



Hardware Vulnerability and Leakage Detection (first version)

PROJECT	
Project Number	101120962
Project Acronym	RESCALE
Project Title	Revolutionised Enhanced Supply Chain Automation with Limited Threats Exposure
Start Date	01.10.2023
Programme	HORIZON-CL3-2022-CS-01-02
DELIVERABLE	
Deliverable Type	Report
Workpackage	WP3
Deliverable Lead	ISI
Editors	Apostolos Fournaris (ISI), Charis Dimopoulos (ISI), Ioannis Morianos (DIE)
Contributors	Evangelos Haleplidis (ISI), Fenia Paraskevopoulou (ISI)
Dissemination Level	Public

Abstract

In this deliverable we summarize the activities of Task 3.3 till M14 of the RESCALE project focusing on the design and initial implementation of the Dynamic Hardware Analyzer component of the RESCALE architecture. This component, as Task 3.3 indicates, provides side channel leakage assessment of some cryptography hardware or software Intellectual Property (IP) block. The designed and developed platform is adopting the latest research directives on Side Channel Attack (SCA) assessment and takes also into account remote SCAs on cloud based Field Programmable Gate Array (FPGA) units that accommodate multiple tenants. The deliverable initially provides an overview of the SotA on the SCA research domain and then analyzes the Dynamic Hardware Analyzer architecture and its components. Furthermore, in the deliverable we map open issues and future activities on hardware vulnerabilities and leakage detection that will be incorporated in the Dynamic Hardware Analyzer in the future (till the end of T3.3) and describe a brief road-map on the next activities to achieve this goal while in parallel also providing some initial results.

Disclaimer

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author’s view – the European Commission is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.



This project has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No 101120962

Internal Reviewers

1. Narges Yousefnezhad (BNR)
2. Christiano Giuffrida (VUA)

Revisions

Version	Date	Partner	Overview
0.1	09/07/2024	ISI	ToC
0.2	10/11/2024	ISI, DIE	First draft
0.3	30/11/2024	ISI, DIE	Finished draft
0.4	15/12/2024	ISI	Reviewer comments collected
0.5	19/12/2024	ISI, DIE	Comments addressed
0.8	23/12/2024	ISI	Final version
0.9	23/12/2024	ISI, AEGIS	Final comments
1.0	30/12/2024	ISI	Final version

Table of Contents

1	Introduction	7
1.1	Scope & Contribution	7
1.2	Relation to Work Packages, Deliverables, and Activities	7
1.3	Contribution to WP3 and Project Objectives	7
1.4	Document Structure	8
2	Overview of Leakage Assessment and Side Channel Attacks	9
2.1	Definition and Importance	9
2.2	State of the Art	9
2.3	Limitations and Challenges	10
3	Dynamic Hardware Analyzer Architecture and Components	12
3.1	Overall Architecture	12
3.2	Trace Collector	13
3.3	Trace Analyzer	17
4	Open Issues	25
4.1	Identified Challenges	25
4.2	Mitigation Solutions	26
5	Roadmap to Final Version	28
5.1	Development Plan	28
5.2	Initial Evaluation	28
6	Main Innovations & Conclusion	35
6.1	Main Innovations	35
6.2	Conclusion	35
6.3	Future Work	36

List of Figures

1	Dynamic Hardware Analyzer Architecture.	13
2	Hardware sensors.	17
3	SCA Trace Analyzer flow.	19
4	Jupyter Notebook input for the SCA Trace Analyzer.	20
5	Configuration of the attack for the SCA Trace Analyzer.	29
6	Trace preprocessing example for the SCA Trace Analyzer.	30
7	Trace preprocessing example for the SCA Trace Analyzer.	31
8	Launch Attack Jupyter cell for the SCA Trace Analyzer.	31
9	SCA assessment report generation example.	32
10	SCA in multi-tenant FPGA block diagram.	33
11	AES leakage plot.	34
12	Pearson's correlation plot.	34

List of Abbreviations

ADC	Analog-to-Digital Converter.	10
AES	Advanced Encryption Standard.	4, 10, 32–34
API	Application Programming Interface.	14–16
CNN	Convolutional Neural Networks.	9
COTS	Commercial off-the-shelf.	12, 14, 15
CPA	Correlation Power Analysis.	9, 10, 12, 18, 23, 28, 32, 33
CPU	Central Processing Unit.	36
CVE	Common Vulnerabilities and Exposures.	25
CWE	Common Weakness Enumeration.	25
DL	Deep Learning.	9, 10, 12, 14, 18, 19, 28, 35
DL-LA	Deep Learning Leakage Assessment.	9
DPA	Differential Power Analysis.	9, 12, 23
DSCG	Dynamic Supply Chain component Guarantee.	7, 28, 36
DuT	Device under Test.	12, 18, 22
FPGA	Field Programmable Gate Array.	1, 4, 9–16, 22, 25, 26, 28, 29, 32, 33, 35, 36
IP	Intellectual Property.	1, 12, 13, 15, 18, 20, 28, 31
JSON	JavaScript Object Notation.	20–22, 29
LUT	Look-Up-Table.	22
ML	Machine Learning.	9, 12, 14, 18, 21–23, 26, 28, 29, 31
MLP	Multilayer Perceptrons.	9
PDN	Power Distribution Networks.	10, 11, 35
PSCA	Power Side Channel Attack.	11, 25, 36
PVT	Process Voltage Temperature.	16
RDS	Routing Delay Sensors.	10
RF	Random Forest.	18, 22, 31
RO	Ring Oscillator.	10, 16, 26

SCA Side Channel Attack. 1, 4, 7–10, 12–22, 25–36

SoC System on Chip. 9, 34

SotA State of the Art. 1, 8, 9

SPA Simple Power Analysis. 9, 12

SVM Support Vector Machine. 18, 22

TBOM Trusted Bill of Materials. 7, 36

TDC Time to Digital Converter. 10, 16, 17, 33

TVLA Test Vector Leakage Assessment. 12, 18, 22

VHDL VHSIC Hardware Description Language. 15

1 Introduction

This deliverable provides the results of Task 3.3 which is complimentary to the dynamic testing module developed in Task 3.2, focusing on detecting Hardware Vulnerabilities and specifically performing Leakage Detection on Side Channel Attacks (SCAs). Note that Task 3.3 is specifically concerned with *physical* side-channel attacks (or simply SCAs hereafter), that is attacks that exploit the physical characteristics (e.g., power consumption) of a target system. In contrast, *digital* side-channel attacks, which exploit digital characteristics (e.g., microarchitectural events) of a target system, are the focus of Task 3.4.

1.1 Scope & Contribution

This document is the outcome of the initial phase of Task 3.3, "Hardware Vulnerability and Leakage Detection". It outlines the initial design of solutions for security testing of software or hardware IP cores against information leakage due to physical characteristics of the utilized hardware execution environment. The main scope of the solutions described in this deliverable targets physical leakage vulnerabilities, especially Leakage Detection that can be exploited using SCAs. As part of the overall RESCALE platform, these solutions contribute as a standalone hardware attack surface analysis and provide reports that will be embedded in the DSCG portion of the Trusted Bill of Materials (TBOM).

1.2 Relation to Work Packages, Deliverables, and Activities

Deliverable D3.5 is the outcome of Task 3.3 within Work Package 3 (WP3). The results from D3.5 will be incorporated into the DSCG report generated by Task 3.2 (WP3), which contributes to the generation of the TBOM in WP4. In addition the tools and methods outlined in D3.5 will be integrated with the rest of the tools from Tasks 3.2 and 3.4, associated with the dynamic testing module, and included into the broader RESCALE solution in WP5.

1.3 Contribution to WP3 and Project Objectives

D3.5 directly contributes to WP3, specifically to Objective O3.4, which focuses on implementing hardware vulnerabilities and information leakage detectors, by developing solutions to detect Leakage that can be exploited using SCAs.

In addition D3.5 contributes directly to Project Objectives O1 and O2 by providing Hardware vulnerabilities to be incorporated into the auditing toolbox. Finally, D3.5 also contributes indirectly to Project Objectives O3 (by providing essential details to be incorporated into a TBOM) and O5 (by providing details for visibility and awareness of hardware components in the supply chain).

1.4 Document Structure

The deliverable structure is outlined in this section as follows:

- **Section 2 - Overview of Leakage Assessment and Side Channel Attacks:** This section provides background and context, summarizing the findings of D2.3 and presenting additional relevant State of the Art (SotA) research on Leakage Assessment and SCAs.
- **Section 3 - Dynamic Hardware Analyzer Architecture and Components:** This section offers an overview of the architecture of the Dynamic Hardware Analyzer with all the internal components, as well as implementation details and interfaces.
- **Section 4 - Open Issues:** This section discusses the identified challenges of the current work along with possible mitigation solutions.
- **Section 5 - Roadmap to Final Version:** This section provides a development plan on how the work of Task 3.3 as well as the evaluation of the tools are going to move forward.
- **Section 6 - Main Innovations & Conclusion:** This section summarizes the key innovations from the work and outlines the next steps.

2 Overview of Leakage Assessment and Side Channel Attacks

2.1 Definition and Importance

2.2 State of the Art

This section provides a comprehensive overview of the SotA for Hardware Vulnerability and Leakage Detection provided on D2.6. It explores advanced SCAs, security assessment platforms, and the emerging challenges posed by multi-tenant Field Programmable Gate Array (FPGA)s. An FPGA is a type of integrated circuit that can be configured by the user after manufacturing to perform specific digital logic tasks. This is achieved through an array of programmable interconnected logic blocks that can be reprogrammed to implement custom hardware functions through dedicated hardware IP Cores. Multi-tenant FPGAs are shared among multiple users or workloads, often in a cloud or data center environment. This approach allows multiple independent applications or tenants to utilize portions of the FPGA's resources simultaneously, optimizing utilization and cost-efficiency.

