# An Inside Job: Remote Power Analysis Attacks on FPGAs

Falk Schellenberg*‡, Dennis R.E. Gnad†‡, Amir Moradi*, and Mehdi B. Tahoori†
*Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
†Institute of Computer Engineering, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
*{falk.schellenberg, amir.moradi}@rub.de †{dennis.gnad, mehdi.tahoori}@kit.edu
‡These authors contributed equally to this work.

***Abstract*—Hardware Trojans have gained increasing interest during the past few years. Undeniably, the detection of such malicious designs needs a deep understanding of how they can practically be built and developed. In this work we present a design methodology dedicated to FPGAs which allows measuring a fraction of the dynamic power consumption. More precisely, we develop internal sensors which are based on FPGA primitives, and transfer the internally-measured side-channel leakages outside. These are distributed and calibrated delay sensors which can indirectly measure voltage fluctuations due to power consumption. By means of a cryptographic core as a case study, we present different settings and parameters for our employed sensors. Using their side-channel measurements, we further exhibit practical key-recovery attacks confirming the applicability of the underlying measurement methodology.**
**This opens a new door to integrate hardware Trojans in a) applications where the FPGA is remotely accessible and b) FPGA-based multi-user platforms where the reconfigurable resources are shared among different users. This type of Trojan is highly difficult to detect since there is no signal connection between targeted (cryptographic) core and the internally-deployed sensors.**

## I. Introduction

Cryptographic devices often deal with secret information as well as privacy of the users. So-called Side-Channel Analysis (SCA) attacks target the implementation of cryptographic schemes and are independent of their mathematical security. For example, [3] exploits the response time of an RSA implementation to retrieve the used secret key. Introduction of Differential Power Analysis (DPA) attacks [16] resulted in extensive research in refining attacks and developing countermeasures. Although timing attacks might even work over the Internet, power analysis attacks are thought to require physical access to the device, i.e., to connect an oscilloscope to measure the power consumption or the electromagnetic emanation in the near proximity. Yet, in the following, we prove this assumption to be wrong. This falls well within the line what has been seen for fault attacks. Before Rowhammer [14], fault attacks were thought to require some sort of physical access to induce a fault into the target. Instead, the attack can lead to pure-software based privilege escalation from an underprivileged user. Furthermore, it can be introduced remotely as well, even at a very high abstraction level [11].

For side-channel attacks, the dynamic power consumption originating from the switching of transistors is usually targeted. Our methodology is based on the work presented in [10], in which a mechanism to capture the fluctuation of the internal supply voltage of FPGAs is shown. It is in fact shown that the supply voltage at different locations of a Power Distribution Network (PDN) is not constant and depends on the activity of the logic. We followed the same principle and built internal sensors to locally monitor the dynamic change in the supply voltage. As a proof of concept, we conducted our experiments on a Spartan-6 FPGA, where an AES encryption module as the targeted cryptographic core is implemented. Indeed, the results show that such sensors enable side-channel attacks retrieving the secret key and is nearly as powerful as when using an external measurement.

We highlight two important properties of the proposed attack enabled by these sensors: a) it does not require a signal connection to the targeted core and b) it can be implemented using general-purpose logic available on any FPGA. These properties lead to a large threat when using 3rd party IP cores, considering both ASIC and FPGA implementations. Furthermore, it enables power attacks in emerging use-cases for FPGAs, such as fabric being shared among multiple users in the cloud [7] or the FPGA being part of a complex System on Chip (SoC). In such scenarios, an attacker might be able to deploy a voltage sensor unnoticed, essentially acting as a hardware Trojan to spy on the power consumption of the unaware victim. This seems contradictory to one of the motivations of using FPGA fabric as accelerator for cryptographic primitives in SoC: although enabling SoCs to receive security patches in hardware well after the design cycle [4], it may open doors to the previously unknown threats and attacks.

Considering the related works, the first powerful hardware Trojan has been presented in [15], where the Trojan inserted at HDL level of a CPU design would give the attacker unlimited access to the CPU resources. Further examples include malicious designs (at netlist and HDL level) made public during the student hardware Trojan challenge ICCD 2011 [23] or stealthy Trojans at the layout level [2]. Other works like [13], [18], [19] have shown methods to build malicious designs which only leak out the secrets when the attacker conducts specific SCA attacks. In [8], a design methodology for building stealthy parametric hardware Trojans and its application to bug attacks has been proposed. Other works, including [17], [24], propose Trojans which are triggered by aging or reduced

supply voltage. In [2], a Trojan is embedded into an SCA-resistant design, and would result in the cryptographic keys leaking through the same side channel but only under a particular condition, e.g., by means of a certain power model.

