IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2023, No. 2, pp. 543–567. DOI:10.46586/tches.v2023.i2.543-567

# RDS: FPGA Routing Delay Sensors for Effective Remote Power Analysis Attacks

David Spielmann*, Ognjen Glamočanin* and Mirjana Stojilović

EPFL, Lausanne, Switzerland
david.spielmann@epfl.ch, ognjen.glamocanin@epfl.ch, mirjana.stojilovic@epfl.ch
* Authors equally contributed to this work

**Abstract.** State-of-the-art sensors for measuring FPGA voltage fluctuations are time-to-digital converters (TDCs). They allow detecting voltage fluctuations in the order of a few nanoseconds. The key building component of a TDC is a delay line, typically implemented as a chain of fast carry propagation multiplexers. In FPGAs, the fast carry chains are constrained to dedicated logic and routing, and need to be routed strictly vertically. In this work, we present an alternative approach to designing on-chip voltage sensors, in which the FPGA routing resources replace the carry logic. We present three variants of what we name a routing delay sensor (RDS): one vertically constrained, one horizontally constrained, and one free of any constraints. We perform a thorough experimental evaluation on both the Sakura-X side-channel evaluation board and the Alveo U200 datacenter card, to evaluate the performance of the RDS sensors in the context of a remote power side-channel analysis attack. The results show that our best RDS implementation in most cases outperforms the TDC. On average, for breaking the full 128-bit key of an AES-128 cryptographic core, an adversary requires 35% fewer side-channel traces when using the RDS than when using the TDC. Besides making the attack more effective, given the absence of the placement and routing constraint, the RDS sensor is also easier to deploy.

**Keywords:** FPGA · Multitenancy · Power Analysis Attack · On-chip sensors

## 1 Introduction

Traditionally, cloud computing platforms are based on central processing units (CPUs). In recent years, however, they have evolved to include a variety of more specialized computing hardware, e.g., graphics processing units (GPUs), Google's tensor processing units (TPUs), and, in particular, field-programmable gate arrays (FPGAs). Today, a number of commercial cloud service providers, such as Amazon AWS, Alibaba, Microsoft Azure, etc., provide their customers with the possibility of remotely accessing cloud FPGAs, to develop and test their hardware accelerators [AWS19, Azu, Hua, Ali].

With FPGAs in the cloud, FPGA-specific virtualization methods are being developed, with the goal of enabling efficient and secure use of FPGA resources. Such methods typically incorporate multitenancy, i.e., the possibility of multiple users sharing the FPGA fabric temporally and spatially. However, cloud FPGA multitenancy is not yet broadly deployed, as the associated security risks are still being investigated. A particular topic of interest is the threat of electrical-level attacks, in which an adversary either injects a disturbance in the shared power delivery network (PDN), with the intention of causing denial-of-service [GOT17] or erroneous computation of the victim [KGT18], or measures the power side-channel leakage to extract a secret (e.g., a cryptographic key) [RPD+18].

On-chip sensors suitable for remote power-analysis attacks can be classified in two groups: time-to-digital converters (TDCs) and frequency counters. The working principle


Received: 2022-10-15 Accepted: 2022-12-15 Published: 2023-03-06

is the same: instead of measuring the voltage directly, they sense the variations of the logic delay caused by the voltage fluctuations, which carry the side-channel information. The primary component in TDCs is a tapped delay line. In frequency counters, that is a ring oscillator (RO).

Ring-oscillator based sensors have a small footprint and are easy to be deployed, whereas TDCs occupy more logic resources and require careful and strict placement constraints. Yet, ROs suffer from low sensitivity, making them suitable for capturing slow-changing signals only. In comparison, TDCs can sense nanosecond-scale voltage variations and, as confirmed by previous studies, they are better suited for remote power side-channel attacks [ZS18, UJS+21, MLS+20, MDL+22]. Finally, attacks based on ring oscillators are easier to detect and prevent in cloud environments [KGT19, GCRS20].

In this paper, we present a novel FPGA on-chip voltage sensor design, fundamentally different from both TDCs and ROs. To pick up voltage variations, our sensor uses the type of FPGA resources that is the most abundant and least constrained: FPGA wires and routing multiplexers, i.e., FPGA routing resources. The routing-delay sensor (RDS) can be implemented in various ways: with or without a tapped delay line, with or without placement constraints. Furthermore, it can be made even more effective than TDC in the context of remote power side-channel attacks.

We begin by designing and implementing two RDS variants working on the principle of a tapped delay line; both use the routing resources only, but one is constrained vertically (VRDS), whereas the other is constrained horizontally (HRDS). Then, we remove the routing and placement constraints to obtain the third variant, which we name simply RDS. As we will demonstrate, the third design is the most performing of the three.

We perform extensive experiments to evaluate the success of a remote power side-channel attack against an AES-128 cryptographic core. To this end, three experimental platforms are used: Alveo U200 (AMD UltraScale+), Sakura-X side-channel evaluation board (AMD Kintex-7) and Digilent Basys 3 (AMD Artix-7). Experiments are repeated multiple times, the placement of the sensor and the AES is varied, as well as the secret key and the environment temperature conditions. We compare the three RDS variants to the TDC implementation commonly used in literature [ZSZF13, GCRS20, MTH+21, GNGT21] and the recently published voltage-fluctuation sensor VITI [UJS+21] (openly available).

Our results show that the final, third variant of our RDS sensor, is more effective not only than VRDS and HRDS, but also VITI and even TDC. When attacking an AES-128 on Sakura-X, the attack with the RDS sensor requires, on average, 35% fewer power side-channel traces to break the entire secret key with the TDC, with this number going as high as 80% in the extreme case. In our experiments, RDS outperforms TDC on an Alveo U200 datacenter card as well.

The remainder of the paper is organized as follows. In Section 2, we first describe the threat model of a remote power side-channel analysis attack on cloud FPGAs. Then, we explain the design and operation of time-to-digital converters for on-chip voltage sensing. Section 3 covers related work. In Section 4, we present the designs of our routing delay sensors and explain how to calibrate them. Section 5 describes the experimental evaluation approach and the corresponding hardware and software setups. Section 6 presents and discusses the results. Finally, Section 7 concludes the paper.

For reproducibility of the experiments and the results in this work, we make the sensor designs and the associated software openly available [SGS23].

## 2 Background

We start this background section by describing the threat model of the remote electrical-level attacks on shared FPGAs. Then, we explain the design and principle of operation of time-to-digital converters for on-chip voltage measurements.

## 2.1 Threat Model

This work follows the common threat model of remote electrical-level attack on multi-tenant FPGAs [GOT17,GCRS20,RPD+18,MS19], in which an FPGA in a datacenter or the cloud is spatially shared by multiple users. The FPGA tenants are given physically separate FPGA regions to deploy their circuits. Furthermore, the assigned regions are logically isolated. To access the external interfaces, such as PCI Express, or the off-chip memory, the tenants use dedicated FPGA logic called *shell*, deployed by the datacenter or the cloud service provider. The tenants are free to implement almost any FPGA circuit (the exception being circuits containing combinational loops [AWS19]) and set placement and routing constraints for their designs. Finally, the FPGA tenant applications share the on-chip power delivery network.

The adversary, in the assigned FPGA region, implements one or more on-chip voltage-fluctuation sensors, together with the control logic and the on-chip buffers, for saving the measurements. The adversary has the possibility of offloading the sensor traces for the off-chip analysis. The victim, on the other side, is performing encryption using a secret key and sending the ciphertexts over a public channel that can be observed by the adversary.

## 2.2 Time-to-Digital Converters

The suitability of FPGA time-to-digital converters for sensing on-chip voltage variations and natural transients in FPGAs was first investigated by Zick et al. [ZSZF13] almost a decade ago. However, it was only after the first works on remote denial-of-service attacks on multi-tenant FPGAs that these on-chip sensors regained the attention of researchers, primarily in the context of power side-channel attacks.

In Fig. 1, we illustrate the typical implementation of a TDC sensor. We can identify three parts: (1) a tapped delay line, commonly implemented using fast carry propagation logic and dedicated routing, (2) an output register (with every carry output driving one flip-flop), and (3) some circuitry (e.g., look-up tables, phase-locked loops, or IDELAY adjustable input delay elements) for tuning the phase shift between the sampling clock of the output register and the clock propagating through the delay line. These two clocks have the same frequency.

Careful tuning of the phase shift and the length of the tapped delay line is critical for correct sensor calibration, i.e., ensuring that only one clock transition is captured in the output register per sampling clock period. Additionally, the delay line must be properly formed by chaining the carry output of one FPGA slice to the carry input of the next one, where all the occupied slices are constrained to one vertical column of the FPGA; and, every carry output must drive precisely the corresponding flip-flop (FF) residing in the same slice. These strict placement constraints are necessary for ensuring the optimal sensitivity of this TDC sensor.

In the absence of any on-chip activity, the sensor output usually is constant (modulo

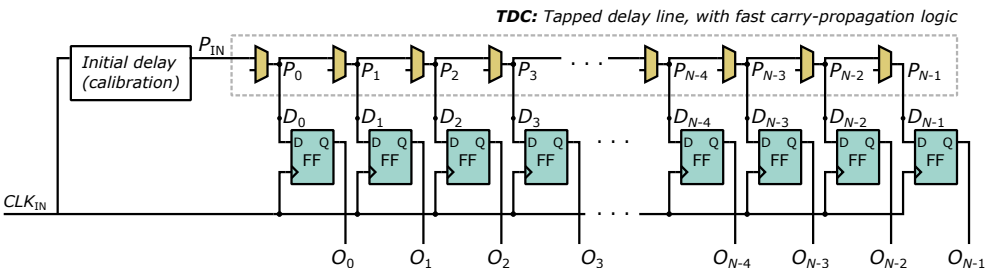



**Figure 1:** TDC sensor.

background noise) and determined by three parameters: the clock frequency, the initial delay (i.e., the phase shift), and the length of the delay line $N$. For a given clock frequency, the initial delay and the parameter $N$ are chosen so that the output register captures a single clock transition in every clock cycle. In other words, the output register should always be filled with a sequence of ones followed by the sequence of zeros—with a possibly imperfect transition—where the location of the flip-flop (FF) with the transition corresponds to the depth of propagation of the clock signal through the delay line. The value in the output register can be converted to the numerical value of one *sensor sample* using a thermometer code [GOT17] or by taking the Hamming weight of the bits in the output register [GCRS20, GNGT21], as we do in this work. Once the on-chip voltage starts fluctuating due to the activity of the victim circuit, the delay of the elements in the sensor change, and consequently, so does the sensor output. We consider the sensor well calibrated (i.e., the initial delay and the length of the delay line are well chosen) if, for the entire duration of the measurement, the Hamming weight of the output register lies in the range $0 < \mathrm{HW}(O) = \mathrm{HW}(O_0, O_1, ..., O_{N-1}) < N$ and only one clock edge is captured.

