REASSURE

Deliverable D1.4

Final portfolio of best methods and improved evaluation techniques

<table>
<thead>
<tr>
<th>Editor:</th>
<th>V. Banciu (Riscure), I. Buhan (Riscure)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable nature:</td>
<td>R</td>
</tr>
<tr>
<td>Dissemination level: (Confidentiality)</td>
<td>PU</td>
</tr>
<tr>
<td>Delivery date:</td>
<td>March 31, 2020</td>
</tr>
<tr>
<td>Version:</td>
<td>1.0</td>
</tr>
<tr>
<td>Total number of pages:</td>
<td>42</td>
</tr>
<tr>
<td>Keywords:</td>
<td>side channels, leakage detection, leakage exploitation, deep learning</td>
</tr>
</tbody>
</table>

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 731591.
Executive summary
This document describes the best techniques and methods created or improved by the REASSURE consortium and aimed at optimizing high-security vulnerability assessment. This deliverable describes the steps for which interesting positive results were found. The techniques are organized according to the steps of our evaluation framework, which is the core of our approach to optimize and streamline evaluation methodology.

Section 2 covers the methodology and best practices for building a measurement setup. Naturally, the choice of equipment and the setup parameters directly impact the outcome of the analysis. However, additional tests should be performed in corner cases of the proper working conditions of a device. An improvement of up to \( \times 100 \) is reported when comparing the best/worst-case scenarios.

Section 3 assesses whether low-cost/low-power processors can potentially level classical infrastructures from a cost-efficiency point of view. The considered distributed leakage detection method is based on the central moments method. The two setups that are compared in this case are a cluster of low-power ARM CPUs on one hand, and a High-Performance Computing (HPC) infrastructure on the other. Experiments indicate that the former setup can leverage a parallel architecture in an industrial environment. From a practical point of view, the speed-up can be more than \( 100 \times \) compared to a fully serial implementation, or around \( 10 \times \) compared to a multi-threaded implementation.

Section 4 is dedicated to the application of neural networks for side channel analysis and describes tools that give an insight into what the neural network is learning, improving explainability and as a happy consequence generalization. The later can be achieved by accurate hyperparameter selection and we show that using the mutual information transferred to the last layer we can improve attack accuracy by a factor of up to \( 30\% \). A second improvement to generalization is achieved by the usage of ensembles, which can lead to up to a 40\% improvement of attack performance compared to using the best model alone.

Section 5 explores multi-channel classifiers which prove useful in cases where the leakage model is unknown and it shows evidence of improvements of one order of magnitude.

Section 6 compares the efficiency of profiled linear regression attacks (LRA) with Scatter, a new technique which aims to improve attacks in the case of misaligned traces. Overall, LRA always reaches a success rate close to 1, while Scatter often fails, even with a high number of traces. When both reach a success rate close to 1, Scatter requires at least twice the number of traces.
## List of authors

<table>
<thead>
<tr>
<th>Company</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCL</td>
<td>D. Bellizia</td>
</tr>
<tr>
<td>UCL</td>
<td>S. Duval</td>
</tr>
<tr>
<td>UCL</td>
<td>W. Wang</td>
</tr>
<tr>
<td>NXP</td>
<td>V. Verneuil</td>
</tr>
<tr>
<td>NXP</td>
<td>P. Kulkarni</td>
</tr>
<tr>
<td>Riscure</td>
<td>V. Banciu</td>
</tr>
<tr>
<td>Riscure</td>
<td>I. Buhan</td>
</tr>
</tbody>
</table>
Revision history

<table>
<thead>
<tr>
<th>Revision number</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>30 March 2020</td>
<td>First version of document</td>
</tr>
</tbody>
</table>
# Contents

**List of authors**  
3

**Revision history**  
4

1 **Introduction**  
7

2 **Measurement setup evaluations and techniques for power analysis**  
8

   2.1 Example of Measurement Setup  
8

   2.2 Target and Experiment Conditions  
9

      2.2.1 Platforms  
9

      2.2.2 Setup Parameters  
10

      2.2.3 Investigation Space  
11

   2.3 Probing Techniques  
13

      2.3.1 Shunt Resistor  
13

      2.3.2 Tektronix CT-1  
14

      2.3.3 FPGA  
15

      2.3.4 ARM Processor  
16

      2.3.5 Impact of Setup Parameters in a Multivariate Setting  
18

   2.4 Conclusion  
19

3 **Distributed Leakage Detection**  
20

   3.1 ARM cluster benchmark  
20

      3.1.1 Compute Module  
20

      3.1.2 Disk  
21

      3.1.3 Operating System and Libraries  
21

      3.1.4 Distributed Memory Benchmark  
21

   3.2 Platform comparison  
22

   3.3 Conclusion  
23

4 **Deep Learning for Side Channel Analysis**  
24

   4.1 Best practice: Salient Features Identifications  
24

      4.1.1 Proposed Method  
25

      4.1.2 Main results on ASCAD Database  
25

   4.2 Best practise: Improving Generalization  
26

      4.2.1 Main results on ASCAD Database:  
28

   4.3 Best practise: Preventing Over-fitting  
28

      4.3.1 Main Results on ASCAD dbabase  
31

   4.4 Conclusion  
31

5 **Multi-Channel based Deep-Learning Analysis**  
33

   5.1 Dependency of DL-based analysis to the soundness of the leakage model  
33

   5.2 Constraints of the multi-channel setup  
33

   5.3 Performance  
34

   5.4 Conclusion  
34

6 **Analysis of Leakage Exploitation Methods for Misaligned Traces**  
35

   6.1 On-the-fly linear regression  
35

   6.2 The Scatter transform  
35

   6.3 Setting #1: a simulated shuffled implementation  
36

      6.3.1 Experiments and results  
36

   6.4 Setting #2: a concrete jittery implementation  
37

   6.5 Conclusion  
37
7 Conclusion

7.1 Deep Learning

7.2 Multi-Channel
1 Introduction

In this document, we give an overview of our contributions and improvements with respect to methods that have been proposed over the past decades and are aimed at optimising the evaluation of side-channel security. We structure this report in accordance to the steps or components that we have identified as the backbones of evaluations in Deliverable D1.1. These steps or components are:

1. **Measurement and preprocessing.** This step provides the adversary/evaluator with leakages (e.g., the power consumption or electromagnetic radiation of a chip, or their simulation in case simulated analyses are considered) based on his input control, and possibly performs data-independent preprocessing in order to improve the quality of these measurements. We summarise our key findings regarding developing sound setups in Section 2.

2. **Leakage detection and mapping.** In leakage detection, the adversary/evaluator aims to detect the presence of any data-dependent leakage (independent of whether this data-dependency is exploitable in a realistic attack). Leakage mapping further aims to connect the detected samples to specific operations performed by the target implementation. We studied leakage detection from the viewpoint of efficiency in this work package, and summarise our best implementation methods in Section 3.

3. **Leakage processing.** In this last step, the adversary/evaluator aims to use the leakages in order to perform an attack (e.g., a key recovery). This step can be subdivided into modelling, information extraction and information exploitation, but methods can blur boundaries. Our research focussed predominantly on deep learning in this context. In Section 4 we explain our novel technique to analyze salient features. Subsequently, the usage of ensembles to improve model generalization in profiled side-channel analysis is analysed. Lastly, a new metric based on mutual information to identify the best epoch during training in the context of profiled side-channel attacks is introduced. Section 5 covers multi-channel based deep learning analysis, a new method which has shown promise in practice already. Finally, Section 6 studies the efficiency of a newly proposed method called Scatter for leakage exploitation of multivariate distributions and compare it with the state-of-the-art method of linear regression analysis.

This deliverable is based upon deliverable D1.2 and tightly connects to deliverable D1.3, which details evaluation strategies that work “backwards” (or top down) starting from a worst case adversary. For the sake of readability and conciseness we do not repeat the methods employed in D1.3 in this deliverable.
2 Measurement setup evaluations and techniques for power analysis

Whilst the design of countermeasures against Side-Channel Analysis (SCA) attacks and the development of secure schemes have grown quite fast, the analysis of good methodologies for testing devices had suffered a lack of investigation and proper definitions. Clearly, the testing part of a secure device can take a large percentage of the time-to-market, since this aspect of the process is intrinsically time-consuming and not trivial, especially for non-experts. Thus, it represents a non-negligible cost in this sense. Emerging sectors as Internet of Things (IoT) could potentially suffer from testing expenses, and they often do in-house validation which may lead to overestimation of the concrete level of security of products due to lack of expertise in the field. Several standardisation bodies have proposed methodologies for testing secure devices against SCAs, which can be cumbersome to follow, or in other cases, the real level of security one can test using those standards could be totally out-of-date (e.g. ISO/IEC 17825:2016). A first step in the formalisation of measurement methodology has been done in ISO/IEC-20085-1 and ISO/IEC-20085-2, describing some aspects of measurement setups, such as a guideline for calibration, and stating some useful definitions (e.g. univariate, multivariate and horizontal attacks) for evaluating secure devices.

Measurement is the initial step of the framework, defined by the REASSURE consortium in Deliverable D1.1, having the important role of producing digital signal representations of side-channel observations while a device is performing some operation involving sensitive or secret data. In this section we discuss in more detail the components and parameters of real measurement setups for power analysis, comparing their impact on the outcome of some key aspect of the evaluation on a real-world device. Several intuitions herein discussed are also suitable for other side channels, such as electromagnetic emissions.