The Trojan we present in this work aims at leaking the cryptographic keys through a side channel as well. However, what makes our work different to the state of the art is its remotely accessible feature. In short, we present how to design power consumption sensors – synthesizable with FPGA primitives – which can be placed in another module next to, or even far from, the cryptographic core.

**Outline:** The remaining paper is structured as follows: In Section II, we elaborate the adversary model in more detail and explain the required background knowledge regarding PDNs and the voltage sensors. Subsequently, in Section III, we explain the implementation of the sensor and provide a discussion of the experimental results in Section IV. Section V concludes our paper.

## II. Preliminaries

### A. Adversary Model and Threat Analysis

Considering Fig. 1, we assume two scenarios for the adversary in our work. In both systems, the adversary's goal is to extract secret information from the other system components, with only access to the PDN, and no signal connections. In the first one, the adversary has partial access to the FPGA fabric, whose resources are shared among multiple users, e.g., FPGA accelerators shared in the cloud [7]. When the sensors are hidden in a complex application – which for instance needs to communicate with the outside of the FPGA – the inspection over the design would not detect any connection between the cryptographic module and the rest of the application, i.e., low chance for the Trojan to be detected. Automated isolation and verification countermeasures for FPGAs with user-controlled logic especially in data centers have already been proposed in [25]. Such techniques usually employ some physical gap between different IP cores with well-formed interfaces [6], [12]. Yet, we later show that such a barrier might be breached with internal voltage sensors, even when the sensors are placed far away from the target.

In the second scenario, an attacker has full access to the FPGA while the FPGA is part of a large system like an SoC where CPUs reside on the same die. For example, we can recall reconfigurable fabric of an SoC, where 3rd party users are allowed to use the FPGA. Any underprivileged user with access to the FPGA fabric can embed the sensors, thereby potentially monitoring the voltage of the whole SoC. This is an increasing threat under the trend of accelerator-use.

Note that, as opposed to the previous works about covert channels passing through this isolation, e.g., by electrical coupling [9] or even by temperature [20], we do not alter the attacked IP core in any form and only monitor the unintentional power consumption. The effects of electrical coupling have been further investigated in [5], [28].

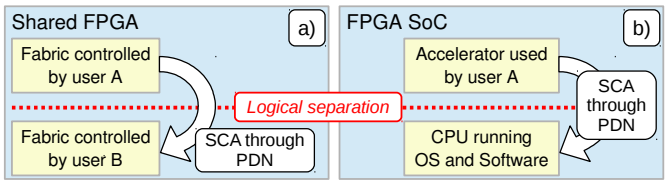


Fig. 1: Two scenarios of SCA attacks, where the circuits are logically separated, but share the same PDN. **a)** In a shared FPGA, one user (A) can attack another (B). **b)** In an FPGA SoC, a user with current access to the FPGA accelerator can attack any software or operating system on the CPU.

### B. Background on Power Distribution Networks (PDNs)

Every modern IC is supplied by power through a complex PDN that starts at the printed circuit board (PCB) and spans to individual logic gates in the IC. These PDNs consist of resistive, capacitive and inductive components, both at board level, and on-chip, in the form of a power mesh. Depending on the operating conditions, and thus power consumption, the voltage in the network is not perfectly stable [1].

Voltage drops in these PDNs can be attributed to a dynamic change in current through inductive components by a $L\,dI/dt$ drop, or a steady state offset, depending on the absolute value of current through resistive components as an $IR$ drop. Together, they lead to a voltage drop in the form of $V_{drop} = L \cdot dI/dt + IR$ [1]. Variations in the operating conditions will be reflected in the power consumption, resulting in a change in the supply current and voltage ($P = V \cdot I$). Because the PDN is not ideal, both changes in $I$ and $V$ can be measured depending on the power.

Thus, the operating conditions of a circuit lead to differences in power, which propagate through the complete PDN as voltage fluctuations. These voltage fluctuations impact circuit delay that can be measured to reveal power consumption information about a running workload. SCA attacks can use this information, for instance to recover secret keys.

