Auto-Aligned Remote Power Analysis through Ring Oscillator-based Sensors

Lilian Bossuet, Anis Fellah-Touta, Carlos Andres Lara-Nino

Auto-Aligned Remote Power Analysis through Ring Oscillator-based Sensors

Lilian BOSSUET, Anis FELLAH-TOUTA, and Carlos Andres LARA-NINO
Univ Lyon, UJM-Saint-Etienne, CNRS, Laboratoire Hubert Curien UMR 5516,
F-42023, SAINT-ETIENNE 42000, France.
carlos.lara@univ-st-etienne.fr

Abstract

In recent years, the field of side-channel analysis has observed a revolution in the design of the attack methodology. Conventional approaches which require the use of highly specialized equipment like oscilloscopes and spectrum analyzers, despite highly-precise, might be regarded as impractical in some scenarios. On the other hand, the use of less-accurate internal sensors which can monitor the power footprint of a circuit has risen in popularity. In particular, delay sensors such as those based in Time-to-Digital converters and Ring-Oscillators have shown promising results. These structures are interesting since they can be implemented from regular hardware resources available in most circuits. This means that components already available in the target might be leveraged to implement a side-channel attack. Moreover, it is not really necessary to have direct access to the platform to carry out such an attack; which implies that if there is a remote link such as Ethernet, an adversary might be able to perform Remote Power Analysis (RPA) of the system. So far, the main challenge for the success of this kind of attacks is cutting and aligning the power traces. This is usually achieved through secondary digital channels which carry some trigger information. In this paper, we propose to use a single channel to encode both the power trace and the alignment information. This is achieved by exploiting architectural vulnerabilities of the platform. Our results demonstrate, for the very first time, that RPA traces can be auto-aligned. As a case study we attempt to perform RPA on a serialized implementation of Photon-Beetle, a finalist in the NIST lightweight cryptography standardization process.

Keywords: Remote Power Analysis, Ring Oscillator Sensors, Trace Alignment, Photon-Beetle.

Please cite as:

```
@InProceedings{BFL22,
title = {{Auto-Aligned Remote Power Analysis through Ring Oscillator-based Sensors}},
author = {Bossuet, Lilian and Fellah-Touta, Anis and Lara-Nino, Carlos Andres},
booktitle = {XVII Reuni{\'o}n espa{\~n}ola sobre criptolog{\'i}a y seguridad de la
informaci{\'o}n.  RECSI 2022},
pages = {37--42},
year = {2022},
publisher = {Ed.  Universidad de Cantabria},
address = {Santander, Spain},
doi = {10.22429/Euc2022.028},
isbn = {978-8-419-02414-5}}
```

1 Introduction

Most of the modern applications of cryptography are considered secure from an algorithmic point of view. It is understood that a formal or statistical proof of security warranties that an adversary cannot compromise the security of the system under reasonable assumptions. However, if the attack model supposes that the attacker has physical access to the platform these notions of security decrease [Sta10]. Under such scenarios, it is necessary to design and implement protections which can mitigate the information that can be retrieved from the hardware platform [MKP12].

A side channel is, in a simplified way, a physical magnitude which can be measured and correlated with the operations being performed in the platform. Electromagnetic emanation, power dissipation, thermal irradiation, energy consumption, and noise are prime suspects for information leakage. Then, we can envision side channel attacks as the application of a sensing strategy with a subsequent information-processing step. For the first part, it is necessary to possess a sensor which can transform the physical magnitude into a digital signal which can be read from a computing system. Such system will then process the samples in a given time. With the adequate quality and volume of data it might be possible to break just any cryptographic implementation. Our work focuses on the first of the two aforementioned tasks, we investigate the process for obtaining these data.

According to Nyquist's theorem [PM06], the first requirement for the appropriate acquisition of samples is to use a sampling frequency at least two times greater than that of the magnitude under analysis. This is the so called sampling theorem. Secondly, the sensor has to perform a quantization step which will encode the sample into a finite array of possible values, generally through binary representation. The more bits we use, the greater the sampling resolution will be, and subsequently we will obtain a greater fidelity to represent the signal of interest. However, more bits and more samples also imply that we must store and transmit more data. In the end, the goal of a sampling scheme is to adequately represent the physical magnitude with the minimal storage and bandwidth costs.

These are not trivial challenges when the targets of the sensing are digital circuits. Most of the time, the operational frequency of these designs is in the order of megahertz (MHz). Furthermore, the signals to be sampled generate transceiver outputs in the order of microvolts (μV). Consequently, a sampling scheme for a digital system must produce a few millions of samples per second with the adequate resolution to represent fluctuations of the millionth part of a volt! Thankfully, the quantization step depends not so much on the scale of the physical magnitude to be measured, but rather on the dynamic range of the signal. Usually less than 20 bits are sufficient to represent such measurements.