Let us define $d_i$ as the time the delayed clock signal takes to propagate from the input of the tapped delay line $P_{IN}$ to the input $D_i$ of the flip-flop $\mathrm{FF}_i$. For the TDC illustrated n Fig. 1, $d_i$ is, therefore, the following time difference:

$$d_i = t(D_i) - t(P_{IN}). \tag{1}$$

The sensitivity of the TDC sensor, i.e., the minimum signal change that can be detected, can be expressed as

$$S = \min(d_i - d_{i-1}),\ \ 0 < i < N. \tag{2}$$

Because the TDC has a tapped delay line, we can rewrite the expression for $d_i$ as the sum of the propagation delay through the first segment, last segment, and all the intermediate segments of the delay line:

$$d_i = \underbrace{t(P_0) - t(P_{IN})}_{\text{first segment}} + \underbrace{t(D_i) - t(P_i)}_{\text{last segment}} + \underbrace{\sum_{1 \le k \le i} t(P_k) - t(P_{k-1})}_{\text{intermediate segments}}, \tag{3}$$

where $P_i$ is at the output of the delay element $i$ in Fig. 1. Combining the expressions in Equations (1), (2), and (3), the sensitivity can be reformulated as:

$$S = \min\left( \underbrace{t(D_i) - t(P_i)}_{\text{routing resources}} + \underbrace{t(P_i) - t(P_{i-1})}_{\text{delay element}} - \underbrace{(t(D_{i-1}) - t(P_{i-1}))}_{\text{routing resources}} \right). \tag{4}$$

Therefore, the sensitivity of the TDC improves if, first, the input signal takes a very short time to pass through a delay element; and, second, if the delays through routing resources ($t(D_i) - t(P_i)$, for $0 \le i < N$) are all approximately equal. These two requirements have a direct impact on the way TDCs are often implemented on FPGAs: the former is satisfied by using fast carry propagation logic, while the latter is achieved by ensuring that each carry output drives only the corresponding flip-flops in the *same* slice. Routing outside of the slice boundaries is, in general, avoided. The exception is when it is necessary to connect the carry output of one slice to the carry input of another. Even then, dedicated carry-specific routing resources are used to minimize the penalty, and consequently, the tapped delay line has to be placed in a single FPGA column.

Another property of a tapped delay line is the monotonic increase of the propagation delay:

$$d_0 < d_1 < \cdots < d_{N-1}. \tag{5}$$

In practice, carry propagation logic in FPGAs is often implemented in the look-ahead fashion, and the delays from the carry input to the carry outputs of the same slice may not be strictly monotonically increasing [GCRS20]. Given that we apply Hamming weight instead of the thermometer code on the output register value, the mentioned lack of monotonicity is of no practical concern.

## 3 Related Work

In literature, the TDCs and the RO-based sensors have been used in the context of power side-channel attacks [ZS18, RPD+18, GCRS20, GDTLM19] and crosstalk side-channel attacks [GRE18, PRP+19]. They have been shown to be effective as covert communication receivers, where the sender is the FPGA, CPU, or even GPU sharing the common power delivery network [GRS20]. The TDC sensors have been used in a correlation power analysis attack on Amazon AWS F1 instances [GCRS20] and to recover the inputs to a neural network deployed on the same cloud FPGA instances [MTH+21]. On-chip sensors were shown effective in capturing the side-channel leakage across integrated circuits sharing the same board [SGMT18b] and against a CPU within the same system-on-chip [GDTLM19]. As ring oscillators are less effective for side-channel attacks [MLS+20] and ring-oscillator structures are relatively easy to detect [LMG+20, KGT19], we focus on the TDC-based sensors in this work.

Zick et al. [ZSZF13] were the first to introduce a phase shift (e.g., with a phase-locked loop) between the clock propagating through the delay line and the clock sampling the output register. This design choice allowed for reducing the length of the observable (i.e., tapped) part of the delay line and helped fit the TDC inside an FPGA column. In the early TDC designs of Zick et al. [ZSZF13] and Gnad et al. [GOT17], transparent latches were used to capture the sensor samples. In later works, latches were replaced by flip flops [GKT+20, GCRS20, MTH+21].

Various implementations of clock phase shifting have been proposed. To reduce jitter, Gnad et al. replaced the phase-locked loop with a chain of look-up tables (LUTs), latches, and fine carry elements [GNGT21]. They implemented a self-calibrated sensor (i.e., with the calibration in hardware): while monitoring the sensor output, the number of coarse delay elements (LUTs, latches) or fine delay elements (carry) is tuned as long as the desired calibration is not achieved. Udugama et al. followed this example and implemented another variant of a voltage-fluctuation sensor calibrated in hardware. For tuning the initial delay, Udugama et al. chose the adjustable input delay FPGA elements (IDELAY). In our work, we perform phase shifting using the coarse and fine delay elements, similarly to Gnad et al. [GNGT21]; however, we control the calibration from the software because it allows us to have much better control and flexibility, which is desirable when experimenting.

Recently, Udugama et al. proposed VITI [UJS+21], a delay-line-based sensor built with the goals of minimizing the use of logic resources and escaping from the strict placement constraints of the carry chains in TDCs. In VITI, the delay line is implemented using LUTs as delay elements. Additionally, LUTs are freely placed, and the connections between them are freely routed. By saving the resources and by using the LUTs instead of the carry-propagation logic, the resolution and the sensitivity of the sensor were sacrificed. Yet, the authors have shown that an adversary armed with VITI can, sometimes partially and sometimes fully, break the secret key of an AES-128 hardware module. Our proposed routing delay sensors are very different because they primarily use the FPGA routing resources to sense the on-chip voltage variations. The third variant of our sensor is free of any placement and routing constraints, similar to VITI, making it simple to deploy and difficult to detect. And this last variant, on average, has better sensitivity than the TDC and allows breaking the secret key faster. In this work, we implement one TDC [GNGT21] and one VITI [UJS+21] and use them as references for comparison with

our implementations of routing delay sensors.

# 4 Routing Delay Sensors

As described in Sections 2 and 3, the principle of operation of the FPGA on-chip voltage sensors is measuring and quantifying the change of the propagation delay of the carry logic (in TDCs) or LUTs (in ring oscillators and VITI). Even though it is not explicitly mentioned in the literature on remote power-side channel attacks, the routing multiplexers in the FPGA interconnect are affected by the on-chip voltage fluctuations and they too contribute, though to a lesser extent, to the delay variations captured by the sensors. Ahmed et al. explored techniques for optimizing FPGA logic circuitry for variable voltage supplies [ASB20]. As part of the study, the authors compared the impact of the power supply voltage on the propagation delay of LUT-dominated and routing-dominated signal paths, to find that (1) both were affected and (2) the LUT-dominated paths were affected more. Motivated by their findings, we ask ourselves the following research question: *if the delay line is built using FPGA routing resources only, how effective would such a sensor be in the remote power SCA attack setting?* To answer this question, we start by designing the following two variants of a routing delay sensor:

- VRDS, with the placement and the routing constrained *vertically*, as in TDCs, and
- HRDS, with the placement and the routing constrained *horizontally*, completely opposite of the TDCs.

In the remainder of this section, we present our VRDS and HRDS implementations and explain their operation. Then, we introduce a third and improved design. Finally, we discuss the calibration procedure and the challenges of portability and detectability of the routing delay sensors.

## 4.1 VRDS and HRDS

In Fig. 2, in the same style as in Fig. 1 for TDCs, we illustrate the design of our routing delay sensor, in which the delay line is implemented with FPGA routing resources (RR) only. We use the label RR to model the global interconnect; the local interconnect is modeled with the direct connection between $P_i$ and $D_i$, for $0 \leq i < N$. It is worth noting that all the expressions in Equations (1), (2), (3), and (4) hold here as well.

When placing and routing, we aim for a modular design and as uniform RR delay as possible across the entire delay line. Fig. 3a shows the placement of two subsequent FFs in the output register and the routing of the delay line for the VRDS. The delay element labeled RR in Fig. 2 is realized with a vertical wire segment of length one (the full line), whereas the segment between $P_i$ and $D_i$ is routed locally (dashed line). The placement of the output register is constrained to a single FPGA column. This placement and routing pattern is then repeated for the entire length of the sensor. Reusing the same connectivity pattern helps improve the sensitivity of the sensor, as the first and the third of the three terms in Eq. (4) become equal, thus canceling each other out.

The implementation of the HRDS is similar, except for using horizontal wires and constraining the FFs to a single FPGA row (Fig. 3b). However, there is one important difference between the HRDS and the VRDS. FPGA columns have a uniform architecture, whereas FPGA rows do not (e.g., in one FPGA row, there are CLBs, DSP blocks, RAM memories, etc.). As a consequence, in the case of the HRDS, the RR blocks in Fig. 2 no longer have a constant delay because the wires connecting two immediately neighboring CLBs are shorter than the wires between two CLBs that are separated by a column of DSP or memory blocks.