2.1 Example of Measurement Setup

A typical measurement setup for power analysis is depicted in Figure 1. The device under test (DUT) is supplied with a power supply voltage level $V_{DD}$, which, in general, is chosen to ensure correct functionality of the DUT itself. A probe converts the current into a voltage signal that the DUT absorbs during the execution of the cryptographic operation. In the case of power analysis, the current absorbed by the DUT is the side-channel observation the evaluator wants to investigate for sensitive data-dependency. The voltage signal is captured by a digital storage oscilloscope (DSO), which has the role to sample and digitize the probed signal. The output signal of the probe can be amplified and/or filtered prior to sampling, in order to enhance the signal strength and filter off unwanted harmonic contents and noise.

---

1 If the operation that takes place inside a device is randomly shuffled or jittered in time/frequency, it is automatically certified with maximum level of security, even if it is still vulnerable to some attack strategies.

2 The usage of out-of-specification power supply voltage level can lead in faulty conditions that may be exploited also (e.g. Fault Attacks), or may damage the DUT.
Figure 2: FPGA implementation of the 32-bit AES-128 encryption core (a) and a power trace example (b).

2.2 Target and Experiment Conditions

2.2.1 Platforms

In the context of IoT, cryptographic primitives are implemented adopting software implementations as well as hardware accelerators, depending on the constraints and needs of the specific application. In our investigation, we have performed a thorough analysis and comparison of different sets of parameters for evaluating AES-128 implementations on both hardware and software platforms, in order to cover both scenarios. For the sake of clarity and to get better intuitions from the comparison between different setup settings, we have adopted only unprotected implementations of the AES-128 block cipher.

**Hardware Implementation - FPGA.** A 32-bit data-path of the AES-128 has been implemented on the Xilinx Spartan-6 LX75 FPGA mounted on the Sakura-G board. The architecture, shown in Figure 2(a), is an iterative loop implementation, where the round computation has been split into two parts. The AddRoundKey has been implemented using 32 2-input XOR LUTs. Only 1/4 of the complete SubsLayer has been implemented, which corresponds to 4 AES S-boxes, and only one MixColumns block has been implemented of the 4 required for a full-parallel design. The ShiftRows operation is obtained in the way we map on the 8-bit FIFO memories each byte of the state, thus not needing combinational logic to be performed. The KeyScheduler has been implemented similarly, adopting a 32-bit data-path also. The round function of the AES-128 is realized in a two-stage pipeline, which means that 5 clock cycles are needed to perform a full round computation.

It has to be noticed that to minimize the noise contribution of non-cryptographic circuitry, the generation of the clock and the SPI interface circuitry for communicating with the PC have been implemented on the control Xilinx Spartan-6 LX9 FPGA, also available on the Sakura-G board and connected with the target LX75. Previous experiments, where the clock generation and PC interfacing along with the AES implementation have been integrated on the target LX75 only, have led to SNR reduced of a factor 2. A trigger signal is generated one cycle before the beginning of the encryption, which is kept at logic value ‘1’ up to the end of the cryptographic operation, to minimize unwanted transient effect on the recorder power trace due to IO activity.

**Software Implementation - ARM Cortex-M0.** The software implementation we have adopted is a simple naive C implementation of the AES-128 block cipher. In this implementation, the round keys are precomputed before the execution of the AES. A trigger signal is generated before the starting of the computation of the AES’s rounds in order not to generate noise during the measurement, as discussed previously. As a physical platform, we have used an STMicroelectronics STMF030R8T6 microprocessor, based on an ARM Cortex-M0 architecture, mounted on a STM32F0 Discovery board. Similarly to the FPGA case, a trigger is generated on one of the GPIO pins at the beginning of the precomputation of round keys, and it is kept at logical value ‘1’ throughout the whole encryption. The board is connected to the PC through a UART-to-USB converter.
2.2.2 Setup Parameters

During the preparation of an experiment, several conditions and parameters have to be taken into account. The number of parameters to check can be very large, regarding the equipment used and the case study. It has to be noted that in some cases it is not trivial to reach an optimal set of parameters to perform an experiment, but “rules of thumb” can be applied in order to minimize the occurrence of systematic errors. In this section we report some practical guidelines useful in the design of a measurement setup for power analysis experiments, that could be intended as a baseline for other side-channels.

Sampling Rate and Analog Bandwidth. The digital oscilloscope model has to be chosen in order to guarantee that it is possible to observe the expected leakage within a certain accuracy in time (and in frequency) and in amplitude (vertical resolution). Assuming that the adversary has some knowledge on the clock frequency $f_{clk}$ used by the DUT, we can expect that the information leakage the evaluator would like to exploit/detect is contained in the bandwidth $[0, n f_{clk}]$, where the coefficient $n \geq 1$ depends strongly on the device under test. To be able to correctly capture and record the observed side-channel we must ensure that the digital oscilloscope can fulfil the following requirements:

- The sampling rate $f_s$ has to be chosen higher than the Nyquist frequency $f_N$ regarding the expected information leakage bandwidth. For simplicity, let us assume that the bandwidth the evaluator is interested in is $[0, f_{clk}]$:

$$f_N = 2 \cdot n \cdot f_{clk} < f_s$$

It is good practice to adopt a sampling rate that would provide at least an oversampling factor of 4 compared to the minimal Nyquist theorem [13]:

$$f_s \geq 4f_N$$

Oversampling is commonly adopted in many fields, in order to provide the possibility to reduce noise by using post-processing techniques (e.g. low-pass filtering).

- The analog bandwidth of the oscilloscope (that is mainly devoted to its front-end circuitry) has to be large enough to ensure that the relevant information leakage in the observed side channel is not filtered off before being sampled:

$$BW_{DSO} > f_N$$

In DSOs the analog bandwidth is always smaller than the maximum sampling frequency, in order to guarantee some minimal oversampling ($f_{s,max} \approx 4BW_{DSO}$ is very common). Usually, both sampling rate and bandwidth of a modern oscilloscope can be reduced from their maximum values and many configurations are often possible. The latter can be reduced internally or externally, by using properly designed and matched analog filters. This aspect can be very important to improve the electrical characteristics of the observed side-channel, as shown in [31]. In electronics, the following expression is often used to provide a theoretical upper bound for the (electrical) signal-to-noise ratio $SNR_{elec}^{upper}$ after sampling for analog-to-digital conversion [13]:

$$SNR_{elec}^{upper} = 6.02dB \cdot N_{bit} + 1.74db + 20log_{10} \left( \frac{f_s}{2BW} \right)$$

where $N_{bit}$ is the number of bits used for the quantization and $BW$ is the analog bandwidth at input of the converter (the front-end of the oscilloscope in our case). Hence, it is clear that the usage of higher sampling rate along with reduced bandwidth is, in general, beneficial to improve the quality of the measurement. On the other hand, we must consider that using a higher sampling rate implies that the memory requirement to store the same time window increases accordingly. The bandwidth reduction with analog filtering provides a good trade-off between memory requirement and improvement factor. As expected, the number of bits used to represent digitally the captured side-channel emission impacts strongly on the quality, and, also for $N_{bit}$, the higher is the better. For a better understanding of Equation 4 and the underlying assumptions, we advise the following reference [13].
### Power Supply Unit

In power analysis measurement setups, the power supply unit (PSU) has to be chosen carefully. In fact, since the aim of an evaluator is to exploit data dependency in noisy power traces, the PSU has to provide a clean and stable voltage (concerning also the load’s requirements), without significant temperature drift and noise. Usually, the noise coming from a power supply has a Gaussian distribution in the amplitude domain, where the mean of this distribution is the desired power supply voltage. We must consider the fact that if we measure the current absorption after the power supply unit, the intrinsic noise of the PSU will likely couple with the measurement equipment, increasing the overall noise of the current trace thus increasing the minimal sampling complexity to reach a certain outcome.

As can be seen on Table 1 comparing the Sakura-G’s on-board power supply (based on the low-noise LT3083 linear regulator chip) and a benchtop Tenma 72-8695 PSU under two different load conditions, it is clear that the standard deviation on a the selected DC voltage is dramatically different. In the optimistic case of 50Ω load, the Sakura-G’s on-board power supply provides 1/3 of the standard deviation provided by the Tenma power supply. This indicates clearly that the first one tends to be less noisy, and thus it represents a better choice compared to the Tenma as power source for power analysis. As a cautionary remark, we have performed this test with the core power supply of the target FPGA of the Sakura-G disconnected. In addition, the output of the Sakura-G’s on-board power supply is loaded by a 1kΩ resistor, to ensure minimal load to the regulator when the FPGA is not powered.

### Impedance Matching

Among bad practices in the design of a measurement setup, impedance mismatch can be considered the most frequent. In electronics, the impedance matching is a fundamental aspect of the design, as well as the accuracy of testing procedures. The impedance matching changes role and definition according to the application, and more importantly the bandwidth and frequency of signals to work with. For the sake of clarity, we exclude the formal definition of impedance matching and its diversification according to the application. The impedance matching is strongly needed to ensure signal integrity along the measurement chain. In fact, the quality of a high-speed signal (which is the case of side-channel signals) available at the front-end of the oscilloscope depends strongly on the fulfilment of this condition. In general, impedance matching requirements of high-frequency equipment, such as amplifiers, cables, probes, etc., are specified in technical datasheets and user manuals. A common value for impedance matching is 50Ω, usually adopted since it represents a good trade-off between transmission lines (such as cables) attenuation factor and signal reflection coefficient. In our opinion, the adversary/evaluator should provide impedance matching at each interface of the measurement chain, in order to guarantee maximal signal integrity.

### 2.2.3 Investigation Space

To show the impact of the choices the adversary/evaluator can make on the outcome of a security evaluation of devices implementing cryptographic primitives, we have chosen to build an investigation space upon the following setup parameters:

- power supply voltage;
- clock frequency;
- sampling rate;
- probing method.