By implementing suitable sensors, a hardware Trojan on the same PDN can thus sense voltage fluctuation in other parts, and perform SCA attacks. This scenario is a threat with respect to hardware Trojans inserted at design time or foundry level. In this work, as shown in Fig. 1, we further extend this scenario to recent use-cases (i.e., shared FPGA) and scenarios with FPGA SoCs, where the Trojans can also be inserted after fabrication time by a third party.

Accordingly, the success of this attack depends on 1) being able to realize these sensors using the FPGA primitives, intended for digital logic, and 2) that enough voltage fluctuation originates from the module under attack. In the following we show that these two requirements are fulfilled and practically prove it by an SCA attack on an AES implementation.

### C. Voltage Drop Sensors

While many FPGAs feature dedicated internal power sensors, they are shown to be inappropriate for side-channel analysis [21]. Regarding the realization of suitable custom sensors based on reconfigurable logic, Zick et al. [27] showed an implementation that uses existing FPGA fabric to sense variation in supply voltage, based on the concept of measuring

the propagation time of a signal with Time-to-Digital Converters (TDCs), as shown in Fig. 2. In [10] it is shown that the activity from synthesized logic can sufficiently stimulate these sensors, and that voltage fluctuations have one of the highest impacts on path delay during runtime, i.e. temperature variation is negligible.

In this work we follow the same concept. The idea is to use a delay line, in which a clock signal propagates through a chain of buffers. As the delay of these buffers depends on the supply voltage, the buffers can be monitored as a surrogate of it. The delay line can be *tapped* by adding latches between these buffers. The latches are enabled with the same clock signal that is connected to the start of the delay line, and thus can show how far the clock can propagate through the buffers within the time the latches are enabled, i.e., half a clock cycle.

When any other circuit on the same PDN becomes active, power is consumed, leading to a voltage drop that slows down the buffers of the delay line, resulting in a reduced numeric value in the TDC's output register. As this unary value can be quite large (i.e., 64 bins in our case), a priority encoder is used to reduce this data to 6 bits. Because of a symptom that higher valued bins can sometimes be faster than lower valued bins, *bubble correction* needs to be applied [26].

To save area of the sensor, usually only the last bits of the buffer chain are tapped, as the delay of the buffers do not change enough to affect the complete delay line. Thus, Fig. 2 shows part of the delay chain to be *observable* and another part to be the *initial* delay. In FPGAs, the observable part is usually implemented using carry-chain primitives (for Xilinx: CARRY4), as they provide the finest resolution per bit. However, the initial delay is then based on elements with less area overhead, but higher delay, like multiple LUT and latch elements, typically available in any FPGA. We show an example floorplan of this sensor in Fig. 3.

Because the real delay of the elements used for the sensor are not known at design time, it has to be ensured that the clock signal reaches the latches through the observable delay line within the respective clock cycle. In addition, it should not reach to the last latches resulting in an output value saturated at the maximum. They need to be calibrated at runtime or through adjusting and re-mapping the design. Thus, either the delay line's length has to be adjusted to the right length, or a second phase-shifted clock has to be used on the latches. In this work, we basically adjusted the initial delay, sufficient for our proof-of-concept. Please note, a more sophisticated attacker would very well be able to use a combination of phase shift and initial delay adjustment.

What is missing in [27] and [10] is an evaluation on how the length of the initial delay will impact the time quantization, i.e. how much time each individual bit in the observable delay line represents, and in effect how detailed a voltage drop will be visible. It is also not discussed how this initial delay can be found, when not using phase-shifted clocks.

The primitives used in the observable delay line have their own delay, but the fluctuations they show are those of the entire delay line (initial and observable) until the respective

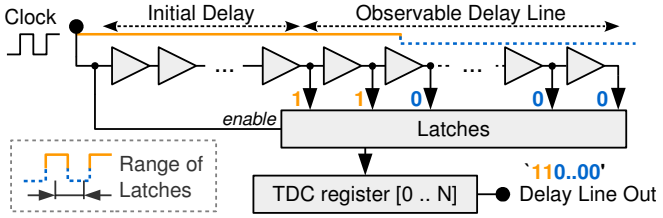


Fig. 2: Principle of the TDC Sensor from [10].