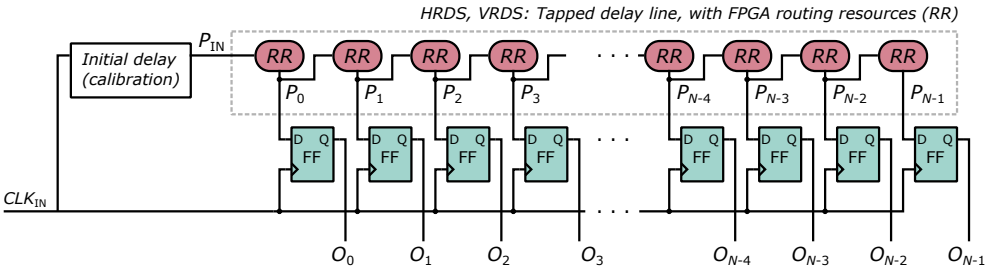
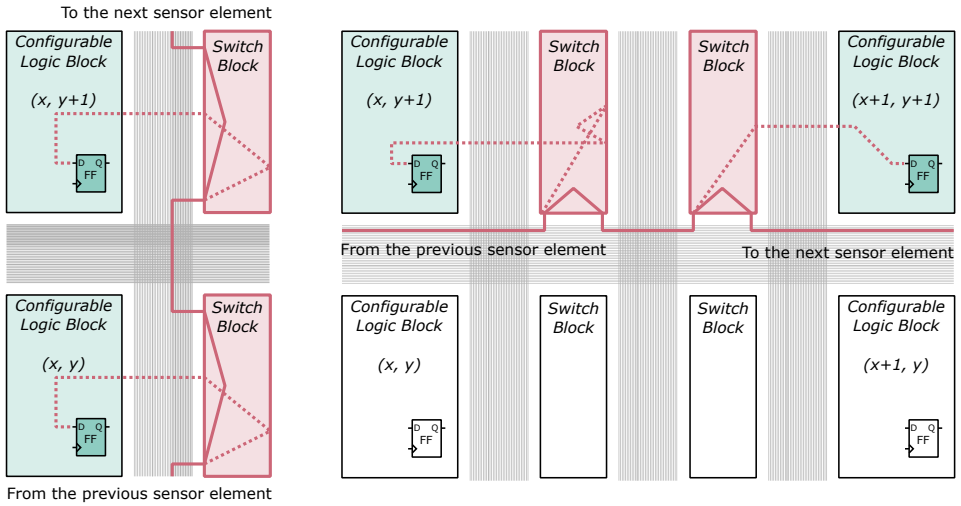



**Figure 2:** Routing delay sensor with a tapped delay line.




**(a)** VRDS **(b)** HRDS

**Figure 3:** Placement and routing of a segment of VRDS and HRDS sensors.

## 4.2 RDS

As the global interconnect wires are longer than the dedicated connections in the carry propagation chains, we expect both VRDS and HRDS to have lower sensitivity than the TDC and, hence, to be less effective for remote power side-channel attacks. When it comes to VITI, it is harder to predict how it compares to VRDS and HRDS without experimenting, as the sensitivity of VITI is considerably lower than the sensitivity of the TDC, because of the LUTs in the tapped delay line.

Is it even possible to build an RDS sensor with better sensitivity than TDC? The answer is *yes*. To explain how, let us again recall the expression for the sensitivity in Eq. (4). The first and the last term are unavoidable, as the connections to the inputs of the FFs in the output register have to exist. Ideally, the first and the third term could cancel each other out, if the exact same last-mile routing path is taken; this is the case in our realization of the VRDS and the HRDS. The middle term in Eq. (4), labeled as delay element, is more challenging to optimize. In TDCs, the carry propagation logic is fast, hence the high sensitivity of TDCs. In VITI, VRDS, and HRDS sensors, the middle term has a more significant impact on the sensitivity.

It is hard to imagine that the middle term in Eq. (4) can be better optimized than it is already with the dedicated carry propagation logic and routing. Therefore, we turn to the alternative expression for sensitivity in Eq. (2) and formulate a new sensor design goal.

**Goal:** *All the paths between $P_{IN}$ and $D_i$ should be routed so that the delays $d_i$, where*

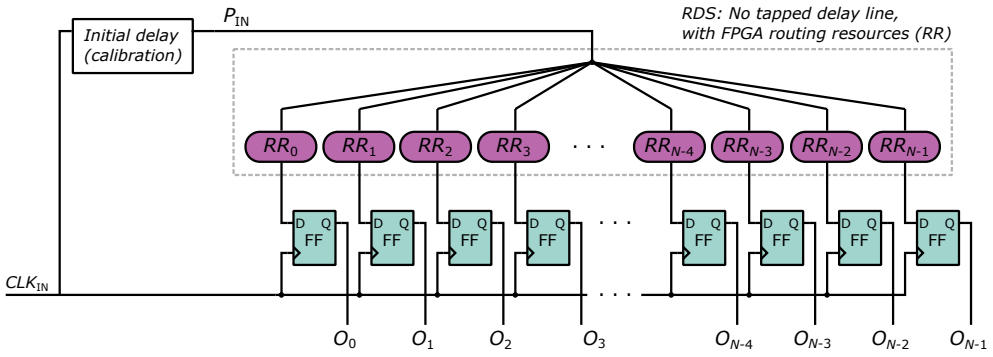



**Figure 4:** RDS sensor.

**Table 1:** Characteristics of the TDC, VITI, and our three variants of the RDS sensor.

| Sensor | TDC [ZSZF13] | VITI [UJS+21] | VRDS | HRDS | RDS |
|---|---|---|---|---|---|
| **Main component** | Carry chain | LUT | Routing resources (wires, multiplexers) | | |
| **Tapped delay line** | Yes | Yes | Yes | Yes | No |
| **Placement and routing** | Vertically constrained | Unconstrained | Vertically constrained | Horizontally constrained | Unconstrained |
| **Sensitivity** | High | Low | Low | Low | High |

$0 \leq i < N$, *are as similar as possible:*

$$d_0 \sim d_1 \sim d_2 \sim \cdots \sim d_{N-2} \sim d_{N-1}.$$

Ideally, the difference between any pair $(d_i, d_j)$, where $0 \leq i, j < N$, should be lower than the sensitivity of a TDC. In practice, the more $(d_i, d_j)$ pairs that satisfy the above goal, the more effective we expect the sensor to be, compared to the TDC, in an attack scenario.

The ideal sensor we formulate above, however, would not be usable in practice because the observable time window

$$W = \max |d_i - d_j|, \;\; 0 \leq i, j < N \tag{6}$$

would be significantly reduced compared to any delay-line-based sensor, where $W = d_{N-1} - d_0$. Such a sensor would not only be challenging to calibrate, but it would also be prone to becoming easily decalibrated, to the extent that even the calibration at run time may not be the solution. With this in mind, we rephrase our sensor design goal as follows.

**Goal:** *Many of the paths between $P_{IN}$ and $D_i$ should be routed so that the delays $d_i$, where $0 \leq i < N$, are as similar as possible. At the same time, the observable time window ($W$ in Eq. (6)) should not be too narrow.*

The approach we take to achieve the above goal is threefold: first, not having a tapped delay line at all; second, letting the FPGA router build the connections between $P_{IN}$ and $D_i$, $0 \leq i < N$; and, third, letting the FPGA placer decide on the locations of the FFs in the output register, in the interest of helping the router find suitable paths. In Fig. 4, we illustrate the resulting RDS design. Finally, it is worth noting that using Hamming weight to convert the binary value in the output register to the integer value of the sensor sample applies perfectly well also to the RDS sensor free of the tapped delay line.

In Table 1, we summarize the main characteristics of the TDC, VITI, and our three variants of the RDS sensor.

## 4.3 Calibration

The goal of the calibration is to ensure that an edge of the clock signal is within the observable delay line when its state is captured in the output register. One can choose either the rising or the falling edge and adjust the calibration accordingly; we opted for the rising edge. As the calibration is a lengthy process of trial and error, it is convenient to automate it. We implement a reconfigurable initial delay line approach proposed in literature [GNGT21]: implementing delay line elements as multiplexers allows a reconfigurable clock entry point in the initial delay line, thus changing the clock delay. Selecting a subset of the available coarse delay elements (in our implementation, LUTs) and fine delay elements (carry propagation logic) allows fine clock delay tuning required for voltage-drop sensors [GNGT21]. We control the calibration from software for better control and higher flexibility.

For the sensors with a tapped delay line (TDC, VITI, VRDS, and HRDS), the calibration procedure is straightforward: increasing (respectively, decreasing) the initial clock delay—an action that moves the rising edge closer to (respectively, further from) the beginning of the delay line (the point $P_{IN}$ in Figs. 1 and 2)—until the rising edge is close to the middle of the observable delay line [SGMT18a]. Considering the Hamming weight of the output register, the rising edge being close to the middle of the observable line translates to the sensor sample being equal to approximately $N/2$.

Given that the RDS does not have a tapped delay line but a tree of routing resources, it requires a somewhat different calibration approach. We can no longer tell where the edge precisely lands, but we can observe and influence the Hamming weight of the output register. We aim to calibrate the sensor so that the power side-channel trace (i.e., the time sequence of sensor samples for a given measurement duration) has a high variance. In other words, the more bits in the output register that toggle when a voltage drop happens, the better. Our RDS calibration approach is illustrated in Fig. 5 and described in Algorithm 1.

The algorithm has two parts that together ensure that many bits of the output register capture the changes in the propagation delay of the rising edge of the clock signal. In the first part (lines 2–13), we incrementally increase the initial delay by including more coarse elements ($IDC$) until the falling edge of the clock exits the observable delay line and all the output bits become '1': $HW(O) = N$. The effects of increasing the initial delay are illustrated in steps 1, 2, and 3 in Fig. 5. We stop including more coarse elements (line 10 of the algorithm) when all the samples in $N_{traces}$ measured traces reach the maximum value, $N$. In the second part of the algorithm (lines 14–28), we gradually increase the number of coarse and the fine ($IDF$) elements, to bring the rising edge carefully into the observable delay line. For every $(IDC, IDF)$ pair, we record $N_{traces}$ traces and compute the maximum value of all the sensor samples. When that maximum value crosses the predefined threshold $\delta$ ($delta < N$), as shown in step 4 of Fig. 5, the calibration is finished. Alternatively, we choose another $(IDC, IDF)$ pair and repeat the procedure.

The user-controlled threshold $\delta$ ($0 \leq \delta < N$) determines the number of output FFs that capture '0' instead of '1'. If $\delta$ is high, a few bits in the output register satisfy this condition: for them, the clock edge has passed to the left of the capture window (shown in the right half of Fig. 5). For the other bits, those equal to '1', the clock edge is to the right of the capture window. It is these bits that, when the on-chip voltage drops due to the victim activity and, consequently, the clock edge moves to the left of the capture window, may change from '1' to '0'. If it happens, this change will be reflected in the lower Hamming weight of the output register (i.e., the lower value of the sensor sample $s$ in Algorithm 1).

