Leaky Noise: New Side-Channel Attack Vectors in Mixed-Signal IoT Devices

Dennis R. E. Gnad, Jonas Krautter and Mehdi B. Tahoori

Karlsruhe Institute of Technology (KIT), Germany.
{dennis.gnad,jonas.krautter,mehdi.tahoori}@kit.edu

Abstract. Microcontrollers and SoC devices have widely been used in *Internet of Things* applications. This also brings the question whether they lead to new security threats unseen in traditional computing systems. In fact, almost all modern SoC chips, particularly in the IoT domain, contain both analog and digital components, for various sensing and transmission tasks. Traditional remote-accessible online systems do not have this property, which can potentially become a security vulnerability. In this paper we demonstrate that such mixed-signal components, namely ADCs, expose a new security threat that allows attackers with ADC access to deduce the activity of a CPU in the system. To prove the leakage, we perform leakage assessment on three individual microcontrollers from two different vendors with various ADC settings. After showing a correlation of CPU activity with ADC noise, we continue with a leakage assessment of modular exponentiation and AES. It is shown that for all of these devices, leakage occurs for at least one algorithm and configuration of the ADC. Finally, we show a full key recovery attack on AES that works despite of the limited ADC sampling rate. These results imply that even remotely accessible microcontroller systems should be equipped with proper countermeasures against power analysis attacks, or restrict access to ADC data.

Keywords: microcontroller · side-channel · leakage assessment · ADC · noise · power analysis · on-chip · remote · software · internet-of-things · mbedtls · freertos · cpa

1 Introduction

Traditional applications of microcontrollers and SoCs are embedded systems with dedicated and limited interfacing capabilities, which typically have to fulfill the requirements of low-cost and energy efficiency. As these devices are recently used in *Internet of Things* (IoT) applications, their security against remote attacks becomes a major concern [MS10, RNL11], as several breaches have already been reported [AAB+17, RSWO17]. Unlike security for highly dependable server systems, the energy efficiency imposes lightweight cryptography [PPK+07]. However, there are potentially new security threats associated with the underlying technology, which is not widely researched yet [CPM+18].

One of these technology-dependent threats is due to their *Mixed-Signal*-integration of analog and digital logic on the same SoC. Inside the SoC, cross-coupling or voltage fluctuations caused by the digital part can bias the analog circuit [SLMW93]. This bias has been shown to even affect radio transceivers, such that electromagnetic (EM) side-channel attacks [GMO01] can be performed in proximity of up to 10 meters [CPM+18]. One of the most widespread analog circuits that is integrated in SoCs is an *Analog-to-Digital Converter* (ADC), essential in many complex modules and applications such as temperature sensors, wireless transceivers or multimedia audio applications, just to name a few. Even more, multiple of such systems are often connected into complete sensor networks made out of *smart sensors* [Sta08], and in so-called Edge-IoT devices [SD16]. This is evident by

IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925, Vol. 2019, No. 3, pp. 305–339
DOI:10.13154/tches.v2019.i3.305-339

the support in Amazon [Ama18] and Microsoft [Mic18] IoT cloud applications for many SoC and microcontroller platforms.

In such mixed-signal SoCs, the analog and digital subsystems typically have a common power supply, and are spatially close on the same die or package, leading to either crosstalk or voltage fluctuations from the digital logic propagating into ADC measurement results, known as one of the challenges in mixed-signal design [AGR99].

For FPGAs, it has already been shown that it is possible to mount power analysis attacks just through software configuration [SGMT18a, ZS18, RPD+18]. Those attacks were recently extended to other chips on the same printed circuit board (PCB) [SGMT18b], possible through a shared power supply. For small wireless systems it has been shown that electromagnetic side-channel leakage can be observed in close proximity of a few meters to the device, without having to probe it directly [CPM+18]. Thus, there is increasing evidence that power analysis attacks, originally considered a local issue, can also be used in remote exploits.

In this paper, we present another type of power analysis side channel that can be exploited through software, potentially remotely. We show that ADC noise, which is usually characterized using statistical methods [LRRB05, Nat15, AR06], is not just statistical noise, but is correlated to the activity in the digital subsystem. To assess the capability of this side-channel, we perform leakage assessment [GGJR+11, SM15] on multiple platforms. Afterwards we show a successful key recovery attack on the Advanced Encryption Standard (AES). In summary, we make the following contributions:

- First assessment of ADC noise as a software-only power analysis side-channel, which could be used remotely.
- Elaborate leakage assessment of this side-channel on a range of systems under different conditions, evaluated on a real-world cryptographic library.
- Successful ciphertext-based key recovery on AES using ADC noise.

In the remaining paper, we first explain preliminaries in Section 2 regarding our adversarial model and the essential background information, including related work. We then explain our experimental setup in Section 3. Our results are presented in Section 4, and discussed in Section 5. Finally, the paper is concluded in Section 6.

2 Preliminaries

2.1 Mixed-Signal Integrated Circuits

Many integrated circuits nowadays are not pure digital or pure analog, but typically contain subsystems of both types of circuits. The more applications a single chip has to support, the more likely the analog and digital blocks will be integrated together. One of the biggest challenges in mixed-signal design is the noise susceptibility of the analog subsystem, which gets affected by the higher-frequency and higher-power digital subsystem.

First of all, if the digital logic consumes high power, a voltage drop in the power supply becomes visible through slightly reduced supply voltage, biasing analog components [AGR99]. Especially a dI/dt-based voltage drop has become predominant in recent years, which does not depend on absolute current, but the ratio of current *change* over time [MF04, ASM07]. For very high frequency current changes, this voltage drop might just be observable inside the chip, with voltage fluctuations traveling through the chip-internal power supply mesh[DWB15, GOKT18], or the common substrate of the whole die [SLMW93].

An additional effect to voltage fluctuations is chip-internal crosstalk from electromagnetic (EM) coupling. Depending on the frequency of a signal pulsed on a wire, the wire

acts as a strong or weak radio transmitter, which also affects nearby wires, biasing wire delays through inductive or capacitive coupling effects [SAHK98]. Digital circuits are designed with a specific noise margin to prevent bit flips during normal operation, but analog circuits can be biased through EM [Fio07].

In summary, it is usually hard to guarantee that digital circuits have zero effect on an adjacent analog circuit. Instead, a mixed-signal chip is designed such that the noise margin is considered sufficient for the application requirements. We will show that security requirements may impose much higher restrictions on the allowed noise levels, at least regarding the noise caused by digital components.

2.2 Analog-to-Digital Converters

Various types of ADCs are implemented in integrated circuits [LRRB05]. One of the most common and cheaper general-purpose designs is a *Successive Approximation Register* (SAR) ADC, which is also the type of ADC utilized in all of the systems evaluated in this paper. We discuss here to which extent ADCs can be influenced by digital noise in mixed-signal chips.

Classical ADC noise characterization analyzes the *effective number of bits* (ENOB) of an ADC over its actual number of bits [LRRB05]. This effective dynamic range depends on a number of distortion parameters, such as the signal-to-noise ratio (SNR) and spurious-free dynamic range (SFDR). These parameters are typically characterized and tested for the ADC circuit itself, and do not specifically involve noise from the digital subsystem of a mixed-signal circuit [Nat15, AR06].

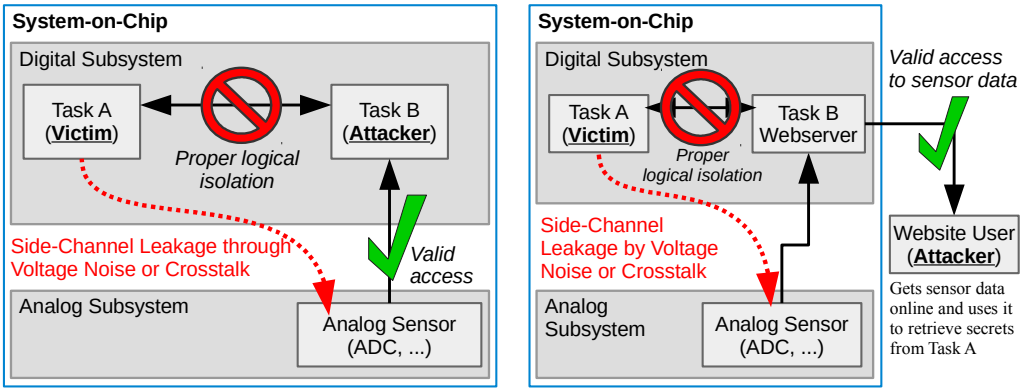
SAR ADCs use multiple discrete timesteps over which the conversion is performed, where each timestep resembles one of the output bits of the ADC, i.e. a 12bit ADC requires at least 12 ADC-internal clock cycles. In each of the timesteps, the current ADC input value is evaluated against a specific voltage level connected to a shared analog comparator. If we assume chip-internal noise affects a fixed voltage in a measurement result, the least significant bits (LSBs) get affected more than the most significant bits (MSBs). Thus, typically only during the conversion of the lower bits, a SAR ADC is susceptible to chip-internal noise.

Another ADC type that is often integrated in SoCs or microcontrollers is a Sigma-Delta-ADC ($\Sigma\Delta$-ADC), which also operates over multiple clock cycles. They perform a sort of approximation over multiple clock cycles. In each clock cycle, the difference of a previously measured value is compared with the input value and is integrated over multiple cycles. This integrated value can be affected by chip-internal noise on any of the integration clock cycles, and affect the LSBs. Through the nature of differential sampling, some of the noise can be rejected. However, noise affects a larger time window of the ADC result than for SAR ADCs, i.e. not just during measurement of the LSBs. So overall, we assume chip-internal noise can affect $\Sigma\Delta$-ADCs about the same as SAR ADCs.

2.3 Adversarial Model

The basic principle of our adversarial model has two variants, which are summarized in Figure 1. We consider a software-based model, in which an attacker has full or partial access to ADC data, and a victim which is running cryptographic code. We assume the ADC is read during the time the cryptographic code is executed, and the adversary has access to that data, either directly through another task in the system, or indirectly through a webserver hosting sensor data. Typically an ADC can be read simultaneously by using a second core, Direct Memory Access (DMA), or interrupt-driven operation, available in most microcontroller architectures.

This adversarial model applies to a broad range of applications and devices. For instance, IoT applications based on smaller microcontrollers or SoCs are within that model,



(a) Variant of the adversarial model in which a malicious task (Task B) could gain knowledge of secret information processed in the victim (Task A), circumventing any access restrictions.

(b) Variant of the adversarial model where side-channel leakage is embedded in sensor data that leaves the system. An external attacker can then use the sensor data to retrieve secrets from Task A.

Figure 1: Basic principle of the two variants of our adversarial model considered in this paper. In both cases an ADC in the Analog Subsystem is biased from Task A in the digital subsystem. This bias can contain secret information that Task A processes.

available from a large range of manufacturers. The *Amazon IoT FreeRTOS* project [Ama18] has direct support for microcontroller boards from various manufacturers. Microsoft Azure also supports various systems, where many smaller mixed-signal microcontroller-based systems are included [Mic18].

2.4 Leakage Assessment

In order to systematically evaluate exploitable leakage in the investigated microcontroller systems, we employ various leakage assessment methodologies based on *non-specific fixed-vs-random t-testing* [GGJR+11, SM15]. Utilizing Welch's t-test for evaluating side-channel security of hardware implementations has been introduced in the seminal work by Goodwin et al. in 2011 [GGJR+11]. The basic principle of all variants of this leakage assessment methodology is to test if statistical differences can be found in recorded side-channel data of the same cryptographic operation with different input values. Typically, two different sets of input data are chosen. For each set, side-channel traces are recorded, followed by an evaluation regarding their distinguishability.

Although a t-test evaluation does not allow an attacker to recover secret keys and break cryptographic implementations, this method is more generic, as it does not require to establish a hypothetical leakage model such as Correlation Power Analysis (CPA) [BCO04]. It avoids the fixation on a specific intermediate value, such as a specific AES round. We briefly explain non-specific test-vector leakage assessment (TVLA), which we used to evaluate basic side-channel attack vulnerability in this work.

For a secret key encryption $\mathrm{Enc}(k, m)$ with secret key k and plaintext m, we choose a key k for all our experiments and generate a set $M_R = \{m_{r1}, m_{r2}, ..., m_{rn}\}$ of random plaintexts m_{ri} as well as a single fixed plaintext m_{fixed}. Then, the tested platform alternatingly computes $\mathrm{Enc}(k, m_{\text{fixed}})$ and $\mathrm{Enc}(k, m_{ri}) \forall m_{ri} \in M_R$, while the ADC is sampled on the same device. On the two sets of traces, the average, variance and higher order moments for every sample time step are computed [SM15]. The obtained values can be used to compute arbitrary order t-values for every sample time step as shown in Equation 1:

$$t = \frac{\mu_r - \mu_{fixed}}{\sqrt{\frac{s_r^2}{n_r} + \frac{s_{fixed}^2}{n_{fixed}}}} \tag{1}$$

In Equation 1, μ_r and μ_{fixed} are the raw averages of the two sets of traces during encryption of m_r and m_{fixed}, respectively, at a specific sample time step for a first order t-test or the higher order central moments for a higher order t-test. Likewise, s_r^2 and s_{fixed}^2 correspond to the respective variances for a first order t-test and the higher order standardized moments for a higher order t-test. The amounts of random and fixed traces are n_r and n_{fixed}.

To prove a design secure against side-channel attacks using leakage assessment, it is usually recommended to select multiple different fixed plaintexts and perform a leakage assessment for each of them. As we do not want to prove security but rather want to show information leakage in ADC noise, we only evaluated a single fixed plaintext m_{fixed}.

Exploitable leakage is assumed for $|t| > 4.5$, a generally accepted threshold [GGJR+11, SM15]. A value of $|t| > 4.5$ relates to a confidence of > 0.99999 that the traces collected from random encryptions and those from fixed encryptions are samples drawn from different populations. In this paper, sampling is synchronized with the beginning of the encryption algorithm and we restrict the leakage assessment to the middle third, as recommended in [GGJR+11].