It should be evident that performing side channel analysis of a chip requires specialized equipment. Multi-channel digital oscilloscopes and high-frequency spectrum analyzers are some of the most popular tools for this task. Yet, their monetary costs are so high that only private companies and large laboratories can acquire these systems. Then, even if the adversary has the resources to acquire the sensing hardware (which would be the case in state-sponsored attacks), they must also gain physical access to the target platform. These two limitations could be sufficient for a designer to declare that their implementation

is totally safe from side channel attacks. Nonetheless, recent works [Gra+20] have demonstrated that neither expensive equipment or physical access to the chip are necessary to perform power analysis on a digital circuit.

Using internal sensors created from digital components is not a particularly novel idea [Fra+10; Hen10]. These constructions have been used to monitor the operation of the circuit and assess whether everything is working as intended. However, until recently it was not considered viable that such sensors could be used to retrieve sensitive information from the platform. It is now known that Time-to-Digital converters (TDC) [Gra+21] and Ring-Oscillator [Gra+19] based sensors (ROS) can be exploited to perform power analysis with moderate accuracy. Moreover, since these circuits can be implemented and operated remotely, performing Remote Power Analysis (RPA) on a chip is just a matter of exploiting some software vulnerability. It is no longer required to assume that the attacker has physical access to the device under attack.

Despite the evident advantages, RPA must still cope with the original sensing problems: obtain more samples with more resolution, while reducing storage and transmission costs. The latter is a particularly interesting problem, since a remote origin of the attack also means that the sampled data must travel back to this origin. Ubiquitous connectivity might provide viability for this approach. The Internet-of-Things (IoT) supposes that everyday objects will be connected to the internet. If the attacker finds a way to access the circuit remotely, then they can leverage hardware components already in the platform [Gra+21] to mount an attack and possibly compromise the security of the system.

A problem so far not addressed in the literature is that RPA, just as classical power analysis, requires some additional information to cut and align the traces. When the attacker has access to the platform we assume that they can poke around until they find some *trigger* signal which can be used to determine the start of a *trace*. However, in a remote-attack model this is not so trivial. We cannot assume that the *start* and *done* signals of the architecture under attack will be connected to our sensor. Therefore, the problem of remotely determining the point for cutting and aligning the traces of significant relevance.

In this work, we propose that the same sensors that are used to perform RPA [Gra+19] can also be used to encode the necessary information for the alignment of the traces. Thus, our attack model considers that the adversary must only retrieve a train of samples from the internal sensor to perform RPA. We achieve this feat by leveraging strategies previously used for covert-channel communications [BB18]. To demonstrate the viability of this approach we propose as case study the RPA of a serialized implementation of Photon-Beetle [Bao+21], a finalist in the NIST lightweight cryptography standardization process.

Our findings suggests that the proposed approach is viable as we managed to align the power traces with statistical validity. However, there are multiple limitations that must be addressed to reach a point where such attack becomes a practical concern, for example for an IoT platform. In particular, the problem of obtaining and transmitting large volumes of data is critical for carrying out a successful attack on the target circuit. Nevertheless, this is a characteristic of remote-attack which falls out of the scope of this work.

The rest of the paper is structured as follows. In Section 2 we describe our methodology and the materials used in our experimentation. In this section we also provide a formal description of the proposed attack scenario. The derived results are subsequently reported in Section 3. Finally, our findings and

conclusions are summarized in Section 4.

2 Materials and Methods

In this Section we provide details regarding our experimental setup. We describe the different components of our system and outline the guidelines for the proposed attack.

2.1 Remote sensors

A delay sensor is a circuit capable of measuring the variation in the delay of an oscillatory digital signal. The propagation delay of an oscillator through digital components will fluctuate as a function of the temperature and voltage of the circuit. Therefore, as the chip performs different processing tasks, these will influence the delay propagation of the target oscillator. This delay will be quantified and then sampled as a digital signal. Both TDCs and ROs can be used to implement delay sensors. In this work, we use the ROS from [Gra+19] to perform the data acquisition. Figure 1 illustrates the general architecture of this circuit.

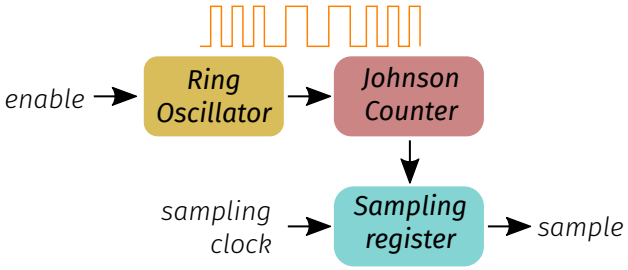


Figure 1: A delay sensor based on a ring oscillator

The main advantage of the ROS from [Gra+19] is their acquisition rate. When implemented in the reconfigurable fabric, these sensors allow to use sampling frequencies up to 250 MHz in the Zynq-7000 family of SoC-FPGAs, and greater in newer technologies. This is achieved thanks to the use of a Johnson counter which mitigates the need for carry propagation found in generic binary counters. The sampling register is 8-bits wide and captures the difference between counter states through a combinatorial function. These values are subsequently accumulated into a 32-bit registers which contains the output of the sensor matrix. The dynamic range of this output depends on the sampling frequency, normally less than 10 bits are used.