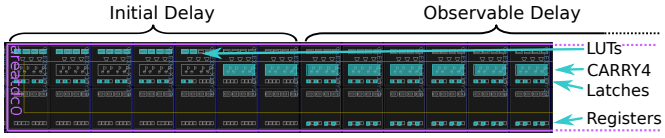


Fig. 3: Floorplan (rotated right) of one TDC Sensor with 18×(LUT, Latch) as part of the Initial Delay.

latch. Thus, the higher the initial delay is in relation to the observable delay, the more variation is zoomed into by the observable part. Thus, more fine-grained quantization levels are seen with higher initial delay, when checking the peak-to-peak variation of a given voltage fluctuation. In Table I, we show the initial delay and resulting variations observed in our experiments.

## III. Implementation of the PDN Trojan

To demonstrate the effectiveness of the internal sensors, we show a successful side-channel attack on an AES-128 implementation using our sensors, and compare it to an attack based on external power measurement. We first start by explaining the AES module and the target platform as well as our sensor and its properties in the following.

### A. AES module

The AES module is a relatively small implementation with a 32-bit datapath, occupying 265 Flip-Flops and 862 LUTs. The 128-bit plaintext, after being XORed with the first roundkey, is loaded into the state registers $s_i$. At every cipher round, which takes five clock cycles, first ShiftRows is performed. Afterwards, as shown in Fig. 4, at each clock cycle, four Sboxes followed by a MixColumn and AddRoundKey are performed while the state register is shifted column-wise. The four Sbox instances are shared with the (not shown) KeySchedule unit while ShiftRows is being performed. By bypassing the MixColumns during the last cipher round — in total after 50 clock cycles — the ciphertext is taken from the state register.

This AES module should generate much less voltage drop than seen in [10], since its footprint in this FPGA is only 0.3% of the total flip flops and 0.9% LUTs, versus 8% of flip flops in [10]. However, we show in the following that we can still gather sufficient information for the attack.

### B. Target Platform and Implementation

Fig. 5 gives an overview of our experimental setup. We ran our experiments on the widely-used side-channel evaluation platform SAKURA-G, featuring a *main* and a *control* Xilinx Spartan-6 FPGA. The main FPGA is a larger XC6SLX75 for

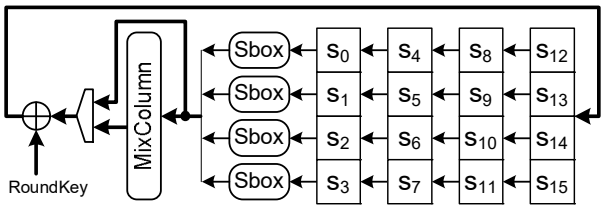


Fig. 4: Architecture of the underlying AES encryption core (ShiftRows and KeySchedule not shown)

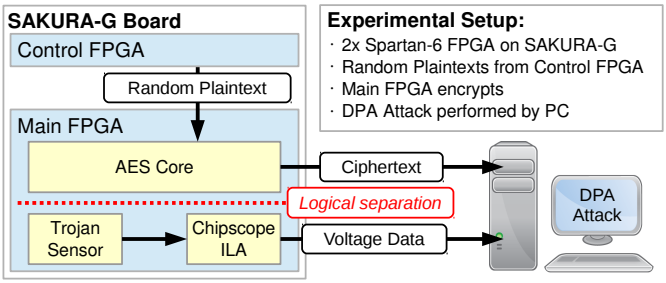


Fig. 5: Experimental setup showing the Sakura-G Board connected to our measurement PC, with Chipscope ILA used for data acquisition.

security implementations, controlled by an auxiliary Spartan-6 XC6SLX9. As a proof of concept, we considered the aforementioned AES encryption module as the targeted cryptographic core, implemented in the main FPGA and ran at a frequency of 24 MHz. The control FPGA generates random plaintexts to be encrypted on the main FPGA. Our Trojan circuit to measure voltage sits in the main FPGA, logically disconnected from the AES module. When the AES module sends out the ciphertext, we also receive the voltage data from our sensors on the workstation, by utilizing the Xilinx Chipscope Integrated Logic Analyzer (ILA). Here, the sensor values are first stored in the internal Block RAM (BRAM) and are then read out using the JTAG interface.

In Fig. 6 (left), we show the entire floorplan of the experimental setup. We only place our design in the lower part of the Spartan-6. In the center region, the AES core is fixed and the sensor is placed on the left side of the AES module. The FPGA slices used for the sensor's delay line, including latches and output register, are not shared with any other logic. For all the experiments, we kept the same placed and routed partition for the AES core, in order to keep the results comparable. However, the logic required for the ILA core are automatically added each time by the synthesis tool.