2.5 Correlation Power Analysis on AES

To recover secret keys of AES through power analysis, Correlation Power Analysis (CPA) with a leakage model is a well-known standard attack [BCO04]. In order to recover a secret AES key, an attacker collects a certain amount of power traces to eventually find a distinct correlation between the collected traces and a power consumption model of the correct key candidate. These power traces are collected during AES encryptions, where either plaintexts or ciphertexts are known to the attacker. Attacks are usually performed on single key bytes of the first or last round key, where the attacker is able to compute power consumption models for all 2^8 possible key byte values in negligible time. The classical correlation model from [BCO04] is based on the assumption that power consumption of computations depends on the Hamming distance between intermediate values, for instance after the *SubBytes* operation of AES:

$$P_{\text{hyp}} = \text{HW}(\text{SBox}^j(K_{\text{hyp}} \oplus S_i)) \tag{2}$$

In Equation 2, $\text{HW}(x)$ is the Hamming weight of x, $\text{SBox}^j(x)$ is the (inverted) *SubBytes* function of the AES algorithm and K_{hyp} is the key hypothesis byte. Depending on whether the first or the last round of the AES encryption is attacked, S_i is one byte from either the input plaintext or the output ciphertext, where $i \in 0, 1, ..., 15$. Moreover, SBox^j is the normal ($j = 1$) AES substitution function when the first round is attacked, and the inverted ($j = -1$) substitution when the last round is targeted.

Another possible Hamming weight model is based on the result of a T-table lookup. The AES algorithm can be optimized by implementing the *MixColumn* and *SubBytes* operations into a single table lookup, where each input byte yields a 32 bit output word. For CPA, the Hamming weight is then based on the output word of the T-table lookup function.

The CPA result for a single byte is based on computing the Pearson's correlation coefficient ρ_j between a hypothetical power model P_{hyp} and the actual measured value P_{trace_j} for every sampling step j, using all collected traces n:

$$\rho_j = \frac{n \cdot \sum P_{\text{hyp}} \cdot P_{\text{trace}_j} - \sum P_{\text{hyp}} \cdot \sum P_{\text{trace}_j}}{\sqrt{n \cdot \sum {P_{\text{trace}_j}}^2 - (\sum P_{\text{trace}_j})^2} \cdot \sqrt{n \cdot \sum {P_{\text{hyp}}}^2 - (\sum P_{\text{hyp}})^2}} \tag{3}$$

With a sufficient amount of traces, the correlation for the correct key byte value will eventually differ significantly at a specific sampling step from the correlations with the incorrect key byte values, allowing the attacker to determine the correct secret key byte.

A successful CPA depends on traces that are collected synchronous to the encryption algorithm. In this paper, we perform an alignment to reduce synchronization inaccuracies. We compute the total average trace over all collected traces and use a normalized cross-correlation based alignment algorithm. Each trace is shifted within a defined range and the normalized cross-correlation with the total average trace is computed as follows in Equation 4:

$$\rho_{cc}(s) = \frac{1}{\sigma \cdot \sigma_t} \sum_i (\mu_i - \mu) \cdot (t_{i+s} - \mu_t) \tag{4}$$

In the above equation, σ is the total standard deviation over all values, σ_t is the standard deviation over the current trace, μ_i is the total average at sampling point i, μ is the total average over all values, t_{i+s} is the current trace value at sampling point $i+s$ and μ_t is the total average of the current trace. The maximum cross-correlation value defines a new trace shifted by s, which is aligned with the total average.

2.6 Related Work

Fault or side-channel attacks that can be exploited even remotely through software are recently increasing. Probably the most impactful findings have been the Spectre and Meltdown speculative execution attacks that use cache timing side-channels as covert information channels [KGG+18, LSG+18]. Yet, even on the lower electrical level, software-based power analysis side-channels and fault attacks have been shown already [SGMT18a, KDK+14]. Fault attacks that can be exploited only through software, and thus often remotely, were already shown in a range of systems[KDK+14, GOT17, TSS17, KGT18], while software-based power side-channels have only been shown for FPGAs so far [SGMT18a, ZS18, RPD+18].

The earliest of these attacks has been *rowhammer* [KDK+14], which demonstrates that faults can be caused in DRAM through malicious access patterns on the memory. These access patterns use repetitive reading of the same memory line, which can cause faults in an adjacent line, leading to bit flips. Because that adjacent memory line might be an important part of protected operating system memory, certain bit flips can eventually lead to privilege escalation and grant root access. This attack has been first shown by programs running on the system, but later it has even been demonstrated through JavaScript in a web browser [GMM15].

Another example in which faults can be caused in software is an ARM-based SoC, secured by *ARM TrustZone*. TrustZone is designed to securely isolate various components of an SoC, and can also establish a *Trusted Execution Environment* (TEE) for trusted applications on the ARM cores. The *CLKSCREW* [TSS17] attack shows how power management can be reprogrammed and exploited by even untrusted applications on the ARM CPU to affect trusted applications or hardware in the SoC which otherwise should not be accessible. Using CLKSCREW, the used RSA signature scheme in TrustZone can also be subverted, leading to execution of self-signed applications.

Fault attacks have also been shown through software access to FPGAs. FPGAs get increasingly adopted as cloud accelerators, with prospects of virtualized multi-tenant FPGAs. In one attack [GOT17], severe faults can be provoked through voltage drops, leading to a denial of service of the complete FPGA accelerator board. These faults can be generated through the re-use of FPGA primitives as ring-oscillators, and enabling them in the right frequency pattern, to target the resonance frequencies of the respective power regulator. In a second work, it has been shown that these faults can also be caused in a more controlled way [KGT18]. Using these more fine-grained fault injections, differential fault attacks on other parts of the FPGA become feasible, which is a high risk for future multi-user FPGA applications.

To perform power analysis side-channel attacks through software, FPGAs provide sufficient configuration flexibility [SGMT18a, ZS18, RPD+18]. The threat model is similar

to fault attacks on multi-tenant FPGAs, in which one of the users attacks another to extract secret data. In these attacks, sensors are synthesized using FPGA logic, which is normally intended for digital circuits. Yet it is feasible to indirectly measure voltage through the speed differences of the components, which are affected by minor voltage fluctuations in the chip-internal power supply. A simple and fairly slow sensor is to count the amount of oscillations of one or more ring-oscillators [ZS18, RPD+18]. In another work, a sensor is used that measures the propagation of the clock signal through a predefined path. By counting the propagation depth into the path, an indirect voltage reading can be acquired [SGMT18a], reaching higher sampling rates than the oscillator-based sensor. Either way, the work showed sufficient side-channel leakage measured by the respective sensor variants for key recoveries of a simple AES [SGMT18a], and textbook RSA [ZS18].

Recently, it was shown how the noise in mixed-signal systems can be exploited by a semi-remote attack [CPM+18], i.e. from a short distance. In their work, small wireless IoT devices are attacked, which integrate the wireless radio circuit together with digital logic. When the digital logic performs cryptographic operations, leakage through the silicon substrate affects the analog radio circuit. If that circuit is used, a fraction of the side-channel leakage from the cryptographic operations can actually be observed in the electromagnetic wave emitted by the radio circuit into the nearby environment. In this way, EM leakage is essentially extended to a larger range than usual [LDMPT15]. This leakage was sufficient to extract the secret keys of a simple *tinyAES* implementation from within a distance of up to 10 meters, and 1 meter for *mbedTLS*.

A similar effect of leakage among chip components was shown for microcontrollers in [SPK+10], in which side-channel leakage could also be observed on digital port pins not connected to the power supply. Thus, it is also an indication that an ADC could observe such leakage if it is connected to a port pin from the inside.

3 Experimental Setup

We show the overview of our common experimental setup in Figure 2. The basic setup consists of a set of software components and different microcontroller boards that all have basic hardware features like ADC or UART modules. We use the software stacks provided by the board vendors themselves, which is very similar across the boards. The stacks are all based on the common security library *mbedTLS* [ARM16] and the real-time operating system *FreeRTOS* [BA19]. These serve merely as a readily available proof of concept platform. We do not exploit any particular vulnerabilities or side effects from this stack. In the following subsections, we explain more details of this experimental setup.

3.1 Hardware Platforms

Three different platforms are evaluated in our experiments, with the same basic experimental setup as in Figure 2. All of the used platforms are evaluation boards for 32-bit microcontroller systems that can be used in IoT applications. These are the ESP32-devkitC, STM32L475 IoT Node, and two copies of the STM32F407VG Discovery, which were bought apart from each other. These two boards were checked in order to see how sample variation affects the results.

In the two STM32 microcontrollers, a Memory Protection Unit (MPU) is integrated to prevent operating system tasks from reading memory outside their allowed range. We did not use that unit, but it shows that these systems actually support a certain level of isolation, which could potentially be broken through the ADC noise side-channel.

All platforms run in an operating frequency range of 80-168 MHz, and respective ADC sampling frequencies were chosen, such that a whole trace of one cryptographic operation

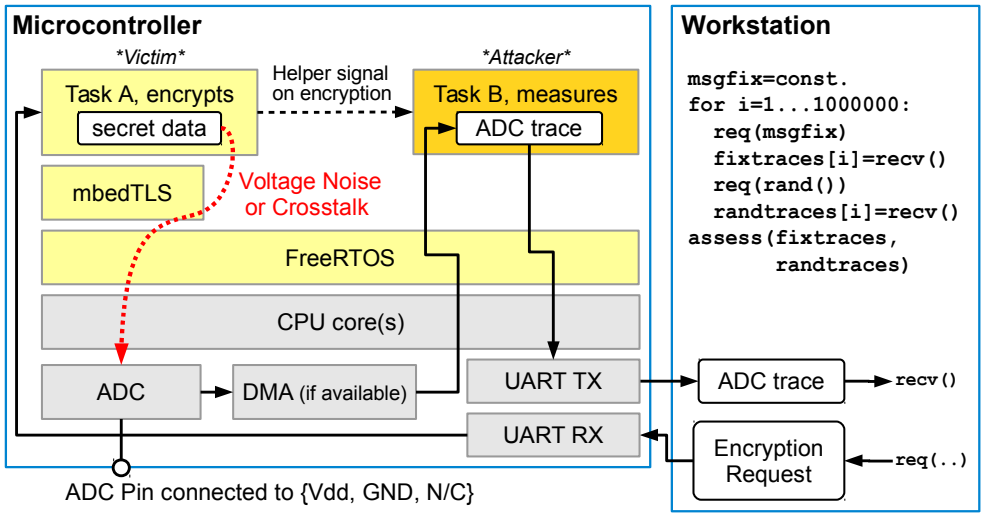


Figure 2: Overview of our common experimental setup, shared among the used platforms.

can be saved in internal SRAM memory. The ADCs of these platforms all support a 12-bit operation mode, which we selected when not noted otherwise.

The power supply on all the microcontroller boards uses the 5 V USB power as input, which we supplied from a standard PC USB output. All boards use a voltage converter to produce a 3.3 V voltage for the V_{dd} of the respective controller. In the STM32F407VG Discovery and STM32L475 IoT Node platform, the manufacturer added a compensation network of capacitors and inductors through which the 3.3 V is connected to the ADC reference pin. We did not do any modifications on any of the boards, and thus also kept this compensation network. In the ESP32-devkitC platform, only an internal ADC reference exists. The ADC of this platform can also be internally connected to Vdd, which we used throughout the results in this paper for 'Vdd', instead of an external connection. The ESP32-devkitC contains three CPU cores, with two Xtensa 32-bit CPUs and a ultra-low-power (ULP) core that can run independently and also collect ADC samples. The STM32F407VG Discovery and STM32L475 IoT Node are single core platforms with Cortex-M4 CPUs, such that DMA is required to sample the ADC in parallel to a running CPU. This information is listed together with details on sampling in Subsection 3.3, Table 2.

3.2 Software Environment

On the software side, we use two operating system tasks in the system. One task encrypts plaintext messages received through UART from an attached workstation, while another has access to ADC data. The detailed operation is explained in the following Subsection 3.3.

In all of the platforms, we used the development environment of the vendor and the respectively provided library versions, of which we provide an overview in Table 1. For the ESP32 platform, the Espressif IoT Development Framework (ESP-IDF) was used, which integrates the listed mbedTLS and FreeRTOS versions. A project is based on a simple Makefile that includes an ESP-IDF makefile. The compiler and assembler versions come from the official setup guide, and we used the default 'Release' compiler optimization for size (-Os). The STM32CubeMX software used for the ST Micorelectronics platforms is a code generator, generating Makefile-based projects, also integrating mbedTLS and FreeRTOS. The official ARM GCC compiler was taken through a package provided for the Eclipse Development Environment and we also used optimization for size (-Os). Only for a single experiment on Correlation Power Analysis (CPA), -O0 was used, which is specifically mentioned in the results later.

Table 1: Used vendor toolchain versions and respective library and compiler versions

Platform	Framework	mbedTLS	FreeRTOS	Compiler(s)
Espressif ESP32-devkitC	ESP-IDF 3.1[1]	2.12.0	8.2.0 Xtensa Port[2]	xtensa gcc 5.2.0[3] esp32ulp 2.28.51[4]
ST Microelectronics STM32F407VG Discovery	STM32CubeMX[5] 4.26.1, 5.0.1	2.6.1	9.0.0	arm gcc 7.3.1[6]
ST Microelectronics STM32L475 IoT Node	STM32CubeMX[5] 4.26.1	2.6.1[7]	9.0.0	arm gcc 7.3.1[5]

[1] Espressif IoT Development Framework https://github.com/espressif/esp-idf/

[2] Espressif explains the Xtensa Port in https://docs.espressif.com/projects/esp-idf/en/v3.1/api-reference/system/freertos_additions.html, which mainly adds multicore support

[3] crosstool-ng-1.22.0-80-g6c4433a-5.2.0 as linked in https://docs.espressif.com/projects/esp-idf/en/v3.1/get-started/linux-setup.html

[4] v2.28.51-esp32ulp-20180809, as linked in https://docs.espressif.com/projects/esp-idf/en/v3.1/api-guides/ulp.html

[5] STM32CubeMX Eclipse plug in https://www.st.com/en/development-tools/stsw-stm32095.html, 4.26.1 was used for leakage assessment, 5.0.1 was used for the CPA attack in Subsection 4.5.