Side Channel Attack (SCA)

SCAs exploit physical characteristics (e.g., power consumption, electromagnetic emissions) to extract sensitive information from systems like SoCs and FPGAs. These attacks can be categorized into two primary methods:

- **Non-Profiling Methods:** Statistical techniques like Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Correlation Power Analysis (CPA).
- **Profiling Methods:** Techniques using prior device access, including Template Attacks [1][12] and Machine Learning (ML)-based approaches.

Deep Learning (DL) has significantly improved SCAs by enhancing attack performance and bypassing countermeasures. Models like Multilayer Perceptrons (MLP) and Convolutional Neural Networks (CNN) require minimal preprocessing and analyze raw traces effectively [11][8][10]. Tools like Deep Learning Leakage Assessment (DL-LA)¹ further enhance leakage detection.

Platforms for Security Assessment

Standardized platforms for SCAs include FPGA-based tools like FOBOS [3], INSTAC boards, Sakura-Sasebo project² boards, and ChipWhisperer systems³. Advanced tools like Riscure Inspector⁴ and DPA Workstation⁵ provide comprehensive evaluation capabilities, but at a higher

¹<https://github.com/Chair-for-Security-Engineering/DL-LA>

²<http://satoh.cs.uec.ac.jp/SAKURA/index.html>

³<https://www.newae.com/chipwhisperer>

⁴<https://www.riscure.com>

⁵<https://www.rambus.com>

cost. Innovations like FlexLeCo [9] and eSHARD⁶ offer scalable and efficient alternatives for accelerated trace collection and analysis[4].

SCAs on Multi-Tenant FPGAs

Multi-tenant FPGAs used in cloud and edge computing enable attackers to exploit shared physical resources. Techniques like Ring Oscillators (ROs) and Time-to-Digital Converters (TDCs) detect voltage fluctuations on FPGA boards, targeting Power Distribution Networks (PDNs) to extract cryptographic keys [14][5]. Hardware Trojans and profiling attacks further highlight vulnerabilities in these systems, including successful AES-128 key extractions [6][13][7].

Gravellier et al. [6] developed an innovative on-chip sensor circuit using Ring Oscillators (RO) to perform remote SCAs on FPGAs by monitoring voltage fluctuations. By applying Correlation Power Analysis (CPA), they successfully decrypted AES encryption keys from a low-frequency core. Similarly, Zhao et al. [16] leveraged an RO circuit embedded within an FPGA's programmable logic block to extract RSA crypto module keys from a CPU processor. Beyond ROs and Time-to-Digital Converters (TDCs), Spielman et al. [15] introduced Routing Delay Sensors (RDS) as an efficient alternative for detecting FPGA voltage fluctuations, reducing the side-channel traces required to break AES-128 encryption by 35% and simplifying deployment process. Furthermore, Sakamoto et al. [13] demonstrated the extraction of complete AES-128 keys from embedded devices using Analog-to-Digital Converter (ADC) data, emphasizing the need for improved sensor security. Finally, Ji et al. [7] employed a DL-based profiling attack on CRYSTALS-Kyber hardware (Kyber768) to retrieve session keys, highlighting vulnerabilities in current post-quantum cryptography implementations and emphasizing the urgency for robust countermeasures.

2.3 Limitations and Challenges

Limitations and Challenges in Side Channel Attack Assessment as a whole

SCAs rely significantly on the capacity of a system to collect side channel traces (i.e., time series data of leaked side channel information associated with sensitive data processing) efficiently and accurately (with sufficient sampling frequency that retains information leakage). However, in many occasions trace collection is time consuming, thus making the assessment extremely slow. Given that elaborate SCAs that bypass existing countermeasures need a huge amount of traces (e.g., in the AES algorithm), the ability or inability to collect such number of traces constitutes a turning point in the SCA success or failure. It should be noted that SCAs are considered not successful (or inapplicable) when they need unrealistically large amount of traces. Methodologies and techniques that can increase the trace collection capacity of an assessment system could significantly change the SCA landscape.

Furthermore, while there are several SCA categories in the relevant literature, most of the attacks even in the same category are manifested differently in each target system, thus making the realization of a comprehensive SCA assessment flow very difficult and diverse. A penetration testing type of approach needs to be followed in order to assess a target against various attack types.

⁶<https://eshard.com/>

Limitations and Challenges in Multi-tenant FPGA environments

Multi-tenant FPGA environments face significant challenges in the context of Power Side Channel Attack (PSCA)s. First, we utilize Hardware Analyzer modules to investigate and demonstrate potential attack scenarios. These tools reveal that, if an attacker has unrestricted time and resources, they can eventually compromise the system, as the shared nature of PDN leaves no permanent mitigation in place. This inherent vulnerability is compounded by the difficulty of isolating power noise contributions from multiple users, which complicates both attack detection and prevention. Second, the sensors used in attacks and analyses introduce additional stress on the FPGA, which can degrade the system's performance, reliability, and stability. Furthermore, ensuring fairness in resource allocation while maintaining security is a significant challenge, as implementing defenses often introduces overhead that impacts other users. These issues underscore the pressing need for innovative strategies to address the unique risks and constraints of multi-tenant FPGA environments.

3 Dynamic Hardware Analyzer Architecture and Components

3.1 Overall Architecture

The Dynamic Hardware Analyzer is designed to evaluate both hardware and software IP cores for vulnerabilities related to information leakage from their underlined hardware structure. This leakage arises from the physical characteristics of the device. The platform is primarily concerned with side-channel leakage, such as power consumption and electromagnetic emissions, which can compromise security. Cryptography IP cores are highly vulnerable to side-channel attacks, making them a primary focus for the analysis within the Dynamic Hardware Analyzer. The platform supports a series of sensors both physical and logical/digital ones in order to collect side channel leakage information, demoted as traces. This allows the platform to assess and analyze both side channel leakage of a software/hardware IP core deployed on a given single user (or tenant) system but also allows the assessment of remotely deployed IP core (mostly hardware IP Cores) on multi-tenant hardware devices, especially FPGA devices shared through cloud infrastructures for hardware acceleration of cryptography operation.

To achieve the above goals, the Dynamic Hardware Analyzer integrates several hardware and software components. These include Commercial off-the-shelf (COTS) hardware such as the ChipWhisperer Lite, open-source side-channel analysis tools such as the Scared library, dedicated hardware platforms like the SAKURA X board, and custom software components such as the FlexLeco v2.0 platform. Together, these elements support the evaluation of various implementation vulnerabilities:

- **Traditional Side-Channel Attacks (SCAs):** These include methods such as Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Correlation Power Analysis (CPA), targeting symmetric and asymmetric cryptography IP cores using common cryptographic algorithms.
- **Machine Learning (ML)/Deep Learning (DL)-based SCAs:** These profiling attacks leverage ML and DL techniques, particularly focusing on asymmetric cryptography.
- **Remote SCAs on Multitenant FPGAs:** These attacks exploit FPGA power distribution networks, using dedicated circuits to gather traces. The trace collection can be done locally or remotely by introducing into a multi-tenant FPGA an attacker circuit that taps onto the power distribution network thus been able to measure the power consumption (indirectly) of a target FPGA design also deployed on the same FPGA.

The Dynamic Hardware Analyzer offers two levels of assessment: a generic view and a refined view. The generic assessment provides an overall evaluation of whether an IP core implementation leaks sensitive data but does not specify how such leakage might be exploited. It uses a technique known as TVLA (Test Vector Leakage Assessment), which checks if the side-channel leakage exceeds a predefined threshold. If the generic assessment yields positive results, the refined view is used. This refined assessment involves a more in-depth, targeted evaluation, treating the cryptography IP core (or device under test, DuT) as a black or grey box. It applies

specific penetration testing techniques and selects the most appropriate side-channel attacks from the Dynamic Hardware Analyzer’s SCA library based on the collected traces.

As shown in Figure 1, the Dynamic Hardware Analyzer consists of two primary modules: the Trace Collection Module, responsible for gathering side-channel data, and the Trace Analyzer Module, which analyzes collected traces to evaluate the collected information and assess of exploitable information leakage exists or specific SCAs can be mounted.