---

<table>
<thead>
<tr>
<th>Picoscope 5244B Impedance (load)</th>
<th>Sakura-G’s LT3083-based PSU (1.2V)</th>
<th>Tenma 72-8695 PSU (1.2V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50Ω (R&amp;S HZ22 adapter)</td>
<td>$\sigma_{DC} = 166\mu V$</td>
<td>$\sigma_{DC} = 499\mu V$</td>
</tr>
<tr>
<td>1MΩ (native)</td>
<td>$\sigma_{DC} = 170\mu V$</td>
<td>$\sigma_{DC} = 1.15\text{mV}$</td>
</tr>
</tbody>
</table>

Table 1: Comparison of standard deviation on the DC voltage output of Sakura-G’s on-board power supply (LT3083-based) and Tenma 72-8695 benchtop PSU on two different load conditions.

---

3As specified in the datasheet of the LT3083 regulator [12].
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>FPGA</th>
<th>ARM Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD}^{PSU}$</td>
<td>{0.8,1.2,1.4} V</td>
<td>{2.0,3.0,3.5} V</td>
</tr>
<tr>
<td>$f_{clk}$</td>
<td>{1,6,24} MHz</td>
<td>{4,8,16} MHz</td>
</tr>
<tr>
<td>$SR$</td>
<td>{62.5,125,250} MS/s</td>
<td>{62.5,125,250}</td>
</tr>
</tbody>
</table>

Table 2: Comparison of standard deviation on the DC voltage output of Sakura-G’s on-board power supply (LT3083-based) and Tenma 72-8695 benchtop PSU on two different load conditions.

Figure 3: Effect of different power supply voltage values on power traces, collected on the Spartan-6 FPGA running at 1MHz.

Table 2 provides a summary of the cases we have investigated for both targets. In the FPGA case, we will refer to passive probing when no preamplifier is used to increase the amplitude of the signal, and active probing otherwise. With the $R_{drop}$ series of experiments, we have used the Sakura-G’s on-board preamplifier, providing a $\times 10$ voltage gain with 300MHz of bandwidth, as reported on the user manual of the board [42]. For the inductive probing experiments, we have used a Rodhe&Schwarz HZ16 [41], providing 20dB of gain in [100kHz;3GHz] band and 4.5dB of noise figure. In all the experiments, we have used a Picoscope 5244B as digital storage oscilloscope, running at 12bit of vertical resolution, 200MHz of bandwidth and Rohde&Schwarz HZ-22 feedthrough 50Ω adapters, where needed to guarantee impedance matching. In the following, we briefly motivate the choices of the particular setup parameters we decided to investigate. For the ARM case we have limited the probing to the inductive probe, as we will discuss further in this document.

**Power Supply Voltage.** It is clear that if we aim to extract information for power consumption, power supply voltage represents a delicate “knob” for a power analysis setup. In fact, varying the power supply voltage not only will vary the overall power consumption of the device, but it will strongly affect its electrical behaviour. In fact, reducing the power supply voltage in CMOS-based circuits leads to an increase of propagation delay, limiting the maximum frequency at which the device can operate properly. More generally, as the voltage is reduced from the nominal condition, the propagation delay will increase following an inverse quadratic behaviour [40], and as a result, the peak of current absorption on a power trace will look smoothed and reduced in magnitude. On the contrary, with an increase of the power supply voltage the propagation delay is reduced, and the peaks of current absorption are shorter and stronger in magnitude.

**Clock frequency.** The clock frequency has a central role in determining some of the choices the evaluator would make designing the measurement setup. In general, the higher the clock frequency, the tighter the requirement for the overall bandwidth and time resolution he/she shall deploy to capture the side-channel correctly, as we have already stated in Section 2.2.2 and in [15]. As a counterexample, in [3] [5], the requirement
of the bandwidth and sampling rate is decoupled from the clock frequency, since the information leakage is pushed towards very high frequency in the Time Enclosed Protocol (>100MHz), even if the clock frequency is in the order of MHz.

2.3 Probing Techniques

The first block in the measurement chain is the probe, as shown in Figure 1. The probe has an important role in the overall experiment, since it represents the interface between the measurement equipment and the DUT, converting the power signal (or, in general, the side-channel signal) into a voltage to be sampled and digitized by a DSO. In this section we briefly recall two main probing techniques used in the context of power analysis.

2.3.1 Shunt Resistor

The most common and easy method to measure power consumption is to obtain the current absorbed by the DUT reading the voltage drop on a resistor in series with its power supply (it can be with the DUT’s ground pin or between the power supply positive lead and the DUT’s $V_{DD}$ pin). This method has some advantages:

- it is very cheap, since usually only a resistor is needed;
- provides very large bandwidth (limited by the quality and parasitic of the shunt resistor);
- the transresistance could be chosen in a range that allows the DUT to work \[27\].

The single-rail measurement (single probe) is based on the simple Ohm’s law, and the expression of the probed signal $v_{meas}(data,t)$ is given as follows:

$$V_{DD}^{DUT}(data,t) = V_{DD}^{PSU} - i(data,t)R_{drop} = v_{meas}(data,t)$$ \hspace{1cm} (5)

where $V_{DD}^{PSU}$ is the power supply voltage, $i(data,t)$ is the current absorbed by the DUT and $R_{drop}$ is the value of the series resistor. An exemplary case of this measurement technique is shown in Figure 4. A variant of this methodology is also described in IEC 61967-4 \[10\].

A first observation is that increasing the value of the resistor $R_{drop}$ we increase $v_{meas}(data,t)$, thus obtaining a stronger signal. On the other hand, we must consider that electronic devices can tolerate down to a minimum supply voltage in order to work properly within the design margins. In some cases, if $R_{drop}$ is too high, even a small current is able to provide an excessive voltage drop across the probing resistor, which may provoke the DUT to stop working. Regarding FPGAs and microprocessors, the value $R_{drop}$ may prevent the device from booting. As a consequence of Equation 5 the actual voltage supply of the DUT is strongly influenced by the value of $R_{drop}$ and by the absorbed current. These fluctuations could impact significantly on the quality of the measurements, as well as triggering some countermeasures that prevent the adversary to use dynamic power to retrieve sensitive information. It has to be noticed, authors in \[27\] have investigated the
role of $R_{\text{drop}}$ in amplifying leakage recombination in a masked implementation, highlighting how increasing its value can help in reducing the effective security order. Considering the signal strength, we must point out that a resistor is a source of white noise due to thermal effect, which may be dominant in some cases.

In literature, as well in some standards (e.g. IEC 61967:2018 [10], ISO-17825:2016 [15]), we can find several different placing options for the probing resistor, which can be sorted into two categories:

- resistor between the power supply and the DUT’s power pins/pads;
- resistor between the DUT and the ground pins/pads.

In modern electronic devices, it is not rare to have multiple power pins as well as multiple ground pins, that could make the power measurements usually non-trivial due to the fact that the delivered voltages have different purposes (e.g. I/O powering, analog references, digital powers, etc.). Of course, the bigger the chip size, the higher the number of power pins. Probing through ground pins or traces can be non-trivial because the ground may be shared among several power distribution networks (PDNs) or it can be harder to access individually a target pin for a specific PDN (e.g., QFN packages with central ground pad). As possible alternative, differential probing (using two probes and a common ground) may be used on a shunt resistor to improve the electrical signal-to-noise ratio (up to $\times 4$), which is formally a benefit for the measurement quality, decreasing the time and the overall cost of the evaluation/attack. Of course, the adoption of differential probing requires using two channels on the oscilloscope or using an external differential amplifier with a single-ended output.

### 2.3.2 Tektronix CT-1

The Tektronix CT-1 [47] is a wide-band inductive probe that acts as a current-to-voltage converter taking advantage of mutual inductance between two coupled circuits. The probe operates in the range $(25kHz;1GHz)$, providing a $TZ_{CT1@50\Omega} = 5mV/mA$ transresistance when terminated into $50\Omega$ (the probe has an internal $50\Omega$ termination resistor to reduce reflections), as shown in Figure 5a. Basically, the CT-1 acts as a current transformer, where the input circuit (the line the evaluator wants to probe) acts as a primary coil source, as shown in Figure 5b. The measured voltage $v_{\text{meas}}$ is expressed as follows:

$$v_{\text{meas}} = i(data,t) \cdot TZ_{CT1@50\Omega}$$

The oscilloscope is connected to the secondary coil of such a transformer, allowing to monitor the current flowing in the primary coil. The probed line and the oscilloscope are galvanically isolated, since there is no electrical connection between them. On the other hand, the mutual inductance provokes a small inductive parasitic on the probed circuit. In order to mitigate this effect, we use not to overwind the probed line in the hole-spot of the CT-1, which can actually increase the signal (while decreasing the bandwidth). Ensuring that the lead wire that is used to monitor the current absorbed by the DUT is short (in most cases, $\sim 5cm$ is sufficient), the introduced parasitic is practically negligible, ensuring a clean probing of the power consumption.

Compared to the shunt resistor technique, the inductive probe allows to have less interaction of the measurement equipment with the DUT, since no significant voltage burden is required to measure the AC current.
signal the adversary/evaluator would like to exploit/detect. On the other hand, with this probing technique it is not possible to monitor static power consumption, since low frequency signals (DC up to 100kHz) cannot be detected by the CT-1.

2.3.3 FPGA

In the FPGA set of experiments, we have targeted byte 5 at the first round of the first S-Box of the computation of the AES-128 algorithm, making no assumption on the power model. For each experiment, $4 \times 10^6$ traces have been collected. In total, we have performed 108 experiments on FPGA among the investigation space aforementioned in Table 2.