[6] GNU MCU Eclipse, based on arm-none-eabi-gcc 7.3.1-1.1-20180724-0637 from https://gnu-mcu-eclipse.github.io/blog/2018/07/24/arm-none-eabi-gcc-v7-3-1-1-1-released/

[7] For this platform, none was provided in CubeMX, but the version from STM32F407VG worked directly

Please note that STM ships an old mbedTLS with their recent development framework. For our evaluation, however, this is not relevant, since we verified that the used functions for our experiments `mbedtls_internal_aes_encrypt` and `mbedtls_mpi_exp_mod` did not contain any significant changes that would defeat the comparability of our results.

We evaluate AES-128 for single encryptions of 128-bit plaintext messages, and respective sliding window modular exponentiation with 512-bit exponentiations. For AES we use basic single encryptions with `mbedtls_internal_aes_encrypt` to prove principal information leakage. Please note that advanced operating modes of AES, like counter-mode, are on principle also vulnerable to power analysis [Jaf07], but require more effort. A knowledgeable attacker could deploy such attacks. For modular exponentiation, we use `mbedtls_mpi_exp_mod`, which is also used in the RSA implementation of mbedTLS, but does not prove its overall vulnerability. Please note that we only use mbedTLS such that we have cryptographic relevant code for the leakage assessment, but not to show any new attack on this specific library.

3.3 Sampling of Data, Transmission and Synchronization

As seen in Figure 2 in our experimental setup, a workstation PC sends encryption requests to the microcontroller system using a simple UART interface, in which one task performs encryptions, while another task has access to the ADC. To be able to record side-channel leakage from the digital to the analog subsystem, the ADC must be sampled during the time of the cryptographic operations of Task A. Task B should not acquire ADC samples while it runs time-multiplexed to Task A, since the ADC would not receive side-channel leakage. Thus, Task B needs to either run in parallel to Task A on a separate CPU, or use hardware accelerated sampling, i.e. through DMA. A separate DMA module is available in many microcontrollers or SoC systems to improve the performance of various tasks. Other controllers offer programmable state machines or specific low-power cores that can control the ADC and other peripherals in parallel to normal task execution on the main CPU(s).

In Figure 3 we show a more detailed description how the two tasks are executing in parallel in our experiments. More detailed example code can be found in Appendix C. At the beginning, Task A receives a message to be encrypted through UART and notifies a sleeping/waiting Task B using a FreeRTOS notification (as the helper signal). Task B

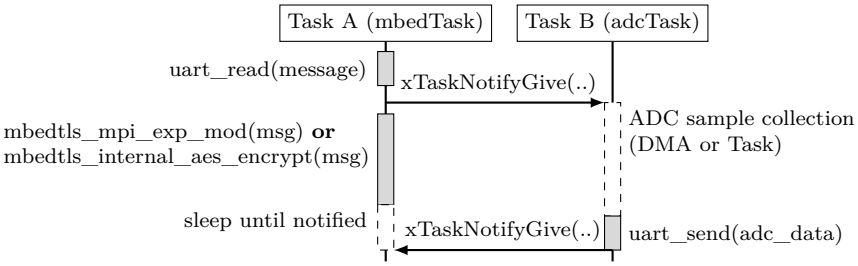


Figure 3: Description of one loop iteration of the two FreeRTOS tasks.

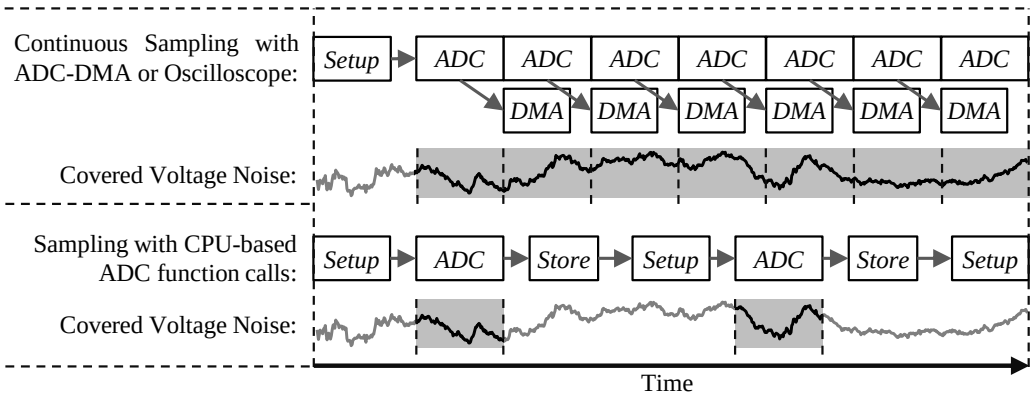


Figure 4: Principle how different ADC sampling styles will cover less or more parts of the voltage noise affecting the ADC result. DMA needs to be used for continuous sampling, while software-based sampling (in the multi-core scenario) will always introduce some gaps.

starts collecting ADC data in parallel to Task A, either in dual-core operation or using DMA operation of the ADC. Task A then encrypts the message using a previously stored key, while the ADC is collecting data. After the encryption, it waits for a notification. Upon finishing the fixed number of ADC samples, Task B sends them to the workstation, and notifies Task A so everything can start afresh for another message. Due to differences in the systems, we use DMA transfer in the two STM32 systems, and dual-core operation in the ESP32. However, after we had performed all experiments, we found the ESP32 also has a sampling mode that does not require the second core, by using its i2s-module.

There is still a fundamental difference between using the ADC with DMA, versus using software on a CPU to acquire individual ADC samples in a loop. This difference is visualized in Figure 4. Running the ADC in a continuous mode with DMA at lower speeds will basically use more time in the ADC conversion itself, but will not reduce the total time range in which the measurement is influenced by noise. In comparison to that, in single-conversion software-based sampling, a certain time between the ADC samples will be used to run software to store data and prepare the next ADC acquisition. If we want to change the sampling rate, we will need to add delay in software. Any analog noise in that time will not affect the ADC result, and thus some side-channel leakage might not be captured in the acquired ADC data. In our experiments we also introduced delays in CPU-based sampling, because of internal memory size limitations. Since sampling with the CPU is usually slower, too, the sampling rate can also be negatively impacted by CPU-based sampling.

Usually a sampling frequency above the CPU or circuit frequency is recommended for power analysis attacks, but is not necessarily required for successful attacks [LDMPT15]. For the platforms we use, an additional limitation is the amount of available internal memory to store all samples. For RSA, we typically had to sample rather slow to not fill the memory (2000-4000 samples per encryption). For AES we had to sample almost as fast

Table 2: Overview of Leakage Assessment Experiments, repeated for ADC Pin = {Vdd, GND, N/C}

Platform and Experiments	Sampling Style	ADC Ref. Filter	Algorithm	Samplerate / #Samples
ESP32-devkitC @80MHz	ULP-CPU	No	AES-128	104 kHz / 16
	CPU		Exp-512	20.4 kHz / 2600
STM32F407VG Discovery @168MHz	DMA	Yes	AES-128	980 kHz / 32
			Exp-512	88 kHz / 4096
STM32L475 IoT Node @80MHz	DMA	Yes	AES-128	684 kHz / 64
			Exp-512	40 kHz / 4096

as possible, to achieve a reasonably high sampling rate (100-1000kHz). For the two STM32 devices this could be achieved by different ADC-DMA setups. For the ESP32-devkitC we had to use the ULP-CPU to sample the ADC fast, and a normal CPU task to sample it slow. That is because the ULP has only access to limited memory below 2048 bytes in total, but also has a dedicated ADC instruction for fast sampling. The finally-used sampling rates depending on the used board and algorithm are shown in Table 2. The shown values were estimated by measuring on an external pin, since it was not always clear from the software to find out the actual sampling rate. It might be feasible to achieve higher sampling rates on the ESP32 with its i2s-module.

3.4 mbedTLS implementation details

This subsection clarifies the implementation details of RSA and AES in the mbedTLS library, which we verified to be unchanged in the relevant parts for this paper, for both library versions (2.6.1 and 2.12.0).

The RSA implementation in mbedTLS is based on CRT (which can be disabled) and can use the exponent blinding side-channel countermeasure if a sufficient source of randomness is provided to the library. The RSA private key function of mbedTLS (`mbedtls_rsa_private`) uses bigint arithmetic, and among other functions calls `mbedtls_mpi_exp_mod`, which performs a sliding window modular exponentiation. That function is where we perform some of our leakage assessments on, with a 512-bit exponent and modulus. We use the message transmitted by UART as the *base* of this exponentiation. The other function we analyze is from the AES algorithm.

The mbedTLS library implements the AES using a T-table lookup based approach. Originally, the AES is a round-based block cipher, where four different operations *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey* are repeatedly applied on a 128 bit data block. A popular optimization is to implement those operations as a combination of multiple table lookups and a subsequent *XOR* operation. This optimization requires the precomputed T-tables, which take an input byte and output a 32 bit word. Besides the addition of the first round key and the last round, the remaining rounds are executed in pairs of two inside each loop iteration. Apart from these optimizations, the mbedTLS AES implementation does not diverge from the textbook AES encryption algorithm and does not include any side-channel countermeasures in particular.

4 Results

Before doing leakage assessment or CPA, we first show that ADC noise correlates with the power consumption of the board. Subsequently, leakage assessment is performed on AES and modular exponentiation of the mbedTLS library.

In all of the tested platforms, side-channel leakage was found in at least one of the tested cryptographic algorithms and operation modes (configurations) of the ADC. In many setups, generic noise was observable on the ADC, even when it was pulled to Vdd or GND. Just in a few setups, we actually observe zero variance in the ADC output, such that information leakage is impossible.

4.1 Preliminary Comparison between Voltage and ADC Noise

For this experiment, we use the STM32F407VG Discovery #1, and run the CPU in high and low activity phases that should be easily distinguishable. In high activity phases, we perform floating point operations, whereas during the intermediate low activity phases, we issue *nop*-commands.

In Figure 5, we show the average of 1000 traces of the supply voltage V_{dd} and supply current I_{dd} measured with an oscilloscope. Concurrently, we show the acquired samples of 6-bit ADC values at maximum sampling rate using DMA mode. Both were recorded during the same activity phases of the CPU on the STM32F407VG Discovery board. The ADC is connected to a floating pin, and neither to GND or V_{dd}. The different phases of workload activity phases can easily be visually distinguished in both of the traces. Our activity pattern can be identified in the externally collected traces for both V_{dd} and I_{dd} in the timeframe from $0\mu s$ to about $160\mu s$, whereas the data transfer of ADC traces to a workstation occurs after $160\mu s$. Likewise, the ADC average values in the bottom diagram reflect the activity in a clearly distinguishable way, albeit not with linear correlation.

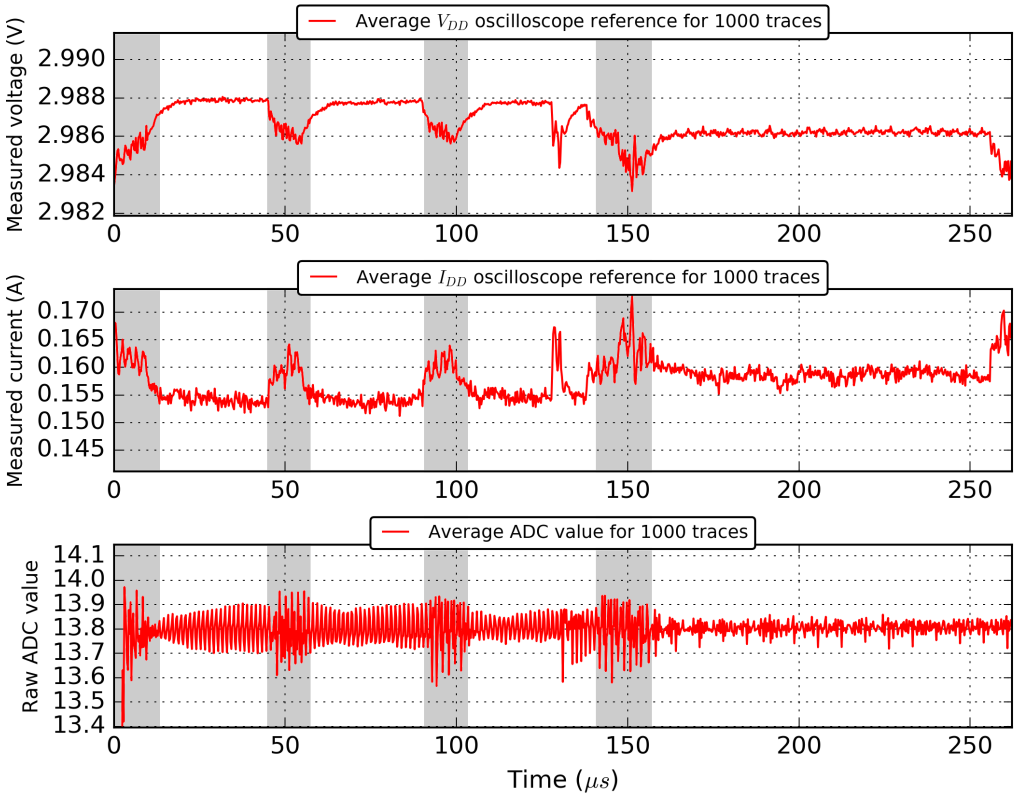


Figure 5: Average over 1000 traces for oscilloscope measurements on V_{dd} and I_{dd} in the first two plots and average of concurrently measured ADC values when the ADC was set to 6-bit and the pin was not connected (N/C) on the bottom. High activity phases are marked grey.