If $\delta$ is low, for many bits in the output register the clock edge will be to the left of the capture window. These bits are unlikely to change their value, as lowering the on-chip voltage moves the edge further away from the capture window. Therefore, low $\delta$ results in fewer bits potentially toggling and, hence, less side channel leakage getting captured. Therefore, we set $\delta$ to a value close to $N$, so that for as many output FFs as possible, the

clock edge is likely to enter the capture window during trace recording. If the calibration fails, then the width of the observable window $N$ needs to be increased, by adding more FFs in the sensor output register.

The described calibration algorithm is not limited to RDS; it can be used for VRDS, HRDS, and even TDC. For these three sensors, we expect the observable window to be wider than for RDS. Consequently, for the same supply voltage variations, fewer bits in their output registers should toggle. From the calibration perspective, fewer bits toggling means that a wider range of $\delta$ values should result in equally correct calibration.

**Algorithm 1** Calibration algorithm

**Input**: $L_{IDC}$, maximum number of coarse elements
**Input**: $L_{IDF}$, maximum number of fine elements
**Input**: $n$, number of samples per trace
**Input**: $N$, observable delay line length (maximum sensor value)
**Input**: $\delta$, calibration parameter
**Input**: $N_{traces}$, number of traces recorded at each calibration step
**Output**: $IDC$, $IDF$, number of coarse and fine elements, respectively

```
1: procedure CALIBRATE(L_IDC, L_IDF, n, N, δ)
2:     for IDC_cnt from 1 to L_IDC do
3:         s_min ← N
4:         IDC ← IDC_cnt; IDF ← 1
5:         SEND_CALIBRATION(IDC, IDF)
6:         for trace from 1 to N_traces do
7:             (s_1, s_2, ..., s_n) ← RECORD_TRACE()
8:             s_min ← MIN(s_min, MIN(s_1, s_2, ..., s_n))
9:         end for
10:        if s_min = N then
11:            break
12:        end if
13:    end for
14:    for IDC_cnt from IDC to L_IDC do
15:        for IDF_cnt from 1 to L_IDF do
16:            s_max ← N
17:            IDC ← IDC_cnt; IDF ← IDF_cnt
18:            SEND_CALIBRATION(IDC, IDF)
19:            for trace from 1 to N_traces do
20:                (s_1, s_2, ..., s_n) ← RECORD_TRACE()
21:                s_max ← MAX(s_max, MAX(s_1, s_2, ..., s_n))
22:            end for
23:            if s_max = δ then
24:                return IDC, IDF
25:            end if
26:        end for
27:    end for
28:    return failure
29: end procedure
```

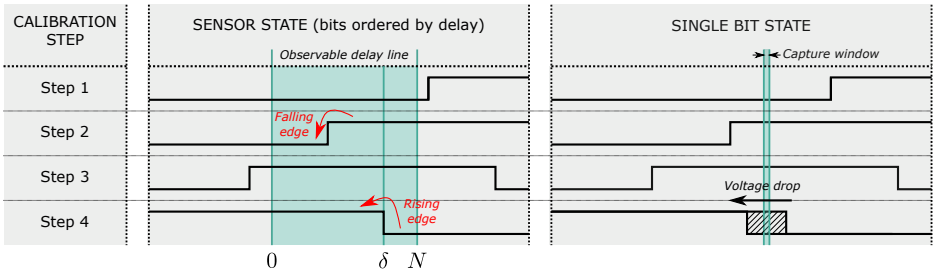



**Figure 5:** Calibration steps for the RDS sensor.

## 4.4 Portability and Detectability

Let us now discuss the challenges of portability and detectability of TDC and routing delay sensors.

As described in Section 2.2, TDC sensors employ carry chain logic and require strict placement constraints, ensuring that the carry chain is correctly and vertically formed and that each carry output drives the corresponding flip-flop residing in the same FPGA slice. These constraints ensure the tapped delay line is fine-grained, as uniform as possible, and that only dedicated connections are used. Depending on the FPGA device on which the sensor is to be deployed, the constraints may need to be adjusted to account for a different sensor location and type of carry logic available (e.g., CARRY4 or CARRY8). As described in Sections 4.1 and 4.2, unlike TDC, neither VRDS, HRDS, nor RDS, require carry-chain logic to sense voltage variations. In the case of HRDS and VRDS, the sensor location, the placement of the flip-flops, and additionally, the resources to route the clock signal through the tapped delay line need constraining. The sensor location aside, in RDS, neither the placement of the flip-flops nor the choice of routing resources needs to be specified by the adversary, which makes RDS easier to deploy and port across FPGA devices.

For the purpose of calibration, both TDC and the routing delay sensors require means to adjust the clock phase. We provide it using LUTs and carry-chain logic (the latter requiring placement constraints). However, if the ease of implementation and portability are of importance, then other approaches, such as a phase-locked loop (PLL) or an adjustable input-delay element [UJS+21], can be used instead. These alternative approaches are equally suitable for TDC as for the routing delay sensors.

Given the high interest in fault and power analysis attacks on cloud FPGAs, researchers have proposed bitstream checking tools [LMG+20, KGT19]. Their principle of operation is to, first, reverse engineer the bitstream into a design netlist and, second, search for potentially malicious patterns. A simple example of a malicious pattern is a combinational loop; some cloud FPGA providers are able to detect it during synthesis and flag it as a design error. This check effectively prevents LUT-based ring oscillators (either as sensors or power wasters) from being deployed on the cloud.

In the case of TDC sensors, bitstream checkers could look for a number of potential issues, starting with timing violations [KGT19], because the clock propagating through the tapped delay line has to violate timing by construction. RDS sensors, similarly to TDC sensors, introduce timing violations in the output register. These timing violations can be bypassed, equally efficiently for TDC and RDS, by using programmable clock-generating circuits (i.e., PLLs or mixed-mode clock managers). It suffices to set a sufficiently low clock frequency at compile time, not to violate the timing constraints, and then change it to the desired value during runtime.

A clock-to-data path [KGT19], inherent to both TDC and RDS, is another flag-raising netlist structure. Yet, if needed, it can be easily avoided; for example, by adding a T flip-flop on the clock path and using its output instead of the clock to drive the delay line.

Precisely constrained carry chains, when detected, can indicate the presence of a sensor. While the carry chains are the basic building blocks of TDCs, RDS can be entirely free of them and, therefore, pass the check.

Finally, connecting the clock signal to the flip-flops of the RDS sensor output register creates a relatively high fan-out signal, yet another feature checked by bitstream scanning tools in literature [LMG+20]. However, these tools look for orders of magnitude higher fan-out, common to power-wasting circuits. The fan-out of 128 (or less) in RDS is not uncommon in FPGA designs (e.g., for enable and reset signals of registers) and, as such, calls for no alarm [LMG+20].

# 5 Experimental Evaluation

This section presents the experimental evaluation methodology. We first describe the architecture of the system used to record power traces. Then, we provide a detailed description of the experiments and, finally, explain the attack metrics used to evaluate and compare the effectiveness of the sensors in the remote power side-channel attack setting.

## 5.1 System Architecture and Floorplan

Our experimental setup consists of three different FPGA boards, allowing us to evaluate the RDS sensor across various FPGA families. For most experiments, we use the side-channel evaluation board Sakura-X [Lab21], equipped with a AMD Kintex-7 FPGA (xc7k160tfbg676-1). Additionally, we perform experiments on AMD Alveo U200 datacenter accelerator card, having an AMD Virtex UltraScale+ FPGA (xcu200-fsgd2104-2-e). Finally, to evaluate the RDS sensor at various external temperatures, we use a smaller Digilent Basys 3 device with a lower-end AMD Artix-7 FPGA (xc7a35t). We use Vivado 18.03 for the Sakura-X and Basys 3 boards, and Vivado 2022.1 for the Alveo U200 design. For compilation, we use the default Vivado synthesis and implementation strategies.

Fig. 6 shows a block diagram of the system architecture used in all three FPGA boards. Despite the FPGA-specific implementation differences, the system architecture of all three setups has the same main components: The shell, responsible for the communication between the FPGA tenants (the adversary or the victim) and the host machine. The victim is an AES-128 hardware module [AU19]; it has an associated controller, for transferring plaintexts and ciphertexts, and for initiating encryption. The adversary, physically isolated from the victim, has a voltage-fluctuation sensor calibrated by a controller module, and a FIFO buffer where sensor samples are stored before they are offloaded to the host machine. To facilitate easy trace collection, we use the encryption initiation signal as a trigger for storing the sensor traces. For temperature-related experiments, we include the system monitor (XADC) to the shell to monitor the on-chip temperature.

Fig. 7 shows the floorplan of the system on the three FPGAs. In all implementations, the

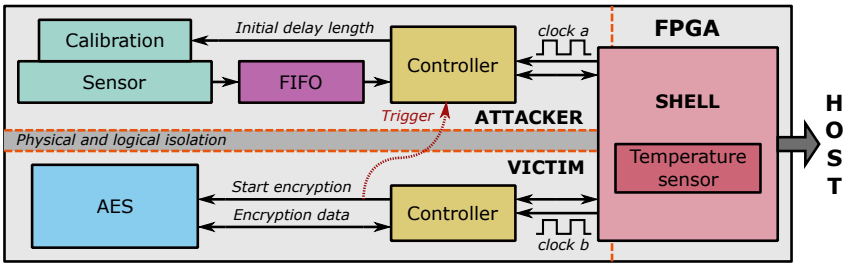
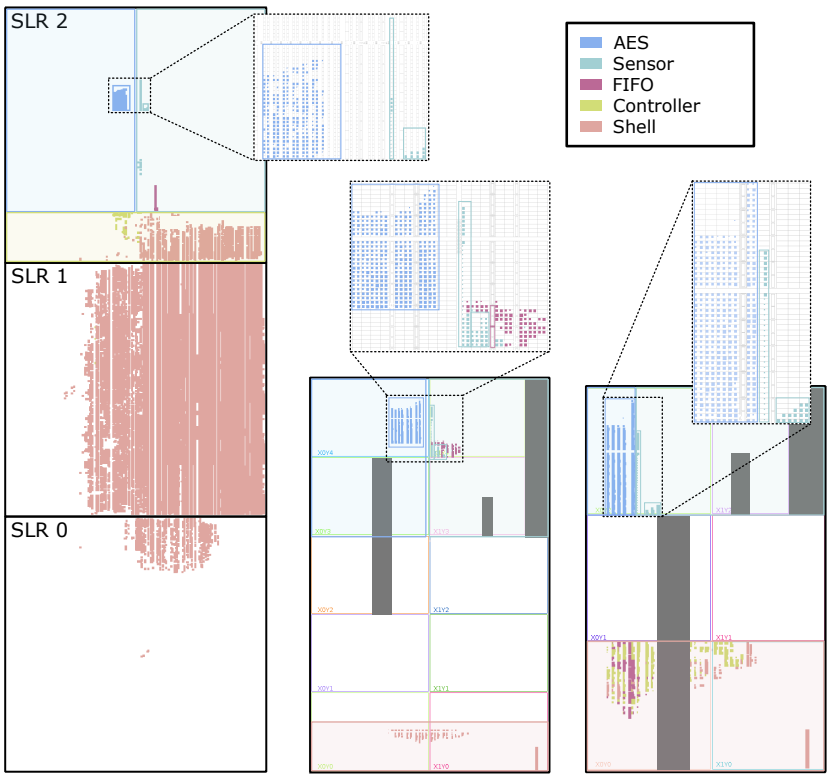