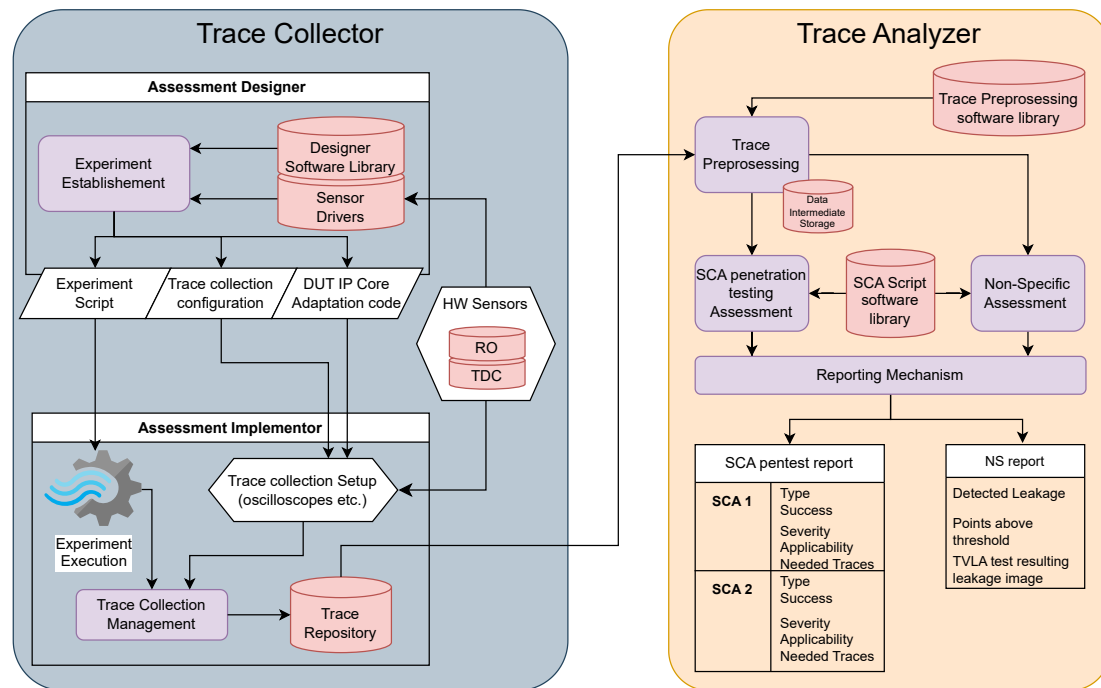


Figure 1: Dynamic Hardware Analyzer Architecture.

3.2 Trace Collector

3.2.1 Features and Supported Processes

The focus of the Trace Collector is the streamlined establishment of leakage assessment experiments, by enabling the collection of power consumption or electromagnetic emission traces as well as remote FPGA extracted leakage traces using remote FPGA sensors. Consisting of two main components, the Assessment Designer and the Assessment Implementor, it allows the user to design and execute trace collection experiments using several platforms as a backend where a deployed assessment target software or hardware can be deployed.

The mechanism supports both hardware and software IP Core evaluations and allow the user to adapt the IP Core design to interface with the supported platforms when required, particularly for hardware cores that need a FlexLeco v.2.0 bus interface [9]. The system also provides scripting and configuration tools to streamline trace collection and store results for further analysis. In general, the trace collector provides an abstraction of the various trace collection approaches in the relevant literature as well as the market.

The main process that is followed in the trace collection is the design and implementation of a controlled environment-based execution flow that, through the use of dedicated sensors, will lead to the collection of SCA traces related to one or multiple types of SCAs. To achieve this the Trace Collection supports several execution flow profiles to much known SCAs. Indicative such flows are the following:

- **Generic Leakage Assessment flow:** Goal of this flow is to identify exploitable leakage of a specific input of a cryptography/security implementation that handles sensitive data. The target implementation is executed a predefined number of times in the Trace Collector Controlled environment. In each time, the targeted input is given either a constant, known value, or a random value. Each one of those two input options are chosen randomly with equal probability. SCA Traces are collected for each execution and are labeled accordingly (constant or random input).
- **Simple SCA flow:** Goal of this flow is the collection of a single (or a small number) of SCA Traces. The target implementation is provided with known inputs, and the outputs are recorded. The experiment may be run a single or a small number of times (with different inputs each) and the SCA traces of each execution are also recorded and associated with the input/output tuples.
- **Advanced SCA flow:** Goal of this flow is the collection of a significant number of SCA traces for a given cryptography/security implementation. The target implementation is executed a predefined number of times in the Trace Collector Controlled environment. Each time, different predefined (or random) inputs are fed to the target implementation with the exception of the input where sensitive information is given. This input has a constant value (known to the assessor) that is meant to be recovered through side-channel analysis. In each execution of the target implementation, the SCA traces are collected and are associated with the input/output tuples used in this execution.
- **Profiling/ML/DL SCA flow:** This flow is similar to the Advanced SCA flow however the goal is to create a dataset of SCA Traces with known sensitive information for at least one of the target implementation inputs that will be used in a profiling (Template or ML/DL based) SCA. Thus each trace, or even a Point of Interest of each trace, is labeled to some part of the sensitive information of a specific execution. The flow requires a significant number of target implementation executions to create the complete data set.

To realize the above indicative flows as well as any variation of them, the Trace Collection offers a generic API that supports random input generation, input and output recording, control environment status check, trace collection start and stop mechanisms as well as control environment configuration. Depending on the type of control environment used for the trace collection, e.g., hardware IP core assessment using FPGAs, software IP Core assessment using embedded system platforms, the realization of this generic API differs using appropriate backend software libraries.

3.2.2 Trace Collection Controlled environment

Trace Collector relies on specialized or COTS hardware platforms and associated sensor equipment to collect SCA traces. Such platforms act as the controlled environment to execute a trace

collection experiment following the above-described execution flows. In such environment, the user is able to control, log and record inputs and outputs of the target IP core to be assessed and also is able, through proper instrumentation to define when the trace collection process will start and end. In parallel, the Trace Collector can potentially be connected to a trace collection oscilloscope so as to collect as fast as possible traces during IP Core execution. Currently, the following three controlled environments may be supported by the Trace Collector:

- **Flexleco v2.0 platform:** The platform originally described in [9] and extended as a git repo⁷ is meant to collect traces of hardware IP Cores that are dynamically deployed on a testing FPGA of the SAKURA X board. A dedicated softcore microprocessor is implemented in the secondary FPGA of the platform that acts as the control mechanism of the various trace collection execution flows that are implemented as C code based binaries executed in this FPGA. The platform supports an abstraction API in the C programming language that can configure the expected interaction with the IP core deployed in the testing FPGA. This can provide inputs and collect properly the outputs of the assessed IP core. The platform requires some instrumentation of the hardware IP Core (in VHDL or Verilog) so as to comply with the bus interface between the two FPGAs (testing and control FPGA) of the SAKURA X platforms.
- **Chipwhisperer lite project platform:** The open source Chipwhisperer platform is used for assessing software IP cores and includes a dedicated board for trace collection that communicates with a target embedded system platform (Chipwhisperer UFO board) through simple serial communication. The Chipwhisperer has a simple serial API to provide inputs, configuration, trace collection start/stop sequences and to collect outputs from the embedded system software running on the UFO board. The platform requires a serial communication interface to be integrated to the software IP core under assessment. This serial communication must be able to comply with the platform's specifications.
- **COTS embedded systems:** To overcome problems of the Chipwhisperer platform like low sampling frequency, small memory buffers, and UFO platform limited memory size, the Trace collector supports a generic mechanism to extract traces from any embedded system board as long as there is support for processor current or voltage measurement (e.g., using a dedicated jumper connector). Using this board infrastructure and ad-hoc circuitry (e.g., series of resistors/capacitors), probes can be mounted on any embedded system and through an oscilloscope traces can be collected. Proper instrumentation is needed in order to assist the trace collection process, practically guiding the trace collector on where to start and stop trace collection.
- **Xilinx/AMD Multitenant FPGA platforms:** As a controlled environment for remote SCAs in a multi-tenant FPGA, we are using various FPGA devices where the FPGA is mounted as a dedicated device locally—thus emulating the remote environment. Xilin FPGAs are used like AMD ZCU 102 or 104 as well as AMD Alveo boards. The FPGAs are mounted with the assessed hardware IP core peripheral and in parallel to this a series of hardware sensors are also mounted to probe the FPGA Power Distribution Network and collect traces accordingly.

⁷<https://github.com/afournaris/FlexLECO2/>

3.2.3 Internal Design/Implementation

Assessment Designer: This component consists of two main software libraries: the Designer Software Library and the Sensor Drivers (firmware) library. The Designer Software Library provides an API that abstracts communication with the controlled environment where the trace analysis takes place focusing on the proper establishment of a trace collection execution including the supply of the assessed IP core with test vectors. This library is responsible for defining the assessment characteristics such as the number of inputs and outputs, control signals, and bit lengths. It also provides functions for starting and stopping trace collection experiments, depending on whether the assessment involved a hardware or software IP core. The Sensor Drivers library enables trace collection from in-house created sensors, such as those used for FPGA-based side channel attacks, and includes firmware to support these sensors. When working with hardware IP cores, the Designer also provides support for integrating the FlexLeco v.2.0 bus interface to manage the control of input, output, and control signals dynamically.

Assessment Implementor: Continuing the work of the Assessment Designer, this component carries out the actual trace collection process, using the configuration scripts and parameters defined by the Assessment Designer. The implementor interfaces with trace collection equipment, such as oscilloscopes and sensors, and stores the collected traces in a centralized SCA trace repository. For remote FPGA attacks, the Assessment Implementor is tasked with gathering power consumption-related traces from hardware sensors as the cryptographic component operates. Commonly, there exist two main types of voltage sensors for mounting remote power side-channel attacks (Figure 2); the Ring Oscillator (RO) based delay sensor and the taped delay-line or Time-to-Digital Converter (TDC) sensor.

- **Ring Oscillators**

RO sensors utilize a NAND gate, with inputs from an external enable signal and the gate's own output. When activated, the gate generates a continuous oscillation whose frequency varies with changes in voltage, as determined by its propagation delay. By monitoring the oscillation frequency through digital counters, RO sensors provide insights into power consumption.