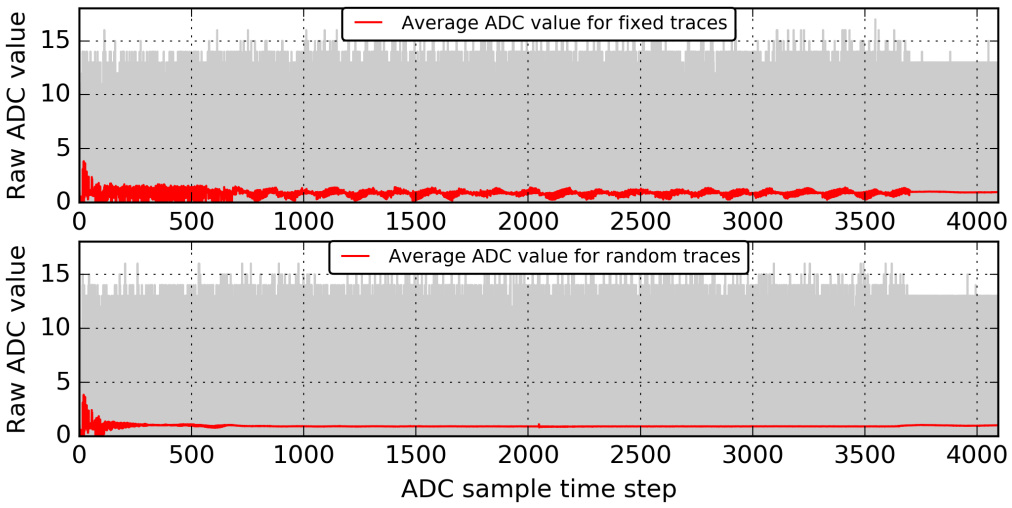


Figure 6: Average over 100k traces for a fixed base exponentiated in a mbedTLS sliding window exponentiation, and 100k traces with each a random base exponentiated with the same secret exponent on the STM32F407VG Discovery #1. ADC connected to GND.

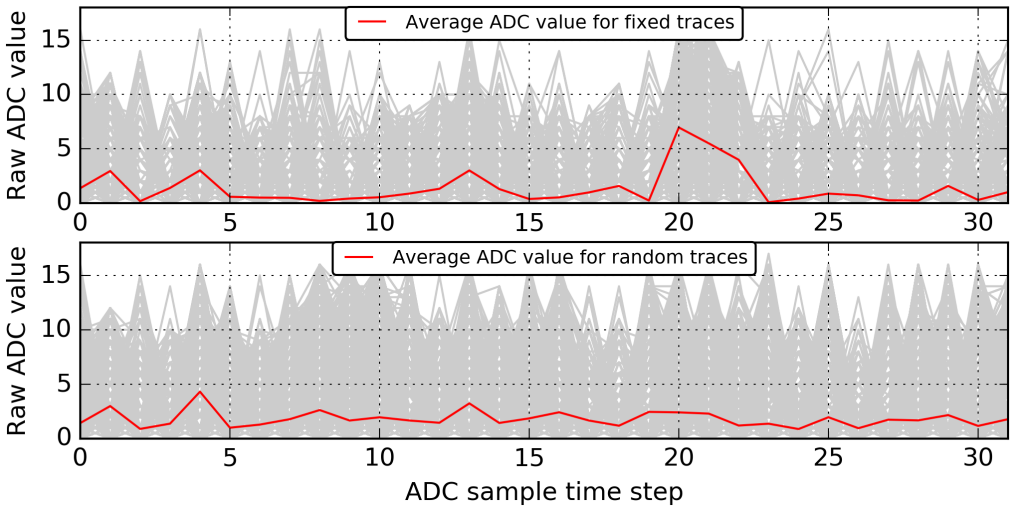


Figure 7: Average over 1M traces for a fixed message encrypted with mbedTLS AES and the FIPS key, and 1M traces with each a random message encrypted with the same key on the STM32F407VG Discovery #1. ADC connected to GND.

4.2 Comparisons on Average ADC Traces of AES and Modular Exponentiation

The preliminary experiment to compare CPU activity phases already shows promising results. However, this experiment is a synthetic test case in which extremely high and low activity phases were chosen on purpose. Subsequently, we prove that even minor differences in the data processed by cryptographic algorithms affects the ADC noise in a systematic way. For that proof, we can already use the data required as a prerequisite for the leakage assessment we explained in Subsection 2.4. We collect two sets of traces while performing encryption operations. One of the sets contains the fixed traces, while the other set contains the random traces. For this example, we connect the pin used for

ADC to GND and again use the same board as used in Subsection 4.1, STM32F407VG Discovery #1.

In Figure 6, we compare the ADC noise that occurs during sliding window exponentiation with a 512-bit secret exponent and modulus. In the first plot, an average over 100,000 (100k) ADC traces is shown, where an exponentiation is performed on the same fixed base. The second plot also shows the average, but with exponentiation on 100k different random bases. The red lines show the averages, while the grey background contains all the single traces. The differences between these plots are indeed distinguishable without further processing. Even more, a pattern is already visible in the average of the fixed traces, which is smoothed out in the case of random traces. This example already shows that a power analysis attack for secret key extraction later on might be feasible.

Additionally to modular exponentiation (Mod-Exp), we also show an average of AES-128 for fixed and random messages in Figure 7. Since AES executes in much shorter time relative to the ADC speed, we can only acquire a few samples, which are at best 2-3 samples per AES round for this specific board. For AES, we collect 1,000,000 (1M) traces for each of the two sets of fixed and random messages. Similar to Mod-Exp, the differences between the traces are visible, with the most distinguishable sample point around sample 20.

4.3 Leakage Assessment on AES and Modular Exponentiation

We analyze the differences between the traces collected during fixed and random modular exponentiations and AES encryptions using the t-test methodology from [GGJR+11, SM15] as explained in Subsection 2.4. Exemplary, we show the results of leakage assessment on the ADC traces collected on the STM32F407VG Discovery, which we presented in the previous subsection. Further details from other platforms can be found in the appendix.

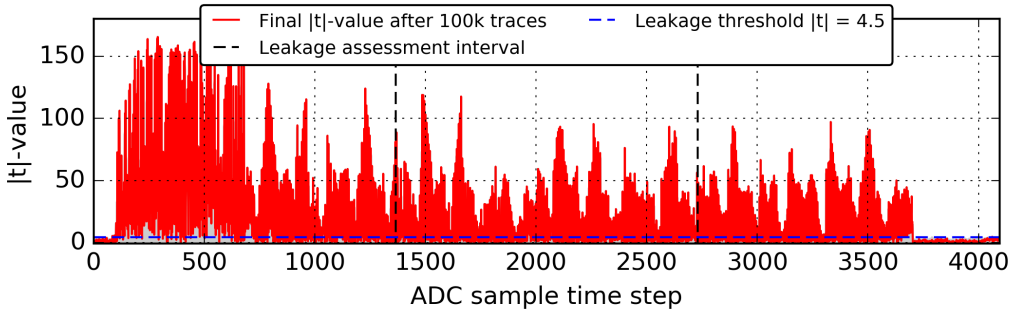


Figure 8: First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #1.

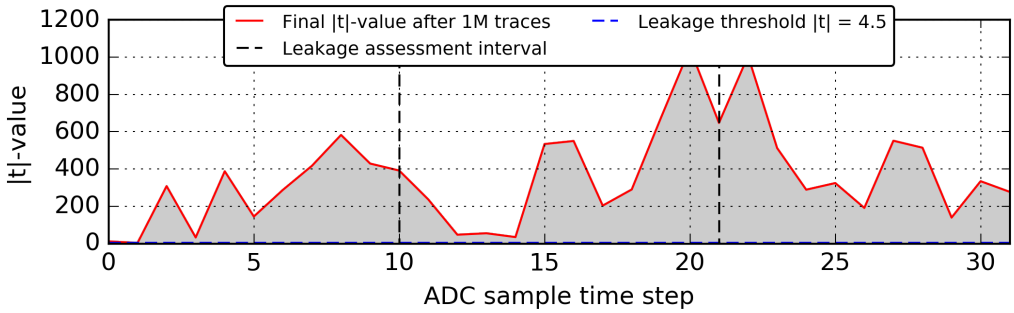


Figure 9: First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery#1.

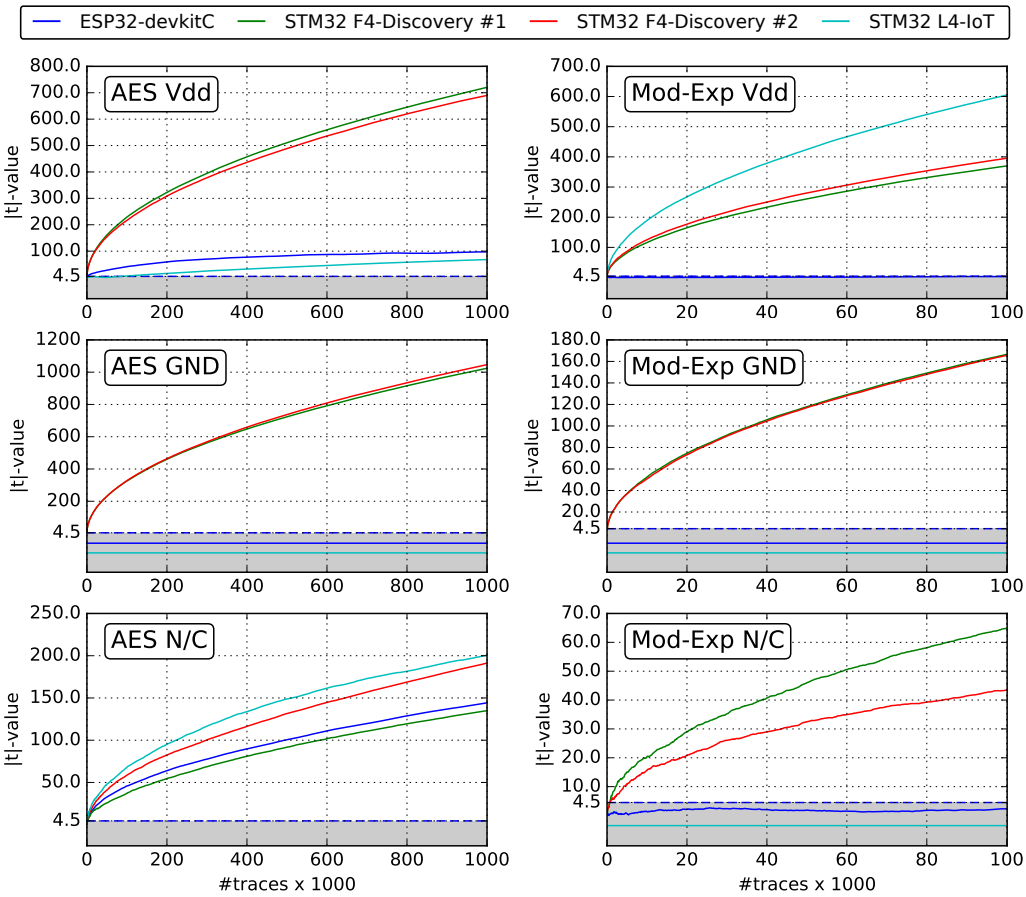


Figure 10: Leakage Assessment on mbedTLS AES and modular exponentiation (Mod-Exp) with {Vdd, GND, N/C} connected to the ADC on all platforms. Flat lines on the bottom are constant zero, shifted for visibility.

Figure 8 shows the first order leakage on the STM32F407VG Discovery for fixed-vs-random traces collected during modular exponentiation. The mathematical evaluation confirms the visual assessment from the previous section: The fixed and random traces are distinguishable, as the absolute $|t|$-value exceeds the threshold limit of 4.5 significantly.

Likewise, in Figure 9 we present the first order fixed-vs-random leakage during AES encryptions. Again, we confirm that the traces for fixed and random encrypted messages are clearly distinguishable, as the $|t|$-value is above the threshold.

Subsequently to the initial experiments, we perform an analysis on the boards introduced in the experimental setup, Section 3. First, we look into the ADC-observable leakages from AES, and secondly into modular exponentiation. This way, we also test the ADC noise characteristics at different operating frequencies. As we presented in the experimental setup in Table 3, for AES we had to use rather high sampling frequencies of the ADC on all the platforms ($104 - 980$kHz). For the ESP32-devkitC, no higher frequency than 104kHz is reachable, which leads to less than 16 samples over the complete AES runtime. For modular exponentiation, the runtime is longer. Thus, we sample slower ($20 - 88$kHz) to collect only as much samples as the memory capacity of the internal SRAM.

In Figure 10, we show six plots of first order leakage assessments on AES and modular exponentiation (marked as 'Mod-Exp'), separated by algorithm and the connection of the ADC being Vdd, GND, or N/C, respectively. In these plots, the change of the highest $|t|$-value inside the leakage assessment interval (cf. Figure 8,Figure 9) is shown, over an increasing amount of traces used for the evaluation.

Table 3: Overview of Leakage Assessment over all tested Platforms and Configurations; ADC connected to {Vdd, GND, not connected (N/C)}. Amount of collected traces are 100k for modular exponentiation, and 1M for AES. The ADC was noise-free when σ=0.

Platform	Leakage detected? ($t > 4.5$)					
	AES-128 (ADC fast)			Mod-Exp-512 (ADC slow)		
	Vdd	GND	N/C	Vdd	GND	N/C
ESP32-devkitC @80MHz	yes	σ=0	yes	no[1]	σ=0	no[1]
STM32L475 IoT Node @80MHz	yes	σ=0	yes	yes	σ=0	σ=0
STM32F407VG Discovery #1 @168MHz	yes	yes	yes	yes	yes	yes
STM32F407VG Discovery #2 @168MHz	yes	yes	yes	yes	yes	yes

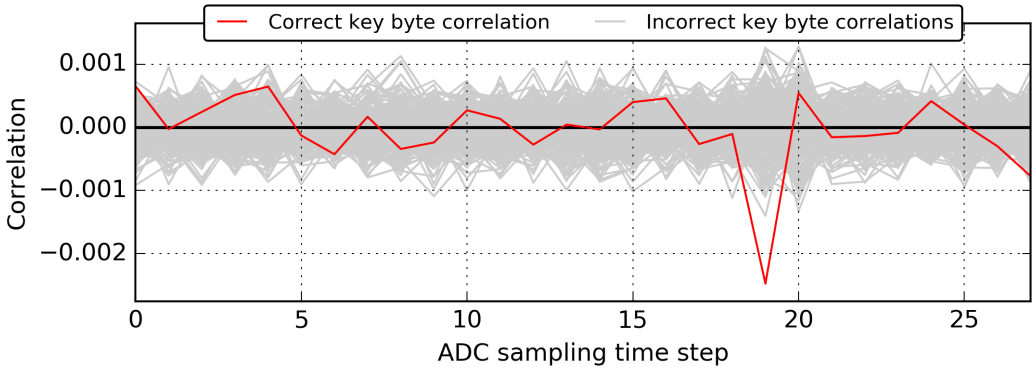
[1] For the center 1/3 trace. For the beginning and/or end of the cryptographic function, |t| was above 4.5. For more details check Appendix A, Figure 14.