To fairly compare the performance of the various measurement setup settings, we have used the SNR as metric, as also used in [21] to compare different acquisition campaigns. In the following analysis, we have used $4 \times 10^6$ traces from each experiment to compute the SNR. In Figure 7, we show the SNR versus time samples for the four probing methods operating at 1.2V and collected at 250MS/s. It is clear that different probing methods provide different “shapes” of the SNR (as well as the $v_{\text{meas}}(data,t)$). In Figure 7a-7b, the inductive probing provides better results than the shunt resistor method, up to a gain of $\times 2$. It has to be noted also that in these two cases, the inductive probe exhibits several peaks also after the main one. This effect is more evident in Figure 7a and it tends to disappear as the clock frequency increases. In the 24MHz case (Figure 7c), the inductive probe does not exhibit any additional peak, and the SNR curve appears very smooth due to overlap between clock cycles. On the contrary, experiments performed with the shunt resistor method exhibit the opposite trend. In addition, we can notice that as the clock frequency increases, the peak value for the inductive probe tends to decrease. A possible explanation to this effect is that the frequency response of the CT-1 probe is strongly affected by the saturation of the magnetic core, as shown in Figure 5a. In our case, we do not have a practical increase of the DC current absorbed by the FPGA as the clock frequency increases (which is anyway counterintuitive). In our opinion, the higher power consumption due to higher clock frequency provokes a strong shift of the power supply voltage, which can be seen, in fact, as a DC shift, which thus tends to reduce the linearity of the probe and its transimpedance gain $TZ_{CT1@50\Omega}$. This effect is observable with both shunt resistor and inductive probe (see Figure 8a), but affects stronger the latter one. In the shunt probe case, clock cycles are still clearly visible, while for the inductive probe it is harder to properly distinguish them, and only the “round grouping” can be spotted without careful inspection. It has to be noted that the power distribution network on the target device acts as a low-pass filter, due to parasitic effects (mostly, but not only, capacitive and resistive) [24]. Hence, at some frequency (which is typically device- and technology-dependent) the current peaks of successive clock cycles overlap, as clearly seen in Figure 8, and adjacent clock cycles are not independent anymore.

In Figure 9, a heatmap of the maximum SNR we obtained among the 108 experiments is depicted. The highest peaks of SNR have been found in experiments adopting the Tektronix CT-1 as probe and the R&S HZ16 preamplifier, which, in average, provided best results, with a maximum of $30 \times 10^{-3}$ of SNR, in the setting $\{1.2V,1MHz\}$ for all sampling rates. Regarding the experiments with the FPGA running at 24MHz, again, the inductive probe outperforms the shunt resistor method in the majority of cases, as also shown in Fig. 7. Compared to slower clock frequency experiments, 24MHz cases provided lower SNR, with a best case of $9.9 \times 10^{-3}$ with $\{1.2V,24MHz,250MS/s\}$ and active inductive probing. It is interesting to notice that the SNR does not increase in all the cases with the power supply voltage for the setting, as it could be intuitive from

Figure 6: Tektronix CT-1 simplified equivalent circuit.
Figure 7: SNR versus time samples of the AES-128 implementation on the Xilinx Spartan-6 FPGA, running at 1MHz (a), 6MHz (b) and 24MHz (c) at 1.2V. The four probing methods are represented with different colors: the passive and active probing curves with \( R_{\text{drop}} = 2\Omega \) are depicted in blue and in orange respectively, while the passive and active probing curves with the Tektronix CT-1 are depicted in green and red respectively.

electronic literature (in general, the dynamic power grows linearly with the power supply voltage). In almost all the cases, a power supply voltage of 1.4V did not provide the highest SNR, while 1.2V experiments have, in average, led to best results. The only outlier we have found is the passive resistor method, where the result obtained with 1.2V and 1.4V were always comparable and stronger than 0.8V experiments. Possibly, this is due to the fact that the signal component of the SNR is dominant over the noise, also if the power supply voltage increases.

Figure 8: Average traces collected on the Xilinx Spartan-6 FPGA running at 24MHz (a) and 1MHz (b).

2.3.4 ARM Processor

In this case, the measurement setup is composed of the STM32F0 Discovery board, which hosts the target STMicroelectronics STM32F030R8T6 processor, powered by a Keithley SMU 2450 source/measurement unit [25] in voltage-source mode instead of the on-board linear regulator, connected to the JP2’s pin towards the target chip. Capacitors C18, C19, C20 and C21 have been removed to completely cut out the off-chip filtering circuitry (close-by the chip), maximizing the bandwidth for our measurement. In addition, the source clock is originated by an 8MHz quartz mounted in X2, and then manipulated by the on-chip clocking circuitry, to provide minimum jitter. To use the crystal as clock source, solder bridges SB16, SB17 and SB18 have been opened. It has to be noted that in this way we aim at minimizing the contribution to the overall noise of the loader/debugger chip also present on the board, that can be used as clock source too. The current absorption of the target has been measured using the CT1 probe, sensing the current provided by the Keithley SMU 2450 through a daughter board (via SMA connectors). To minimize reflections, we have also mounted a 50Ω SMD resistor on the daughter board between the input connected to the source and the ground.

The univariate analysis for the ARM case has been carried out on a smaller exploration space compared

---

\[ f_{\text{clk}}=24\text{MHz}, V_{\text{DD}}=1.2\text{V}, S=250\text{MS/s} \]

\[ R=2\Omega \]

\[ R=2\Omega+\text{AMPL} \]

\[ \text{Sakura} \]

\[ \text{CT-1} \]

\[ \text{CT-1+AMPL} \]

\[ \text{CT-1+AMPLHZ16} \]
to the FPGA case, but using the same intermediate variable for our analysis, also making no assumptions on the power model. For each case study we have collected 100k traces. In our experiments we have tweaked the voltage power supply of the device, going from the minimum allowed by the device to boot correctly 2.0V, up a maximum of 3.5V. As a remark, the voltage range declared as standard by the manufacturer is 2.4-3.6V [46]. Two sampling rates have been considered, 62.5MS/s and 125MS/s, while three clock frequencies have been chosen, 4MHz, 8MHz and 16MHz, in order to investigate no overlap, partial overlap and strong overlap between clock cycles in terms of current absorption. This effect is shown in Figure 10, where average traces for three different clock frequency are shown. As we can see, moving from 4MHz (a) to 16MHz (c), clock cycles overlap due to different responses of the internal power distribution network of the processor at different frequencies, which tends to naturally integrate the absorbed current, as discussed previously. Increasing the clock frequency, the contribution of individual clock cycles to the overall power consumption is less distinguishable, as observed in FPGA experiments with $f_{clk}=24MHz$.

In Table 3, the maximum SNR is reported for all evaluated cases. It has to be noted that the values are up to three orders of magnitude higher than the FPGA case. In general, hardware implementations offers a lower SNR due to specialized resources (and optimized in case of ASICs) for computations, which implies also smaller signal. Recently, Gao et al. [18] highlight how the generic architecture of some ARM processors may exhibit unwanted leakage also for masked implementations, that are not captured in common probing models (for which architecture-agnostic models of physical processors are often used).

In general, SNR evaluations from software implementations exhibit more peaks than hardware implementations, as shown in Figure 11. In fact, design limitations due to smaller data-path widths (8, 16 and 32bits for most IoT and smart-card applications) and generic microarchitecture/ISA imply that the same intermediate variables may be manipulated several times along the computation. It is interesting to notice that the
### Table 3: Maximum SNR obtained from the STM32 processor in different setup conditions over 1 byte.

<table>
<thead>
<tr>
<th>Sampl. Rate</th>
<th>Clock Freq.</th>
<th>V&lt;sub&gt;DD&lt;/sub&gt; 2.0V</th>
<th>3.0V</th>
<th>3.5V</th>
</tr>
</thead>
<tbody>
<tr>
<td>62.5MS/s</td>
<td>4MHz</td>
<td>4.34</td>
<td>4.40</td>
<td>5.97</td>
</tr>
<tr>
<td></td>
<td>8MHz</td>
<td>3.94</td>
<td>3.67</td>
<td>4.68</td>
</tr>
<tr>
<td></td>
<td>16MHz</td>
<td>6.08</td>
<td>4.05</td>
<td>4.19</td>
</tr>
<tr>
<td>125MS/s</td>
<td>4MHz</td>
<td>5.11</td>
<td>5.15</td>
<td>5.13</td>
</tr>
<tr>
<td></td>
<td>8MHz</td>
<td>4.63</td>
<td>4.69</td>
<td>5.78</td>
</tr>
<tr>
<td></td>
<td>16MHz</td>
<td>11.35</td>
<td>5.52</td>
<td>5.89</td>
</tr>
</tbody>
</table>

Table 3: Maximum SNR obtained from the STM32 processor in different setup conditions over 1 byte.

Figure 11: SNR plots of experiments performed on the STM32 ARM processor, and collected at 125MS/s.

{16MHz,2.0V,125MS/s} setting offers the best result, providing a peak value of 11.35. SNR analysis has shown that moving from a sampling rate of 62.5MS/s to 125MS/s, a moderate gain can be found for all case studies. A possible explanation that given a certain fixed analog bandwidth, increasing the sampling rate allows to decrease the noise density spectrum in the bandwidth we observe. This effect is particularly evident at 2.0V for the 16MHz case.