- **Time-to-Digital Converters**

TDC-based sensors, on the other hand, operate by assessing how far a signal can propagate along a path with latches inserted between logic elements, such as a chain of buffers. Since buffer delays are sensitive to changes in process, voltage, and temperature (PVT), these delays are detectable by reading the latches. In this setup, a signal is sent simultaneously to the input of the first buffer and the latch enable signals, so the signal on the wire outruns the signal through the buffer chain. With a precisely timed clock signal, the latches become disabled at half the clock period, preserving the clock's reach within the chain for further analysis. The initial delay is set to be less than half of the clock period and latches along the buffer chain fall within the delay variation range of interest. This range is selected to balance acceptable area overhead and operating values.

In comparison, RO sensors require simpler circuitry and consume fewer resources of the fabric—though they are less accurate due to the limitation of counters, which cannot achieve nanosecond-level sampling. RO sensors are also more easily flagged as malicious, making them less desirable for attack purposes. TDC sensors, however, enable nanosecond-level voltage fluctuation

measurements and can function as thermal sensors. Additionally, their implementation is harder for design tools to block, making TDC sensors a more effective choice for power side-channel attacks.

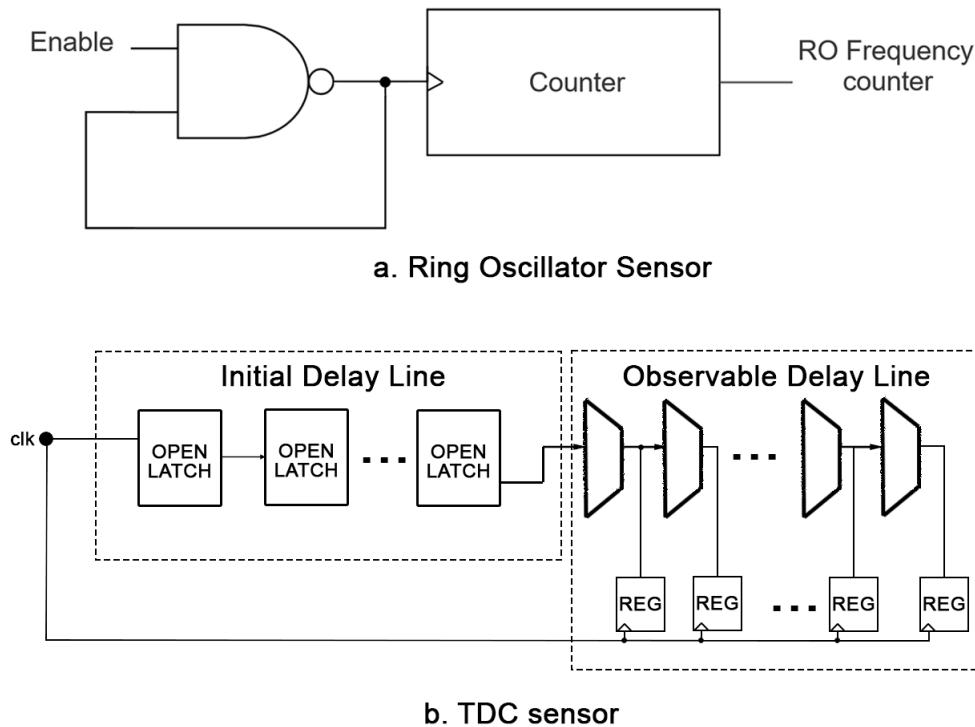


Figure 2: Hardware sensors.

3.2.4 Interconnections

The Assessment Designer interacts with users via a scripting interface, enabling them to configure and execute trace collection experiments following a trace collection flow profile. It connects to the trace collection equipment, including sensors and oscilloscopes, as well as custom-built sensors when necessary. The Assessment Implementor uses the configurations defined by the Designer to manage the trace collection process and store data in the SCA repository, making it available for further analysis. Additionally, for hardware IP cores, the Assessment Designer ensures seamless integration by extending the IP core with the FlexLeco v.2.0 bus interface, facilitating smooth communication during trace collection.

3.3 Trace Analyzer

3.3.1 Features and Supported Processes

The Trace Analyzer component is responsible for providing leakage assessment functionality in a generalized and flexible manner. It is designed to offer the SCA analysis for a variety of platforms, both software and hardware and in order to achieve that it encapsulates a unified API that can be easily adaptable. Utilizing a number of leakage assessment functions, the Trace

Analyzer has the ability to make SCA evaluations of the resistance of a given implementation of a cryptographic library with a relatively high degree of confidence. The two high-level processes this component supports can be summarized below:

- **Non-specific leakage assessment:** In cases where a deep understanding of the Device under Test (DuT) is not necessary or feasible, a non-specific leakage assessment approach like TVLA t-test based techniques can help detect leakage points of interest.
- **Specialized SCAs:** Fully fledged attacks that abide to penetration testing principles and adopt various threat models can challenge the resistance of a specific IP core or software implementation against a list of SCAs. A very popular category of techniques is profiling attacks, and by extension the ML/DL approach that enhances their performance in attacking the chosen points of interest that are unique and specific for each assessed target.

In a more explicit manner, the processes presented below for each category are characteristic examples of the included SCA functionality:

Non-Specific Leakage Assessment

The Welch t-test is a popular statistical tool that attempts to highlight possible differences between the means of two different target groups. In the context of SCA, its purpose is mainly to make comparisons between traces (electromagnetic emissions, power consumption) that usually represent different conditions, for example random vs. fixed numerical values as inputs for the cryptographic algorithm. Possible differences in means between these groups are identified by the t-test and can signify a potential leakage of critical information. Despite its ease of use and implementation as a first step for leakage detection, this technique has difficulties in detecting subtle or non-linear dependencies between the traces and requires in many cases very large datasets for reliable leakage detection.

The evolution of the Welch t-test for SCA is the Test Vector Leakage Assessment (TVLA). It computes the t-statistic for traces collected under different conditions, but compares it against a predefined threshold (usually at ± 4.5) in order to identify possible leakage. The TVLA technique constitutes a standardized approach to leakage assessment, as it offers semi-automated detection of first-order leakage. Similarly to the Welch t-test, it struggles in identifying more complex leakage forms and requires large amounts of data to do so.

Specialized SCA Assessment

A more advanced SCA attack is the Correlation Power Analysis (CPA), in which as the name suggests, the correlation between the real dataset (power consumption traces) and a chosen hypothetical power model based on guessed intermediate values is measured. Using these predicted values for different key guesses, CPA identifies through the highest correlation which guess was the correct one, thus further enabling the recovery of the secret key. It is quite effective for many types of SCAs when paired with an accurate power model, but its efficiency is hampered by an incorrect power model or noisy captured traces. For the profiling type of SCAs, where an IP Core is uniquely assessed, techniques based on ML, such as Support Vector Machines (SVMs), Random Forest (RFs), Neural Networks (Deep/Convolutional) have quickly gained popularity in a SCA context. Their ability to learn patterns of leakage that might be non-linear and complex on already labeled training data and adapt them into unknown traces makes

them a powerful tool for modern side channel analysis, since the variety of data they can handle is pretty large.

3.3.2 Internal Design/Implementation

Architecturally, the Trace Analyzer consists of two main submodules, the SCA script library and the trace preprocessing library. The main functionality of the analyzer is offered by the SCA script library, since it facilitates all the leakage assessment functions under a unified API. It is also responsible for the initialization of the SCA assessment flow. To support the SCA analysis, the trace preprocessing library provides implementations of useful signal processing functions to enhance the performance of the submodule. Below, the SCA penetration testing assessment flow is presented in Figure 3:

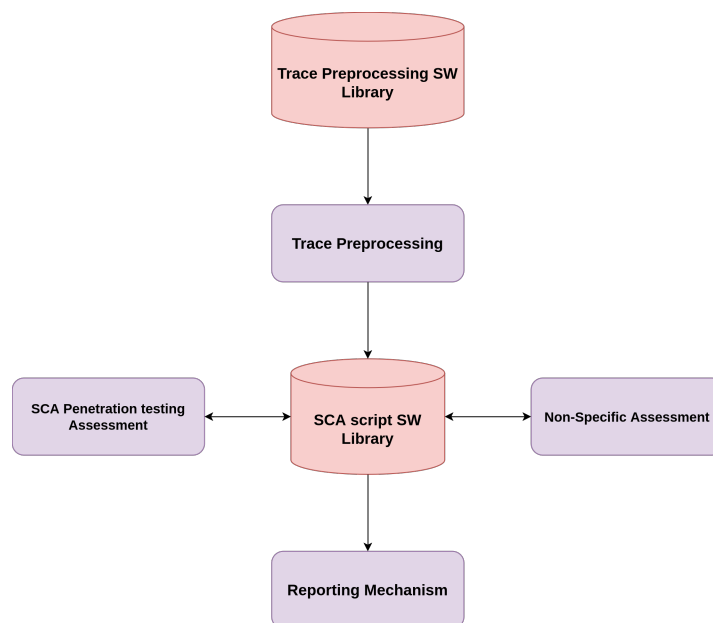


Figure 3: SCA Trace Analyzer flow.

The SCA script software library is a flexible tool for the user, since it offers fast implementation of various common SCAs to assess a specific cryptographic implementation. The high level overview of these functions have been discussed in subsection 3.3.1. Based on the type of the attack and the configurations provided by the expert user, the SCA script library is assisted by the trace preprocessing library to prepare the trace dataset tailored for each attack.