We start with the AES algorithm in the left column of Figure 10. We compare the platforms when the ADC is connected to Vdd. In this configuration, $|t|$ reaches values clearly beyond the confidence threshold of 4.5, suggesting that all platforms are leaking the information processed in the AES algorithm. For the case of a connection to GND, both samples of the STM32F407VG Discovery leak, but not the other boards. For the other boards, the ADCs actually output a constant value, such that Ground-Noise does not occur. In the case, when we have no connection (N/C) on the ADC, i.e. the pin is in a so-called floating state, all of the boards exhibit leakage ($|t| \gg 4.5$). We also looked into second order leakage. However, there were no changes with respect to the conclusion, if leakage is observed ($|t| > 4.5$), or not ($|t| \leq 4.5$).

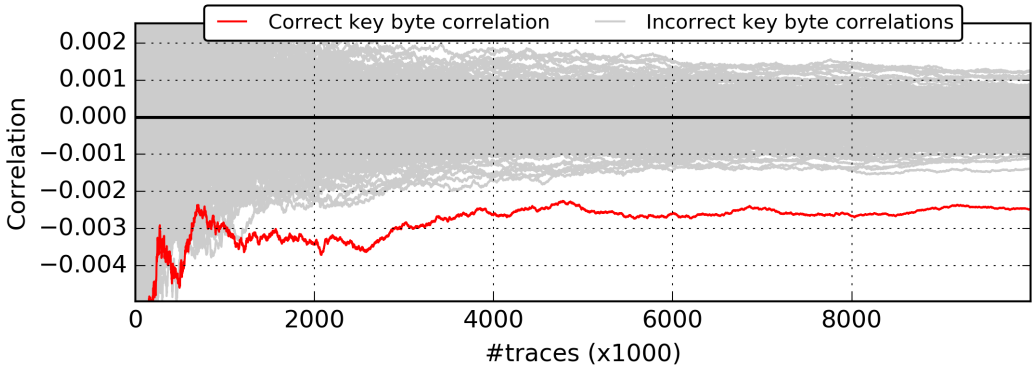
In the case of Mod-Exp, we only show first order leakage again in Figure 10, since there was no difference in the second order, similar to AES. We still compare the platforms with ADC connections to {Vdd, GND, N/C}. For Mod-Exp, again both STM32F407VG Discovery #1 and #2 leak in all of the tested cases. The other boards leak less than for AES. While the ESP32-devkitC t-value reaches beyond $|t| > 4.5$ for Vdd and N/C, it only does during the beginning or end phase of the modular exponentiation. We thus conclude negatively for the leakage assessment. This result might be due to an input value being copied or recoded inside the function. Interested readers can check the appendix, Figure 14, for this detail. For the STM32L475 IoT Node, the ADC was sufficiently noisy at faster sampling rates for AES, except when the ADC was connected to GND. However, for modular exponentiation it also became entirely noise free when being not connected (N/C).

4.4 Summary of Leakage Assessments

In summary, the ADC settings such as the sampling frequency and connection to Vdd, GND or N/C levels affect the observable side-channel leakage. This relation exists, because the sampling frequency also changes the inherent noise characteristics. The connection of the ADC affects how it is coupled to the remaining parts of the system, particularly the digital (CPU and memory) subsystem. In most cases when the ADC shows any noise (sample data with $\sigma > 0$), the t-test detects leakage in the ADC data. In other cases, when the ADC is completely noise free ($\sigma = 0$), surely no information leakage is possible. The two boards that we used of the same type lead to almost the same results, leading to the conclusion that sample variation only has a minor effect. We summarize these results in Table 3. For reference, Appendix A shows all leakage assessments across all the boards in detail, including the assessment over the complete time window when ADC samples were acquired (cf. Subsection 3.2).



(a) Total correlation after 10M traces for all 256 key byte candidates; The correlation with the correct key byte is marked red



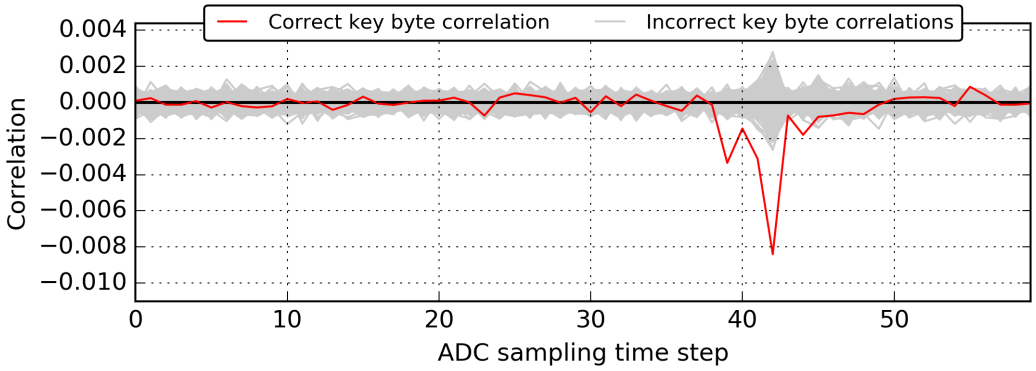
(b) Correlation progress over 10M traces for all 256 key byte candidates; The correlation with the correct key byte is marked red

Figure 11: Results of a CPA attack on the 6th byte of the last secret round key of AES on the STM32F407VG Discovery #1 @168MHz with the ADC connected to GND and the program compiled with the -Os optimization option (like for leakage assessment).

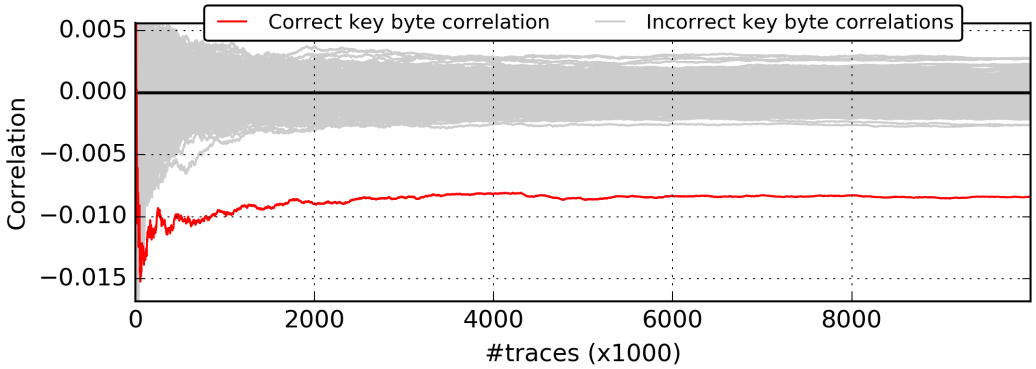
4.5 Correlation Power Analysis Attack on AES

In this subsection, we present results of CPA attacks on secret AES round keys in different setups. We perform a ciphertext-based CPA on the last round of AES over 10M ADC traces and show both the final correlations for each key byte candidate with the entire set of traces as well as the correlation progress over the amount of traces. We evaluate the different preprocessing and power model variants, which are explained in Subsection 2.5. Pre-aligning traces with a shift of ±2 using normalized cross-correlation, and performing CPA with the standard S-box Hamming distance model is the most successful variant. The CPA experiments are performed on the STM32F407VG Discovery, which shows the most promising results during leakage assessment. Although we attacked all 16 bytes of the AES round key, only the best results for the respective setup are presented here. We state the total amount of recovered bytes for each setup and display the correlation plots for all key bytes in the appendix.

We first evaluate a CPA attack on 10M traces using the same parameters as for leakage assessment, when the ADC pin is connected to GND. Here, AES takes about 13μs, leading to an estimated 17 samples during the complete AES function call to `mbedtls_internal_aes_encrypt(..)`. In this setup, we are able to recover 2 secret key bytes in total, where the best correlation is seen for the 6th byte. In Figure 11a, the total correlation with all of the 256 possible values of the 6th byte of the last round key after 10M ADC traces is



(a) Total correlation after 10M traces for all 256 key byte candidates; The correlation with the correct key byte is marked red



(b) Correlation progress over 10M traces for all 256 key byte candidates; The correlation with the correct key byte is marked red

Figure 12: Results of a CPA attack on the 12th byte of the last secret round key of AES on the STM32F407VG Discovery #2 @56MHz with the ADC connected to Vdd and the program compiled with the -O0 optimization option.

shown. A small peak in the last part of the correlation values indicates the sampling point that corresponds to the last AES round operation as well as the correct value of the last round key byte, which has the maximum correlation value at that point. However, the differences in correlation between the correct and the incorrect key bytes and the peak in general are very small.

In addition to evaluating CPA attacks on the previous setup for leakage assessment, we also adapt the compilation and frequency parameters to achieve a higher relative ADC sampling rate. Additionally we found to be more successful when the ADC pin is connected to Vdd. In that case, the MCU is running at 56MHz and the program is compiled with -O0 optimization, making AES take about 40μs to compute. Due to the on-chip clock network, the ADC is running slower, but faster relative to AES. This relative improvement leads to about 43 ADC samples for the entire AES encryption. Furthermore, we activate the DMA-based ADC sampling from the encryption task directly. This avoids any misalignment and synchronizes exactly with the beginning of the AES computations. In this setup, 6 secret key bytes can be recovered and the best correlation appears when attacking the 12th byte. The results can be seen in Figure 12. A peak during the last part of the encryption is clearly visible, again indicating the last AES round. Furthermore, we see that the correct key byte, which is marked red, correlates much clearer with the collected traces than the incorrect key bytes.

We conclude that key recovery attacks on the AES with data from ADC traces are generally possible, albeit the success depends on the overall system parameters, such as the clock speed, the ADC pin connection, and possibly even the code optimization at compile time. For data collected with the ADC pin left unconnected, we were unable to recover secret key bytes successfully.

5 Discussion

Our results prove the existence of a correlation between the data processed in a microcontroller, and the noise that can be observed in their integrated ADCs. The correlation is strong enough to distinguish the data processed in cryptographic algorithms, running on a CPU in the system. By leakage assessment it was shown that this observation is valid in most cases. Furthermore, we proved that the leakage can be sufficient to perform a CPA-based key recovery attack on AES. Due to the protection mechanism that can be enabled for the full RSA implementation of the used mbedTLS library, we assume that more advanced attacks are required for that, but are generally feasible [Wal01, SI11, PC16].

The performed experiments reveal an underlying problem of highly integrated mixed-signal systems that consist of analog and digital components in a single chip. Such a level of System on Chip (SoC) integration is an increasing trend in many hardware platforms for IoT applications and beyond. In these systems, both power supply based coupling effects as well as crosstalk can cause the noise in the analog part, which can then be exploited by anyone with access to the data measured in the analog circuit.

5.1 Practical relation to the adversarial model

Applications based on microcontrollers or SoCs use the ADC for various tasks, such as audio streaming or other sensor measurements. For power saving applications, the respective ADC pin is often pulled to GND, the supply voltage (V_{dd}) or left unconnected. Any of these ADC pin configurations can be the recommended way to save power, depending on the manufacturer and device. Often, it is possible to define by software to which pin the ADC is connected, or even internally connect it to GND or Vdd, such as Vref for the tested ESP32 chip. Some of these IoT or mobile applications make their data publicly available, or execute untrusted code in a memory protected area. In both ways, an adversary with the right access can get the necessary side-channel data to perform an attack.

Accidentally allowing a high sampling rate in publicly accessible IoT sensors can already be an attack vector from which sufficient leakage can be collected. For instance, a reference application for one of the tested platforms contains a web server which accepts any ADC sampling rate from its website. Any website hosted on that web server can control the sampling rate through JavaScript. JavaScript in turn can be manipulated by any user that accesses the website. In effect, if this reference application is used as the basis of a product, it introduces a threat. On the other hand, 3rd party smartphone applications often need to use various analog sensors of the system, which could contain the required side-channel information to perform an attack. Even a typical audio sampling rate is already fast enough to distinguish differences in modular exponentiation on the tested platforms, and it was already reported that such a low sampling rate can be sufficient to attack RSA [GST14].

Our experimental setups were chosen for comparability across different vendor systems, and additionally to allow methods for leakage assessment to be performed easily. Yet, these setups are sufficient to prove that ADC noise is generally a possible source of side-channel information. Using the aforementioned example, this could actually be exploitable in some existing products, and adds a dangerous new way to acquire power side-channel information completely remotely.

If an attack on a real system can succeed depends on additional aspects. Next to the basic requirement of access to ADC data, it is also required that the data can be sampled during cryptographic operations, and the collected traces to be aligned properly. However, it was shown that even in full commercial implementations, many of such obstacles can still be circumvented, and complex attacks can be performed by considering more aspects of a full system [EKM+08, BGV+12, TSS17, KGG+18]. For instance, it is often still possible to find a way of synchronization. At least an estimation at which time an encryption starts can often be achieved through the behavior of the overall application. A remote user can thus estimate which part of sensor data might contain exploitable side-channel leakage.

Another aspect in real systems is the side-channel data being modulated on top of other sensor measurement data, or the available sampling rate. However, this should only increase the number of traces required for differential attacks, which are specifically suited to cope with such situations. Often the sensor data is not much of a problem if it changes rather slowly, for instance a temperature sensor. It was also shown that a sampling rate below the expected side-channel leakage can still be sufficient to perform a full attack [GST14, LDMPT15].

Our results imply that even sensor data that a microcontroller measures and sends over a digital connection made available online can leak sensitive information which is accessible from anywhere in the world. In many scenarios, leaking the secret keys of a sensor node might be just a small issue, but in other scenarios a more sophisticated attacker could use such information as part of a larger scale attack, for instance on SCADA systems [CRS09, Lan11].