### 2.3.5 Impact of Setup Parameters in a Multivariate Setting

Investigating the role of each samples in a trace, and how they are related to each other, represents a quite complex task. This aspect has already been reported in many works in literature, especially regarding the possibility to exploit jointly multiple samples in so-called horizontal attacks. Nowadays oscilloscopes can reach easily GS/s performance, and their cost is sensibly reduced compared to a decade ago, opening evaluators/adversaries to new challenges in exploiting side-channel observations. As a matter of fact, we need to consider that higher sampling rate is usually avoided in the side-channel community, since usually it is preferred to reduce instead of increase the number of samples per trace in order not to over-constrain the minimum memory and time requirements for an experiment.

On the other hand, we want to investigate if it is possible to take advantage of a higher sampling rate to

---

On the Picoscope 5244B the available bandwidth of the front-end is 200MHz, but it can be reduced to 20MHz by an internal filter. We did not make use of this internal filter, and we have performed all experiments for both targets using full bandwidth.
extract more information from the same time windows. A first intuition in this sense has been proposed in \cite{6}, where authors show how increasing the number of intra-cycle dimensions/samples in a practical scenario can theoretically lead to an increase of the information that can be extracted from power traces. Results on practical cases in \cite{6} have been discussed only in one setup setting, and for the sake of good evaluation methodology, multiple corner cases have to be taken into account.

2.4 Conclusion

The probe choice and all the other setup parameters impact on the outcome of the analysis, as shown through the SNR evaluation. Optionally, in the perspective of reaching a (close to) worst-case evaluation, additional tests should be performed in corner cases of the DUT’s proper working conditions, as shown for the SNR evaluations on both hardware and software implementations. It has to be noted that in the FPGA case, the SNR improvement we have from the worst to the best case is approximately $\times 30$, while for the STM32 experiments, the improvement between best/worst-case is approximately $\times 2$. Previous experiments on STM32 where the power supply was provided by on-board regulators and the source clock was generated by the on-chip RC-oscillator had given way worse results than the worst-case SNR we have shown in Table \ref{tab:snr}. Considering this, the best/worst-case improvement is in the order of $\times 100$. Hence, this setup investigation is clearly a useful approach in the direction of performing a calibration and fine-tuning of the setup, and to guide the evaluator to choose the best tools and settings to perform an experiment, similarly to what was reported in \cite{17}. We have also performed a multivariate information theoretic analysis on some selected cases that provided best results after SNR evaluation, considering as points of interest time samples corresponding to the maximum peak of SNR and its neighbouring points (within the same clock cycle). We have found that the role of correlated noise in a trace is very critical to extract information, depending on whether time samples we observe are related or not to the same intermediate. If the multivariate evaluation is performed on Non-Independent Operation Leakage (NIOL) samples (i.e. samples from the same cycles performing a single operation), correlated noise should be minimized to increase the information we can extract. To achieve this, we have found that lowering the supply voltage and increasing the sampling rate are effective in reducing the correlated noise between samples, especially for the FPGA case. Adopting the logarithm of the effective variance on standardised variables \cite{25} as metric to represent easily the correlation matrix characterizing a large number of samples (in other words, “how much” they are correlated) with a single number, we have observed that FPGA experiments performed with the active inductive probing benefit of lower voltage increasing this metric from $-2.33$ at 1.4V to $-1.26$ at 0.8V, as shown in Figure \ref{fig:correlation_matrix}. Considering the asymptotic mutual information (taken as the average between the hypothetical and the perceived information), we have found that for a 10-dimension multivariate analysis, 0.8V provided an improvement of $\times 2.25$ compared to its univariate counterpart, while 1.2V and 1.4V cases provided much smaller improvement. Similar conclusions can be drawn for ARM experiments, with different margins of improvement, since some of the overall setup parameters’ optimal values are strongly device-dependent.

Figure 12: Heatmap of correlation matrices for the AES running on the Spartan-6 FPGA at 6MHz, at 0.8V (a), 1.2V (b) and 1.4V (c).
3 Distributed Leakage Detection

Leakage detection is a major stage of side-channel evaluation, as detailed in the Deliverable 1.1. For evaluation labs and vendors, it represents therefore a very significant amount of the time and cost of a side-channel evaluation – roughly comprised between 30% and 70%, depending on the product being evaluated and the certification scheme. Optimizing the leakage detection performance is thus a key point to enable more efficient and cost-effective security evaluations.

Several leakage detection strategies have been discussed in detail in the Section 3 of the Deliverable 1.2 and we have presented implementations of distributed memory leakage detection computation using a High-Performance Computing (HPC) facility in an industrial context. Namely, two computation methods have been investigated: one based on central moments and one based on histograms. To suit industrial constraints, we ensure that measurements do not need to be repeated, therefore online formulas for the computation of the central moments are not considered. Also, the histogram-based leakage detection was abandoned due to its huge memory requirements.

In this section, we present our results on using a cluster of low-power ARM CPUs for efficient and cost-effective leakage detection.

3.1 ARM cluster benchmark

High-Performance Computing (HPC) infrastructures are usually based on fast and complex x86 processors, as historically CPU manufacturers have first focused on increasing the clock frequency and operation throughput of single cores. With the advent of low-cost/low-power processors designed for mobile devices, the question arise if such CPUs could beat the classical architectures from a cost-efficiency point of view. In other fields such as imaging and web servers, it was found that a cluster using many low-power ARM CPUs provided a better cost-efficiency than a few Intel server-grade CPUs [56, 58].

We have therefore investigated how suitable a low-power ARM CPUs cluster would be in the context of side-channel evaluation.

3.1.1 Compute Module

The compute modules are single boards with 64-Bit Quad Core ARM CPUs A53 running at 1.5GHz. Each module has a DDR3 SODIMM form factor (which can be connected to a baseboard) with build-in 2GB LPDDR3 RAM and a microSD slot. Compute modules can get extremely hot when doing number crunching on all cores. We used simple aluminium heat sinks and a USB fan to dissipate the heat, which lead to acceptable working temperatures. Figure 13 shows a single SoPine compute module.

![Figure 13: The SoPine compute module](image)

A PINE64 clusterboard can be used to cluster the single SoPine module. One clusterboard can host up to 7 SoPine A64 compute modules, expanding its functionality as a fully featured cluster server. The clusterboard has a built-in 8 Gigabit Ethernet port unmanaged switch. In this experiment, we used two clusterboards which were connected using a small 5-port switch. The switch was then connected to another SOC (a baseboard with graphic support) for server management. Figure 14 shows the clusterboard with seven compute modules installed.

---

7 The memory requirements of the histogram-based leakage detection could possibly be significantly improved by storing the histograms as sparse arrays; unfortunately this feature couldn’t be implemented in the project’s timeframe.
3.1.2 Disk

For the IO, each compute module is equipped with a 64GB microSD card. A directory on one of the modules is shared using NFS for all other compute modules. Moreover, an external hard drive is attached to the clusterboard and it is mounted for all the compute modules. The hard drive supports USB 3, but the compute modules only support USB 2.0 and therefore cannot take advantage of the USB 3 speed-up.

3.1.3 Operating System and Libraries

The Armbian GNU/Linux distribution is installed on the compute modules. It is based on Debian and does not basically support graphics. However, we found it quite easy to configure to act as an ARM server or client. We compiled Python 3.6.5 from source to run the python software. We used MPICH 3.2.1. as our MPI implementation compiled from scratch. The mpi4py library is also linked to the MPICH to run the distributed python application.

3.1.4 Distributed Memory Benchmark

Our test side-channel analysis framework is based on Python. The computing kernels are written in Cython to gain similar speed as compiled C/C++. The benchmark tests in this report are all done to compute central moments of a 1GB trace file. Figure 15 shows the results of the benchmark as speed-up versus number of cores. As it can be seen, the scaling is not fully satisfactory, e.g. with 50 cores the speed-up is about 23.8 only. One reason is that for this experiment the data is only read from one process and distributed to others. We could not use a shared memory buffer because the hard disk containing the trace file had only one connection to the IO process and did not allow multiple connections from other compute modules. Moreover, the network used here is slow compared to very fast network connections available in high-end clusters. Nevertheless, we could achieve a 50% efficiency on a low-cost platform, which demonstrates the feasibility of the computation distribution on an ARM cluster. Therefore we expect that a high efficiency (i.e. $\geq 90\%$) is achievable on a server-grade ARM cluster and with some more software optimisation.

---

$^8$The speed-up for a given number of cores $n$ is defined at the quotient of the computation time for 1 core by the computation time for $n$ cores. Therefore the speed-up for a single core is always 1 by definition. An ideal speed-up scaling would be linear, meaning that the computation with $n$ cores would be $n$ times faster than with a single core. In practice, a linear scaling is usually not achievable due to overheads that cannot be parallelised, but the aim is to get as close to it as possible.
3.2 Platform comparison

We have implemented the distributed leakage detection based on the central moments methods on two very different platforms: on one hand an industrial-grade HPC facility using Intel Xeon processors, high-speed network links between nodes, and fast RAID storage, as presented in the Section 4 of D1.2, and on the other hand the low-power ARM-based platform using consumer-grade components presented above. The major differences between the two CPU architectures are recalled in Table 4.