**Figure 6:** System architecture.




a) Alveo U200 b) Sakura-X c) Basys 3

**Figure 7:** Floorplan of all three boards used in the experimental evaluation.

attacker and victim reside in separate regions (Pblocks). The AES and the sensor are placed close to one another, simulating the worst-case scenario for the victim. Because of different FPGA/host communication constraints, the shell and the controller implementation differ for each FPGA. On Sakura-X, the shell occupies the smallest area, as the communication happens through a serial connection and an additional control FPGA [Lab21], which reduces the noise of the communication process. On Basys 3, communication is achieved via UART. On Alveo U200, the automatically inserted shell occupies the largest share of the FPGA: it is static and prevents using a custom, smaller shell. Users are constrained to use the AXI-4 shell interface, making the design and implementation of the controllers more complex.

The clock frequencies of the sensor and the AES are set to 200 MHz and 20 MHz, respectively. The exception are the experiments where the FPGA is subjected to variable external temperatures, where we increased the AES frequency to 50 MHz, to reduce the trace size and speed up the acquisition of a large number of traces.

## 5.2 Experimental Setup

In our experiments, we aim to break the secret key of an open-source AES-128 cryptographic module implementation [AU19], using correlation power analysis (CPA). We record $N_{TRACES}$ power side-channel traces per experiment (the exact value of $N_{TRACES}$ depends on the experimental platform). For each trace acquisition, we send the key $K$ and the plaintext $PT$ to the AES core and record all the sensor samples captured during the AES encryption. We use the current ciphertext as the next plaintext, to avoid plaintext

**Table 2:** Key and plaintext values used in the experimental evaluation.

| Key | Key value | Plaintexts |
|---|---|---|
| K1 | 0x7d266aecb153b4d5d6b171a58136605b | $PT_0 = 0$<br>$PT_{i+1} = CT_i$ |
| K2 | 0xe3fb107fa4aaeb7130f411d4c88dbf6c | |
| K3 | 0xa89e2fd6926dc2478402b717631d08ce | |
| K4 | 0xa3a03d60c06457dc65d8afd5815f629c | |
| K5 | 0xe1055ac2abadea4fc7fc6be1310448d9 | |

repetition. For simplicity, as seen in Fig. 6, we use the start encryption signal to trigger the collection of a sensor trace. In real attacks, this signal is absent, and the attacker resorts to trace alignment and automatic triggering techniques described in previous work [SGMT18a]. We repeat the experiments with five different keys to draw more substantial results. The keys and the plaintexts used in the experimental evaluation are listed in Table 2.

### 5.3 Attack Metric

Correlation power analysis was introduced by Brier et al. [BCO04]. In this statistical attack, the attacker correlates the measured power side-channel traces with a precomputed power model using Pearson's correlation coefficient. When there are more than a million power traces to analyze, the CPA attack becomes computationally expensive and can take hours or days to complete. In this work, we use the open-source code [HHR15, TNP+14] to run the CPA attack on a GPU and save on the computation time (the attack on $\sim 2$ million traces can be completed in a matter of hours).

Most side-channel attacks take a divide-and-conquer approach and independently attack individual bytes (sub-keys) of the 128-bit AES key. However, evaluating the security of the sub-keys does not necessarily illuminate the security of the entire key. In particular, when an attacker fails to break all the key bytes, the remaining effort for a full key recovery through trial and error is not quantified. For this reason, we opt for the key rank estimation metric, which uses the correlation computed with the CPA to estimate the remaining effort for an attacker. For example, if an attacker has no side-channel information, then the key rank equals to the entire key space, i.e., $2^{128}$ in the case of AES-128. Alternatively, when the entire key is broken, the key rank drops to zero. While there are several ways of computing the key rank, in this paper, we use the histogram-convolution-based algorithm of Glowacz et al. [CVR+15] where the key rank is upper and lower bounded.

## 6 Results and Discussion

In this section, we present the results of the experiments. We start by comparing the power side-channel traces measured with the TDC and the RDS. Then, we compare the RDS sensor with the TDC [GNGT21] by performing a power side-channel attack and a statistical analysis of the characteristics of the sensor traces. After evaluating the impact of the size of the RDS output register on the success of the attack, we examine how RDS behaves under different temperature conditions. Finally, we compare the RDS with the TDC across a range of different placements for the sensor and the AES.

Table 3 lists the FPGA resource utilization of our implementations of the TDC, VITI, and RDS sensors. As described in Section 4.3 and similarly to previous work [GNGT21], we use the coarse (LUTs, latches) and fine (carry) elements for calibration. With 32 LUTs and latches, and 24 fast carry elements for fine phase shift tuning, every sensor in Table 3 can be calibrated. Because of the high sensitivity of the TDC and the RDS, we set the output register size to 128 bits ($N = 128$). Unlike the TDC, VITI has a short observable

**Table 3:** Resource utilization of TDC, VITI, and RDS.

| Sensor | Initial delay (calibration) | | | Other resources | | |
|---|---|---|---|---|---|---|
| | LUT | CARRY | Latch | LUT | CARRY | FF |
| **TDC** | 32 | 0 | 32 | 0 | 32 | 128 |
| **VITI** | 32 | 0 | 32 | 4 | 0 | 4 |
| **VRDS** | 32 | 0 | 32 | 0 | 0 | 32 |
| **HRDS** | 32 | 0 | 32 | 0 | 0 | 16 |
| **RDS** | 32 | 24 | 32 | 0 | 0 | 128 |

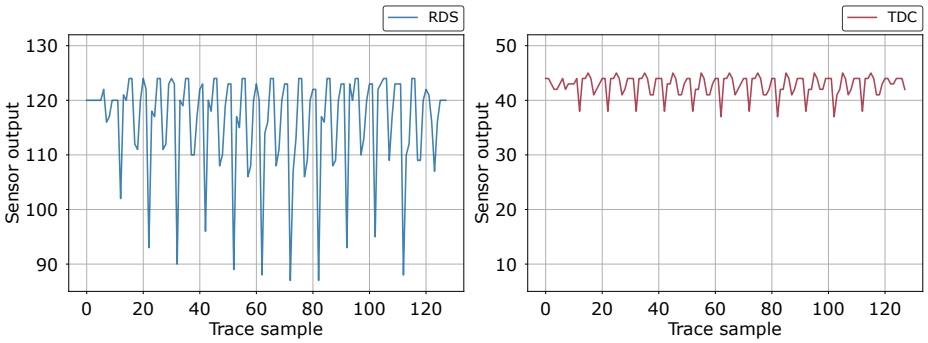



**Figure 8:** One side-channel trace recorded with the RDS (left) and the TDC (right) sensor.

delay line and uses only 4 LUTs and FFs [UJS+21]. For VRDS and HRDS, we set the output register size, $N$, to 32 and 16, respectively.

We constrain the RDS register to a Pblock having $\approx 2\times$ more flip-flops than required and let Vivado complete the placement and routing (P&R). We conjecture that if the assigned Pblock is not overly resource-limited, the P&R will not be a hard task and, hence, Vivado will place the flip-flops and route the signals in a close to optimal way.

## 6.1 RDS Sensors Versus TDC and VITI

As a first step in the experimental analysis, in Fig. 8 we visualize the waveforms of the RDS and the TDC traces recorded during the encryption of a single plaintext on Sakura-X. The traces are placed side by side for easier comparison. In both of them, the AES rounds are clearly visible. The traces have 128 samples, covering the entire duration of one AES-128 encryption. Because of two independent calibrations, the vertical offsets of the traces differ. What is more important to notice is that the peak-to-peak amplitude of the RDS trace is higher (37) than the peak-to-peak amplitude of the TDC trace (8). The higher RDS trace variation suggests that an attacker with the RDS may be able to break the secret key faster. To evaluate this hypothesis, we run additional experiments.

We record 100k traces (corresponding to the encryption of 100k plaintexts) for each of the five keys in Table 2, as described in Section 5.2. We run the experiments for the TDC, VITI, and the three variants of our RDS sensors. Fig. 9 visualizes the results of the statistical analysis of the sensor samples recorded for the key $K1$. On the left, we compare the number of output bits with nonzero variance. For VITI, VRDS, and HRDS, there are only one or two bits that toggle. In the case of TDC, there are 11 bits. Finally, the RDS has the highest number of bits toggling: 47. This result explains the waveforms in Fig. 8, as the higher peak-to-peak value is in direct relation with the number of bits toggling.

The right part of Fig. 9 depicts the variance of the output bits of the RDS and the

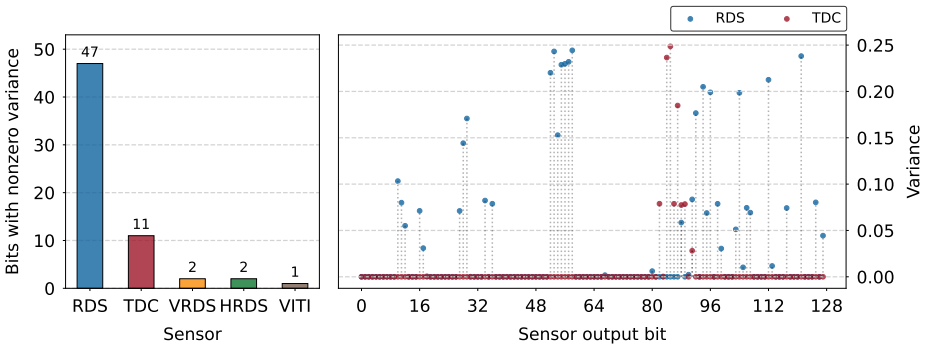



**Figure 9:** Number of bits toggling during trace acquisition for every sensor (left), and the variance of the bits for RDS and TDC sensors (right).