5.2 Mitigating the threat

As our results suggest, it is of high importance that designers of embedded systems or integrated circuits for critical applications are aware of a potential security vulnerability through integrated analog components. That fact is the most important take-away message from the investigations made in this paper. We believe that noise of analog components should not be just characterized as a noise margin, but needs to be assessed for information leakage from security-related digital components in the system.

When designers are aware of this possible threat, it is achievable to design proper mitigations, depending on the application. The task of mitigation, for instance, can be considered in the hardware design of the ADC module to reduce or completely remove the noise-related leakage. In small IoT systems in which all executed code can be trusted, information leakage and possible attacks require a certain level of care, and we suggest either of the following options as a possible isolation practice:

- It must be guaranteed that any analog measurements can only take place mutually exclusive to security-related computations.
- If exclusive measurements cannot be guaranteed, a leakage assessment needs to be performed on all analog measurement data that is made accessible to possible attackers. If the data contains leakage, it has to be handled with the same security level as the secret data that is processed in the system.
- Filter the ADC data in a way that leakage cannot be assessed anymore. For instance, a filter for noise or specific frequencies might make attacks infeasible, or reduce effective sampling rates.

However, in multi-user systems in which memory protection is used among unprivileged tasks that are not fully trusted, more serious care has to be taken in order to prevent one of the tasks from extracting the secrets of another task. For instance, underprivileged tasks should not be allowed to get arbitrary ADC measurements. This could mean that

in systems like a smartphone, a task that records the microphone on one of the ADCs could potentially run power analysis side-channel attacks on the remaining system. More research is needed to evaluate how widespread this threat is in existing systems.

6 Conclusion

Mixed-signal systems such as microcontrollers and other SoCs are increasingly adopted in Internet-of-Things and personal computing appliances. In these systems, digital logic such as a CPU causes noise in the analog components, for instance in the ADCs used for a microphone, or any environmental sensors. In this paper, leakage assessment was performed on the noise of ADC data for various platforms. In the majority of cases, leakage was detected. In one case we demonstrated a successful key recovery attack on AES proving that the leakage can be exploited. Previous works have explored a similar software-based power analysis side-channel, and this paper generalizes those results. It is now confirmed that power analysis side-channels are not just a threat for attackers with physical access to the device. Any sensory data that leaves a system might contain enough correlated noise that could be exploited to perform power analysis attacks. Thus, this paper stresses the importance to either use better isolation between analog and digital subsystems, or protect cryptographic implementations against power analysis attacks, even if local attackers are not considered in their threat model.

Acknowledgements

The authors thank Kevin Schäfer from Rutronik Elektronische Bauelemente GmbH, Germany, for providing us some of the platforms used in our experiments. Additionally he provided us with valuable knowledge on ADC noise characteristics and power distribution.

References

[AAB+17] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the Mirai Botnet. In *USENIX Security Symposium*, pages 1092–1110, 2017.

[AGR99] Xavier Aragones, Jose Luis Gonzalez, and Antonio Rubio. *Analysis and solutions for switching noise coupling in mixed-signal ICs*. Springer Science & Business Media, Dordrecht, 1999.

[Ama18] Amazon-FreeRTOS. Cloud-native IoT operating system for microcontrollers, 2017-2018. https://github.com/aws/amazon-freertos.

[AR06] Sadok Aouini and Gordon W. Roberts. A Predictable Robust Fully Programmable Analog Gaussian Noise Source for Mixed-Signal/Digital ATE. In *IEEE International Test Conference*, pages 1–10, October 2006.

[ARM16] ARM MBED. SSL library mbed TLS / PolarSSL, 2008-2016. https://tls.mbed.org/.

[ASM07] Karim Arabi, Resve Saleh, and Xiongfei Meng. Power supply noise in SoCs: Metrics, management, and measurement. *IEEE Design & Test of Computers*, 24(3):236–244, May 2007.

[BA19] Richard Barry and Amazon Web Services (AWS). FreeRTOS, 2003–2019. https://www.freertos.org/.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, 2004.

[BGV+12] Josep Balasch, Benedikt Gierlichs, Roel Verdult, Lejla Batina, and Ingrid Verbauwhede. Power analysis of atmel cryptomemory–recovering keys from secure EEPROMs. In *Cryptographers' Track at the RSA Conference*, pages 19–34. Springer, 2012.

[CPM+18] Giovanni Camurati, Sebastian Poeplau, Marius Muench, Tom Hayes, and Aurélien Francillon. Screaming channels: When electromagnetic side channels meet radio transceivers. In *Proceedings of the Conference on Computer and Communications Security (CCS)*. ACM, October 2018.

[CRS09] Alvaro A Cardenas, Tanya Roosta, and Shankar Sastry. Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems. *Ad Hoc Networks*, 7(8):1434–1447, 2009.

[DWB15] Shidhartha Das, Paul Whatmough, and David Bull. Modeling and characterization of the system-level power delivery network for a dual-core ARM Cortex-A57 cluster in 28nm CMOS. In *International Symposium on Low Power Electronics and Design*, pages 146–151, 2015.

[EKM+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 203–220, Berlin, Heidelberg, 2008. Springer.

[Fio07] Franco Fiori. On the susceptibility of analog circuit to EMI. In *Analog Circuit Design*, pages 183–202. Springer, 2007.

[GGJR+11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[GMM15] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer.js: A remote software-induced fault attack in javascript. *CoRR*, abs/1507.06955, 2015.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 251–261. Springer, 2001.

[GOKT18] Dennis R. E. Gnad, Fabian Oboril, Saman Kiamehr, and Mehdi B. Tahoori. An experimental evaluation and analysis of transient voltage fluctuations in FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(10):1817–1830, October 2018.

[GOT17] Dennis R. E. Gnad, Fabian Oboril, and Mehdi B. Tahoori. Voltage drop-based fault attacks on FPGAs using valid bitstreams. In *International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, September 2017.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 444–461, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[Jaf07] Josh Jaffe. A first-order DPA attack against AES in counter mode with unknown initial counter. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, pages 1–13, Berlin, Heidelberg, 2007.

[KDK+14] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *International Symposium on Computer Architecture (ISCA)*, pages 361–372, Piscataway, NJ, USA, 2014. ACM/IEEE.

[KGG+18] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *ArXiv e-prints*, January 2018.

[KGT18] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. FPGAhammer: remote voltage fault attacks on shared FPGAs, suitable for DFA on AES. *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, 2018(3), 2018.

[Lan11] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[LDMPT15] Jake Longo, Elke De Mulder, Daniel Page, and Michael Tunstall. SoC it to EM: Electromagnetic side-channel attacks on a complex system-on-chip. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 620–640, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[LRRB05] Bin Le, Thierry W. Rondeau, Joy Lynn H. Reed, and Charles W. Bostian. Analog-to-digital converters. *IEEE Signal Processing Magazine*, 22(6):69–77, November 2005.

[LSG+18] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *ArXiv e-prints*, January 2018.

[MF04] Andrey V. Mezhiba and Eby G. Friedman. Scaling trends of on-chip power distribution noise. *Trans. VLSI Syst.*, 12(4):386–394, April 2004.

[Mic18] Microsoft. Azure device catalog, 2018. https://catalog.azureiotsolutions.com/.

[MS10] Carlo Maria Medaglia and Alexandru Serbanati. An overview of privacy and security issues in the internet of things. In Daniel Giusto, Antonio Iera, Giacomo Morabito, and Luigi Atzori, editors, *The Internet of Things*, pages 389–395, New York, NY, 2010. Springer.

[Nat15] National Instruments. ADC dynamic characteristics measurement reference design, 2015. http://www.ni.com/example/8319/en/.

[PC16] Guilherme Perin and Łukasz Chmielewski. A semi-parametric approach for side-channel attacks on protected rsa implementations. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications*, pages 34–53, Cham, 2016. Springer International Publishing.

[PPK+07] Christof Paar, Axel Poschmann, Sandeep Kumar, Thomas Eisenbarth, and Leif Uhsadel. A survey of lightweight-cryptography implementations. *IEEE Design & Test of Computers*, 24:522–533, November 2007.

[RNL11] Rodrigo Roman, Pablo Najera, and Javier Lopez. Securing the internet of things. *Computer*, 44(9):51–58, September 2011.

[RPD+18] Chethan Ramesh, Shivukumar B Patil, Siva Nishok Dhanuskodi, George Provelengios, Sebastien Pillement, Daniel Holcomb, and Russell Tessier. FPGA side channel attacks without physical access. In *International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages paper–116. IEEE, May 2018.

[RSWO17] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. Iot goes nuclear: Creating a zigbee chain reaction. In *IEEE Symposium on Security and Privacy (S&P)*, pages 195–212. IEEE, 2017.

[SAHK98] Tilmann Stöhr, Markus Alt, Asmus Hetzel, and Jürgen Koehl. Analysis, reduction and avoidance of crosstalk on vlsi chips. In *International Symposium on Physical Design (ISPD)*, pages 211–218, New York, NY, USA, 1998. ACM.

[SD16] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, May 2016.

[SGMT18a] Falk Schellenberg, Dennis R.E. Gnad, Amir Moradi, and Mehdi B. Tahoori. An inside job: Remote power analysis attacks on FPGAs. In *Proceedings of Design, Automation & Test in Europe (DATE)*, March 2018.

[SGMT18b] Falk Schellenberg, Dennis R.E. Gnad, Amir Moradi, and Mehdi B. Tahoori. Remote inter-chip power analysis side-channel attacks at board-level. ICCAD 2018. Cryptology ePrint Archive, Report 2018/881, 2018.

[SI11] Werner Schindler and Kouichi Itoh. Exponent blinding does not always lift (partial) spa resistance to higher-level security. In Javier Lopez and Gene Tsudik, editors, *Applied Cryptography and Network Security*, pages 73–90, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[SLMW93] David K. Su, Marc J. Loinaz, Shoichi Masui, and Bruce A. Wooley. Experimental results and modeling techniques for substrate noise in mixed-signal integrated circuits. *IEEE Journal of Solid-State Circuits*, 28(4):420–430, April 1993.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 495–513. Springer, 2015.

[SPK+10] Jörn-Marc Schmidt, Thomas Plos, Mario Kirschbaum, Michael Hutter, Marcel Medwed, and Christoph Herbst. Side-channel leakage across borders. In *International Conference on Smart Card Research and Advanced Applications*, pages 36–48. Springer, 2010.

[Sta08] John A. Stankovic. Wireless sensor networks. *Computer*, 41(10):92–95, October 2008.

[TSS17] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. CLKSCREW: Exposing the perils of security-oblivious energy management. In *26th USENIX Security Symposium*, pages 1057–1074, Vancouver, BC, 2017. USENIX Association.

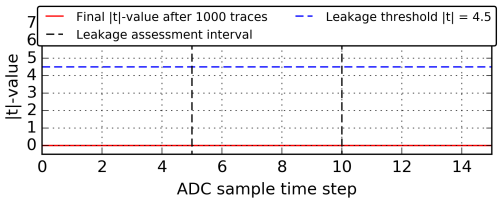
[Wal01] Colin D. Walter. Sliding windows succumbs to big mac attack. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 286–299. Springer, 2001.

[ZS18] Mark Zhao and G. Edward Suh. FPGA-based remote power side-channel attacks. In *Symposium on Security and Privacy (S&P)*, pages 805–820, May 2018.

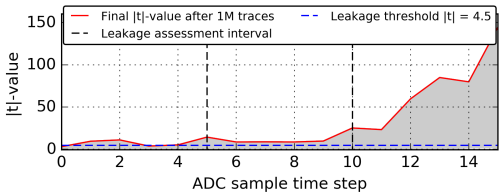
A Results of all Leakage Assessments

A.1 ESP32-devkitC

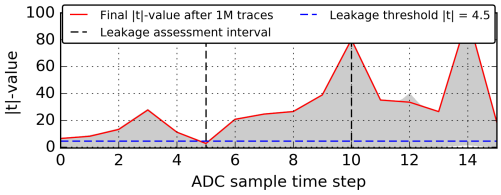
A.1.1 AES



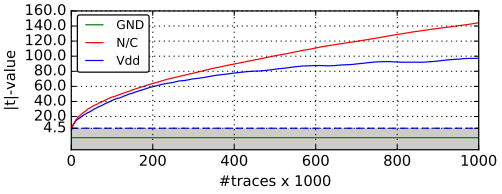
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1k traces collected during AES encryptions on the ESP32-devkitC with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the ESP32-devkitC with the ADC pin disconnected (N/C).



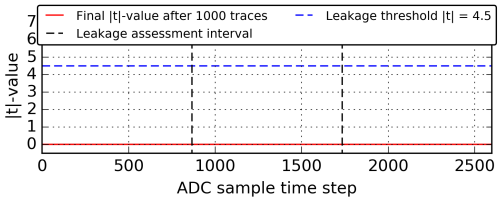
(c) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the ESP32-devkitC with the ADC pin connected to V_{dd}.



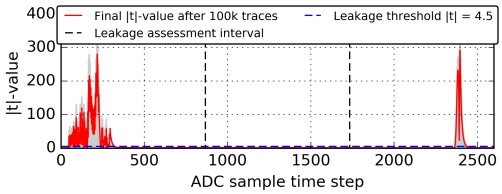
(d) Leakage Assessment progress on mbedTLS AES with {Vdd, GND, N/C} connected to the ADC on the ESP32-devkitC.

Figure 13: Results of Leakage Assessments on ESP32-devkitC for mbedTLS AES.

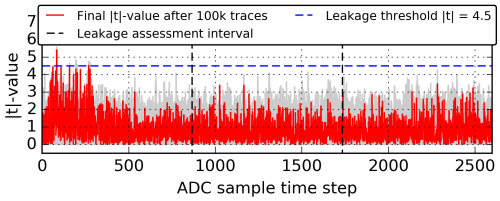
A.1.2 Sliding Window Modular Exponentiation



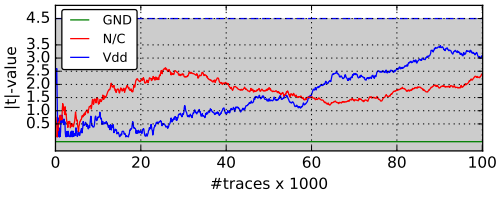
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1k traces collected during modular exponentiation on the ESP32-devkitC with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the ESP32-devkitC with the ADC pin disconnected (N/C).