Under normal operation, a sampling frequency is chosen to clock the sampling register and *read* the resulting values. However, this oscillator can be modified and (for some frequencies) the result will be samples with a different offset but that still convey the change in delay propagation derived from the internal operation of the circuit. We employ this oscillator to produce a frequency-encoding which is then used to align the traces.

2.2 Covert channels

Frequency-based covert channels have been explored in [BB18] with the goal of bypassing Trust-Zone protections in a Zynq-7000 SoC-FPGA. The authors demonstrated that it was possible to modify the output of Phase Locked Loops (PLL) in the circuit through different modulation strategies in order to encode a message. This would create a covert channel between different components that were not supposed to communicate with each other. For example, a trusted application would exchange information with an untrusted hardware accelerator.

FPGA-enabled SoCs such as the Zynq-7000 boards feature different clocks which flow from the processing system into the programmable logic. These are sourced from a group of main PLLs and through a set of multipliers and dividers produce the desired frequency output. These multipliers and dividers are simply digital values stored in registers which can be modified from the processors of the SoC. In our work, we employ a Zynq Ultrascale+ SoC-FPGA as implementation platform. For these systems the clock modulation can be performed in a similar way as in previous generations of the technology, see Fig. 2.

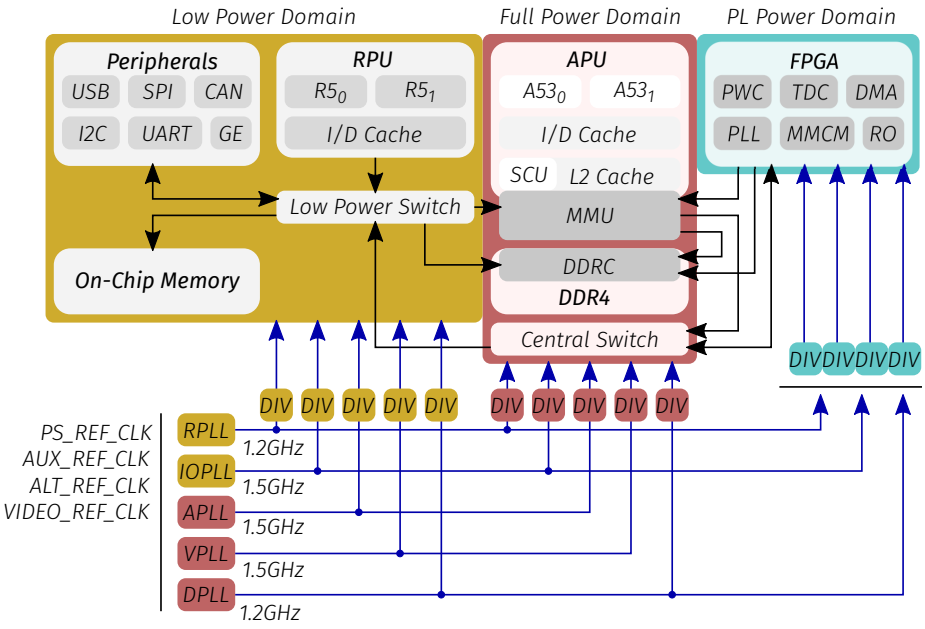
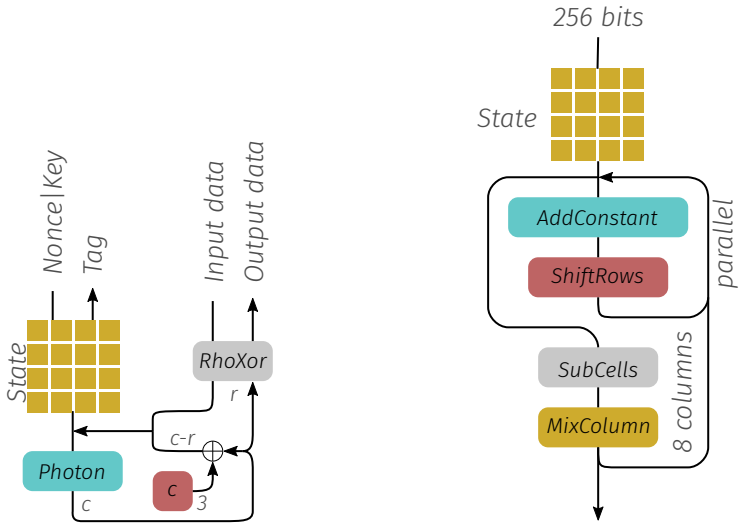


Figure 2: The clock tree of Zynq Ultrascale+ SoC-FPGAs

However, we do not encode a complex message in the covert channel. We use frequency modulation to modify the sampling clock of the ROS and produce a discernible pattern in the resulting sample train. As the circuit under attack is called from the processor, the malicious application modifies the target clock with each call. This pattern will encode the necessary information to perform the alignment of the traces.