<table>
<thead>
<tr>
<th></th>
<th>Intel Xeon</th>
<th>ARM Corex A53</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Width</td>
<td>64 bit</td>
<td>32/64 bit</td>
</tr>
<tr>
<td>Issue Width (µOP/clock)</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Pipeline Depth</td>
<td>20-24</td>
<td>8</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>individual</td>
<td>shared 4 cores</td>
</tr>
</tbody>
</table>

Table 4: Intel Xeon v. ARM A53 architecture

The overall cost of the experimental 56-core ARM cluster was approximately 925 Eur (989 USD). In comparison, the cost of a single CPU workstation equipped with the Intel Xeon E5-2690V3 CPU (the one used on the HPC facility where we ran our tests) was approximately 5600 Eur at the same period. Comparing the performances on a single core, we measured that the leakage detection runs about 12.5 times faster on the Intel Xeon CPU. Assuming an ideal linear scaling that a more optimized implementation would tend to achieve, the 12-core Xeon E5-2690V3 would still be about 3.15 times faster than the ARM cluster under the same conditions (and assuming a 15% speed-up for the Intel Xeon to account for the extra virtual hyperthreading cores). Therefore, for a same level of performance the cost of the Intel Xeon platform is 92% higher than the one of the ARM cluster.

Clearly the above comparison is not totally fair, since we compare a low-cost equipment for the ARM cluster to high-end server-grade equipment for the Intel Xeon platform. Our comparison also does not use the last generation of processors. Moreover, the cost-efficiency ratio of the Intel-based platform would improve if we would consider a cluster with dozens of CPUs.

Nevertheless, this case study shows that it is feasible to assemble an ARM cluster for a low price and get competitive leakage detection performances compared to a classical x86-based solution. In general, we estimate that the cost-efficiency of a low-power CPU cluster is around 30% to 50% higher than the one of classical x86-based clusters, which is in line with other public results [2, 53].
3.3 Conclusion

As leakage detection becomes an increasingly time and resource-consuming part of side-channel evaluations, parallelised architectures represent the natural solution to carry out the required calculations and meet the tight evaluation time-frame constraints. In this respect, we have demonstrated that a distributed implementation using the Message Passing Interface (MPI) standard of the central moments-based leakage detection can leverage a parallel architecture in an industrial environment. From a practical point of view, the speed-up can be tremendous: more than 100× compared to a fully serial implementation, or around 10× compared to a multi-threaded implementation, using a relatively small cluster (4 to 16 CPUs).

Another parameter that impacts the cost-efficiency of distributed systems is the choice of the underlying CPU architecture. Until recently, the classical x86 architecture was a natural choice for HPC, but the developments of low-power CPUs in the last decades call this choice into question. Therefore, we have compared two clusters based on different types of CPUs to check what architecture provides the best cost-efficiency in the context of side-channel leakage detection. Our experiments show that low-power ARM CPUs are an efficient alternative to classical x86 architectures and can provide a significantly higher cost efficiency; we estimate that the order of magnitude of the potential improvement lies in the range of 30% to 50%.
4 Deep Learning for Side Channel Analysis

The adoption of deep neural networks in profiled side-channel attacks improves the capabilities of key recovery as demonstrated by several recent publications [8][26][39]. At the same time, deep neural networks are able to implement highly parametric solutions, which means that the configuration of such complex systems require the careful selection of hyper-parameters for each specific problem.

In side-channel attacks, every data set is collected from a different target of evaluation (TOE) (cryptographic hardware or software design). In consequence, every profiled attack that is mounted on the newly collected trace set requires a different optimization problem starting from training a deep neural network from scratch. A scenario when a trained neural network for target $A$ could be reused for a different target $B$ is very unlikely. The selection of hyper-parameters needs to be reconsidered based on the features presented in every side-channel trace set.

The large number of hyper-parameters makes the selection of hyper-parameters a very difficult problem. In the end, optimizing the hyper-parameters for a deep neural network in the context of side-channel analysis can be considered as a multi-objective problem. Due to the hyper-parametric solution, there will be several optimal solutions for the problem. Literature suggests the adoption of optimization algorithm such as genetic algorithms or reinforcement learning to find an optimal neural network. Even if such solution sound promising, these methods require a large amount of time and computation power, which could prove infeasible for side-channel evaluations. Related works on deep learning for side-channel analysis propose several neural networks configurations that are able to recover the target key in a profiled setting. For that, convolutional neural networks and multiple layer perceptrons are usually applied and the number of layers vary according to the number of traces. Therefore, these related works can be used as sort of guidelines for a feasible neural network configuration.

In this section we focus on the training optimization of deep neural network for profiled side-channel analysis. Our investigation is not focused on determining specific hyper-parameters, e.g., layer-wise configuration, but more generally on determining a suitable metric for side-channel analysis and implicitly the determination of the correct number of training epochs.

As the main goal is to train a deep neural network that is able to learn the leakage in side-channel traces, we propose a method to assess which samples in a trace set the network is considered for its predictions. These methods are usually related to sensitivity analysis or interpretability of neural networks. Our method identify what we called salient features in side-channel traces.

The main best methods we will be discussing in this section are:

- We propose a technique to analyze what are the learned side-channel samples from traces. This technique is salient features.
- We propose the usage of ensemble to improve model generalization in profiled side-channel analysis.
- We propose a new metric based on mutual information to identify the best epoch during training in the context of profiled side-channel attacks.

4.1 Best practice: Salient Features Identifications

In the context of side-channel analysis, researchers have focused on identifying which input features or trace samples are more relevant for the neural network predictions. Here, we define this technique as model assessment since we verify what input samples the neural network is learning during training in order to make its predictions.

Deep neural networks can implement highly complex models. For every trace set collected from a specific TOE, there will be different sets of hyper-parameters that define several local minima and a global minimum in the landscape (defined by the error/loss/cost function). The local minima can also be defined as being one of the optimal solutions for the current optimization problem. It is generally assumed that as soon as the learning model finds a local minimum, the neural network would be actually learning from data-dependent leakages. Ideally, the network should also discard non-leaking samples in its decision process. Of course, this task is accompanied by the correct selection of a leakage model. The task of making the model to converge to one of the local minima or, ideally, to the global minimum is difficult due to the large amount of possible
combinations of hyper-parameters. However, training metrics (accuracy, loss function) could indicate such performances during training and validation phases. The remaining question is related to what exactly the network has learned from side-channel traces.

If the task relies on the identification of what are the optimal values for two or three of the hyper-parameters, a grid search could cover a reasonable range of possibilities in a feasible amount of time. If more hyper-parameters need to be found, the usage of an optimization algorithm would be necessary, such as evolutionary algorithms or reinforcement learning. However, optimization algorithms can lead to unrealistic situations where the convergence to a good set of hyper-parameters is unfeasible within a reasonable amount of time. When evaluating the security of cryptographic implementations, leakage certifications or even security assessments on the manufacturer’s side, the available time and budget may not afford for a very expensive hyper-parameters search.

To clearly understand how a trained model is actually making the distinction between data-dependent samples from non-leaking samples, we need to appeal for model assessment techniques. Part of this model assessment can be the identification of input features (i.e., input samples in side-channel traces) that are decisive for the classification task. Some works [45] give directions in showing how information is conveyed through the hidden layers, and others try to explain how the features in the input data affect the activation in hidden layers [9]. Techniques that highlight what convolution layers learn from images (e.g., segmentation learning), enable the location of objects in input data. Some techniques, based on occlusion sensitivity, are useful in the direction of identifying how the convolution layers treat separate classes from the input data. For side-channel analysis, the works presented in [29], [23] and [50] provide good coverage of model assessment techniques.

4.1.1 Proposed Method

Once a neural network is trained, the training traces are provided to the network and, one by one, a neuron is activated with the higher value in the output or softmax layer is taken into account. This single neuron represents one of the classes defined in the input data set. For side-channel analysis, it is well-known that the number of classes is derived from the selected leakage model and the number of neurons in the output layer is equal to this number of classes.

A neuron in the output layer is fully connected to all the neurons in the previous layer through a set of connection weights. Furthermore, the activation value for each neuron depends on all their previous weight connections and activation values of the previous layer. Every connection between two neurons is defined by a weight value, which remains unchanged after the training stops as well as the bias value for every layer. The values of the input data (or input trace) are processed by all the neuron connections, producing different activation values for all the neurons inside the network. The main goal of training a neural network for side-channel analysis is to learn the parameters (weights and biases) in a way that network can distinguish input trace samples that represents data dependent leakage. If the neural network is able to create activation paths based on input samples that are actually points of interests (i.e., samples that contains data dependent information) we assume that our model is trained in a way that it is learning the actual leakages from training traces.

We are interested in identifying exactly the sample(s) from each input trace that the network is selecting as the most important one(s) for its classification task. By doing this assessment, we are able to observe whether the trained model is actually selecting leaking samples (points of interest). The proposed method can be applied to training data as well as validation data. When the method is applied to training data, it is possible to visualize what input samples the trained network is taking into better consideration for its predictions. By processing validation data, where the labels are known, it is possible to visualize the generalization capabilities as an additional information to the conventional metrics.

4.1.2 Main results on ASCAD Database

In [39] the authors provide a full description of the AES implementation that was used to generate the ASCAD database for side-channel traces and it is available on GitHub (https://github.com/ANSSI-FR/ASCAD). The authors suggest the usage of 50k traces for training and 10k traces for validation/test.

The implemented AES is protected with first-order masking countermeasure and the side-channel traces contain information of electromagnetic emanation related to the first round. The operations on sub-keys bytes 1 and 2 are not masked. Therefore, we select the trace interval containing 700 samples with respect to the
REASSURE Deliverable D1.4

Figure 16: Backward propagation path result (left) and correlation with $S_{box}$ output values for $k1$ (right).

Figure 17: Backward propagation path result (left) and correlation with $r[3]$ and masked $S_{box}$ output (right).

processing of the key byte 1. Figure 16 provides the correlation analysis for the S-box output (right) and results for the proposed salient features method. As we can observe, salient features indicates that the trained network is actually selecting the samples that are actually points of interest (high correlation) as the main input features. In this case, we can assume that the network is actually recognizing with a high precision the location of points of interest.