The trace preprocessing library is a necessary component that aids with the side channel assessment. Even for more advanced DL attacks that can handle noisy data in a more efficient manner, preprocessing the power traces is of paramount importance. Examples of the functions that the trace preprocessing library offers are:

- filtering (high-pass, low-pass, band-pass)
- cropping
- averaging

- scaling
- general techniques to increase the Signal-to-Noise ratio

After the SCA Assessment is completed, the Trace Analyzer generates an assessment report through a reporting mechanism. This report is a JavaScript Object Notation (JSON) file, containing all the necessary information about the SCA and its structure is explained in Section 3.3.3.2. The report of the tool can act as the building block of the CycloneDX based input to the overall Dynamic Testing Module of the RESCALE architecture further analyzed in Deliverable D3.3.

3.3.3 Interconnections-Input/Output

3.3.3.1 Input

Universally collected traces with the aid of the Trace Collector from various controlled environment setups can be considered the data input to the platform. This type of data is usually structured as .csv or .mat files, depending on the configuration of the lab setup. Thus, after the trace collection and correct annotation, the trace dataset is being tested for its SCA resistance by the Analyzer submodule. The user can access the functionality of the Analyzer through an easy-to-use Jupyter Notebook interface.

HW Platfor...

SW Version:

OS:

Algorithm:

Choose an existing ML model

Choose a pre-computed features file

Choose the output report directory

☐ Use existing attack configuration file

Figure 4: Jupyter Notebook input for the SCA Trace Analyzer.

The SCA security expert begins the assessment by selecting all the necessary information to conduct the testing of the implementation as is shown in Figure 4. This process is available through two methods, the first of one is to manually input the information with the Jupyter Notebook functionality. Apart from the mandatory fields that categorize the cryptographic algorithm to be assessed information and the platform the assessed IP Core has been implemented to, the user has additional options related to the attack itself. More specifically, the user can

choose an existing pre-trained ML model for classification of the trace dataset that adheres to the same preprocessing and labeling of the data. On top of that the user can use a pre-computed features file of the same data in order to gain a performance boost, since the processing of the various features of the trace dataset can take a significant amount of time. Additionally, the user can choose the output report directory, where the SCA Assessment report the Trace Analyzer generates will be stored.

For the sake of simplicity, there is an additional option available for the user, where an already existing attack configuration file can be used for inserting all the necessary fields required by the Analyzer. This is a JSON file describing the attack in detail. An example of such a file is presented below:

```
{
  "type": "",
  "target_settings": {
    "assessed_system": {
      "hardware_platform": "",
      "software_version": "",
      "operating_system": "",
      "algorithm": ""
    },
    "report_directory": "",
    "model_file": "",
    "trace_file": "",
    "features_file": "",
    "retrain": boolean
  },
  "attack_settings": {
    "cnn_settings": {
      "layers": ,
      "filters": [],
      "kernel_size":
    },
    "rf_settings": {
      "n_estimators": ,
      "max_depth":
    },
    "svm_settings": {
      "kernel": "",
      "C":
    },
    "mlp_settings": {
      "hidden_layers": [],
      "activation": ""
    },
    "t_test_settings": {
      "alpha": ,
      "tail": ""
    }
  },
}
```

```
    "cpa_settings": {
      "number_of_traces": ,
      "hypothesis": ""
    },
    "dpa_settings": {
      "number_of_traces": ,
      "threshold":
    }
  }
}
```

The JSON format offers a quick and flexible way of setting up a new experiment or reiterate one with variations in its parameters. The basic JSON fields of the attack configuration file are explained below:

- **type**: The type of SCA assessment or attack to be executed for this specific attack instance. It can range between simple leakage indicator tests like TVLA or ML attacks (SVM, RF, Convolutional Neural Networks). The type also indicates if the attack is related to remote SCAs (using multi-tenant FPGAs) or local SCA
- **target_settings**: Various information regarding the target of the attack, its hardware or software platforms and the underlying cryptographic implementation.
- **report_directory**: The output directory of the SCA assessment report.
- **model_file**: A pre-trained ML model that is ready to use for inference.
- **trace_file**: The file containing the trace dataset.
- **features_file**: This field is relevant in cases when the feature preprocessing has already been completed and the relevant features are saved in a separate file.
- **retrain**: A boolean value indicating the retraining of a new ML model based on the trace dataset or the reuse of an existing one.
- **attack_settings**: These JSON subfields are utilized as a LUT from the SCA script library, which automatically selects the correct settings based on the type of the attack the security expert has chosen to conduct.

3.3.3.2 Output

During the attack process, the user of the platform constantly interacts with the Jupyter Notebook to proceed to the next semi-automated steps of the acrsShortSCA vulnerability assessment. Based on predefined thresholds that are unique for each type of attack, a final level of leakage severity is generated for the attack instance. This severity indicator, alongside information about the attack itself, the DuT and possible recommendations for strengthening the underlying cryptographic implementation are included in the acrsShortSCA Assessment Report. Its JSON format is explained below:

```
{
  {
    "title": "SCA Toolbox Leakage Assessment Report",
    "date": "YYYY-MM-DD",
    "assessed_system": {
      "system_UUID": "",
      "hardware_platform": "",
      "software_version": "",
      "operating_system": "",
      "algorithm": ""
      "Mitigations": {
        "type": "",
        "description": "",
      },
    },
    "side_channel_attack": {
      "SCA_UUID": "",
      "type": "",
      "description": "",
      "tools_used": [],
      "leakage": {
        "severity_level": "",
        "impact_description": ""
      },
    },
  },
  "leakage_metrics": {
    "type_of_metrics": [
      "threshold"
    ],
    "metric_values":
  },
  "recommendations": ""
}
```

- **assessed_system:** The information relevant to the target system to be assessed, including fields like the hardware platform, software version, the operating system if present and the algorithm under attack.
- **side_channel_attack:** This field gives a brief pre-generated description of the attack to be executed (CPA, DPA, ML etc.). In addition, information is collected about all the possible tools or software libraries that were necessary for the execution of this specific attack.
- **leakage:** This field provides information about the leakage severity level that was attributed after the leakage assessment has been completed. This severity metric is an integer value and indicates the potential vulnerability from side channel attacks the target system to be assessed is under. Thus, it constitutes a quick and easy way to raise potential suggestions about new countermeasures required against such attacks.

- **leakage_metrics:** In this field, technical information related to the specific attack and its corresponding leakage are presented, including various metric values.
- **recommendations:** Possible courses of action to be taken as next steps in order to protect the target system to be assessed when we have high leakage severity.

4 Open Issues

4.1 Identified Challenges

4.1.1 Generic SCA Assessment Challenges

Defining generic assessment strategies of side-channel leakage for software or hardware IP Cores is very difficult to achieve. In fact, the relevant literature relies on custom assessment flows that follow common generic principles per attack type, but are realized in practice differently. This makes it challenging to provide a unified way of assessing multiple different IP cores (i.e., IP Cores with different cryptography functionalities). Creating an exhaustive list of attacks per type of cryptography scheme implementation is not always realistic, hence the challenge of creating an expandable assessment flow remains. Thus, the Dynamic Hardware Analyzer system will have to be structured in such a way that can support a corpus of indicative SCAs that can be used for security testing a given cryptography IP core. However, such a corpus cannot be complete. Instead, proper structures within the Dynamic Hardware Analyzer must be provided to include in the future new SCAs (or SCA updates).

Another challenge to consider is the fact that well-known SCAs are not directly reflected in CWE/CVE databases. They are often considered the means to facilitate some more high-level Exploit/Vulnerability. However, in RESCALE we consider that discovering exploitable leakage and pointing to possible exploits of such leakage towards an SCA constitute important information to be considered when an IP Core component needs to be trusted in the Hardware or Software supply chain. Hence, in the RESCALE Dynamic Hardware Analyzer, we need to provide a reporting mechanism, in line with the CycloneDX initiative (promoted in the project) that will provide a measurable indication of the risk and attack severity associated with SCA leakage. An initial attempt towards this end is provided in paragraph 3.3.3.2 and also in the Dynamic Hardware Analyzer CycloneDX declaration of Deliverable 3.3.

4.1.2 Multi-tenant FPGA PSCA challenges

Researchers face significant challenges in the context of PSCAs on multi-tenant FPGA environments. In the RESCALE project, we use the Hardware Analyzer module to evaluate potential attacks, demonstrating that, if attackers have sufficient time and resources, they can eventually mount a successful attack. Assessment becomes more complex when several tenants are sharing the same FPGA platform—thus introducing noise in the measurements. This, however, does not mean that an SCA cannot be mounted, but rather that an existing assessment flow must be modified to take into account such noise. Countermeasures on PSCA can increase the amount of time and resources that an attacker needs to mount an attack. However, they cannot prevent a determined attacker with significant resources/time from actually bypassing mitigation mechanisms. Another challenge is that the sensors employed for these attacks and analyzers can impose additional stress on the FPGA, potentially degrading its performance and stability. These challenges emphasize the critical need to understand and address security vulnerabilities in multi-tenant FPGA systems. Furthermore, ensuring fairness in resource allocation while maintaining security is a significant challenge, as implementing defenses often

introduces overhead that impacts other tenants of the FPGA platform.

4.2 Mitigation Solutions

Although the RESCALE project is focused on the security assessment of software and hardware components of an application supply chain and not on mitigation, it is important to conceptually describe the mitigation mechanisms of SCAs and how they can alter the assessment flow. SCAs are mitigated using two primary methodologies, masking and hiding.