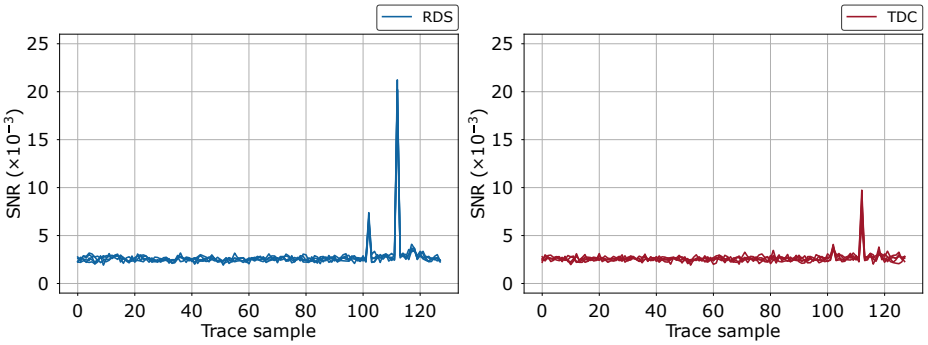



**Figure 10:** Signal-to-noise ratio for the RDS (left) and TDC (right), computed on the least-significant byte of the output of the ninth AES round.

TDC. As expected, the TDC has a cluster of bits with nonzero variance, where the rising clock edge lands in the delay line. This figure also highlights the difference between the RDS and the TDC: replacing the delay line with free routing in the RDS results in a higher number of bits toggling. Additionally, the toggling bits are not necessarily clustered closely together.

Let us now compare the signal-to-noise ratio (SNR) for the RDS and TDC. The SNR is a side-channel evaluation metric defined as the ratio between the useful signal, i.e., the variance of the data-dependent power consumption, and noise. It can be obtained from the power side-channel traces without performing an attack and is most commonly used to identify trace samples with significant leakage (i.e., samples that are commonly linked to the secret key). To compute the SNR, we follow the procedure outlined by Papagiannopoulos et al. [PGA+22]. Fig. 10 shows the results corresponding to the least-significant byte of the output of the ninth AES round, for the keys in Table 2. For both sensors, we can observe two peaks: in sample 102 (the beginning of the last AES round) and in sample 112 (the end of the last round, i.e., the moment when the ciphertext is saved in the state register). RDS is clearly superior to TDC, as the SNR approximately doubles in these two points of interest. Across all the experiments and every byte of the intermediate value, SNR in sample 112 for RDS is consistently higher than for TDC, by a factor of 1.57× on average, with a maximum of 2.87×.

To compare the sensors in the power side-channel attack scenario, we attack the traces using CPA and the key rank estimation metric, and repeat the experiment five times (each time with a different key). Fig. 11 shows the attack results. As outlined in Section 5.3,

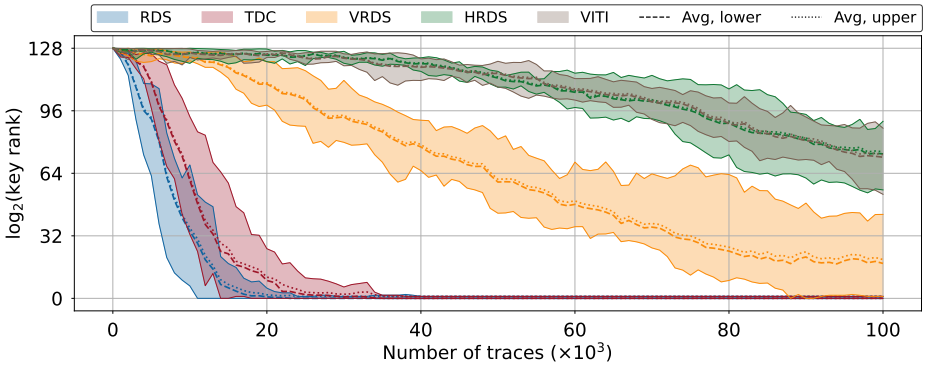



**Figure 11:** Key rank estimation for TDC, VITI, and our three RDS variants.

the key rank is estimated as a range. The dashed and dotted lines represent the lower and the upper bounds of this range, averaged over all the experiments. The shaded areas indicate the entire range of the key rank (min, max), observed across all the runs. The results demonstrate that the RDS and the TDC, as predicted, are superior. The coarse delay lines of VITI, HRDS, and VRDS result in lower sensor sensitivity, making it difficult for the sensor to capture small voltage fluctuations. HRDS and VITI give very similar results, suggesting that both the horizontal wires and the LUTs have a similar response to voltage fluctuations. VRDS, however, is superior. This result is not surprising, as vertical routing uses shorter wires than horizontal, thanks to the absence of heterogeneous blocks (DSPs, memory) within an FPGA column.

Finally, and most interestingly, Fig. 11 shows that, on average, the RDS sensor outperforms the TDC. As the RDS sensor has more output bits with nonzero variance and the RDS trace has higher peak-to-peak amplitude, an attack with the RDS requires fewer traces to recover the full key.

## 6.2 Impact of The RDS Size on The Attack Success

The number of FFs in the output register of the TDC does not impact its performance—provided that the sensor is correctly calibrated. This is not the case for the RDS. The left side of Fig. 13 shows the number of bits with nonzero variance for RDS with 128, 96, 64, and 32 FFs in the output register. The right side shows the variance of each bit in the output register, computed across 100k traces (corresponding to 100k AES-128 encryptions on Sakura-X, with the key K1). We see that reducing the output register size results in fewer bits that toggle.

Fig. 12 shows the results of the key rank analysis. In general, increasing the number of bits in the output register leads to fewer number of traces needed to break the full AES key. However, for 96 and 128 bits in the output register, there is no notable difference. These results correlate well with the per-bit variance shown in Fig. 13 and the intuition that the more bits with the nonzero variance, the more effective the attack.

## 6.3 RDS Versus TDC on The Alveo U200 Datacenter Card

To evaluate the RDS sensor on a cloud-scale FPGA, we deploy our system on the AMD Alveo U200 datacenter accelerator card (as explained in Section 5.1). Cloud-scale FPGAs are large in size, which makes it more difficult to sense switching activities on the shared PDN [GCRS20]. Therefore, we record 1.8 million traces, and repeat the trace collection for each of the keys in Table 2. Furthermore, we repeat all the experiments five times.

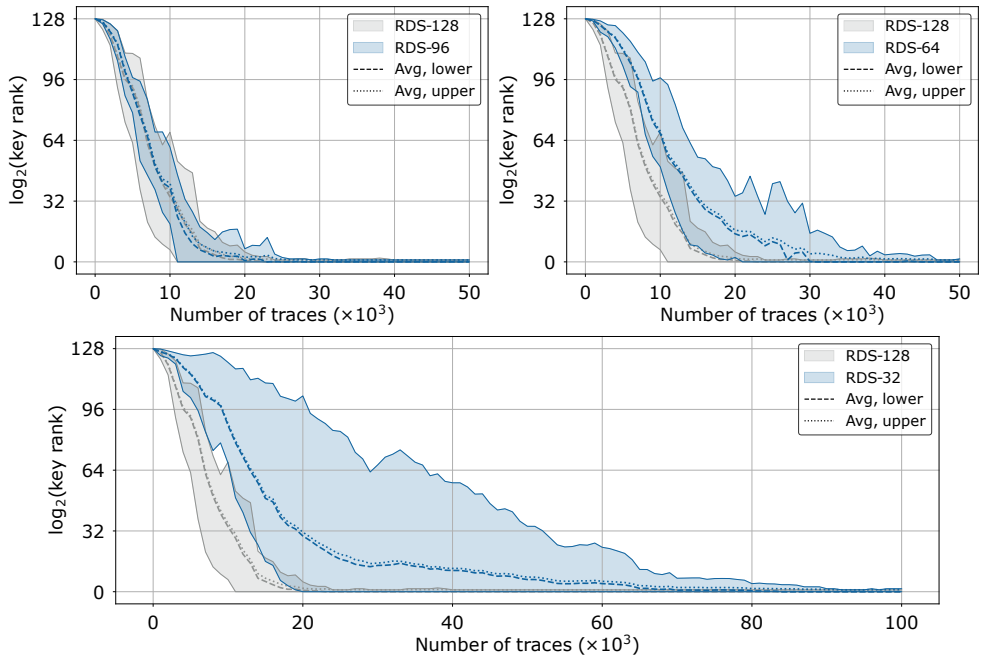



**Figure 12:** Comparison of RDS sensors with the number of bits used.

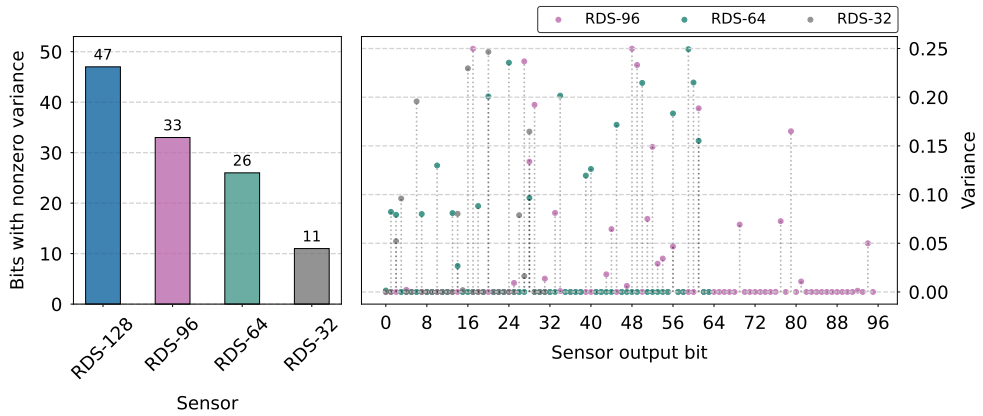



**Figure 13:** Left, the number of bits toggling during 100k trace acquisitions for the RDS with 128, 96, 63, and 32 bits in the output register. Right, the variance per bit.

