

1.4 Guessing Entropy and Bruteforce

In the context of this paper, the Guessing Entropy (GE) metric [7] is a natural candidate to compare the performance of different attacks. Its computation is lightweight, but requires knowledge of the secret key. It corresponds to the number of key candidates to examine before finding the secret key, usually considering an ideal key enumeration algorithm.

The GE is thus a conceptualization of the required computing workload to bruteforce the key, assisted by the knowledge of scores of the attack. Indeed, the attack results allow the search space to be reduced in case the secret key is not ranked first. Therefore, it is generally possible to exhaust the candidate efficiently and recover the correct key with a reasonable remaining effort. For example, using a Score-based Key Enumeration Algorithm (SKEA) [4].

Table 1 gives an overview of the computation power provided by commodity hardware. Based on these values, we decided to choose a reasonable GE threshold at 2^{32}. In the example given in Fig. 4, the attack will succeed in about 255 000 traces, but given only 100 000, an attacker should be able to recover the full secret in less than a minute using SKEA method.

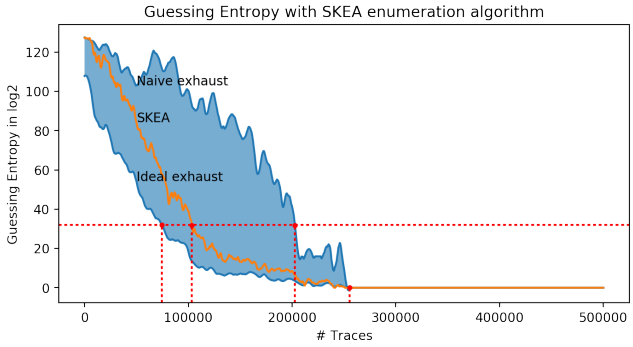


Figure 4: Example of Guessing Entropy bounds (with an ideal, state-of-the-art and naive enumeration algorithm)

1.5 Contribution

In this work, we present the first public side-channel attack targeting the cryptoprocessor of a mobile phone SoC.

We detail how this attack path can be applied to compromise a secure boot, by using SCA as an entry-point for reverse-engineering the boot firmware. To illustrate this scenario, we conclude that the secure boot of the targeted hardware can be bypassed with a software vulnerability.

Moreover, as the arbitrary code execution allows direct access to the cryptographic core, in-depth analysis of the leakage can be achieved. We also performed a dedicated study on the thermal response of the SoC when the cryptoprocessor is operating. Thus, the focus of the paper is on the analysis methodology on such complex devices, as many techniques might be applicable to other systems.

Even if the practical realization is tedious, the stakes of breaking the secure boot definitely justify the mitigation of such risks.

To proceed beyond the attack, some countermeasures are discussed. The proposed low-cost solutions are shown effective to protect this device against our side-channel attack.

2 ATTACK STRATEGY

This section describes the target cryptosystem and the path behind our SCA. Fig. 2 recalls the architecture being examined. This description should cover most devices protected by a secure boot. As a side note, this analysis ignores scenarios involving fault injection attacks [34].

The target software triggering the firmware decryption cannot be modified as it is written in the device ROM. It has been written in factory masks and is supposed to be flawless. As it is protected by ARM TrustZone, this code and the encrypted parts of the firmware cannot be read nor reverse-engineered.

Nevertheless, its execution roughly follows the necessary steps of secure loading of the next stages of bootloaders. Pseudo-code 1 summarizes the BootROM sequence responsible for loading the first stage bootloader (BL1). This code copies BL1 from an external memory in its own internal SRAM. The loaded data can be of arbitrary length, but are usually limited, as with our practical example of 16 KiB. After the copy, the CPU verifies the integrity and authenticity of BL1 using asymmetric cryptography. After this signature verification, data is trusted and BL1 can be decrypted in-place by a dedicated hardware cryptoprocessor.

In general, the firmware decryption key is identical for all phones of the same model. This assumption can be confirmed by checking that the same firmware can be flashed on multiple devices. Which means that any device can be used to perform the attack and retrieve the manufacturer secret key.

Listing 1: Sketch of the BootROM code

```
void reset(){
        hardware_init();

        int sb_en = read_efuse(SECURE_BOOT_ENABLED);
        int bootmedia = read_gpio(BOOT_RESISTORS);

        BL1_Copy( bootmedia );
        BL1_Verify_Checksum();

        if (sb_en == 1){
                BL1_Verify_Pubkey();
                BL1_Verify_Signature();
                BL1_Decrypt();
        }

        BL1_jump();
}
```

The target uses an AES-128 in Cipher Block Chaining (CBC) mode to decrypt the firmware. Unlike CBC encryption, the input data processed by the block cipher are known to an adversary, which is mandatory to perform any SCA.

However, this architecture processes verification before decryption, which mitigates the device exposure to SCA. It forces the adversary to work in a known ciphertext context: as input data are verified before decryption, an attacker is not able to choose or modify the ciphertexts that will be decrypted. Thus, the number of unique inputs is limited to the length of a firmware in block size,

Table 1: AES-256 exhaust time by order of magnitude, using an Intel 4790k with AES-NI instruction set

Number of keys	2^{32}	2^{38}	2^{40}	2^{42}	2^{44}	2^{46}	2^{48}
Number of computers	1	1	10	10	10	10	10
Time	1 min	1 hour	22 min	1.5 hours	6 hours	24 hours	4 days

multiplied by the number of firmware packages available.