2.3 Target architecture

As a case study we attempt to perform RPA on a serialized implementation of Photon-Beetle [Bao+21]. The algorithm under attack is illustrated in Fig. 3. The main operations of this authenticated cipher are similar to those of AES. In the underlying permutation (P_{256}), the state, arranged as a matrix of eight rows and eight 4-bit columns, is first XOR-ed with some round constants (*AddConstant*), then substituted (*SubCells*), shifted (*ShiftRows)* and finally mixed (*MixColumn*). These operations are repeated over 12 rounds. Photon-Beetle uses P_{256} to process each block of the message. The rate for the architecture under study is 32-bits.



(a) Photon-Beetle (b) Photon (P_{256})

Figure 3: The specification for Photon-Beetle

In the architecture under attack, the application of the *ShiftRows* and *SubCells* operations have been swapped in order to merge *AddConstant* and *ShiftRows* into *AddShift* and *SubCells* and *MixColumn* into *SubMix*. This change in the order of operations is applied in order to serialize *SubMix* in a single step. Given the characteristics of the *SubCells* transformation and the granularity of *ShiftRows*, it does not have any effect in the output of the algorithm or its resilience against power analysis.

The *AddShift* operation is performed in a single cycle and the *SubMix* operation is serialized over eight cycles. The additional operation in Photon-Beetle is the mathematical component ρ which is an XOR and permutation (*RhoXor*) performed before every call to the underlying permutation.

Every round of P_{256} takes 9 cycles, since there are 12 of such rounds, the latency of this permutation is 108 cycles. Counting the application of *RhoXor*, a total of 109 cycles are required per call. Then a total of $109 \times (\lceil a \rceil/32 + \lceil m \rceil/32 + 1)$ cycles are spent to process an m-bits message with a-bits of associated data. Our experiments take 1,853 cycles for a 256-bits message with 256-bits of associated data.

2.4 Attack scenario

Our attack scenario consists of two main actors, see Fig. 4. These are two C-language applications running on bare metal. First, we have an application ($A0$) on the ARM CortexA53-0 processor which can query the hardware architecture under attack. This actor can also modify the frequency of the FPGA clocks. The second party is another application ($A1$) on the ARM CortexA53-1 which can query the ROS.

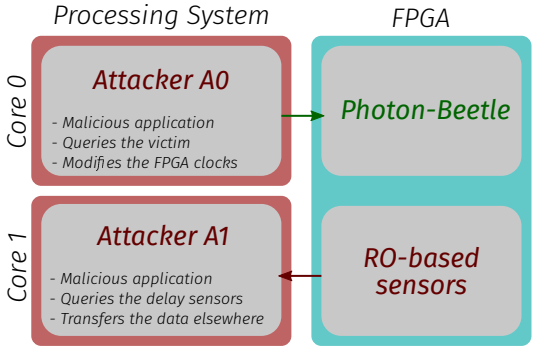


Figure 4: The proposed attack scenario

The proposed scenario assumes that $A0$ has the necessary access level to modify the control registers of the SoC. We assume also that $A0$ can query the hardware accelerator, which holds the cryptographic secrets but only replies to legitimate requests. For $A1$ the assumptions are more usual. We have an application which comes with a custom hardware acceleration and will perform some given task. Except that the accelerator contains a bank of ROS and $A1$ can retrieve the samples and transfer them to the remote origin. The exact mechanism for transferred the data is not addressed for brevity.

3 Experimental results

We used a TE0802 development board for our experimentation. This platform features a Zynq Ultrascale+ SoC-FPGA (xczu2cg-sbva484-1-e). We used the AMD-Xilinx 2022.1 toolchain, creating the hardware specification in Vivado and programming and launching the applications through Vitis.

3.1 PLL's transition delay

We first studied the PLL response times to assess whether it was viable to modify the frequency of these components from the processing system. Figure 5 shows the results for this experiment. We first enabled a digital trigger (TRIGGER) from the processor and then requested a frequency change from 100 MHz to 150 MHz. We sampled the PLL response, as well as the MSB in the output of the sensor which would indicate that the new frequency had been detected. We estimated that it takes approximately $400ns$ for the PLL to start the process to modify its output. Then $200ns$ are used to perform the requested change; during

this time the output of the PLL is unstable. Finally, it takes a few additional nanoseconds for the processor to be notified of the change.