The results are summarized in Fig. 14. Similarly to Fig. 11, the average of the lower and the upper bounds of the key rank are shown with dashed lines. The shaded areas correspond to the minimum and the maximum key rank values observed across all the attacks. Again, we see that the RDS outperforms the TDC, even more clearly than in the experiments discussed in Section 6.1. Therefore, a remote adversary equipped with the RDS instead of the TDC can steal a secret with fewer measurements.

## 6.4 Varying the Placement

To further evaluate and compare the RDS and the TDC, we record the power side-channel traces for a number of different sensor and AES placements. Fig. 15 provides a conceptual overview of the Sakura-X floorplan, with the sensor and AES positions marked in green

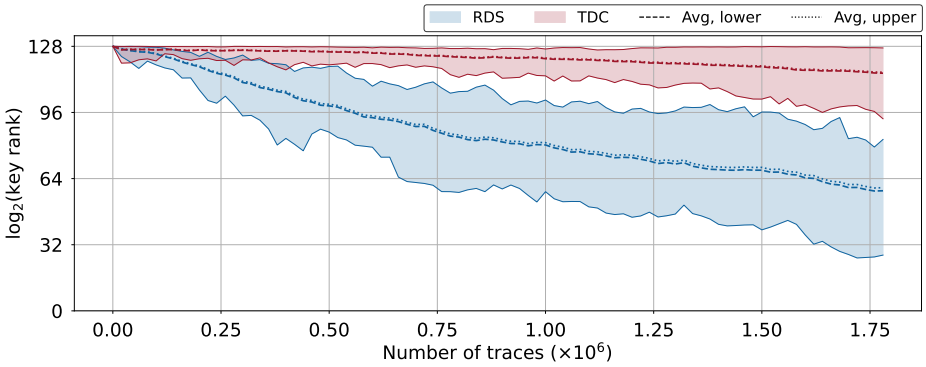



**Figure 14:** Key rank estimation for the TDC and RDS sensors on the Alveo U200 board.

and blue, respectively. The FPGA has ten clock regions, out of which the shell, which is always kept at a constant location, occupies the bottom two. In the remaining eight clock regions, we define eight sensor locations (in regions 2, 3, 6, and 7) and four AES locations (in regions 1, 4, 5, and 8). In all experiments, AES and the sensors are in separate regions, in line with the threat model. For a pair of sensor-AES placements, we record 100k traces, once for the key K1 and once for the key K4 in Table 2.

In the first set of experiments, we fix the AES to the location labeled as M1 and vary the sensor placement. In total, we collect the data from 32 experiments (eight sensor locations, two secret keys, and two sensors). Then, we compute the key rank metric and record the number of traces required for breaking all the bytes of the secret key (i.e., the number of traces for which the log(key rank) first time drops to zero). To compare the effectiveness of the RDS with the TDC, we normalize the results and visualize them in Fig. 16 (left part). We see that, in most cases, the RDS outperforms or performs equally well as the TDC. On average, the number of traces that the RDS requires to break the key is 60% of the number of traces that the TDC requires (in other words, the attack with the RDS needs 40% fewer traces than the attack with the TDC, to break the entire secret key). Finally, we observe that the number of traces when the log(key rank) drops to zero can vary across the experiments (irrespective of the chosen key). The same can be observed in Figs. 11 and 14: When the key rank drops to zero and the slope of the curve significantly reduces, likely due to the low SNR of many bits of the secret key, the range of the obtained results widens. Figs. 11 and 14 also show that the extent of this variability, for both TDC and RDS, reduces for the bits of the secret key which are less impacted by noise and, consequently, broken earlier.

In the second set of experiments, we fix the location of the sensors at M2 and vary the placement of the AES. The results are shown on the right side of Fig. 16. Again, in most cases, the RDS requires fewer or an equal number of traces to break the key. In this experiment, the average ratio is 0.76 (i.e., the RDS needs 24% fewer traces than the TDC to break the key, on average). If we take all the experiments into account, then we can say that an attack with the RDS requires 35% fewer traces than the attack with the TDC, for all the bytes of the secret key to be recovered.

Table 4 lists the obtained results: the first two rows contain data per sensor and location (averaged across the corresponding experiments with two different keys). The third row aggregates the results per region, both sensors considered. Looking at the last row of data, we can note the correlation between the number of traces to break the key and the locations of the sensor and the AES with respect to one another: When the AES is in the first region, and the sensors change places, the attack is fastest with sensors in the second, third, then sixth, and finally the seventh region; with respect to the AES, regions

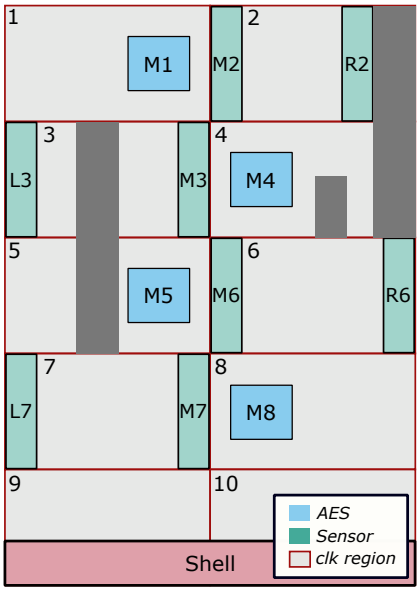



**Figure 15:** Floorplans for the chosen sensor and AES placements.

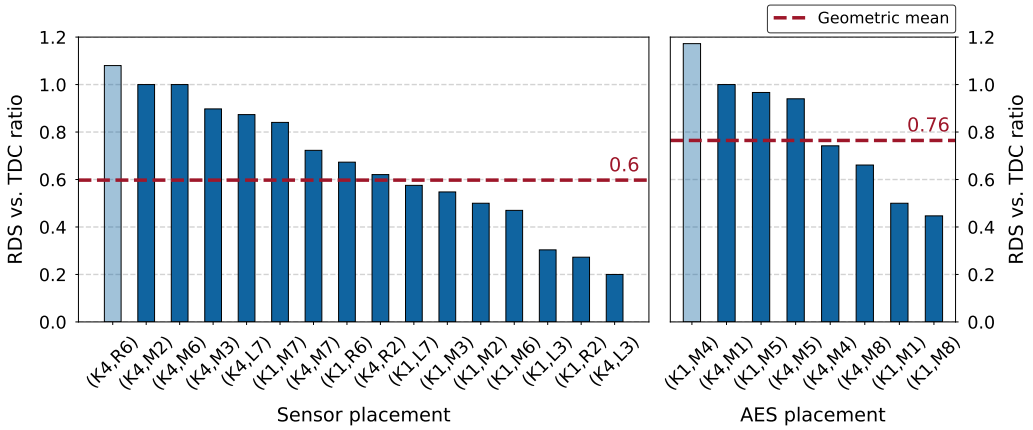



**Figure 16:** Ratio between the number of traces needed to break the key with the RDS and the TDC sensor when varying the placement of the sensor (left) or the AES (right). In dark blue are the results where an attack with RDS is at least as efficient as with TDC. The dashed red line corresponds to the geometric mean.

**Table 4:** Number of traces required to break the full AES 128-bit key, for different sensor placements (AES placed at M1) and different AES placements (sensor placed at M2). The last row shows the average number of traces to break the key per FPGA region, both RDS and TDC sensors considered.

| Sensor | Traces to break key ($\times 10^3$) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Sensor placement | | | | | | | | AES placement | | | |
| | M2 | R2 | L3 | M3 | M6 | R6 | L7 | M7 | M1 | M4 | M5 | M8 |
| RDS | 16 | 18 | 17 | 29 | 46 | 31 | 56 | 59 | 16 | 29 | 91 | 41 |
| TDC | 23 | 48 | 71 | 41 | 73 | 39 | 76 | 76 | 23 | 30 | 95 | 77 |
| Average | 26.3 | | 39.5 | | 47.3 | | 66.8 | | 19.5 | 29.5 | 93.0 | 59.0 |

two and three are close by, while six and seven are further away. When the sensors are in the second region, and the AES changes place, the attack is fastest when AES is in the first, then fourth, then eighth, and finally the fifth region; with respect to the sensors, the first and the fourth region are close by, while the fifth and the eighth regions are further away.

## 6.5 Temperature Impact

The results in the previous sections have demonstrated the effectiveness of the RDS sensor at room temperature. Here, we evaluate how the environment temperature and its variations during the trace acquisition affect the attack success. For these sets of experiments, we use a cost-efficient Digilent Basys 3 board and the design floorplan in Fig. 7. To measure the on-chip temperature, the temperature sensor (XADC) is used, as described in Section 5.1.

To evaluate the transient temperature effects on the RDS sensor, we record 70k traces using key K1 while keeping the calibration constant throughout the trace acquisition. First, we record the baseline results at a stable room temperature. Then, we turn to the following three temperature-varying scenarios: one temperature increase and two temperature drops. In all the experiments, we first record 9k traces at room temperature and only then start warming up or cooling the device.

Fig. 17 shows how the transient temperature changes impact the success of the attack. We can observe that the slight temperature drop of ∼17°C (green line) does not significantly impact the attack success when compared to the baseline. By contrast, cooling the FPGA by ∼47°C for 15k traces (yellow line) increases the attack effort to break the entire key by the additional 10k–15k traces. Finally, heating the FPGA by ∼37°C (red line) has an even more pronounced impact on the success of the attack. Even though the key is not fully broken, the key space after 70k traces is greatly reduced. These results are not surprising; similar observations have been reported for the delay-line-based sensors [UJS+21]. Importantly, despite the significant temperature changes, the RDS sensor remained calibrated, and the clock edge did not drift away from the observable time window.

In the final set of experiments, we evaluate the impact of different stable environment temperatures on the RDS sensor traces. To this end, we record the traces with the FPGA in a thermal chamber. We set the chamber temperatures to 40, 45, 50, 55, and 60°C. At each temperature, we acquire 70k traces using K1 while keeping the calibration constant and repeat the trace acquisition ten times. Fig. 18 shows the average upper and lower bounds of the key rank estimation metric. We see that, at higher environmental temperatures, the attack effort increases. However, the RDS remained calibrated under all the tested conditions and always resulted in a successful CPA attack.