(c) First order leakage assessment result based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the ESP32-devkitC with the ADC connected to V_{dd}.

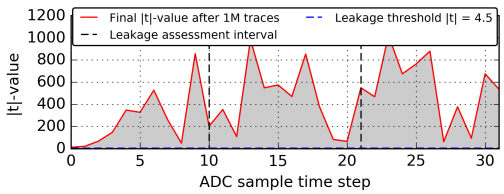


(d) Leakage Assessment progress on mbedTLS modular exponentiation with {Vdd, GND, N/C} connected to the ADC on the ESP32-devkitC.

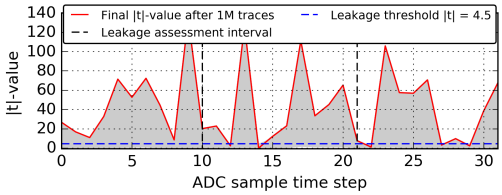
Figure 14: Results of Leakage Assessments on ESP32-devkitC for mbedTLS modular exponentiation.

A.2 STM32F407VG Discovery #1

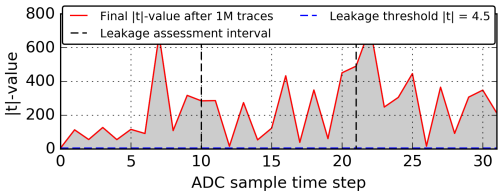
A.2.1 AES



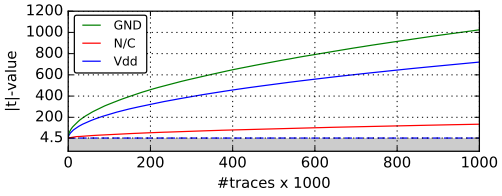
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #1 with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #1 with the ADC pin disconnected (N/C).



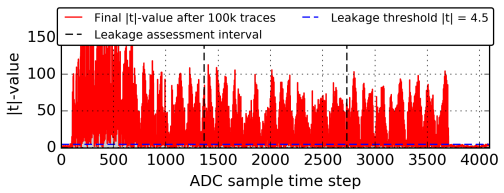
(c) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #1 with the ADC pin connected to V_{dd}.



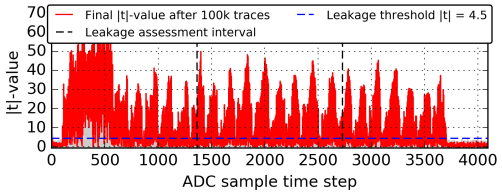
(d) Leakage Assessment progress on mbedTLS AES with {Vdd, GND, N/C} connected to the ADC on the STM32F407VG Discovery #1.

Figure 15: Results of Leakage Assessment on STM32F407VG Discovery #1 for mbedTLS AES.

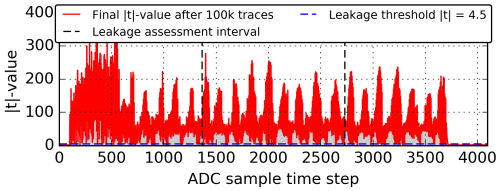
A.2.2 Sliding Window Modular Exponentiation



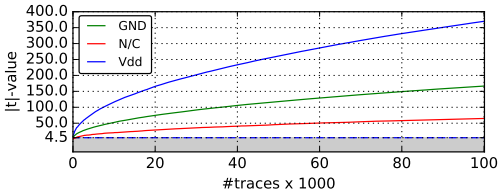
(a) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #1 with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #1 with the ADC pin disconnected (N/C).



(c) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #1 with the ADC pin connected to V_{dd}.

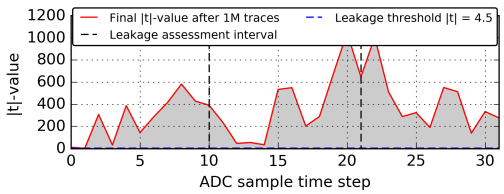


(d) Leakage Assessment progress on mbedTLS modular exponentiation with {Vdd, GND, N/C} connected to the ADC on the STM32F407VG Discovery #1.

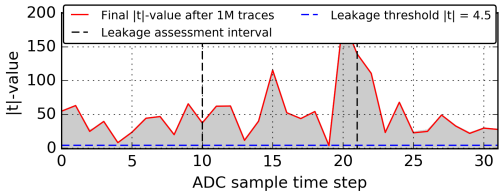
Figure 16: Results of Leakage Assessments on STM32F407VG Discovery #1 for mbedTLS modular exponentiation.

A.3 STM32F407VG Discovery #2

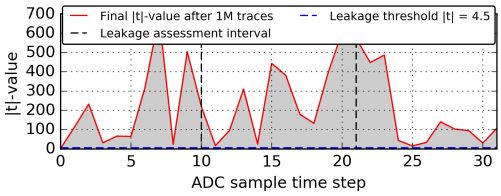
A.3.1 AES



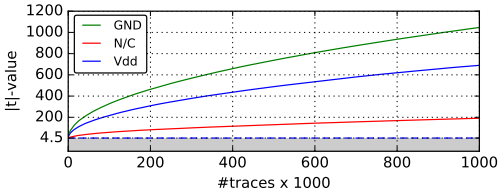
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #2 with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #2 with the ADC pin disconnected (N/C).



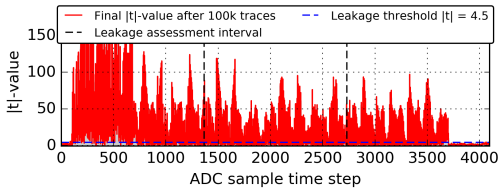
(c) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #2 with the ADC pin connected to V_{dd}.



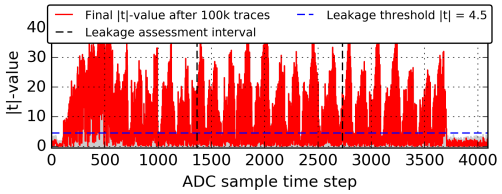
(d) Leakage Assessment progress on mbedTLS AES with {Vdd, GND, N/C} connected to the ADC on the STM32F407VG Discovery #2.

Figure 17: Results of Leakage Assessments on STM32F407VG Discovery #2 for mbedTLS AES.

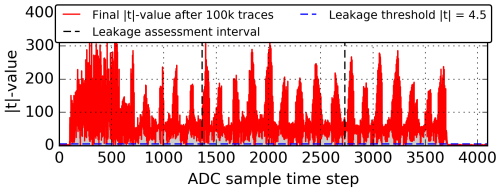
A.3.2 Sliding Window Modular Exponentiation



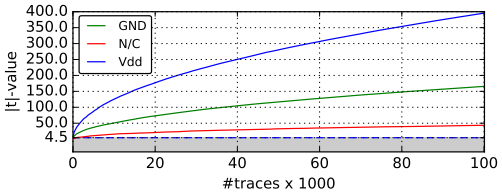
(a) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #2 with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #2 with the ADC pin disconnected (N/C).



(c) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #2 with the ADC pin connected to V_{dd}.

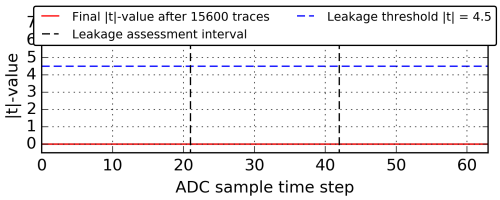


(d) Leakage Assessment progress on mbedTLS modular exponentiation with {Vdd, GND, N/C} connected to the ADC on the STM32F407VG Discovery #2.

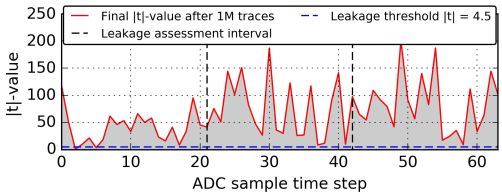
Figure 18: Results of Leakage Assessments on STM32F407VG Discovery #2 for mbedTLS modular exponentiation.

A.4 STM32L475 IoT Node

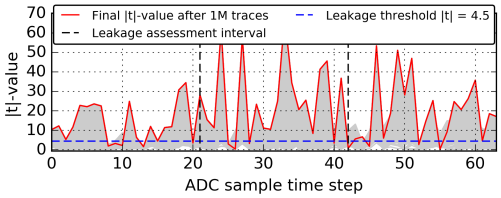
A.4.1 AES



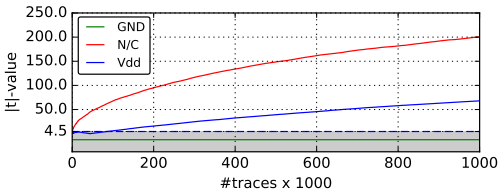
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32L475 IoT Node with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32L475 IoT Node with the ADC pin disconnected (N/C).



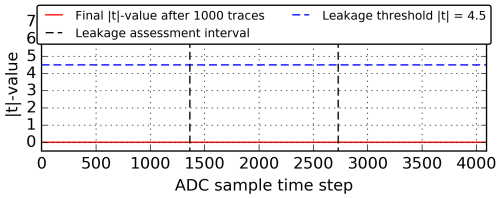
(c) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32L475 IoT Node with the ADC pin connected to V_{dd}.



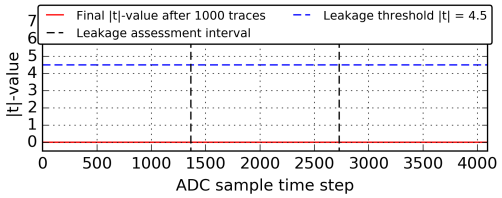
(d) Leakage Assessment progress on mbedTLS AES with {Vdd, GND, N/C} connected to the ADC on the STM32L475 IoT Node.

Figure 19: Results of Leakage Assessments on STM32L475 IoT Node for mbedTLS AES.

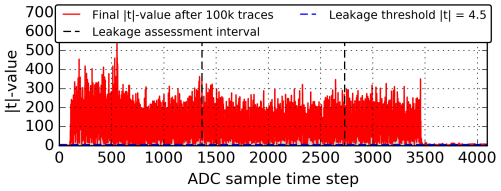
A.4.2 Sliding Window Modular Exponentiation



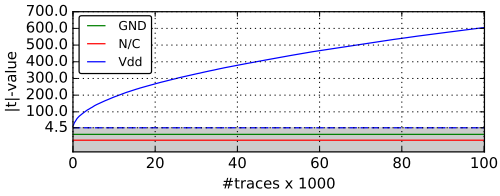
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1k traces collected during modular exponentiation on the STM32L475 IoT Node with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1k traces collected during modular exponentiation on the STM32L475 IoT Node with the ADC pin disconnected (N/C).



(c) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32L475 IoT Node with the ADC pin connected to V_{dd}.

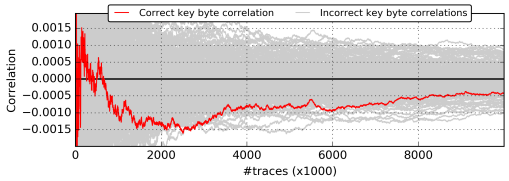


(d) Leakage Assessment progress on mbedTLS modular exponentiation with {Vdd, GND, N/C} connected to the ADC on the STM32L475 IoT Node.

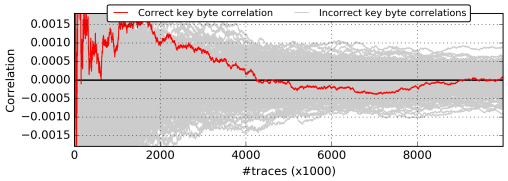
Figure 20: Results of Leakage Assessments on STM32L475 IoT Node for mbedTLS modular exponentiation.

B Results of CPA for all secret AES key bytes on Vdd and GND

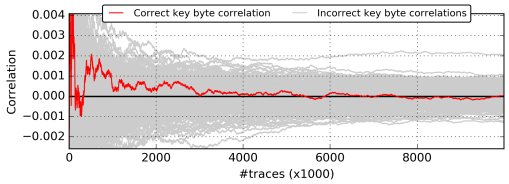
B.1 ADC pin connected to GND



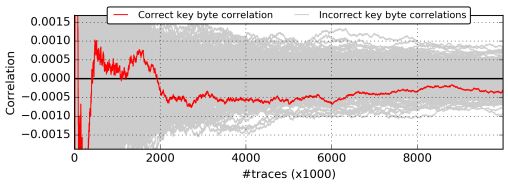
(a) CPA progress for the 0th secret key byte



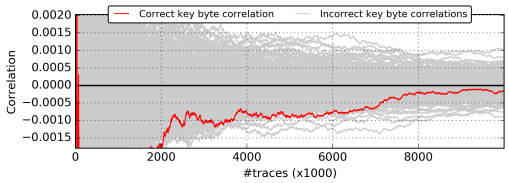
(b) CPA progress for the 1st secret key byte



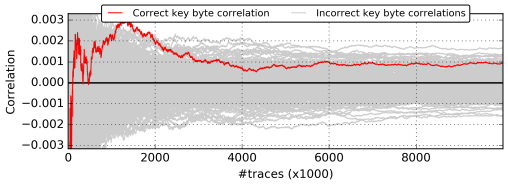
(c) CPA progress for the 2nd secret key byte



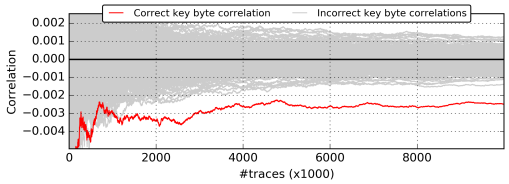
(d) CPA progress for the 3rd secret key byte



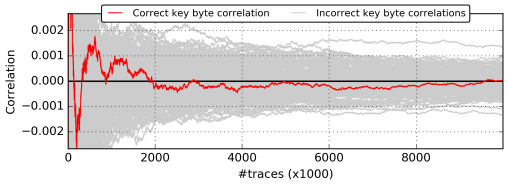
(e) CPA progress for the 4th secret key byte



(f) CPA progress for the 5th secret key byte

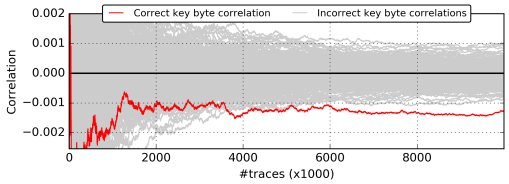


(g) CPA progress for the 6th secret key byte

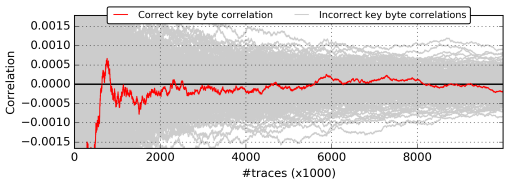


(h) CPA progress for the 7th secret key byte

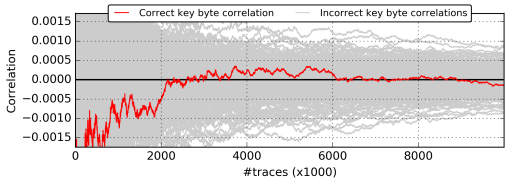
Figure 21: Results of a CPA attack on the last secret round key (bytes 0 to 7) of AES on the STM32F407VG Discovery #1 @168MHz with the ADC connected to GND and the program compiled with the -Os optimization option. Each plot shows the correlation progress of all 256 key candidates for a specific key byte over $10M$ traces and the respective correct key candidate is marked red.