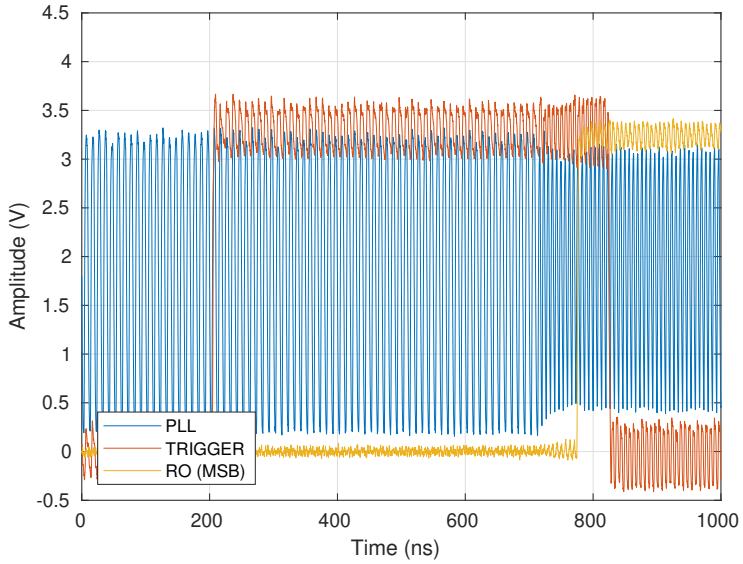


Figure 5: The step response of a PLL in the Zynq Ultrascale+ SoC-FPGAs. Obtained with a digital oscilloscope at 10 GSps.

These findings suggest that it is possible to employ frequency modulation to align the traces. Since the PLL calls are blocking, it is not necessary to account for the transition delay in the acquisition. Nonetheless, if the latency of the architecture under attack is much smaller than $600ns$, the traces will contain mostly spurious data.

3.2 Acquisition rate

As illustrated in Fig. 4, we assumed that the *A1* application could retrieve the samples from the ROS. These two modules were connected through an AXI-HP channel clocked at 300 MHz; a typical link in Zynq Ultrascale+ chips.

Recall from Subsection 2.3 that the underlying operations of Photon-Beetle include *RhoXor* (one cycle), *AddShift* (one cycle), and *SubMix* (eight cycles). To determine the acquisition rate of the processor we sampled some digital triggers that correspond with the processing of these operations. The details for this experiment are shown in Fig. 6.

By clocking the architecture under attack at 1 MHz we could retrieve five samples per cycle. These results indicated that the processor could retrieve 32 bits of data with a delay of 200 ns. Since we only included one sampler per transaction, our sampling rate was approximately 5 MSps.

3.3 The covert channel

We used the algorithm in Fig. 7 to encode the synchronization information into the covert channel. Before each call to the cipher, the *A0* application

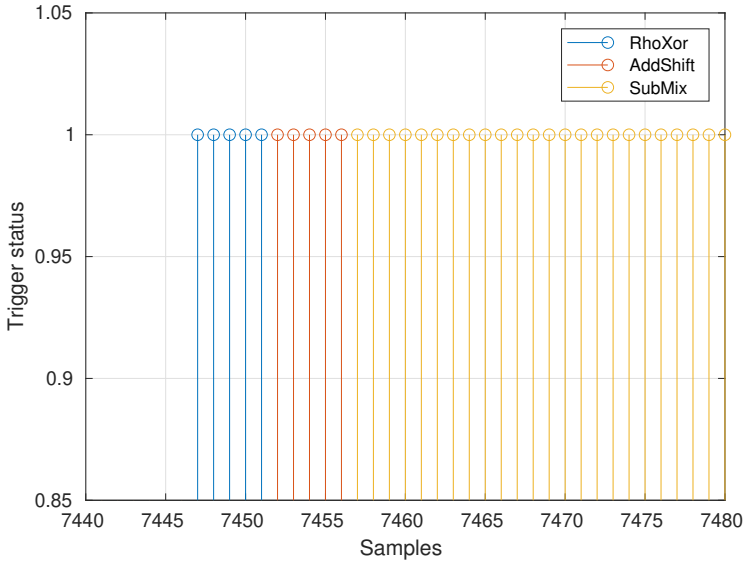


Figure 6: The sampling rate of the processor on the digital triggers of the Photon-Beetle architecture

modified the sampling frequency of the ROS. This value would iterate between two predefined values. For simplicity, *A0* would perform this task in an infinite loop.

Require: f_1, f_2 a pair of sampling frequencies
  $f_{RO} = f_1$
  while TRUE **do**
    $f_{RO} \leftarrow f_{RO} = f_1 ? \quad f_2 : f_1$
    `PhotonBeetle`(ENCRYPT)
  end while

Figure 7: The channel modulation strategy

Figure 8 illustrates the results from this modulation strategy. In this graph we plot a series of 50,000 samples retrieved by the *A1* application from the ROS. On first sight it was possible to clearly differentiate between the iterative stages of the operation of the chip. We could identify fragments of the channel which had a mean of ∼140 counts and others with a mean of ∼60 counts. Nonetheless, we could also note that there was a significant overlap between the sample windows. These outliers could be mitigated with the application of a moving average filter over 16 samples. This filer was only used for segmenting the channel into traces, however. Thanks to the evident offset in the windows a simple threshold-detection method could be used for this task.