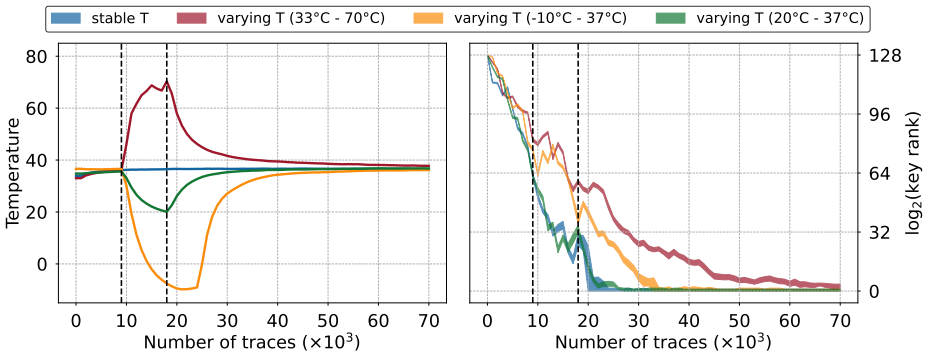



**Figure 17:** Key rank estimation for the RDS at different transient temperatures.

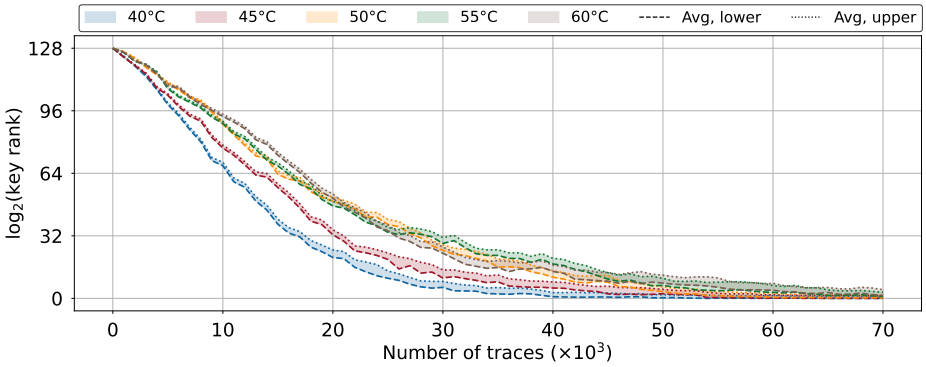



**Figure 18:** Key rank estimation for the RDS at different, but stable, environment temperatures.

# 7 Conclusions

This work presents a novel FPGA-based voltage sensor design, fundamentally different from TDC and RO sensors. Our new routing delay sensor leverages routing resources for sensing voltage variations. We present three variants of the new sensor: one vertically constrained (VRDS), one horizontally constrained (HRDS), and one free of any placement or routing constraints (RDS). We evaluate the performance of the RDS sensor using the correlation power analysis attack and the key rank estimation metric, in an attack against an AES-128 hardware cryptographic module. The results, computed for a number of different sensor and AES placements on the Sakura-X board, show that the RDS outperforms the TDC: on average, an attack with the RDS requires 35% fewer traces to break the secret key. The observation that the RDS is superior is confirmed with extensive experiments on the cloud-scale FPGA (the Alveo U200 datacenter card). Finally, we show that the RDS is not significantly impacted by the changes in environmental temperature. Future work will investigate the avenues for further improvements of the RDS sensor performance.

# 8 Acknowledgments

We thank the anonymous reviewers and our shepherd, Dr. Thorben Moos, for their valuable feedback. This work is partially supported by the Swiss National Science Foundation (grant No. 182428).

# References

[Ali] Alibaba. Compute optimized instance families with FPGAs. alibabacloud.com/help/doc-detail/108504.htm. Accessed: 2022-10-10.

[ASB20] Ibrahim Ahmed, Linda L. Shen, and Vaughn Betz. Optimizing FPGA logic circuitry for variable voltage supplies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(4):890–903, April 2020.

[AU19] AIST and Tohoku University. AES Encryption Core. http://www.aoki.ecei.tohoku.ac.jp/crypto/, 2019.

[AWS19] Amazon AWS. Amazon EC2 F1. https://aws.amazon.com/ec2/instance-types/f1/, 2019. Accessed: 2022-10-10.

[Azu] Microsoft Azure. Machine learning. https://azure.microsoft.com/en-us/pricing/details/machine-learning/. Accessed: 2022-10-10.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge*, pages 16–29, Camebridge, MA, USA, November 2004.

[CVR+15] Glowacz Cezary, Grosso Vincent, Poussier Romain, Schüth Joachim, and Standaert François-Xavier. Simpler and more efficient rank estimation for side-channel security assessment. In *International Workshop on Fast Software Encryption*, pages 117–29, Istanbul, Turkey, March 2015.

[GCRS20] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. Are cloud FPGAs really vulnerable to power analysis attacks? In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pages 1–4, Grenoble, France, March 2020.

[GDTLM19] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, and Philippe Loubet-Moundi. High-speed ring oscillator based sensors for remote side-channel attacks on FPGAs. In *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, page 1–8, Cancun, Mexico, December 2019. IEEE.

[GKT+20] Dennis R. E. Gnad, Jonas Krautter, Mehdi B. Tahoori, Falk Schellenberg, and Amir Moradi. Remote electrical-level security threats to multi-tenant FPGAs. *IEEE Design & Test*, 37(2):111–19, January 2020.

[GNGT21] Dennis R. E. Gnad, Cong Dang Khoa Nguyen, Syed Hashim Gillani, and Mehdi B. Tahoori. Voltage-based covert channels using FPGAs. *ACM Transactions on Design Automation of Electronic Systems*, 26(6):1–25, November 2021.

[GOT17] Dennis R. E. Gnad, Fabian Oboril, and Mehdi B. Tahoori. Voltage drop-based fault attacks on FPGAs using valid bitstreams. In *27th International Conference on Field-Programmable Logic and Applications*, pages 1–7, Ghent, Belgium, September 2017.

[GRE18] Ilias Giechaskiel, Kasper B. Rasmussen, and Ken Eguro. Leaky wires: Information leakage and covert communication between FPGA long wires. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 15–27, Songdo, Incheon, Republic of Korea, June 2018.

[GRS20] Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. C$^3$APSULe: Cross-FPGA covert-channel attacks through power supply unit leakage. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1728–41, May 2020.

[HHR15] Gamaarachchi Hasindu, Ganegoda Harsha, and Ragel Roshan. The A to Z of building a testbed for power analysis attacks. In *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*, pages 501–6, Peradeniya, Kandy, Sri Lanka, December 2015.

[Hua] Huawei. FPGA accelerated cloud server-huawei cloud. https://www.huaweicloud.com/en-us/product/fcs.html. Accessed: 2022-10-10.

[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*, pages 44–68, August 2018.

[KGT19] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 12(3):1–26, August 2019.

[Lab21] Satoh Lab. Sakura-X side-channel evaluation board. https://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-X.html, 2021. Accessed: 2021-9-20.

[LMG+20] Tuan Minh La, Kaspar Matas, Nikola Grunchevski, Khoa Dang Pham, and Dirk Koch. FPGADefender: Malicious self-oscillator scanning for Xilinx UltraScale + FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 13(3):1–31, September 2020.

[MDL+22] Shayan Moini, Aleksa Deric, Xiang Li, George Provelengios, Wayne Burleson, Russel Tessier, and Daniel Holcomb. Voltage sensor implementations for remote power attacks on FPGAs. *ACM Transactions on Reconfigurable Technology and Systems*, 16(1):1–21, December 2022.

[MLS+20] Shayan Moini, Xiang Li, Peter Stanwicks, George Provelengios, Wayne Burleson, Russell Tessier, and Daniel Holcomb. Understanding and comparing the capabilities of on-chip voltage sensors against remote power attacks on FPGAs. In *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 941–44, Springfield, MA, USA, August 2020.

[MS19] Seyedeh Sharareh Mirzargar and Mirjana Stojilović. Physical side-channel attacks and covert communication on FPGAs: A survey. In *29th International Conference on Field-Programmable Logic and Applications*, pages 202–10, Barcelona, Spain, September 2019.

[MTH+21] Shayan Moini, Shanquan Tian, Daniel Holcomb, Jakub Szefer, and Russell Tessier. Remote power side-channel attacks on BNN accelerators in FPGAs. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, Grenoble, France, February 2021.

[PGA+22] Kostas Papagiannopoulos, Ognjen Glamočanin, Melissa Azouaoui, Dorian Ros, Francesco Regazzoni, and Mirjana Stojilović. The side-channel metrics cheat sheet. *ACM Computing Surveys*, October 2022. Just Accepted.

[PRP+19] George Provelengios, Chethan Ramesh, Shivukumar B Patil, Ken Eguro, Russell Tessier, and Daniel Holcomb. Characterization of long wire data leakage in deep submicron FPGAs. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 292–97, Monterey, CA, USA, February 2019.

[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 *26th Symposium on Field-Programmable Custom Computing Machines*, pages 1–8, Boulder, CO, USA, May 2018.

[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 the Design, Automation and Test in Europe Conference and Exhibition*, pages 1111–16, Dresden, Germany, 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. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 114:1–114:7, New York, NY, USA, November 2018.

[SGS23] David Spielmann, Ognjen Glamočanin, and Mirjana Stojilović. RDS: FPGA routing delay sensors for effective remote power analysis attacks. Artifacts (v1.0). https://doi.org/10.5281/zenodo.7534002, January 2023. Accessed: 2023-1-13.

[TNP+14] Swamy Tushar, Shah Neel, Luo Pei, Fei Yunsi, and Kaeli David. Scalable and efficient implementation of correlation power analysis using graphics processing units (GPUs). In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy*, pages 1–8, New York, NY, USA, June 2014.

[UJS+21] Brian Udugama, Darshana Jayasinghe, Hassaan Saadat, Aleksandar Ignjatovic, and Sri Parameswaran. VITI: A tiny self-calibrating sensor for power-variation measurement in FPGAs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(1):657–78, November 2021.

[ZS18] Mark Zhao and G. Edward Suh. FPGA-based remote power side-channel attacks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 805–20, San Francisco, CA, USA, May 2018.

[ZSZF13] Kenneth M. Zick, Meeta Srivastav, Wei Zhang, and Matthew French. Sensing nanosecond-scale voltage attacks and natural transients in FPGAs. In *Proceedings of the 21st ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 101–4, Monterey, CA, USA, February 2013.