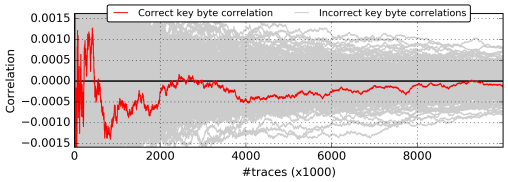
(a) CPA progress for the 8th secret key byte



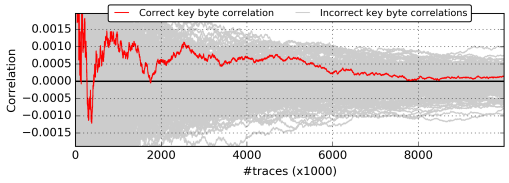
(b) CPA progress for the 9th secret key byte



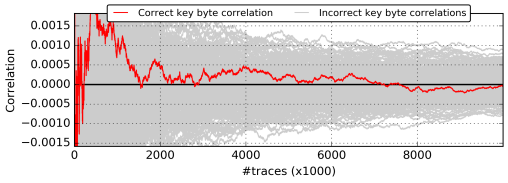
(c) CPA progress for the 10th secret key byte



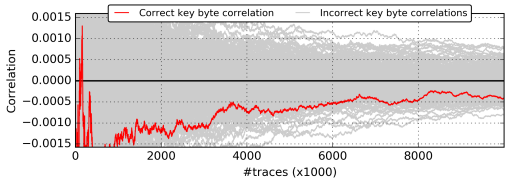
(d) CPA progress for the 11th secret key byte



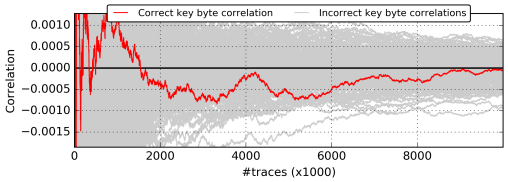
(e) CPA progress for the 12th secret key byte



(f) CPA progress for the 13th secret key byte



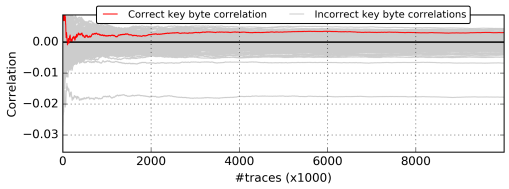
(g) CPA progress for the 14th secret key byte



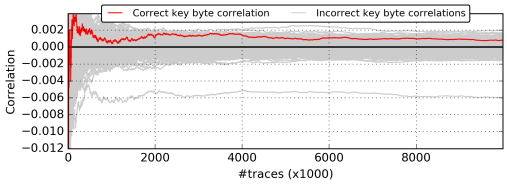
(h) CPA progress for the 15th secret key byte

Figure 22: Results of a CPA attack on the last secret round key (bytes 8 to 15) of AES on the STM32F407VG Discovery #1 @168MHz with the ADC connected to GND and the program compiled with the -Os optimization option. Each plot shows the correlation progress of all 256 key candidates for a specific key byte over $10M$ traces and the respective correct key candidate is marked red.

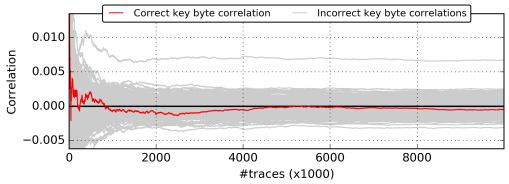
B.2 ADC pin connected to Vdd



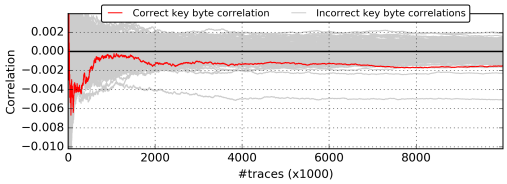
(a) CPA progress for the 0th secret key byte



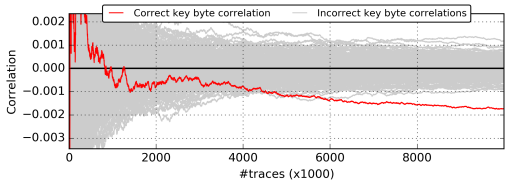
(b) CPA progress for the 1st secret key byte



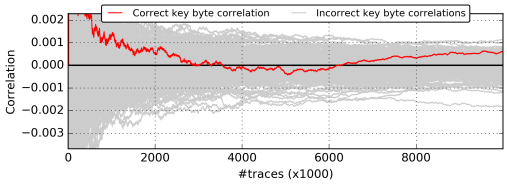
(c) CPA progress for the 2nd secret key byte



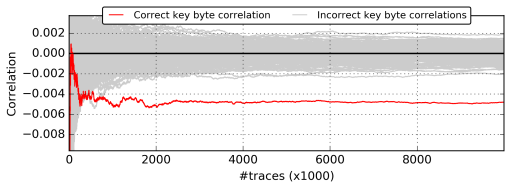
(d) CPA progress for the 3rd secret key byte



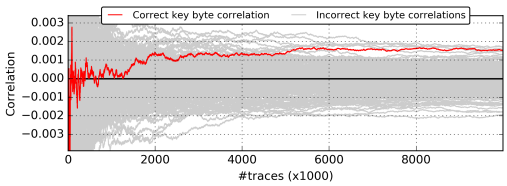
(e) CPA progress for the 4th secret key byte



(f) CPA progress for the 5th secret key byte

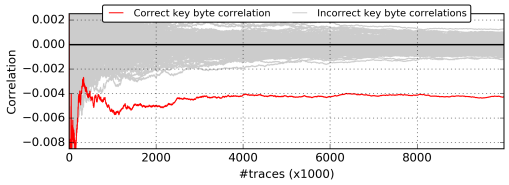


(g) CPA progress for the 6th secret key byte

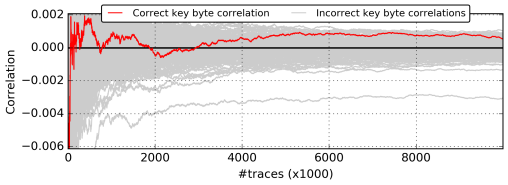


(h) CPA progress for the 7th secret key byte

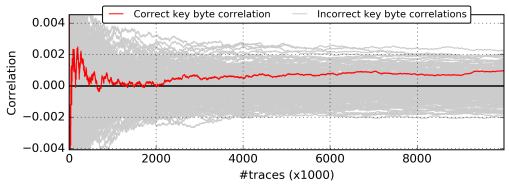
Figure 23: Results of a CPA attack on the last secret round key (bytes 0 to 7) of AES on the STM32F407VG Discovery #2 @56MHz with the ADC connected to Vdd and the program compiled with the -O0 optimization option. Each plot shows the correlation progress of all 256 key candidates for a specific key byte over $10M$ traces and the respective correct key candidate is marked red.



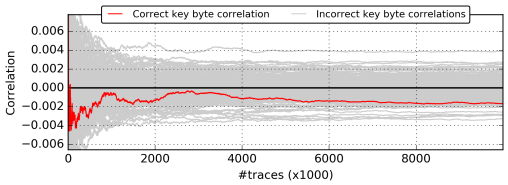
(a) CPA progress for the 8th secret key byte



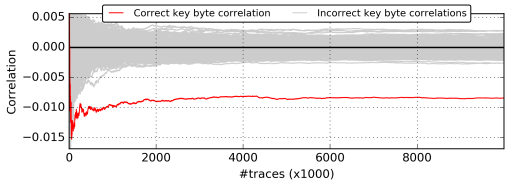
(b) CPA progress for the 9th secret key byte



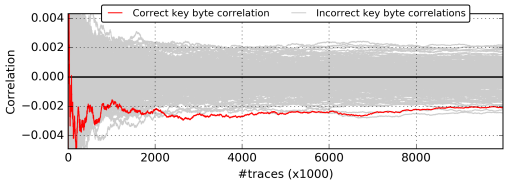
(c) CPA progress for the 10th secret key byte



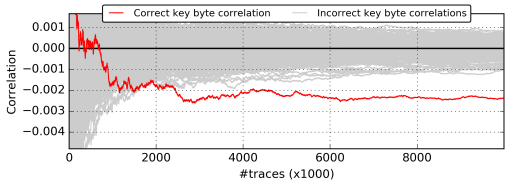
(d) CPA progress for the 11th secret key byte



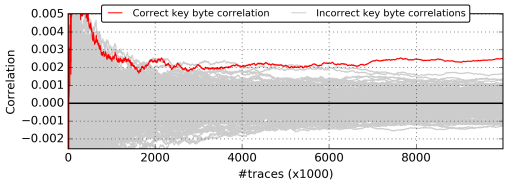
(e) CPA progress for the 12th secret key byte



(f) CPA progress for the 13th secret key byte



(g) CPA progress for the 14th secret key byte



(h) CPA progress for the 15th secret key byte

Figure 24: Results of a CPA attack on the last secret round key (bytes 8 to 15) of AES on the STM32F407VG Discovery #2 @168MHz with the ADC connected to Vdd and the program compiled with the -O0 optimization option. Each plot shows the correlation progress of all 256 key candidates for a specific key byte over $10M$ traces and the respective correct key candidate is marked red.

C Simplified Source Code of the Experiments

Listing 1: generic adcTask to sample the ADC for traces

```
// adcHandle = ..
void adcTask(void * pvParameters) {
  while(1) {
    // wait for notify from mbedTask:
    while (ulTaskNotifyTake(pdTRUE, portMAX_DELAY) == 0);
    adc_get_samples(); // see details below (esp32: CPU, stm32: DMA)
    uart_send(adc_data, sizeof(adc_data));
    xTaskNotifyGive( mbedHandle ); // notify mbedTask
  }
}
```

Listing 2: generic mbedTask to run mbedTLS AES or modular exponentiation

```
// mbedHandle = ..
void mbedTask(void * pvParameters) {
  // [...] init contexts/secrets for mbedtls here
  while(1) {
    // read message from uart
    uart_read(msg, sizeof(msg));
#ifdef EXP
    mbedtls_mpi_read_string(&g, 16, msg)
#endif
    // start ADC in other task:
    xTaskNotifyGive( adcHandle ); // notify adcTask
#ifdef EXP
    mbedtls_mpi_exp_mod(&dummy, &g, &e, &modulus, NULL)
#else // AES
    mbedtls_internal_aes_encrypt(&ctx, msg, dummy);
#endif
    // wait for notify from adcTask:
    while (ulTaskNotifyTake(pdTRUE, portMAX_DELAY) == 0);
  }
}
```

Listing 3: adc_get_samples for ESP32 CPU Task-based

```
static inline void adc_get_samples() {
  for (int i=0;i<ADC_WORDS;i++) {
    adc_data[i] = adc1_get_raw(ADC_CHAN_SEL);
  }
}
```

Listing 4: adc_get_samples for ESP32 ULP-based, plus ULP assembly code (adc.S). Please note, in the actual implementation we directly used the ULP from the mbedTask instead of a separate adcTask.

```
static inline void adc_get_samples() {
  adc1_ulp_enable();
  ulp_load_binary(0, ulp_main_bin_start,ulp_bin_size);
  ulp_set_wakeup_period(0, 1000);
  while(((volatile typeof(ulp_sync_back)) ulp_sync_back) == 0);
  *((volatile typeof(ulp_sync_back)*) &ulp_sync_back) = 0;
  uint32_t* p_ulp_adc_data = &ulp_adc_data;
  for (int i=0;i<ADC_WORDS;i++) {
```

```
    adc_data[i] = (uint16_t) (*p_ulp_adc_data++) & 0xffff;
  }
}

adc.S:
    move r0, adc_data
  measure:
    adc r2, adc_nr, adc_channel + 1
    st r2,r0,0
    add r0, r0, 1
    jumpr measure, adc_data+ADC_WORDS, lt
    // sync back to main cpu, which spinlocks:
    move r1, sync_back
    move r2, 0x0001
    st r2, r1,0
    halt
```

Listing 5: adc_get_samples for STM32 and ADC-DMA interrupt handler

```
static inline void adc_get_samples()
{
  // start to acquire ADC samples through DMA
  HAL_ADC_Start_DMA(&hadc3, adc_data, ADC_WORDS)
  // wait for ADC/DMA to finish while mbedTask executes
  osSignalWait(0x0001, osWaitForever);
  // make sure DMA is stopped
  HAL_ADC_Stop_DMA(&hadc3);
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
  osSignalSet(adcHandle, 0x0001);
}
```