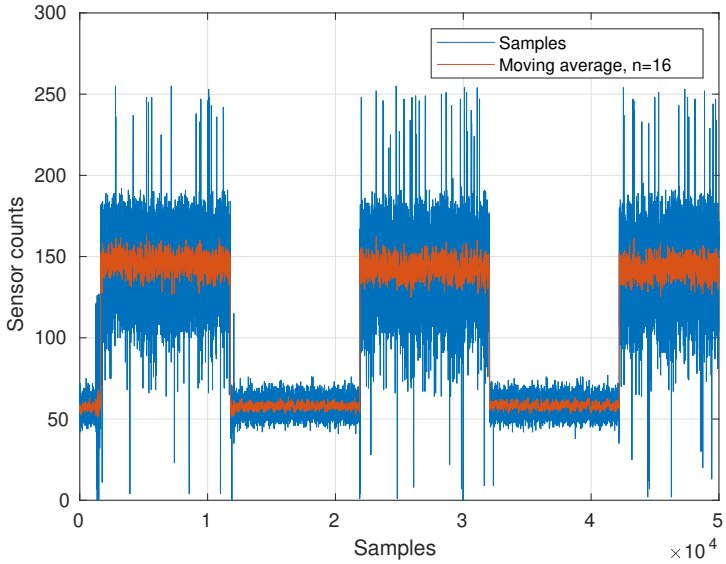


Figure 8: A sequence of samples when the sampling frequency of the ROS is modified from the application

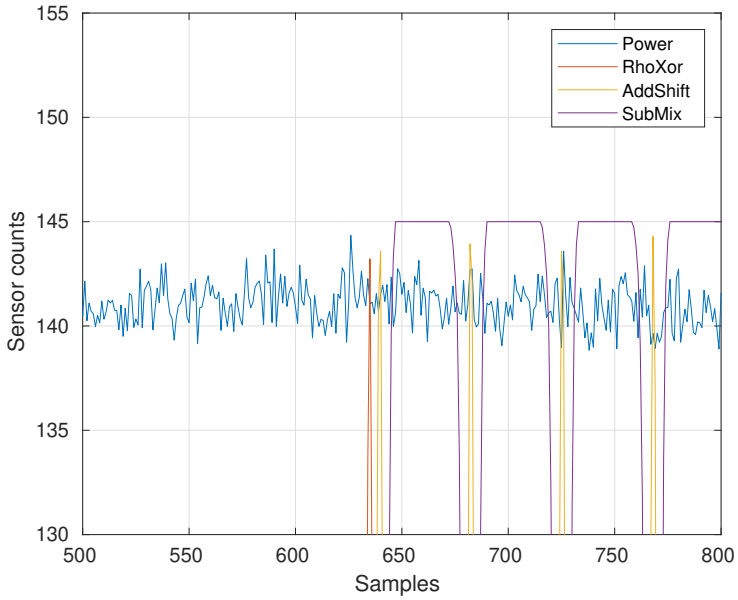
3.4 Two sets of traces

A result of modifying the sampling rate of the ROS is that we obtained two sets of samples which were fundamentally different. Yet, thanks to the sub-sampling process performed by the processor, in the end we obtained traces of the same length. Since cryptanalysis methods like differential power analysis do not depend on the magnitude or period of the samples, it would be possible to process the traces as a single set. However, to be rigorous about their study we processed both sets separately. In the following, we use the notation $traces_H$ to identify the set with $\bar{x} \approx 140$ and $traces_L$ to identify the set with $\bar{x} \approx 60$.

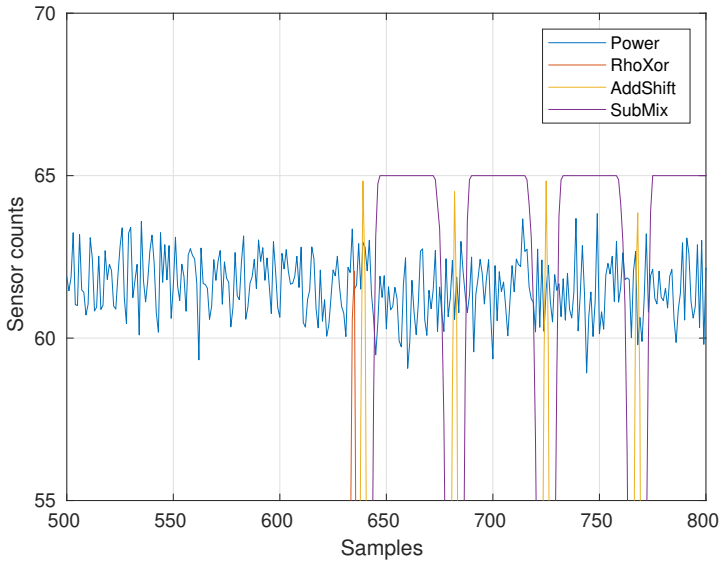
3.5 Validation and analysis

To determine whether the alignment was successful we sampled the digital triggers shown in Fig. 6 and attempted to perform the automatic alignment of these traces. In the same samples, we included the output of the ROS and also performed their automatic alignment. By automatic alignment we mean that we only had to cut the sample into segments and classify the traces into the corresponding set. Then we could obtain the average value for each set. Recall that the processor could retrieve 32-bit samples from the FPGA, but the output of the sensor was under 10 bits; thus we could retrieve the triggers from the exact sampling intervals by appending them to each sample.