To perform the attack, an adversary would typically flash a firmware on the device, boot it several times and observe the decryptions being handled by the cryptoprocessor. The challenges involved in signal acquisition are discussed in Section 3.

To obtain more measurements, flashing another firmware allows adversaries to observe different data being decrypted. Then, statistical analysis of the data is applied to retrieve the manufacturer secret key. The quantity of data required to succeed our attack is detailed in Section 4. Note that in order to validate the retrieval of the manufacturer secret key, it is mandatory to decrypt few blocks of a firmware and confirm that the data are not random but legitimate ARM instructions. Without knowledge of IV, the first block cannot be decrypted.

Finally, knowledge of the firmware encryption key enables several malicious behaviors. First, as explained in Section 5, it becomes possible to reverse-engineer the code that was previously encrypted. Secondly, it is extremely valuable in case the attacker found a way to craft a valid signature or bypass its verification [22], and wants to rebuild his own firmware. However, due to the structure of CBC, an attacker without the key can still craft a firmware that will be decrypted into chosen and executable instructions [33].

3 SIGNAL ACQUISITION

3.1 Probe Position

The biggest challenge of this attack remains the signal acquisition in order to perform a side-channel analysis. Indeed, this Section details the solution used to overcome the Package-on-Package issue.

First, the firmware decryption lasts for 100 µs over the 0.5 s of the boot sequence. Usually, the goal is to find a trigger signal such as the communications with the bootmedia, that is not necessarily leaking sensitive information, but allows to roughly frame the cryptographic event in time. Then, it is possible to search for a signal emitted from the computation of the target AES decryptions at different locations.

On the whole surface of a smartphone, several components can contain sensitive information about the computation:

Power Management Integrated Circuit (PMIC): This circuit, and its surrounding capacitors and inductors, are generating voltage levels for all power domains of the SoC. Among them, one DC voltage regulator is powering directly the ARM cores. However, the hardware cryptoprocessor often works on another power domain, that is unknown to the attacker and shared with other peripherals.

External decoupling capacitors: Their role is to stabilize power supplies and reduce noise for the different ICs of the board. The main SoC should have many capacitors, often located just under the chip. They are dedicated to each power level of the PMIC, and radiate EM waves related to the power consumption of the different peripherals of the chip.

Sense resistors: The board might contain test points used for factory testing of the device. Sometimes, there are sense resistors dedicated to current consumption measurement of the ARM cores. Or, an attacker may also add a sense resistor in the power delivery path of the target components.

System-on-Chip die: As silicon chips are mainly composed of transistors and metal wires, any transient current flowing through the logic gates is generating an EM field [11, 31]. The simultaneous switching of thousands of gates leads to the emission of high energy peaks. This radiated signal, which is directly relates to the device activity, can be measured with adequate EM probes, in the vicinity of the chip.

Of all side-channel sources priorly listed, near-field measurement over the die is the most promising. Indeed, it enables separation of signals emitted from a single peripheral, or even a part of it, from those of other functional blocks. In practice, EM side-channel attacks tend to be more precise and localized than other methods such as power analysis [29], especially for chips as complex as modern SoCs.

However, on high-end devices, memory speed and miniaturization push manufacturers to use Package-on-Package (PoP) [15], as depicted in Fig. 2 in which the DRAMs are placed on top of the main SoC. This configuration can be a show-stopper for side-channel signal acquisition. Indeed, the sensitive emissions of the SoC are blocked and absorbed by the upper package. On top of that, the DRAM emits more than the SoC itself, as it is usually closer to the EM sensors and is typically comprised of 4 memory chips, hundreds of wire bondings, and a PCB interposer with metallic routing lines.

While CPU activity can often be observed with all methods described, our target is a smaller peripheral: the cryptoprocessor. During the boot sequence, the device's power consumption is fairly high, typically up to 2 W. In contrast, the sole consumption induced by the AES computation is likely to be negligible. We experimentally estimated it to be around 2 mW.

To a certain extent, decoupling capacitors and power measurements will be hard to leverage. The cryptoprocessor power domain is shared with many peripherals and they are all decoupled through the same bank of capacitors. In practice, on this device, we were not able to find anything that could look like an AES in these signals.

Additionally, leakage assessment tools such as the Welch t-test or related techniques [13] cannot be used without synchronous observations of the cryptographic signals. This point is also highlighted in [25], where the authors are not able to identify the HW AES signal unless they somehow modify the chip behavior to fallback

in a CPU mode for memory copy.

Physical alteration of the device can address the issue of PoP assembly. As the target cryptographic operation happens at the very first stage of the boot sequence (ROM), the device does not have to boot completely in order to decrypt BL1. An attacker can take advantage of this particularity and desolder the PoP stack to allow a much simpler physical access. The device will continue booting normally until the initialization of the DRAM produces a deadlock in the second stage bootloader (BL2).

Indeed, our only goal is to retrieve the secret key allowing firmware decryption. As it is a static key, identical for all devices, it does not matter if several boards are broken in the process.