- **Masking**

Masking countermeasures aim to reduce the sensitive information exposed in the side channel, or, in other words, decrease the correlation between the sensitive information been processed and the side channel information captured by the measurements. Masking is typically applied by adding, multiplying, or XORing random values to sensitive values before processing. Usually, this happens in combination with a split of sensitive data (like cryptographic keys) into multiple independent shares using random values, ensuring that no single share reveals any information about the original data. The impact of masking, as expected, is significant in the success of SCAs. However, it can be thwarted by high-order SCAs that combine leakage from multiple points of the cryptographic processing (more than one order). Therefore, SCA security testing should also implement high-order SCA for a very comprehensive IP Core assessment.

- **Hiding**

Hiding countermeasures focus on reducing the observability of side channel leakage through measurements. This can be done by reducing the signal strength or by increasing noise or interference in the signal. Usually, hiding is done in hardware, since the defenders need to exploit the hardware structure to, for example, equalize power consumption or time, so as to hide sensitive information leakage. In contrast to masking which is application-specific (i.e., is manifested differently on each cryptography scheme implementation), hiding can be generic and implementation-agnostic. Advanced SCAs that involve deep statistical analysis of signals or rely on ML can potentially overcome hiding countermeasures by including noise reduction techniques and ML training under the presence of noise.

When it comes to multi-tenant FPGA SCAs, some of the above generic methodologies can become more focused on the FPGA hardware structure. For example a common countermeasure is the use of "Passive Fences," which involve isolating designs with empty FPGA slices. While this technique aims to prevent physical-level interactions between neighboring designs, it has proven ineffective against electrical-level attacks. Power distribution networks can still be exploited to launch SCAs, regardless of the size or configuration of Passive Fences.

In addition to passive fences, "Active fences" offer a more dynamic approach to security. These fences employ ROs as a countermeasure to introduce controlled noise into the power supply network [2]. By carefully adjusting the frequency and amplitude of the ROs, the noise can obscure sensitive power consumption patterns, making it significantly harder for attackers to extract valuable information. This technique provides a more robust defense against power

SCAs compared to passive fences, as it actively counters the attacker's efforts to monitor power fluctuations.

The above generic and FPGA focused countermeasures can significantly delay the computation and overall performance of a cryptography IP Core. While they can make a SCA significantly more difficult to realize in practice, they cannot guarantee complete SCA resistance. Therefore, the SCA security assessment goal is to minimize the risk of successful attacks rather than removing it.

5 Roadmap to Final Version

5.1 Development Plan

Given the overall concept and architecture of the hardware vulnerability and Leakage Detection solution (i.e., consisting of two modules/components, the Trace Collector and Trace Analyzer), the development plan is focused firstly on the finetuning and prototype development of the Trace Analyzer (since this component provides the core SCA assessment) and secondarily on the trace collection component (with the main focus being the multiple platform integration). Eventually, the Trace Collector and Trace Analyzer components will be offered as two different building blocks. In this way, trace collection, which requires the setup and operation of a laboratory controlled environment, is practically disassociated from the trace analysis process, which is unrelated to the lab (it only requires the traces and proper configuration). Thus, the Trace Analyzer will be provided as a container where all the necessary setup prerequisites are met and libraries are installed. The Trace analysis container is going to be integrated into the Dynamic Testing Module and contribute to the generation of the DSCG report.

5.2 Initial Evaluation

To evaluate the platform functionality and end-to-end usage (from trace collection to trace analysis and reporting) we have performed some initial evaluation tests. In the following subsections, we provide demonstration of indicative tests performed for ML/DL SCA assessment showcasing the Trace Analyzer component in action. This is followed by complementary trace collection and analysis based on CPA assessment for remote multi-tenant FPGA based hardware IP cores.

5.2.1 ML-based SCA

Focusing more on the SCA Trace Analyzer flow that was introduced in Section 3, in Figure 5 presents an overview of the attack configuration JSON file. The attack is thereby set up with the necessary information for the execution.

Load attack configuration file

```
In [11]: config = load_attack_config(config_path)
```

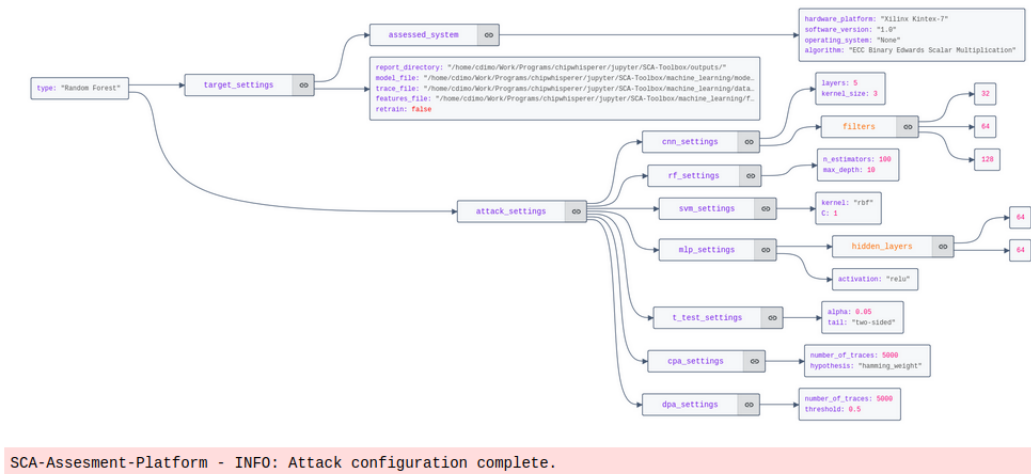


Figure 5: Configuration of the attack for the SCA Trace Analyzer.

As an example, in this attack instance, a Binary Edwards Elliptic Curve HW accelerator based on a 233-bit curve is the target of a ML-based attack. Before implementing the attack and training the possible models, the traces that were captured with the use of a SAKURA-X FPGA evaluation board and a PicoScope 5000 Series oscilloscope⁸ need to be preprocessed for better attack efficiency. Thus, Figures 6 and 7 display a preprocessing and feature extraction process, where the relevant features for every key-bit trace are computed. This process is an optional step in the overall flow, as the relevant file containing the various features can be reused in subsequent attacks directly and, if necessary, through the JSON attack configuration functionality.

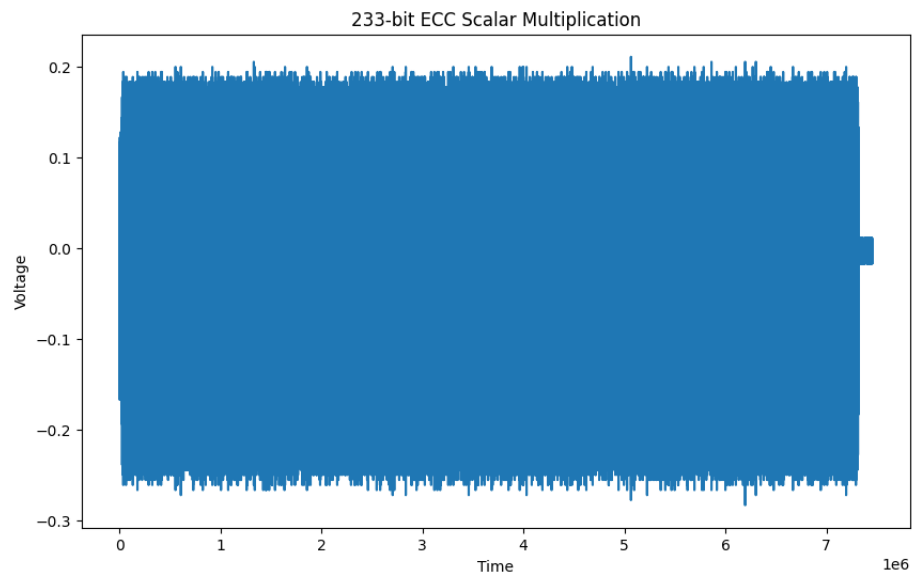
⁸<https://www.picotech.com/oscilloscope/5000/flexible-resolution-oscilloscope>

ML Preprocessing

Trace Splitting and Feature Extraction

```
In [3]: features_filepath = os.path.join(ml_directory, 'features/bec_split_features.csv')
preprocessing.compute_features(features_filepath)
```

```
SCA-Assesment-Platform - INFO: Dataset preprocessing...
SCA-Assesment-Platform - INFO: Loading trace file...
```



```
SCA-Assesment-Platform - INFO: Splitting into key-bit power traces...
SCA-Assesment-Platform - INFO: Trace bit-splitting...DONE
SCA-Assesment-Platform - INFO: Displaying trace sample...
```

Figure 6: Trace preprocessing example for the SCA Trace Analyzer.