AES rounds from sub-key bytes 3 to 16 are masked and the corresponding mask values are also available for each trace in the database. This information enables the calculation of the masked S-box output value. Based on that, we select the interval representing the processing of sub-key byte 3 and compute the correlation for $r[3]$ (mask for $k3$) as well the masked S-box output value. Figure 17 shows these correlation results. Next, we train a convolutional neural network over these data set and also implement the model assessment with salient features method. The analysis is performed in a black-box scenario, where the training traces are labelled based on the Hamming weight of S-Box output without any masking into consideration.

Figure 17 also shows the results for the salient features on masked AES. Comparing correlation with salient features, it is possible to verify that the neural network considers the input samples related to the mask $r[3]$ and the masked S-box output as the main input or salient features. This confirms that the trained neural network is actually combining two or more samples for every trace as a second-order attack.

In conclusion, salient features detection helps in the optimization of neural network training because it indicates whether a neural network is being able to learn from actual leakages in profiled side-channel analysis.

4.2 Best practise: Improving Generalization

In this section, we provide results from an investigation that aims at improving deep learning generalization for profiled side-channel attacks. To deal with problematic generalization of trained deep neural network models, there are four main alternatives:

- **Data generation**: it includes the acquisition of more training traces from a device under control, where the key is known and the input and/or outputs can be controlled by the attacker.
• **Data augmentation:** to deal with misalignment and noise, one can modify the training traces in order to produce artificial traces by implementation the effects that are expected in the test data.

• **Regularization:** these are techniques that prevent the model from overfitting the training data. Dropout, L1, L2 and noisy (Gaussian) layers are some of the examples of regularizers that can be used for side-channel analysis.

• **Ensembles:** this is a technique that combines the results from multiple models in order to improve the test accuracy. Stacking, averaging, boosting and bagging are examples of techniques adopted for ensemble learning.

In this work, we focus on improving the generalization of deep learning-based profiled side-channel attacks with the usage of ensembles. Ensemble learning is a technique that combines the predictions from several models in order to reduce the generalization error. The expected result is that the prediction obtained from ensemble models is superior if compared to a single model. This is motivated by the statistical intuition that averaging measurements (or predictions) can lead to a more stable and reliable estimate because we reduce the influence of random fluctuations in single measurements.

Ideally, the hyper-parameters for each model should vary in order to learn different features from the same training set. If the models have configurations, it is very likely that the neural network will learn similar representations from the training set and provide similar classification results for the same test set. Another option, which we follow in this work, is to randomly split the training set into smaller sets and train a separate model for each one of them. In this case, the single models can have the same structure. On the other hand, we can build an ensemble of slightly different models that use the same training data, which can reduce the influence of random fluctuations in single models.

Here, the main goal of the ensemble learning application is to improve the distinguishability of the likelihood in the key recovery or test phase. When a deep learning-based profiled side-channel attack is successful, the main reason for that comes from the likelihood being larger for the correct key candidate. As a result, successful ensembles should increase the likelihood for the correct key candidate while averaging out the variations for the incorrect candidates.

Note however, that we are not assuming that ensemble learning is always better if compared to a single learning model. The main goal of this analysis is to demonstrate that the chances of success in terms of key recovery are higher for ensembles when compared to single models. For a very large training set, one expects that a complex deep neural network needs to be defined. This actually requires a careful selection of hyper-parameters and to try each new configuration may take a significant amount of time and computation power. Ensemble learning makes use of smaller training sets, which reduces the complexity for defining appropriate hyper-parameters. Moreover, because several models are combined into one, if few of the models contain hyper-parameters that do not provide good learnability from training sets, the fluctuations introduced by these models will be removed by having models that are actually generalizing. The experiments provided in this section demonstrate that ensembles are more likely to succeed if compared to single learning models for a specific amount of trained models.

Ensemble learning methods do not exclude the importance of correct hyper-parameters selection for each single model. The selection of these hyper-parameters can vary for each model (heterogeneous case) or remain the same (homogeneous case). The idea of using heterogeneous models is the possible removal of errors imposed by the wrong selection of hyper-parameters if a sufficient amount of models is trained and if most of the neural networks are configured with reasonable hyper-parameters (the hyper-parameters that were already proved to work well in side-channel analysis). Hyper-parameters search methods, like random search, grid search, Bayesian optimization or genetic algorithms, also provide solutions to define the best configuration for a neural network. However, these types of search would consider a large training set and several iterations (e.g., generations in the genetic algorithm). This could render the analysis impractical due to computational or time constraints. Ensembles can work on smaller training sets and they require a limited amount of models, which reduces the complexity. Main methods for ensemble learning include boosting, bagging and stacking. In this work, the implemented ensembles are more similar to bagging methodology. We decided to use ensembles with averaged probabilities. Therefore, the objectives are twofold:

• To verify the benefits of ensemble learning in side-channel analysis with ensembles are built from all trained single models;
- To verify the benefits of ensembles in side-channel analysis when ensembles are built from N best trained
  modes. The best models are selected based on the key rank for the validation set, as this metric provides
  more consistent results in comparison to conventional deep learning metrics (accuracy, loss or recall).

4.2.1 Main results on ASCAD Database:

Figure [18] provides results for the ensemble method. In this case, the same model configuration is trained 10
different times and the ensemble result is computed from all models and for the 3 best models. The results are
provided for different training sets of 10000, 25000, 50000 and 75000 traces.

Figure [19] provides results for ensemble computed from 10 models having different hyper-parameters. The
results are provided for different training sets of 10000, 25000, 50000 and 75000 traces. We see that for all
cases, the more traces are available during the attack phase, the better the performance of ensemble based
modeling. In our experiments, the most impressive results are obtained when using 25k traces for training at
data point 500. The best model per campaign reaches a success rate of 44%, while the ensemble from the three
best methods reaches a success rate of approximately 84%.

The above figures confirm ensembles of 3 best models are a more reliable solution in comparison to simply
selecting the best model among 10 different trained models.

4.3 Best practise: Preventing Over-fitting

One way to explain generalization and training aspects of a deep neural network is through the lens of informa-
tion theory. In [45], the authors propose a new methodology to interpret the training of multi-layer perceptron
(MLP) through a theory called the Information Bottleneck (IB) [51]. The authors demonstrate that the training
of an MLP provides two distinct phases – fitting and compression. These phases are determined by computing
the mutual information between the intermediate representations (activations from hidden layers), and input
(raw data) and output (labels). This way, an output from a hidden layer can be seen as a summary of statistics
containing information about the input and output. The fitting phase is usually very fast, requiring only a few
epochs, while the compression phase lasts longer. The compression phase is also the one responsible for the
generalization of the neural network, i.e., its ability to perform on unseen data.

In this section, we consider the mutual information between output layer activations (Softmax) and the data
labels (as given by the leakage model) as a metric to identify the epoch at which the neural network achieves
its optimal generalization capacity. We empirically demonstrate that this metric provides better attack perform-
ance for profiled attacks against masked AES implementations when compared to usual metrics like accuracy
or loss. More specifically, our results emphasize that training a neural network for too many epochs may af-
fect generalization, and early stopping based on the mutual information metric is a reliable way to avoid this
scenario. To the best of our knowledge, this is the first result providing a reliable attack performance metric
different from conducting an actual attack (key ranking). While key ranking is a reliable validation metric to
optimize the generalization of a deep neural network for side-channel attacks [8, 35], it brings significant com-
putational overheads when using large validation sets. Mutual information, on the other hand, offers remarkable
performance at a fraction of computational cost as it does not have to be computed for all key hypothesis. To
facilitate reproducible research, we make the source code publicly available [1].

A common technique to determine the optimal training point in neural networks is early stopping. This
technique is based on a metric, usually training/validation loss or accuracy that determines at which epoch the
training must stop. The main idea is to avoid the continuation of the training process after the best possible
value for a reference metric is found. The assumption is that the neural network achieved the best generalization
and will start overfitting and deteriorate the generalization after this point, which is an undesired behavior. For
side-channel analysis, deep learning metrics have demonstrated to be inconsistent as reference for the validation
process [38]. As such, one can see that implementing early stopping based on, e.g., loss function or accuracy
could lead to inconsistent results. An alternative would be to compute the key rank for the validation set at
the end of each epoch. Unfortunately, calculating key rank will lead to a significant time overhead for larger
datasets.

The information transferred to the output network layer is important for measuring generalization. At
the same time, [45] mentioned that the beginning of the compression phase usually coincides with the best
generalization. Typically, this is a moment when the mutual information between labels and the output layer
Figure 18: Success rates from averaged ensemble learning on masked AES traces (ASCAD database). This experiment considers the ensemble of 10 single homogeneous models.
Figure 19: Success rates from averaged ensemble learning on masked AES traces (ASCAD database). This experiment considers the ensemble of 10 single heterogeneous models.
probabilities $I(T,Y)$ achieves its maximum value. Additionally, [43] provides empirical results demonstrating that the compression phase does not necessarily improve generalization.

Consequently, we propose to use the mutual information $I(T,Y)$ in the output or Softmax layer as a reference metric for the early stopping. The maximum value for $I(T,Y)$ happens when the fitting phase is usually finished and compression is starting. This means that the training phase usually does not need to proceed to the compression phase in order to achieve best generalization. The calculation of $I(T,Y)$ gives minimal overheads during the training process mainly because we make the computation for a small fraction of the validation set. Our estimation provided that the time overhead to compute the mutual information at the end of each epoch is less than 2%. We demonstrate that, for general datasets, the success rate is commonly higher at the epoch that indicates the higher information value in the output layer.