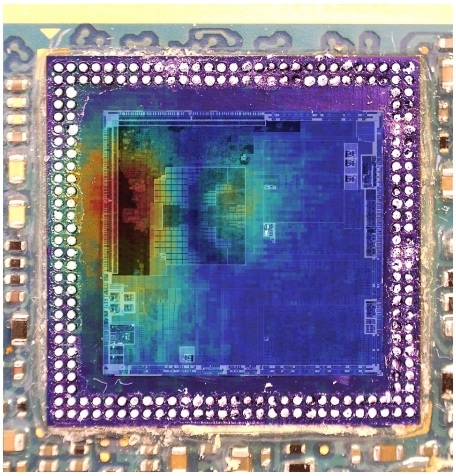
Removing the stacked memory can be performed manually with low-cost tools such as an hot air station [34]. After some practice, there is a quite high chance of success to not damage the SoC. As seen in Fig. 1, the use of heat-resistant adhesive on the component surrounding the SoC is of great help to avoid further rework on the board. Alternatively, a CNC milling machine could be used to finely carve out the solder balls at the edges of both packages.

In order to flash different firmware on the prepared device without its DRAM, few options are available. First, it is possible to cold-flash the external memory with dedicated recovery tools [27]. Else it is also possible to make the SoC boot on other external media, such as an SD card or a serial port. Indeed, disabling the flash often force the device to fallback in other boot modes. Alternatively, using the board schematics, it is possible to modify the resistors determining the bootmedia in Pseudo-code 1. With these techniques, we were able to conveniently load different firmware from an external SD card.

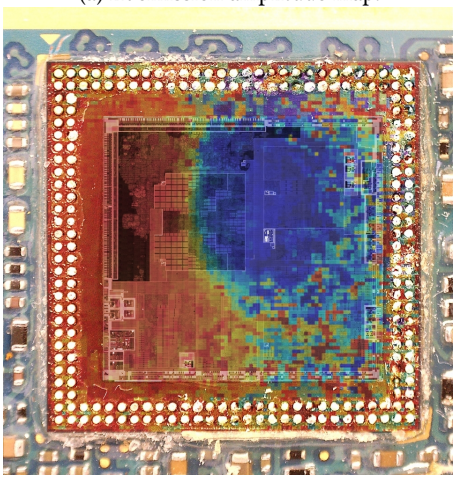
Thanks to these tricks, near-field EM measurements can be performed directly over the cryptoprocessor, which greatly simplifies the signal acquisition process. Our experimental bench, visible in Fig. 3, is comprised of a Lecroy WaveRunner 8-bit oscilloscope, a Langer H-Field probe (RF-B 0.3-3) and its 30dB low-noise amplifier (PA303). Fig. 1 shows the EM probe performing near-field acquisition over the SoC die. The chip backside is directly exposed, which allows to put the probe on contact with the silicon substrate and capture the fields emitted by logic layers underneath.

Still, finding the precise location of the AES computation over the SoC remains challenging and highly depends on the target hardware. A block cipher logic is typically comprised of few thousands transistors while the SoC actually contains several billions of them. We did not succeed in finding the cryptoprocessor using visual inspection. However, searching for an EM signature that could correspond to the cryptographic operations took several hours but was achievable. The goal was to find a repeated signal happening after copy of BL1 from flash, with main components at low frequency (hundreds of MHz). In the meantime, CPU activity was expected to be lower than usual.

To exhaustively find the locations of interest over the die, a thermography was also performed.



(a) IR emission amplitude map.



(b) In-phase with AES in red, Out-of-phase in blue.

Figure 5: Lock-in Thermal Imaging of the SoC during AES

3.2 Lock-in Thermography

In some cases, the attacker can have access to the cryptoprocessor hardware outside of the secure boot context. If so, lock-in thermography [5] of the SoC can help at identifying areas of interest precisely [10].[1]

Lock-in thermography is based on thermal analysis of the die using infrared (IR) measurements. For that, the nominal thermal behaviour is modulated at a known frequency. In this case, the modulation is software based and realized by cycling through phases of encryption (warming up) and CPU wait (cooling down). A Fourier transform applied on the infrared signal at the modulation frequency provides the main characteristics of the cartography: amplitude and phase. Scanning the chip and acquiring measurements at each location allow precise thermal maps of the die to be built.

Fig. 5 represents the results of an IR analysis conducted over the SoC with (a) the amplitude map and (b) the phase map. On

[1] The secure boot vulnerability allowed to perform this analysis with custom software, but its applicability to the boot sequence remains an open question.

both images, warm colors show respectively areas of power consumption and in-phase activity. Conversely, cold colors are used for weak power consumption and out-of-phase activity. Looking at the amplitude map, two main areas are distinguishable. The area of highest consumption, on the left, is confirmed by the phase image to be the cryptoprocessor. Indeed, it is warming up during phases of encryption. The second area therefore corresponds to the CPU performing timer checks during cooling phase with respect to the modulation timings.

Consequently, this thermal lock-in scan allows an attacker to reduce the search space and directly target the EM emissions of the cryptoprocessor. It could probably be applied to identify the location of other peripherals if needed.

3.3 Near-field Electromagnetic Emissions

Once a correct probe position over the cryptoprocessor is found, it is possible to trigger the oscilloscope directly on any cryptographic event. In addition, the AES computations can be identified in the acquired signals by counting the number of blocks manipulations. In Fig. 6, the EM signal is composed of a steady 200 MHz clock, where multiple AES decryptions occur, visible in the signal envelope.

A close-up of this signal in Fig. 6 illustrates the computation flow of the firmware decryption. Each clock cycle of the active area is associated to an AES round computation. During idle phase, a DMA is responsible for moving data in and out of the cryptoprocessor. Even if the AES is identified clearly in the signal, the overall signal-to-noise ratio remains low, as the peaks amplitude only increase by about 10% during cryptographic activity. This weak signal might explain why power and far-field side-channels were unsuccessful.