As shown in Fig. 9, the triggers were easily identifiable after the automatic alignment, even those with only five samples (*RhoXor*, *AddShift*). However, as it was also evident, the proposed alignment method was not perfect. Nonetheless, as the number of observations is increased, the similarity between the sets of auto-aligned traces and a *golden* set of aligned traces should grow. To measure



(a) Automatic alignment of $traces_H$



(b) Automatic alignment of $traces_L$

Figure 9: Visual results for the automatic alignment, the graphs shown represent the average of 500 traces

this similarity we used the digital triggers to perform the alignment and create two golden sets of traces ($golden_H$ corresponding to $traces_H$, and $golden_L$ corresponding to $traces_L$). Then we evaluated their similarity to our resulting

sets. First we obtained simple statistics such as their mean ($\bar{x}$) and standard deviation (σ_x). These results re provided in Table 1. Despite the similarity in the results, these values are not really meaningful for unstructured signals.

Metrics	$\bar{x}$	σ_x
$traces_H$	140.8311	0.0071
$golden_H$	140.8315	0.0071
$traces_L$	61.6604	0.0150
$golden_L$	61.6600	0.0151

Table 1: Basic statistical analysis

Next, we used cross-correlation to determine whether the alignment proposed was optimal or not. For each one of the traces in the auto-aligned sets we computed their cross-correlation with the respective golden-aligned trace. For this kind of experiment we expected our results to be close to zero, as that would imply that the traces were aligned properly. Our preliminary results showed that the traces in the $traces_H$ set were better aligned than those in $traces_L$. This suggested that the transition from a lower to a higher sampling frequency was more consistent than the inverse operation. Therefore, we simply aligned all the traces in $traces_L$ to the rising edge.

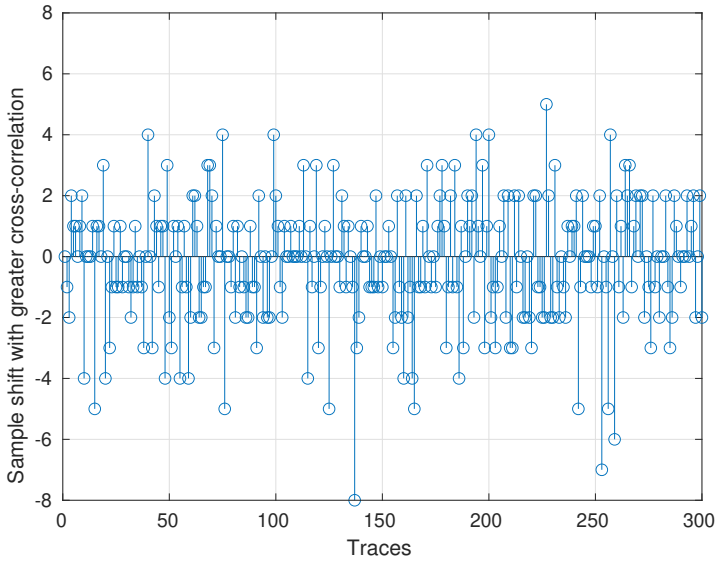
The results of this analysis, illustrated in Fig. 10, indicate that a 81% of the traces in $traces_H$ were well-aligned with only a miss-alignment of two samples or less. The 73% of the traces in $traces_L$ satisfied the same condition. Bear in mind that five samples represent a single cycle of the architecture under attack, and that our case study has a processing latency of 1,853 cycles. Therefore, if we are more lenient and accept a miss-alignment of under a cycle of this architecture (five samples or less) we obtain that over 99% of the traces in both sets are properly aligned.

4 Conclusions and Future work

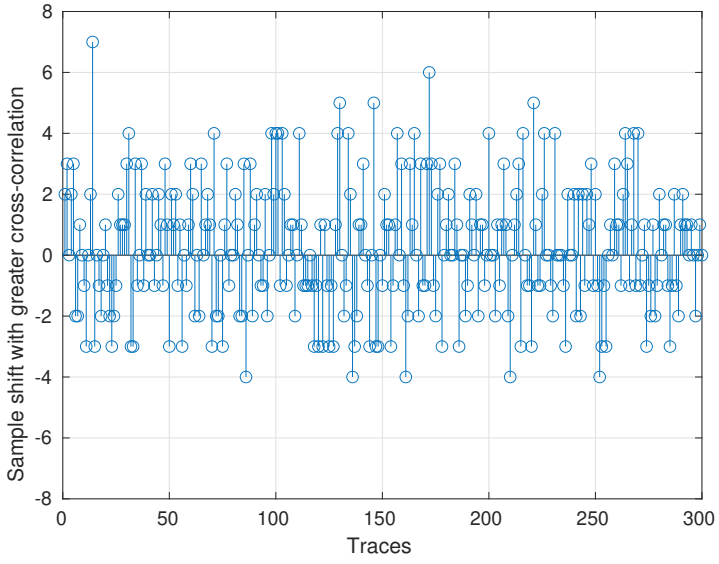
In this paper we have described a new strategy for the automatic alignment of traces in the scope of RPA attacks. Our findings suggest that the proposed method is viable under certain assumptions. For example, the processing latency of the architecture under attack should not be smaller than the transition delay of the PLL. As case study we attempted to perform RPA on the authenticated cipher Photon-Beetle. At this point, the limitations of RPA made it difficult to conduct a power analysis attack, nonetheless the proposed alignment method can brings us closer to this end. See Appendix.