### C. Data Acquisition

We compare the efficiency of our developed sensor to a traditional measurement setup. To this end, we measured the voltage drop over a $\approx 0\,\Omega$ shunt resistor[1] in the Vdd path using a Picoscope 6403. Fig. 7 (top) depicts the resulting trace, measured at 625 MS/s showing approximately 120 quantization levels. Note that the round structure of the underlying AES implementation can be observed through ten similar patterns, each including five smaller peaks of each individual step, respectively. A 24 MHz clock is externally given to the

[1]The built-in shunt resistor of the SAKURA was shorted with a jumper.

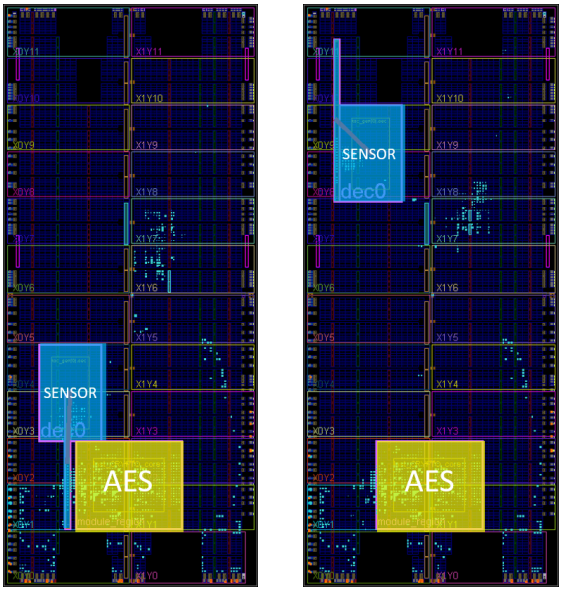


Fig. 6: Floorplans showing the Experimental Setup with all the relevant parts. **Left:** the internal sensor is placed close to the AES module. **Right:** the internal sensor is placed far away from the AES.

main FPGA which supplies both the AES module and our developed sensor. Thus, in contrast to the oscilloscope that has an independent time base, the internal sensor can sample the power consumption synchronously. The side-channel information is expected to be amplitude-modulated over the clock signal, i.e., it is visible at the clock peaks. Therefore, it would be enough to sample the power consumption (only) at this exact moment when the side-channel information leakage occurs. This drastically lowers the required sample frequency for a successful attack [22]. To verify this, we conducted different experiments by supplying the sensor with different frequencies (24 MHz, 48 MHz, 72 MHz, and 96 MHz) while the AES module always runs at 24 MHz. To this end, we used a Digital Clock Manager (DCM) to generate the desired clock frequencies based on the external 24 MHz clock.

### D. Sensor Feasibility Discussion

In our experiments we used Xilinx Chipscope to read the sensor values. Note that besides using the same clock source, there is no connection made between the AES core and the sensors, or the logic belonging to Chipscope. However, our developed sensor has the additional advantage that even if it is not synchronized with the AES clock, it catches all the variation that occurs in half of each clock cycle, since the clock traverses the delay chain during half of the clock period in which the latches are enabled (see Section II-C).

When we use a 180° phase-shifted clock (or negative latch-enable signal) for either the latches or a complete second sensor, the average of all variation in the time between two samples can be covered. This is an advantage over oscilloscope based samples, so even when the sensors clock domain would be separated, enough information can be inferred.

Although we use JTAG to connect to Chipscope in our experimental setup, an actual attacker would easily be able

TABLE I: Overview of different sensor's sampling frequency with AES module @ 24 MHz.

| Sampling frequency (MHz) | 96 | 72 | 48 | 24 |
|---|---|---|---|---|
| No. of primitives used for initial delay | 10 | 14 | 22 | 46 |
| Observed peak-to-peak variation | 6 | 6 | 8 | 15 |