In this context, let us define the following terms: an input is a unique ciphertext, being a piece of correctly authenticated firmware to be decrypted by the cryptoprocessor; a trace corresponds to the recorded side-channel signal of a whole firmware decryption, and an observation is a part of the signal corresponding to a sole execution of AES.

Finally, by repeatedly acquiring the signal of AES-CBC decryption and extracting each individual block cipher from the traces, a statistical analysis of the relationship between processed data and physical observations can be performed. Similarly to usual SCA, this analysis can be done with a really large set of observations. However, because the number of inputs is limited by authentication prior to the decryption, there are many observations of the same deciphering. Therefore, the set of inputs processed is extremely reduced. This is a key difference that should limit the efficiency of SCA by reducing the exploration of the statistical space from 2^{128} to only a few thousands.

In order to finely characterize the cryptoprocessor side-channel leakage, we performed an acquisition campaign of 2500 traces and 5000 inputs, which corresponds to 12.5 million AES observations overall. It can be performed in about 3 h thanks to the cryptoprocessor efficiency.

As taking control of the hardware cryptoprocessor is enabled by the arbitrary code execution presented in Section 5, we redeveloped a driver and modified the secret key. This dataset can thus be shared with interested researchers, without disclosing the manufacturer's secret key.

4 SECRET KEY RECOVERY

4.1 Leakage Analysis

In the context of this attack, two solutions are available. First, the analyst can work directly on the trace set; or average all observations sharing the same input data, and attack this reduced trace set, which is lighter and faster from a computational point-of-view.

The two solutions turn out to be equally efficient at retrieving the secret key. It is expected when looking at Pearson correlation formula, as the covariance (numerator) is identical in the two cases.

A classical correlation analysis [6] is able to retrieve each of the 128 bits of secret key. Fig. 7 (top) depicts the correlation found for the correct byte coming apart from all the other candidates. The evolution of the score in convergence traces shows that very few inputs are required to deduce that this byte is legitimate.

The overall attack can retrieve the key with only two firmware when averaging 2500 traces together, as shown by the GE in Fig. 8. When limited to a single firmware of 1000 inputs, only a small bruteforce of 2^{32} candidates is required to find the key. Otherwise, it is also possible to acquire more traces to increase the amount of observations of the same decryption and reduce measurement noise.

Without dedicated side-channel protections, it is straightforward to conduct an attack on this device. That being said, the challenges of signal acquisition had to be overcome beforehand.

4.2 Attack Extrapolation

This section is focused on a methodology to better estimate the number of inputs that the device is allowed to decrypt without threatening the secret key.

When limited to few unique inputs, the previous SCA seems to struggle at retrieving the key. Our testing showed that even when averaging more traces, the attack is no more successful.

Our hypothesis, which is confirmed experimentally, is that the cryptoprocessor is leaking information about the 128 bits of the AES state simultaneously. As a result, when attacking a key byte, algorithmic noise coming from all the others will disrupt the signal. When adding new traces of the same decryption, the algorithmic noise remains. Nevertheless, the device can boot as many times as the attacker wants, which allows for a very large number of traces/observations that remove measurement noise.

As the cryptoprocessor is focused on performance, it is possible to capture hundred millions observations within hours. The question is then to know if an attacker could retrieve the secret key with 1000 inputs, 500, or less, given an unlimited number of observations. If not, it will be possible to leverage this limitation as a countermeasure.

To address this issue, we deeply analyzed the leakage model of the device. This characterization requires knowledge of the secret key. Fig. 9 shows the relationship between the estimated value (i.e.

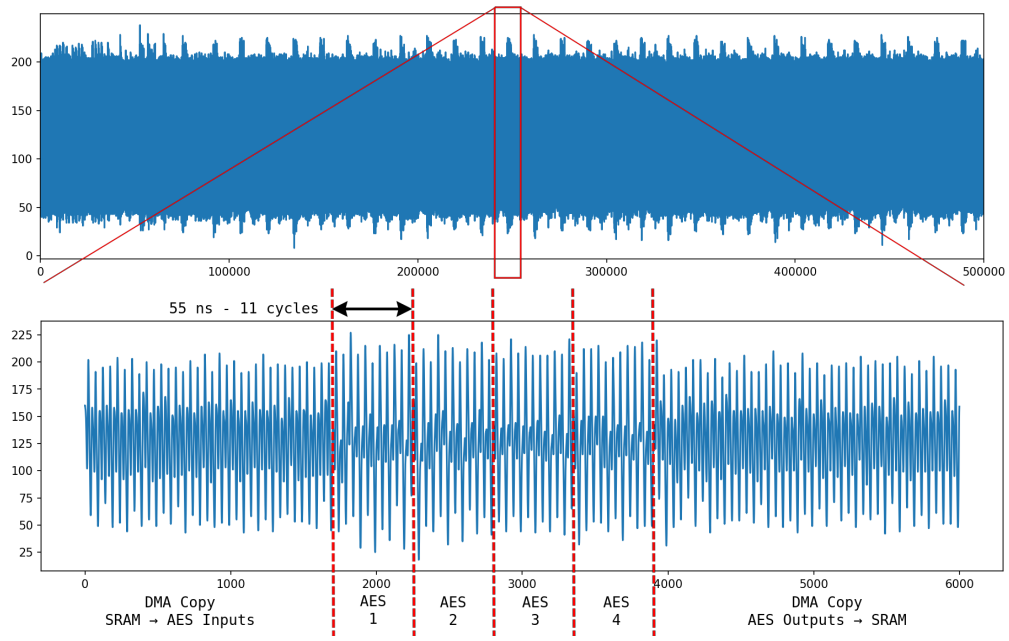


Figure 6: (top) Signal signature of firmware decryption - (bottom) Close-up on AES computation