Acknowledgements

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-19-CE39-0008 (project ARCHI-SEC).



(a) Between $traces_H$ and $golden_H$



(b) Between $traces_L$ and $golden_L$

Figure 10: Cross-correlation of the auto-aligned traces

References

[Bao+21] DZhenzhen Bao et al. *PHOTON-Beetle Authenticated Encryption and Hash Family*. NIST Lightweight Cryptography – Finalists. National Institute of Standards and Technology, 2021.

[BB18] El Mehdi Benhani and Lilian Bossuet. “DVFS as a Security Failure of TrustZone-enabled Heterogeneous SoC”. In: *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2018, pp. 489–492.

[Fra+10] John J. León Franco, Eduardo Boemo, Encarnación Castillo and Luis Parrilla. “Ring oscillators as thermal sensors in FPGAs: Experiments in low voltage”. In: *2010 VI Southern Programmable Logic Conference (SPL)*. 2010, pp. 133–137.

[Gra+19] 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: *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. 2019, pp. 1–8.

[Gra+20] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, Philippe Loubet Moundi and Francis Olivier. “Remote Side-Channel Attacks on Heterogeneous SoC”. In: *Smart Card Research and Advanced Applications*. Ed. by Sonia Belaïd and Tim Güneysu. Cham: Springer International Publishing, 2020, pp. 109–125. ISBN: 978-3-030-42068-0.

[Gra+21] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia and Philippe Loubet Moundi. “SideLine: How Delay-Lines (May) Leak Secrets from Your SoC”. In: *Constructive Side-Channel Analysis and Secure Design*. Ed. by Shivam Bhasin and Fabrizio De Santis. Cham: Springer International Publishing, 2021, pp. 3–30. ISBN: 978-3-030-89915-8.

[Hen10] Stephan Henzler. “Time-to-Digital Converter Basics”. In: *Time-to-Digital Converters*. Dordrecht: Springer Netherlands, 2010, pp. 5–18. ISBN: 978-90-481-8628-0.

[MKP12] Amir Moradi, Markus Kasper and Christof Paar. “Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures”. In: *Topics in Cryptology – CT-RSA 2012*. Ed. by Orr Dunkelman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–18. ISBN: 978-3-642-27954-6.

[PM06] John G. Proakis and Dimitris K. Manolakis. *Digital Signal Processing (4th Edition)*. USA: Prentice-Hall, Inc., 2006. ISBN: 0131873741.

[Sta10] François-Xavier Standaert. “Introduction to Side-Channel Attacks”. In: *Secure Integrated Circuits and Systems*. Ed. by Ingrid M.R. Verbauwhede. Boston, MA: Springer US, 2010, pp. 27–42. ISBN: 978-0-387-71829-3.

A Power Analysis of Photon-Beetle

With roughly 4,000 traces we can distinguish some patters which are assumed to be correlated with the operation of the circuit under analysis. This is illustrated in Fig. 11.

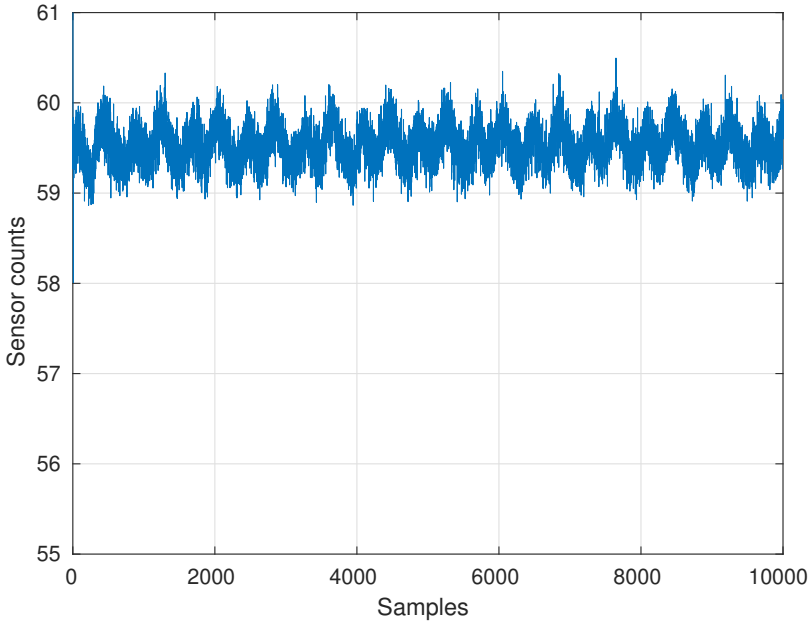


Figure 11: The average of ∼4,000 traces obtained remotely