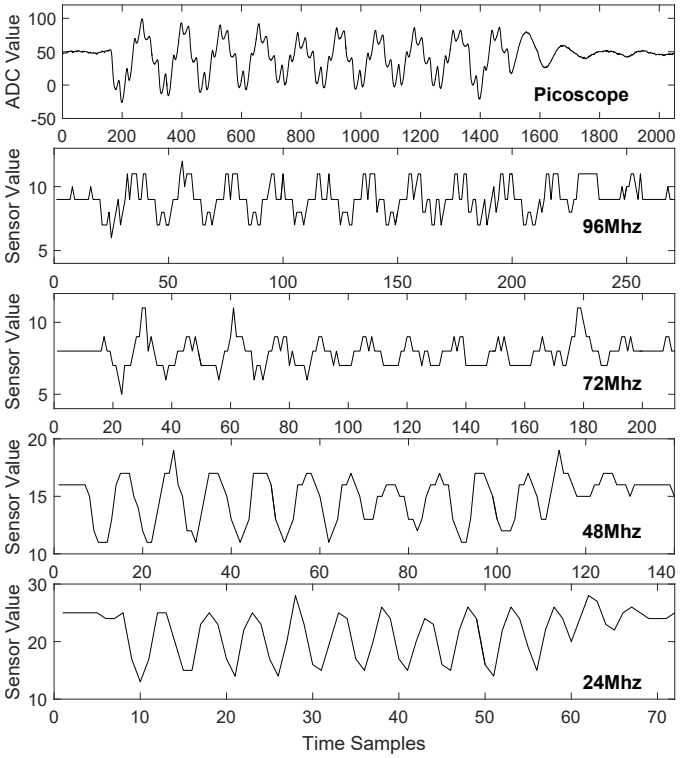


Fig. 7: Single traces measured using an oscilloscope (top) and using our developed sensor at different sampling frequencies (below). Time samples refer to the individual samples captured at the respective sampling rate.

to use whatever remote connection he has, to transmit the sensor values from internal BRAM to the outside. Since no logical signaling between the attacked module and the sensor is desired, the attacker would need to adjust a mechanism to trigger the start of saving the samples e.g., into the BRAM. This can be achieved by observing the measured signal itself and trigger by detecting a large peak. Indeed, the sensor value varies only slightly, indicating that the AES is inactive (cf. Fig. 7). The power consumption of the first round of the AES module results in a large negative peak in the sensor value, enabling a stable reference point for aligning the traces.

As described in Section II-C, for each sensor frequency, the initial delay of the sensor has to be adjusted. This leads to different levels of quantization, and thus the observed peak-to-peak variation. This relationship is verified by our experimental data in Fig. 7, where sensors at lower operating frequencies show higher peak-to-peak variation (cf. Table I).

## IV. Results

In the following, we provide experimental results showing a successful attack using the traces measured by the internal sensor. We compare the results to a traditional measurement setup, i.e., measuring the power consumption externally.

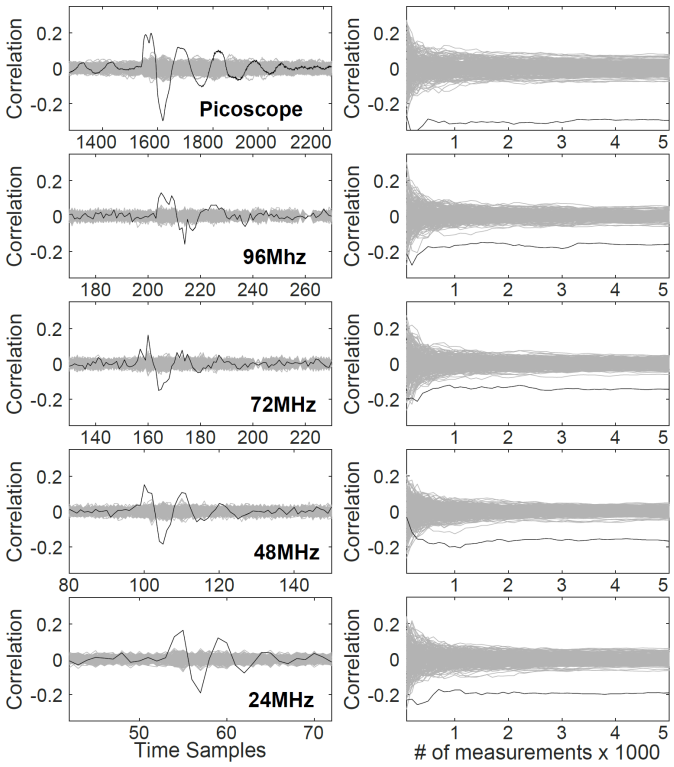


Fig. 8: Results using the oscilloscope (top row), using the internal sensor at different sampling frequencies (rows below), for each the correlation by means of 5 000 traces (left) and the progressive curves over the number of traces (right). The correct key hypothesis is marked in black. Time samples refer to the individual samples captured at the respective sampling rate.