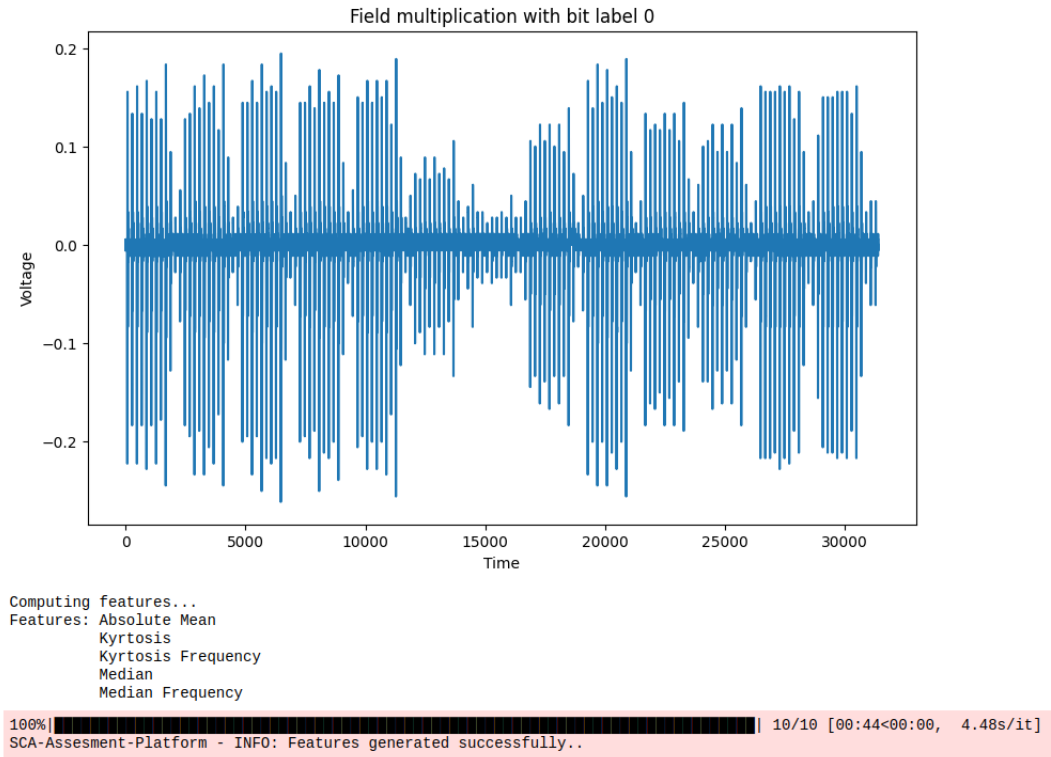


Figure 7: Trace preprocessing example for the SCA Trace Analyzer.

In Figure 8, the example ML-based attack is executed through a simple command call. As chosen in the configuration file, the type of attack to be assessed is a RF ML attack targeting a binary classification on the pre-processed traces, which are classified as bit-0 or bit-1. In this case, the hardened IP Core forces the RF model to a low classification accuracy.

Launch Leakage Assessment

Non Specific Leakage Assessment / SCA Penetration Testing

The type of leakage assessment is specified in the configuration file

In [12]:

```
attack = Attack(config)
leakage, leakage_metrics = attack.execute()
```

```
SCA-Assessment-Platform - INFO: Executing Random Forest attack...
SCA-Assessment-Platform - INFO: Using existing RF model for inference...
SCA-Assessment-Platform - INFO: --- Classification report ---
```

	precision	recall	f1-score	support
0	0.62	0.72	0.66	542
1	0.32	0.23	0.27	310
accuracy			0.54	852
macro avg	0.47	0.47	0.47	852
weighted avg	0.51	0.54	0.52	852

Accuracy is 0.539906103286385

Figure 8: Launch Attack Jupyter cell for the SCA Trace Analyzer.

The result collected by the attack process flow of the Trace Analyzer is used to generate a report filled with the necessary information for a semi-automated assessment report against SCAs, as shown in Figure 9. The relevant fields are automatically generated based on the leakage met-

rics, providing possible countermeasures or general courses of action in case of high leakage severity. However, this step of the process should be manually curated and monitored by the security expert that validates the final report.

SCA Assessment Report Generation

```
In [13]: generate_report(leakage, leakage_metrics, save_directory)

Side Channel Analysis report saved to: /home/cdimio/Work/Programs/chipwhisperer/jupyter/SCA-Toolbox/outputs/sca_report_20240905_191039.json
{
  "title": "SCA Toolbox Leakage Assessment Report",
  "date": "05/09/2024 - 19:10",
  "assessed_system": {
    "hardware_platform": "Xilinx Kintex-7",
    "software_version": "1.0",
    "operating_system": "None",
    "algorithm": "ECC Binary Edwards Scalar Multiplication"
  },
  "side_channel_attack": {
    "type": "Random Forest",
    "description": "A machine learning algorithm that builds multiple decision trees and merges them together to get a more accurate and stable prediction.",
    "tools_used": [
      "Scikit-Learn",
      "NumPy",
      "Pandas"
    ]
  },
  "leakage": {
    "severity_level": "medium",
    "impact_description": "Moderate leakage detected with a possible risk of key recovery under certain conditions."
  },
  "leakage_metrics": {
    "type_of_metrics": [
      "accuracy"
    ],
    "metric_values": 0.54
  },
  "recommendations": "Strengthen countermeasures by incorporating random delays and masking techniques."
}
```

Figure 9: SCA assessment report generation example.

5.2.2 Multi-tenant FPGA SCA

For performing CPA on Multi-tenant FPGA, the open-source SCABox-App⁹ Python application, that is currently partially integrated into the Trace Collector component, is used in tandem with the trace analysis libraries. SCABox-App application can operate on any computer connected through a USB UART interface to an FPGA board emulated a multi-tenant FPGA. The Python application conducts several AES iterations in chunks, collecting sensor measurements during each cycle. After each chunk, it performs a CPA on the final cycle of each AES iteration, using the sensor data alongside the resulting ciphertext. Since the encryption key used in the attack is known, it's possible to determine when enough data has been collected for a successful outcome. The application also includes a framework that visualizes correlation analysis through plots of the correlation against time samples and the number of traces. The user can specify parameters such as the number of AES iterations (N), the number of chunks (C), and the attack target. The total number of iterations in a single attack is calculated as $N \times C$. The attack can target either the board's USB port where the design runs or a directory of stored binary files from previous attacks. Although filtering can be applied to the data collected, it is not essential for the attack's success.

CPA operates on the hypothesis that power consumption measurements are correlated to data and operations processed within the cryptographic device. Using this approach, the analyzer

⁹<https://emse-sas-lab.github.io/SCABox/>

builds a hypothetical power model of the cryptographic process, estimating power consumption based on different possible values of the secret data. By calculating the Pearson correlation coefficient between predicted power values (from the model) and actual measured power traces, the analyzer assesses the linear relationship between them. The Pearson correlation coefficient, which ranges from -1 to +1, quantifies this relationship, with +1 indicating a perfect positive linear correlation and -1 a perfect negative one. If the traces are strongly correlated with a specific model prediction, this signals that the hypothetical model closely matches the actual cryptographic operation, allowing the secret data to be inferred. This statistical approach makes CPA a powerful method for extracting sensitive information from power consumption patterns. The assessment process is sketched in Figure 10.

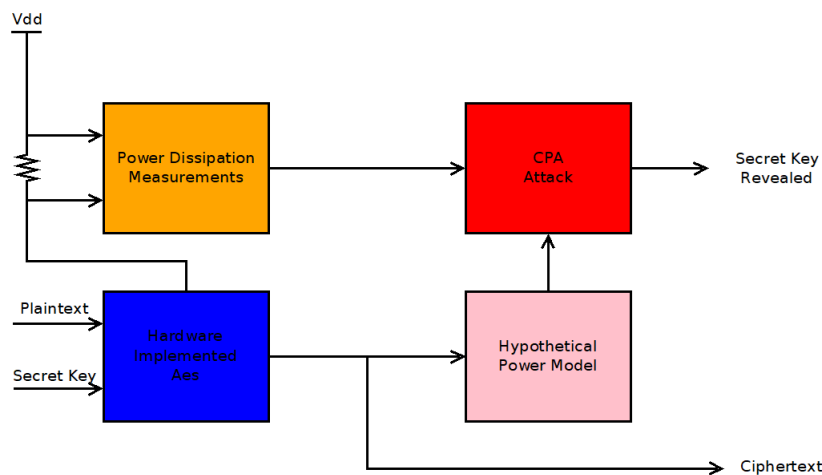


Figure 10: SCA in multi-tenant FPGA block diagram.

An example of hardware leakage with TDC sensors is shown in Figure 11. There are 11 distinct spikes, one for every AES cycle. The measurements between those spikes show a similar, repeated pattern. This is due to the same calculations that happen at each AES cycle and the divergence between them is because of the different data that are being processed. This means that the AES functionality is capturable by the TDC sensors and that, each time the AES 128-bit output register is overwritten at the end of each cycle, a high voltage drop occurs and is captured by the sensor. Although the cryptographic module's behavior is observable in the plot of the TDC sensor measurements, this does not mean that the statistical evaluation of the data can successfully lead to the extraction of the key. Multiple AES iterations need to be captured for the CPA to have a good result for any of the key's bytes. After the CPA, the result in Figure 12 shows the most probable byte value of the key (red line). The sooner the first true byte of the key has the highest correlation and is determined as the most probable by the CPA attack, the more vulnerable the system is.

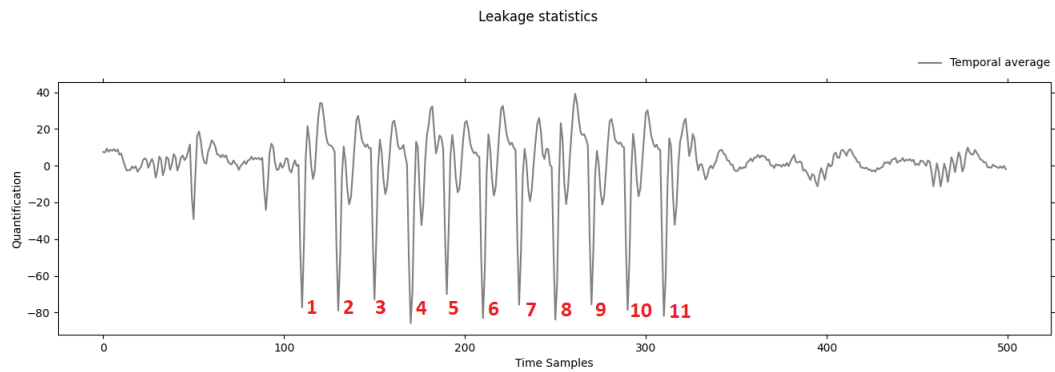


Figure 11: AES leakage plot.