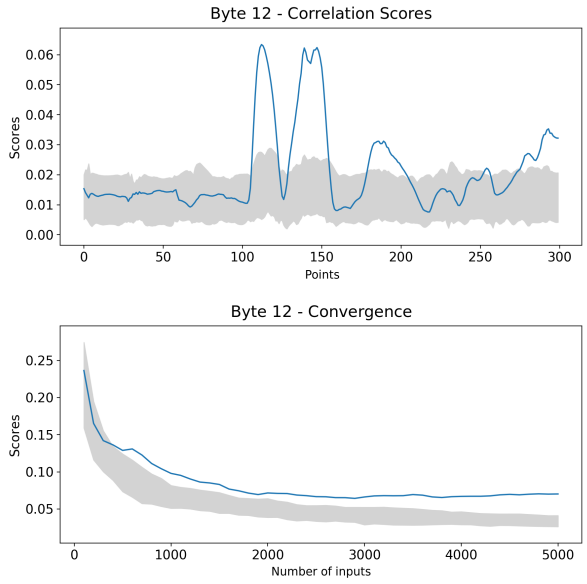


Figure 7: Attack results on byte 12 - (a) CPA scores (b) Convergence traces

the intermediate data targeted by the attack) and the value that is observed in the signal of that trace. The violin plot gives a clue of the distribution of observed values for each processed data; while the middle dash of each segment represents the average observed value. A linear regression in orange confirms that the cryptoprocessor is leaking all 128 bits of information simultaneously, with a strong correlation $\rho = -0.73$.

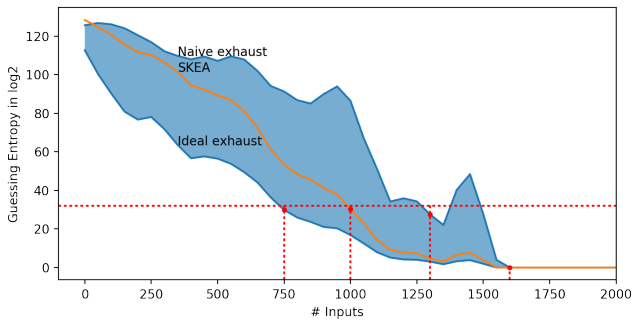


Figure 8: Guessing Entropy of the attack targeting the cryptoprocessor AES decryption, with 2500 averaged traces

The leakage strength, though, is very small: the signal amplitude decreases by 0.15 point for each additional bit set in the intermediate state of the computation. Which also indicates that an 8-bit oscilloscope might not be precise enough to capture fine changes in this configuration.

Thanks to this model, an empirical dataset can be crafted, containing the same input data as the one used for the attack, but removing all measurement noise and imprecisions. This dataset ensures perfect correlation with the model, and is left only with algorithmic noise induced by simultaneous leakage of all bytes.

Attacking such a dataset corresponds in practice to an extrapolation of the attack with an infinite number of traces, or in other words, infinite signal-to-noise ratio.

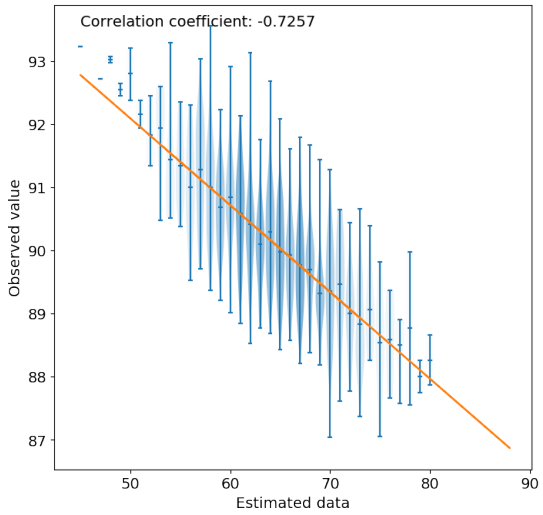


Figure 9: Linear leakage model of the device over 128 bits, using averaged traces

Fig. 10 gives an overview of the impact of algorithmic noise. A correlation analysis that would normally retrieve the key in about 10 traces requires around 700 in the context of simultaneous leakage of 128 bits.[2]

This graph will help gauge the security level achieved by a key usage limitation countermeasure, as explained in Section 6.3.

5 FIRMWARE ANALYSIS

Firmware encryption is the strongest barrier against reverse engineering of the boot sequence. Finding the decryption key is therefore an entry point for different kinds of attacks. Previous work, targeting higher level of the boot sequence, have shown that most implementations have flaws that could potentially be exploited [30].

Recovering the firmware decryption key allows an attacker to reverse-engineer the boot sequence at very early stages. This new entry-point can reveal major security issues in the secure boot sequence. For example, an attacker can quickly study the differences found between firmware upgrades in order to find what vulnerabilities were fixed, and deploy a malicious exploit targeting the previous versions.