As an example, we use a standard Correlation Power Analysis (CPA) attack on the AES module. Only a few bits within each byte of the internal state showed a strong leakage. Hence, we chose to predict only a single bit $b$ to evaluate our key hypothesis $k_{hyp}$. Note that this was identical both for the oscilloscope as well as the internal sensor. We ran the attack on all bits of the state. The results in the following correspond to the bit position $bitpos$ showing the highest correlation. We have chosen the state just before the SubBytes operation at the last round. Based on a ciphertext byte $c_i$, our model is

$$b = \mathsf{Sbox}^{-1}\left(k_{hyp} \oplus c_i\right) \wedge \left(2^{bitpos}\right).$$

### A. Sensor placed close to the AES core

Fig. 8 depicts the results using the oscilloscope as well as placing the internal sensor close to the AES core, with a gap of just 4 FPGA slices to avoid potential crosstalk. In all cases the correlation curves using 5 000 traces and the progressive curves over the number of traces are shown. Starting with the result using the oscilloscope, we observed the maximum correlation of approximately $-0.3$ for the correct key hypothesis. As shown, the attacks using the internally-measured traces by the sensor are also successful. The correct key hypothesis is clearly distinguished from the others, but with a slightly lower maximum correlation of about $-0.2$.

Comparing the results of the sensor at different sampling frequencies, we do not observe a large deviation. This is

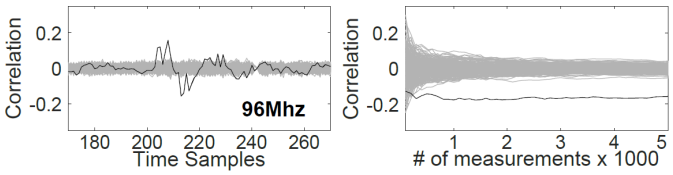


Fig. 9: Correlation using 5 000 traces (left) and progress of the maximum correlation over the number of traces (right) using the internal sensor at 96 MHz sampling frequency, placed far away from the AES module.

caused by the synchronous sampling as most of the information is contained in the respective peak anyway. Finally, we can observe that the higher resolution (more quantization steps) slightly improves the maximum correlation.

### B. Distant Sensor

We further investigated whether we still can detect any side-channel leakage in case the sensor is placed far away from the cryptographic block. We placed the sensor in the opposite region as far away as possible from the AES module. The right part of Fig. 6 depicts the corresponding layout. We examined this situation only with 96 MHz sampling frequency, i.e., the worst case in Fig. 8. The corresponding CPA results are depicted in Fig. 9, indicating that the successful attack is still possible with only a slight decrease in the correlation. This highlights the high risks involved when sharing an FPGA among multiple users. Note that for a real-world design, additional logic might be placed in between the AES and the sensor, resulting in noise and an increased number of required traces for a successful attack. Anyhow, such effects are also present for an external measurement. As stated, for the presented results we made use of a SAKURA-G board optimized for SCA evaluations. However, we were able to collect similar traces on standard Artix-7 and Zynq-7000 FPGA evaluation boards as well.

## V. Conclusion

We have shown a Trojan which exploits the Power Distribution Network (PDN) as side channel to successfully retrieve secret keys. To this end, we have developed sensors using reconfigurable resources of FPGAs to internally capture the dynamic power consumption. We analyzed the feasibility to sense minor variations through sufficient quantization. This relies on the characteristics of the PDN of an FPGA: When the FPGA logic toggles, the supply voltage fluctuates, which is observable through the PDN. The calibrated delay sensors allow inferring the power consumption indirectly from delay changes due to such voltage fluctuations.

The Trojan can be inserted remotely without requiring physical access and with no signal connection to the attacked module. Further, it provides a very strong side channel to the entire device, even if the sensor is not placed in proximity of the attacked module. In fact, our work is a proof of concept and warns that even with 100% logical separation between the modules, the PDN carries SCA information which makes many security threats and attacks possible. This reveals a major vulnerability in emerging applications of FPGAs, such as FPGA fabric being shared between multiple users. While we have used an FPGA for our experiments, this type of attack can be transferred to other ICs and SoCs as well.