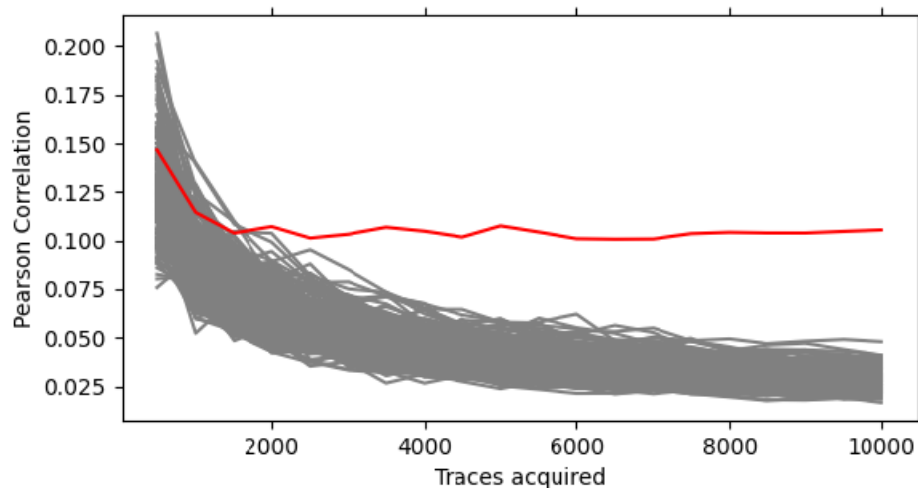


Figure 12: Pearson's correlation plot.

This initial experiment for the SCA in an AES module was conducted on a ZedBoard¹⁰ platform with AMD Xilinx Zynq®-7000 SoC acting as a controlled environment for the trace collector. However, the Hardware Analyzer can be evaluated in several other platforms, with different types of SCA scenarios, targeting multiple modules.

¹⁰<https://www.xilinx.com/products/boards-and-kits/1-8dyf-11.html>

6 Main Innovations & Conclusion

6.1 Main Innovations

The goal of the Dynamic Hardware Analyzer with its two components (Trace collector and Trace Analyzer) is to provide a comprehensive SCA leakage assessment approach in a unified, comprehensive way. To this end, we introduce several innovations for both trace collection and trace analysis. The trace collection approach followed in the RESCALE platform is meant to be simple and flexible, especially when it comes to hardware IP Core assessment. The provided APIs can be easily used in order to create an assessment flow and remain, up to a point, controlled environment agnostic.

Regarding the Trace Analyzer component, while we cannot provide a complete assessment library (i.e., for every SCA type and for every cryptography scheme implementation type), we provide a unified mechanism for assessment and reporting. Beyond that, Trace Analysis is focused on the most potent SCAs (i.e., DL profiling attacks) and how to apply this approach to new and popular cryptography scheme implementations like Public Key Cryptography—and more specifically Postquantum Cryptography (for standardized schemes). The outcome of this research innovation is to introduce new SCAs in order to assess such target systems.

Furthermore, in the RESCALE project, our work extends traditional SCA scenarios on multi-tenant FPGAs by considering the involvement of multiple other users in addition to the attacker and the victim. These additional users contribute to the system by generating voltage noise within the PDN and altering various environmental variables, which can introduce complexity into the analysis. Additionally, in another scenario, the user implementing the cryptographic algorithm introduces deliberate noise to mask the process and obscure side-channel leakage. This expanded scenarios aim to better reflect real-world conditions in multi-tenant cloud environments, where multiple workloads and mitigations can influence the vulnerability of cryptographic systems and affect the accuracy of side-channel leakage detection.

6.2 Conclusion

In this deliverable we described the overall concept, architecture, and functionalities of the hardware vulnerability and leakage detection mechanism introduced in the RESCALE project. This mechanism is consolidated into the Dynamic Hardware Analyzer solution and its components were also described at a high level. We also described open issues that we plan to consider after the submission of the deliverable till the end of Task 3.3. It can be concluded that there are several research directions to be explored in terms of SCA assessment, but also several obstacles to overcome—both practical and theoretical. The SCA research domain is huge and diverse, which highlight the need to focus on specific concepts in Task 3.3 and integrate them into the Dynamic Hardware Analyzer platform. Finally, in this Deliverable, we describe how the overall solution will be provided within the RESCALE platform (i.e., split into two different blocks, trace collection and trace analysis).

6.3 Future Work

The Dynamic Hardware Analyzer tool will be upgraded to aggregate a wide range of SCA outcomes from various subplatforms, including FPGAs and CPUs focusing on the further integration of the multi-tenant FPGA assessment approaches. This enhancement will involve creating a robust report management system designed to simplify the analysis process, enabling users to consolidate, compare, and interpret data across different attack vectors and hardware configurations. The reporting mechanism will be made more fine-grained to ensure that it is both comprehensive and detailed, while remaining adaptable to changing requirements. This design will facilitate seamless integration into advanced frameworks like DSCG and TBOM. The Hardware Analyzer platform will be further evaluated in more complex environments with multiple users to assess its effectiveness and adaptability in realistic multi-tenant scenarios as well as in traditional controlled environments for existing hardware and software IP Core cryptography implementations (mostly Public Key cryptography). Furthermore, since PSCAs are often difficult to fully mitigate by allowing an attacker, in many cases, to eventually succeed with sufficient time and resources, the platform will also focus on analyzing the severity of vulnerabilities over time. By incorporating time-based standards, the Dynamic Hardware Analyzer will help quantify how quickly a system could be compromised, offering critical metrics to prioritize security measures and assess the practical risks of such attacks. This quantification will be consolidated in severity level metrics that resemble traditional risk analysis metrics in the relevant research literature. Task 3.3 will be collaborating with the RESCALE use-case pilots (especially the PST pilot) to identify how the Dynamic Hardware Analyzer can be evaluated by assessing IP Cores on the IoT platform or cloud infrastructure of the pilots.

References

- [1] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 13–28, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [2] Christos Diktopoulos, Konstantinos Georgopoulos, Andreas Brokalakis, Georgios Christou, Grigorios Chrysos, Ioannis Morianos, and Sotiris Ioannidis. Assessing the effectiveness of active fences against scas for multi-tenant fpgas. In *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, pages 391–396, 2022.
- [3] Eduardo Ferrufino, Luke Beckwith, Abubakr Abdulgadir, and Jens-Peter Kaps. Fobos 3: An open-source platform for side-channel analysis and benchmarking. In *Proceedings of the 2023 Workshop on Attacks and Solutions in Hardware Security, ASHES '23*, page 5–14, New York, NY, USA, 2023. Association for Computing Machinery.
- [4] Apostolos P. Fournaris, Athanassios Moschos, and Nicolas Sklavos. *Side Channel Assessment Platforms and Tools for Ubiquitous Systems*, pages 147–163. Springer International Publishing, Cham, 2021.
- [5] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. Are cloud fpgas really vulnerable to power analysis attacks? In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1007–1010, 2020.
- [6] 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)*, pages 1–8, 2019.
- [7] Yanning Ji, Ruize Wang, Kalle Ngo, Elena Dubrova, and Linus Backlund. A side-channel attack on a hardware implementation of crystals-kyber. In *2023 IEEE European Test Symposium (ETS)*, pages 1–5, 2023.
- [8] Xiangjun Lu, Chi Zhang, Pei Cao, Dawu Gu, and Haining Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):235–274, Jul. 2021.
- [9] A. Moschos, A. P. Fournaris, and O. Koufopavlou. A flexible leakage trace collection setup for arbitrary cryptographic ip cores. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 138–142, Los Alamitos, CA, USA, may 2018. IEEE Computer Society.
- [10] Guilherme Perin, Lichao Wu, and Stjepan Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):828–861, Aug. 2022.
- [11] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Computing Surveys*, 55(11), feb 2023.

- [12] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Jingdian Ming, Chengbin Jin, and Huizhong Li. Practical public template attacks on crystals-dilithium with randomness leakages. *IEEE Transactions on Information Forensics and Security*, 18:1–14, 2023.
- [13] Junichi Sakamoto, Kazuki Tachibana, and Tsutomu Matsumoto. Application of profiled analysis to adc-based remote side-channel attacks. In *2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 115–121, 2023.
- [14] Falk Schellenberg, Dennis R.E. Gnadt, Amir Moradi, and Mehdi B. Tahoori. An inside job: Remote power analysis attacks on fpgas. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1111–1116, 2018.
- [15] David Spielmann, Ognjen Glamocanin, and Mirjana Stojilovic. Rds: Fpga routing delay sensors for effective remote power analysis attacks. Cryptology ePrint Archive, Paper 2023/043, 2023. <https://eprint.iacr.org/2023/043>.
- [16] Mark Zhao and G. Edward Suh. Fpga-based remote power side-channel attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 229–244, 2018.