[2] This resulting entropy depends on the input data distribution. In practice, as the firmware is ciphered, the input can be considered as uniform, which leads in average to the presented results. Still, some variations from this graph are to be expected.

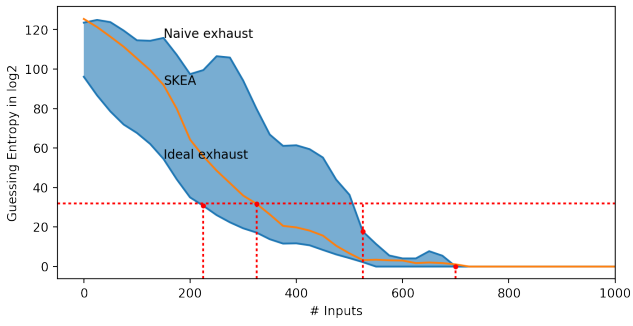


Figure 10: Forecast security bounds of the attack

In our practical example, several issues could be witnessed thanks to analysis of the decrypted software:

Decryption Key Reuse:
The secret key found via SCA can quickly be tested to decrypt various firmware packages provided by the manufacturer. It turns out that the same master key was used many times: all firmware updates for this particular phone model, as well as many mobiles based on the same SoC, including development boards. Such a decision exposes more code to be reversed, and implies that a single attack can affect a wider scope of devices. It also eases SCA, as it allows a wider statistical exploration of the link between ciphered data and observations. In our example, using multiple firmware enables retrieving the key without bruteforce.

Incautious Signing:
When looking at development or early released firmware, it appears that the security features, such as signature verification, are not fully implemented. Still, the development code was signed with the same private key and packaged in the same format than for any smartphone on the field. Without anti-downgrade protections, it is then straightforward to run malicious code by simply using an insecure firmware, not intended to be run on the final product. This allows arbitrary code execution in secure world, at bootloader stage.

Software Resilience:
In addition, the overall firmware security does not appear to be hardened. For example, the data parsers involved in BL2 loading are not resilient to unexpected length configurations. A software exploit might be possible, but was actually not necessary, thanks to the previously mentioned flaw.

Key Sanitization:
The secret key is not erased from the cryptoprocessor after its purpose is fulfilled. Hence, the following entities of the chain-of-trust might be able to reuse the AES without reconfiguration of the key. It becomes even more problematic if the device exposes an API to the user, for example through a Linux driver or a TEE service.

To sum up this analysis, it is extremely difficult to guarantee a firmware to be flawless, especially during development and product launch phases. Therefore, firmware encryption plays a decisive role in the overall security of the boot sequence, and should not be neglected.

Being able to read several iterations of the bootloader revealed critical flaws in its design, which result in arbitrary code execution in secure world. Such tool can definitely be exploited for disreputable operations. This can threaten trusted services of the TEE, such as mobile payment, DRM, or user data encryption. But can also be used to deeply establish a powerful malware in secure world, that can be made undetectable.

6 MITIGATIONS

The design issues and software weaknesses described in Section 5 can be addressed to strengthen the secure boot. But as it is difficult to guarantee flawless firmware all along the device life cycle, mitigating the side-channel attack scenarios can greatly enhance its security.

6.1 Decorrelation

Efficient countermeasures against SCA have been developed in the smartcard industry since [8]. In practice, delicate hardware design efforts are required to reduce the information leakage of the cryptographic operations [18]. These countermeasures will complicate the attack but might not prevent it.

However, decorrelation of the leaking physical quantity with the manipulated data can also be achieved at logical level. Applying a masking scheme on the entire combinatory logic of the algorithm is laborious though, as it requires to adjust the circuit to handle randomly masked data. Besides, it is necessary to validate experimentally that the protection is also effective at hardware level, as physical effects can introduce unexpected bias [9], that can be exploited by an adversary [23]. We believe that Boolean masking is not mandatory on this particular use case, as other mitigation techniques are also viable, and might be more efficient.

In the following Sections, the paper describes two countermeasures particularly suited to prevent the presented attack scenario.

6.2 Shuffling

In the smartcard industry, it is common to see up to 15 fake executions of an algorithm along with a genuine one hidden among them. Such hiding countermeasure is efficient against statistical attacks, as the analyst has no clue of where the information stands. He is forced to consider all fake operations as legitimate, which significantly increases noise. This technique, though, implies a severe performance drawback.

In the case of AES-CBC, the sequence of block decryptions can be shuffled randomly[3] without performance loss. Doing so, the attacker is not able to tell which block is being handled when. Given the length of a firmware, and assuming that the attacker is not able to recover the disposition of the AES blocks, the leakage will be significantly diluted within the entire decryption process.

From a theoretical point of view, when the leakage is uniformly located over t samples, the attack will require *at least* t times more traces to succeed [32]. At a rate around 1 trace per second enforced by device reboot time, the presented attack can be performed in around 1 h. After shuffling, it would then take more than 40 days of continuous acquisition, which implies new challenges, such as environmental stability of the measurements. The goal of such countermeasures is simply to seek for impracticality of the threat.

6.3 Key Usage Limitation

Manufacturers should consider using a firmware-dependent key derivation algorithm, seeded for example on a firmware version number. Such mechanism limits the key usage to a single firmware. It will not be possible to use different firmware in the same statistical attack.

Furthermore, it would reduce the scope of the attack significantly. Indeed, retrieving the key would only affect a single firmware, and no other version or device.