## References


[1] K. Arabi, R. Saleh, and X. Meng. Power supply noise in socs: Metrics, management, and measurement. *Des. Test. Comput.*, 2007.

[2] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson. Stealthy Dopant-Level Hardware Trojans. In *CHES*, volume 8086 of *LNCS*. Springer, 2013.

[3] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.

[4] L. Bossuet, M. Grand, et al. Architectures of Flexible Symmetric Key Crypto Engines—a Survey: From Hardware Coprocessor to Multi-crypto-processor System on Chip. *ACM Comput. Surv.*, Aug. 2013.

[5] T. D. Cnudde, B. Bilgin, et al. Does Coupling Affect the Security of Masked Implementations? In *COSADE*, volume 10348 of *LNCS*. Springer, 2017.

[6] J. D. Corbett. The Xilinx Isolation Design Flow for Fault-Tolerant Systems, 2013.

[7] S. A. Fahmy, K. Vipin, and S. Shreejith. Virtualized FPGA Accelerators for Efficient Cloud Computing. In *CloudCom*. IEEE, 2015.

[8] S. Ghandali, G. T. Becker, D. Holcomb, and C. Paar. A Design Methodology for Stealthy Parametric Trojans and Its Application to Bug Attacks. In *CHES*, volume 9813 of *LNCS*. Springer, 2016.

[9] I. Giechaskiel and K. Eguro. Information Leakage Between FPGA Long Wires. *CoRR*, 2016.

[10] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori. Analysis of transient voltage fluctuations in FPGAs. In *FPT*. IEEE, 2016.

[11] D. Gruss, C. Maurice, and S. Mangard. Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript. In *DIMVA*, volume 9721 of *LNCS*. Springer, 2016.

[12] T. Huffmire, B. Brotherton, et al. Moats and Drawbridges: An Isolation Primitive for Reconfigurable Hardware Based Systems. In *IEEE S&P*. IEEE Computer Society, 2007.

[13] M. Kasper, A. Moradi, et al. Side channels as building blocks. *J. Cryptographic Engineering*, 2012.

[14] Y. Kim, R. Daly, et al. Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In *ISCA*. ACM/IEEE, 2014.

[15] S. T. King, J. Tucek, et al. Designing and Implementing Malicious Hardware. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[16] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO*, volume 1666 of *LNCS*. Springer, 1999.

[17] R. Kumar, P. Jovanovic, W. P. Burleson, and I. Polian. Parametric Trojans for Fault-Injection Attacks on Cryptographic Hardware. In *FDTC*. IEEE Computer Society, 2014.

[18] L. Lin, W. Burleson, and C. Paar. MOLES: Malicious off-chip leakage enabled by side-channels. In *ICCAD*. ACM, 2009.

[19] L. Lin, M. Kasper, et al. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In *CHES*, volume 5747 of *LNCS*. Springer, 2009.

[20] R. J. Masti, D. Rai, et al. Thermal Covert Channels on Multi-core Platforms. In *USENIX*. USENIX Association, 2015.

[21] C. Nagl. Exploiting the Virtex 6 System Monitor for Power-Analysis Attacks. Master's thesis, IAIK - Graz University of Technology, 2012.

[22] C. O'Flynn and Z. Chen. Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection. *Journal of Cryptographic Engineering*, Apr 2015.

[23] J. Rajendran, V. Jyothi, and R. Karri. Blue team red team approach to hardware trust assessment. In *ICCD*. IEEE Computer Society, 2011.

[24] Y. Shiyanovskii, F. G. Wolff, et al. Process reliability based trojans through NBTI and HCI effects. In *AHS*. IEEE, 2010.

[25] S. Trimberger and S. McNeil. Security of FPGAs in Data Centers. In *IVSW*. IEEE Computer Society, 2017.

[26] J. Wu. Several key issues on implementing delay line based tdcs using fpgas. *IEEE Trans. Nucl. Sci.*, June 2010.

[27] K. M. Zick, M. Srivastav, W. Zhang, and M. French. Sensing Nanosecond-scale Voltage Attacks and Natural Transients in FPGAs. In *FPGA*. ACM, 2013.

[28] L. Zussa, I. Exurville, et al. Evidence of an information leakage between logically independent blocks. In *CS2@HiPEAC*. ACM, 2015.