In our example, a single firmware is already vulnerable to side-channel analysis. Therefore, dynamic re-keying protocols could be considered to further reduce the amount of data processed per key [19, 26]. With such technique, each secret key is used so little that it is no longer possible to extract it by SCA. Yet, the protocol should be designed with care and expertise, otherwise opening new attack paths, for example on the key derivation algorithm.

More specifically, looking at the extrapolated security bounds of Fig. 10: with 400 inputs and infinite amount of observations, an attacker should be able to extract the secret key with our side-channel attack. However, if the key is used only to handle 100 input blocks, it should be safe from this attack. The guessing entropy is reducing so quickly that even limiting to 200 blocks corresponds to a risk that should not be undertaken.

7 RESPONSIBLE DISCLOSURE

Our results have been communicated to the SoC manufacturer following their responsible disclosure guidelines. The confidentiality of BL1 is not a critical issue in their security policy. Indeed, it does not directly allow bypass of the secure boot. Moreover, the proposed countermeasures imply modifications of the BootROM, which cannot be patched. As a result, no further action was taken, but the reference of the chip was not exposed.

8 CONCLUSION

The scenario unfolded in this paper demonstrates that side-channel attacks can enable decryption of closed-source firmware. Practical results on a mobile phone illustrate the use of near-field measurements to extract the key from the System-on-Chip cryptoprocessor. The complexity of the attack mostly resides in the acquisition setup, and requires some tricks to get close access to the die.

Such tool is especially valuable when no software exploit can be found, as it exposes new entry-points for reverse engineering and analysis of the boot sequence.

Although complex, our attack scenario is not out of range. Many examples on popular devices show that the strong software security barriers of secure boot, and the stakes of breaking it, have been motivating malicious individuals to attempt physical attacks.

Our results show that side-channel threat should be seriously taken into consideration when designing a product, especially for assets as strategic as the secure boot sequence. The described methodology is intended for a large number of IoT devices but particularly applies to security demanding endpoints such as mobile phones.

Low-cost protections are suggested and could have been used to significantly mitigate the risk. However, as with the security industry standards, practical evaluation should be systematically performed on the final product to make sure the protections are efficient.

[3] This property does not apply to CBC encryption.

REFERENCES

[1] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. 2002. The EM Side-Channel(s). In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers (Lecture Notes in Computer Science)*, Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar (Eds.), Vol. 2523. Springer, 29–45. https://doi.org/10.1007/3-540-36400-5_4

[2] Monjur Alam, Haider A. Khan, Moumita Dey, Nishith Sinha, Robert Locke Callan, Alenka G. Zajic, and Milos Prvulovic. 2018. One&Done: A Single-Decryption EM-Based Attack on OpenSSL's Constant-Time Blinded RSA. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. 585–602. https://www.usenix.org/conference/usenixsecurity18/presentation/alam

[3] Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. 2015. DPA, Bitslicing and Masking at 1 GHz, See [14], 599–619. https://doi.org/10.1007/978-3-662-48324-4_30

[4] Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, and Marc Witteman. 2015. Fast and Memory-Efficient Key Recovery in Side-Channel Attacks. In *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers (Lecture Notes in Computer Science)*, Orr Dunkelman and Liam Keliher (Eds.), Vol. 9566. Springer, 310–327. https://doi.org/10.1007/978-3-319-31301-6_19

[5] Otwin Breitenstein, Wilhelm Warta, and Martin Langenkamp. 2010. *Lock-in thermography: Basics and use for evaluating electronic devices and materials*. Vol. 10. Springer Science & Business Media.

[6] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings (Lecture Notes in Computer Science)*, Marc Joye and Jean-Jacques Quisquater (Eds.), Vol. 3156. Springer, 16–29. https://doi.org/10.1007/978-3-540-28632-5_2

[7] Christian Cachin. 1997. *Entropy measures and unconditional security in cryptography*. Ph.D. Dissertation. ETH Zurich. http://d-nb.info/950686247

[8] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. 1999. Towards Sound Approaches to Counteract Power-Analysis Attacks, See [35], 398–412. https://doi.org/10.1007/3-540-48405-1_26

[9] Thomas De Cnudde, Maik Ender, and Amir Moradi. 2018. Hardware Masking, Revisited. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018, 2 (2018), 123–148. https://doi.org/10.13154/tches.v2018.i2.123-148

[10] Maxime Cozzi, Jean Marc Gallière, and Philippe Maurine. 2018. Thermal Scans for Detecting Hardware Trojans. In *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings (Lecture Notes in Computer Science)*, Junfeng Fan and Benedikt Gierlichs (Eds.), Vol. 10815. Springer, 117–132. https://doi.org/10.1007/978-3-319-89641-0_7

[11] Franco L. Fiori. 2008. Reducing SoC electromagnetic emissions by design. In *15th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2008, St. Julien's, Malta, August 31 2008-September 3, 2008*. IEEE, 422–425. https://doi.org/10.1109/ICECS.2008.4674880

[12] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. 2016. ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 1626–1638. https://doi.org/10.1145/2976749.2978353

[13] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. 2011. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, Vol. 7. 115–136.

[14] Tim Güneysu and Helena Handschuh (Eds.). 2015. *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. Lecture Notes in Computer Science, Vol. 9293. Springer. https://doi.org/10.1007/978-3-662-48324-4

[15] Keith Gutierrez and Gerald Coley. 2009. *PCB Design Guidelines for 0.4mm Package-On-Package*. Application Report SPRAAV1B. Texas Instruments. http://www.ti.com/lit/an/spraav1b/spraav1b.pdf Accessed Online (Sept. 2019).

[16] Michael Hutter and Jörn-Marc Schmidt. 2013. The Temperature Side Channel and Heating Fault Attacks. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers (Lecture Notes in Computer Science)*, Aurélien Francillon and Pankaj Rohatgi (Eds.), Vol. 8419. Springer, 219–235. https://doi.org/10.1007/978-3-319-08302-5_15

[17] Auguste Kerckhoffs. 1883. La cryptographie militaire. *Journal des sciences militaires* vol. IX (January and February 1883), 5–38 and 161–191. https://www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf Accessed Online (Sept. 2019).

[18] Hyunmin Kim, Seokhie Hong, Bart Preneel, and Ingrid Verbauwhede. 2017. STBC: Side Channel Attack Tolerant Balanced Circuit with Reduced Propagation Delay. In *2017 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2017, Bochum, Germany, July 3-5, 2017*. IEEE Computer Society, 74–79. https://doi.org/10.1109/ISVLSI.2017.22

[19] Paul Kocher. 2005. Design and validation strategies for obtaining assurance in countermeasures to power analysis and related attacks. In *Proceedings of the NIST Physical Security Workshop*, Vol. 46.

[20] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2018. Spectre Attacks: Exploiting Speculative Execution. *CoRR* abs/1801.01203 (2018). arXiv:1801.01203 http://arxiv.org/abs/1801.01203

[21] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis, See [35], 388–397. https://doi.org/10.1007/3-540-48405-1_25

[22] Ulrich Kühn, Andrei Pyshkin, Erik Tews, and Ralf-Philipp Weinmann. 2008. Variants of Bleichenbacher's Low-Exponent Attack on PKCS#1 RSA Signatures. In *Sicherheit 2008: Sicherheit, Schutz und Zuverlässigkeit. Konferenzband der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 2.-4. April 2008 im Saarbrücker Schloss. (LNI)*, Ammar Alkassar and Jörg H. Siekmann (Eds.), Vol. 128. GI, 97–109.

[23] Itamar Levi, Davide Bellizia, and François-Xavier Standaert. 2019. Reducing a Masked Implementation's Effective Security Order with Setup Manipulations And an Explanation Based on Externally-Amplified Couplings. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019, 2 (2019), 293–317. https://doi.org/10.13154/tches.v2019.i2.293-317

[24] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. 2018. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. 973–990. https://www.usenix.org/conference/usenixsecurity18/presentation/lipp

[25] Jake Longo, Elke De Mulder, Dan Page, and Michael Tunstall. 2015. SoC It to EM: ElectroMagnetic Side-Channel Attacks on a Complex System-on-Chip, See [14], 620–640. https://doi.org/10.1007/978-3-662-48324-4_31

[26] Marcel Medwed, Christophe Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. 2011. Fresh Re-keying II: Securing Multiple Parties against Side-Channel and Fault Attacks. In *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers (Lecture Notes in Computer Science)*, Emmanuel Prouff (Ed.), Vol. 7079. Springer, 115–132. https://doi.org/10.1007/978-3-642-27257-8_8

[27] Multi-COM. [n.d.]. VR-Table, eMMC JTAG FBus. https://vr-table.com/features.php Accessed Online (Sept. 2019).

[28] Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard. 2016. DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, Thorsten Holz and Stefan Savage (Eds.). USENIX Association, 565–581. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/pessl

[29] Jean-Jacques Quisquater and David Samyde. 2001. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings (Lecture Notes in Computer Science)*, Isabelle Attali and Thomas P. Jensen (Eds.), Vol. 2140. Springer, 200–210. https://doi.org/10.1007/3-540-45418-7_17

[30] Nilo Redini, Aravind Machiry, Dipanjan Das, Yanick Fratantonio, Antonio Bianchi, Eric Gustafson, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. 2017. BootStomp: On the Security of Bootloaders in Mobile Devices. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, Engin Kirda and Thomas Ristenpart (Eds.). USENIX Association, 781–798. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/redini

[31] Etienne Sicard (eds.) Sonia Ben Dhia, Mohamed Ramdani. 2006. *Electromagnetic Compatibility of Integrated Circuits: Techniques for low emission and susceptibility* (1 ed.). Springer US. http://gen.lib.rus.ec/book/index.php?md5=daff750da7c9e67635c190e9274becd7

[32] Youssef Souissi. 2011. *Optimization methods for side channel attacks. (Méthodes optimisant l'analyse des cryptoprocesseurs sur les canaux cachés)*. Ph.D. Dissertation. Télécom ParisTech, France. https://tel.archives-ouvertes.fr/pastel-00681665

[33] Albert Spruyt and Niek Timmers. 2017. Constructing AES-CBC Shellcode. *International Journal of PoC||GTFO* 0x17 (December 2017), 5–8. https://www.alchemistowl.org/pocorgtfo/pocorgtfo17.pdf

[34] Aurélien Vasselle, Hugues Thiebeauld, Quentin Maouhoub, Adele Morisset, and Sebastien Ermeneux. 2018. Laser-Induced Fault Injection on Smartphone Bypassing the Secure Boot. (2018). https://doi.org/10.1109/TC.2018.2860010

[35] Michael J. Wiener (Ed.). 1999. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Lecture Notes in Computer Science, Vol. 1666. Springer. https://doi.org/10.1007/3-540-48405-1