4.3.1 Main Results on ASCAD database

The empirical validation on the ASCAD database (key byte 3) considers 99k traces for training, 500 traces for validation, and 500 traces for the test. Both validation and test sets have a fixed key. The neural network is trained for 100 epochs and validation accuracy, recall, and loss are used to determine the epoch with the best corresponding metric. Additionally, the validation set is used to determine the epoch with the best key ranking (based on the number of traces to achieve key rank 1) and the maximum $I(T_n; Y)$. After identifying the best epoch for each metric, the corresponding machine learning models are applied to the test set.

Figure 20 shows the guessing entropy and success rate for the test set obtained for each validation metric. These results were obtained from averaging of 100 trained models with the same hyper-parameters. The configured neural network is a multiple layer perceptron with four dense layers containing 400 neurons each. As results in Figure 20 indicate, the best success rate is achieved when the machine learning model is selected from the epoch when the metric is the maximum value of $I(T_n; Y)$. More precisely, around the processing of 400 traces, the success rate reaches 100% if the model is selected from the epoch determined by the maximum $I(T_n, Y)$ value. At the same data point, the key ranking success metric reaches a success rate of approximately 80%, accuracy success rate is approximately 64%, while the recall metric barely reaches 30% success rate. For more experimental evidence on the value of this method, we refer the reader to [36].

4.4 Conclusion

This section offers a summary of optimization techniques for deep learning applications to profiled side-channel analysis. The first contribution refers to the identification of the features learned from side-channel traces during training phase. We show that the proposed method, salient feature detection, helps in visualizing whether the neural network is learning correctly the leakage or simply overfitting. The second contribution refers to the improvement of generalization for side-channel attacks based on the usage of ensembles. We demonstrated
that selecting best models based on final key ranking metric and building an ensemble from these best models provides higher success rates. The third contribution is a new metric for early stopping where the mutual information between labels and output class probabilities indicates the epoch where the neural network generalizes the better. Empirical results show that using mutual information as a metric during training improves the success rates of profiled side-channel attacks.
5 Multi-Channel based Deep-Learning Analysis

A novel strategy to DL-based side-channel evaluation, so-called multi-channel approach, has been presented in the deliverable 1.2. It aims at improving the usage of Neural Networks (NN) for the task of side-channel analysis compared to the standard classifier strategy that has been extensively discussed in the previous section. To do so, the multi-channel approach utilizes a NN in a different way than it has been proposed so far in the literature: instead of training a NN to classify the leakages based on single traces, it is trained to recognize a correct (or wrong) key hypothesis based on a permutation of input traces. For a detailed introduction to the multi-channel based Deep-Learning Analysis, the reader can refer to Section 6 of D1.2.

We recall hereafter the main aspects in which the multi-channel approach differs from the classical one, and how it could alleviate some of the difficulties encountered in DL-based side-channel analysis.

5.1 Dependency of DL-based analysis to the soundness of the leakage model

The research on the multi-channel classifier initially stems from the observation that the standard classifier strategy depends highly on the leakage model assumptions. In particular, the leakage model labeling partition must be close enough to the actual leakage classes – i.e. the clusters of observed values if the real leakage model is not known – otherwise, the training of the classifier is hampered. For instance, if the observations corresponding to a single label can take different distinct values (aside from the noise influence), the training of the NN is partly destroyed every time an observation with a different value is processed for that label, and it will struggle to converge towards a mean value. On the other hand, if several labels are not actually distinguishable in the measurements, the predictions will randomly favor one or the other label depending on small random variations of the output values of the NN and the noise in the measurements, due to the design of the last layer of the NN – usually using softmax as activation function. Nonetheless, the first issue seems the most problematic, as it (partially) prevents the NN to learn to classify the leakages correctly.

One of the main advantages of the multi-channel design is to relax the leakage model assumptions. Since there is no need to guess the leakage model of the device, the values of the targeted intermediate value can be directly used, which solves the first problem mentioned above. Also, if several values of the intermediate are not distinguishable from each other – the cause of the second issue –, it simply creates redundant information among the input channels that is just ignored by the NN. Because the model is not forced by design to differentiate the values, the multi-channel design is a better approach when the leakage model is not known a priori.

5.2 Constraints of the multi-channel setup

In both the training and the inference phases, the measurement data (traces) must be presented to the NN as separate channels, as many as the number of possible values for the targeted intermediate value. For input traces containing \( s \) sample points, the input of the NN is therefore a vector of size \( n_i \times s \). The first obvious constraint of the multi-channel approach is the memory it requires to train and use the model. In general this means that the number of samples that can be considered in the analysis is limited to a few hundreds at most. On the other hand, the fact that \( n_i \) input traces are combined to create a single input offers the possibility to present more than \( n \) distinct input to the NN – both in the training and inference phases – and opens the door for implicit data augmentation, a promising feature for cases where the number of measurement is limited (e.g. because of bounded measurement time, or because of some limit set at the application level).

Another constraining aspect of the multi-channel setup lies in the data pipeline complexity required by a real attack. Since the model is trained to distinguish a correct input ordering from a wrong one – labeled as “correct order” vs. “wrong order” since the ordering depends on the key hypothesis –, the inference phase must be repeated \( n_k \) times, where \( n_k \) denotes the number of possible values that the targeted key word can take (e.g. 256 for many block ciphers, such as AES). For each guess, the attacker must re-order the traces of the data set to create the corresponding multi-channel input, requiring a complex input pipeline where computation complexity can easily get out of hand if not implemented with great care. However, in an evaluation context, this constraint seems to be avoidable. Indeed, the training metric (typically the classification accuracy) is a strong indicator of the effectiveness of the multi-channel analysis, which is generally not true with the classical approach due to its low accuracy problem – an issue inherent to the standard classifier design in a noisy environment. As a consequence, a multi-channel NN could be used for leakage detection purposes by
REASSURE Deliverable D1.4

Figure 21: Results of attacks using the multi-channel and standard classifier approaches on the ASCAD data set assuming an Hamming-weight leakage model

implementing the training phase only and monitoring the training accuracy as a leakage indicator. Sensitivity analysis techniques such as gradient visualization [30] could be used afterwards to map the detected leakages to the input sample points, or a real attack can be implemented if necessary to demonstrate the exploitability of the leakage.

5.3 Performance

To illustrate the performance of the multi-channel classifier, we take the example of the ASCAD public data set [39], which is known to have a biased leakage model: the least significant bit is leaking predominantly (see D1.2).

DL-based side-channel attacks are performed, on one hand using the standard classifier and on the other hand using the multi-channel approach. The Hamming-weight leakage model is used in both cases, to illustrate how each approach performs when the leakage model is unknown. The hyper-parameters of both models have been optimized using state-of-the-art methods. Figure 21 shows the results of the two attacks averaged over 100 experiments.

The mean guessing entropy drops much more quickly using the multi-channel classifier than the standard classifier. The former one achieves a mean rank of 0 with approximately 300 traces, while the mean rank converges much more slowly for the standard approach: a mean rank of 0 is reached with more than 3000 traces (not shown of the graphic for readability reasons).

5.4 Conclusion

The standard classifier approach and the multi-channel one are not interchangeable and they differ in several aspects as presented above. Both have their place in the toolbox of the side-channel evaluator and more research is necessary to make definitive recommendations. However, it seems already appropriate to advise the use of a multi-channel classifier in cases where the standard approach fails and the leakage model is unknown. In such cases, the performance of the multi-channel classifier may beat the standard one by an order of magnitude (about 10 times). Moreover, the multi-channel approach might be also suitable for leakage detection purposes, as the training metric is a reliable indicator, contrary to the case of a standard classifier; however more research is required to confirm this with certainty.
6 Analysis of Leakage Exploitation Methods for Misaligned Traces

The efficiency of side-channel exploitation can be significantly reduced in case of misaligned traces, which may be due to jitter in the measurements or to dedicated countermeasures such as shuffling [22, 55] or random delays [11]. In such traces, the information is disseminated as a multivariate distribution, therefore multivariate side-channel methods are required. In this work, we compare the efficiency of a classical methodology, namely the on-the-fly variant of the (profiled) linear regression attack (LRA) [14], and a recent proposal for improving the efficiency of attacks with misaligned traces called Scatter [48]. A more detailed version is published at COSADE 2020 [59].

6.1 On-the-fly linear regression

The use of linear regression for (profiled) side-channel attacks was introduced by Schindler et al. [44]. It was then extended to non-profiled key-recovery attacks in [14]. We next denote this non-profiled extension as LRA.

6.2 The Scatter transform

From a high-level view, Scatter is a multivariate preprocessing of traces used with “generic-emulating” distinguishers [57] such as Mutual Information Analysis (MIA) [19] or the Chi$^2$ test [32]. The authors of Scatter claim that it efficiently deals with synchronization issues and may improve higher-order side-channel attacks (for instance against masked implementations) [49].

The Scatter transform is depicted in Figure 22. It is combined with either a Chi$^2$ or an MIA distinguisher. In short, the Scatter approach is to transform each $d$-dimension trace into $d$ one-dimension samples. All the testing and exploitation is then performed on the $d$ one-dimension samples. In the Scatter approach, the exploitation works with histogram-based distinguishers: the attacker or evaluator tries to estimate the distribution of the $d$ one-dimension samples using histograms. The histograms are then partitioned according to the key guess and a hypothetical leakage model (e.g., Hamming weight of an AES S-box output). The Chi$^2$ or MIA distinguishers are then used to search for the correct key guess.

In more details:

1. Estimate histograms based on the amplitude of the sample points within a window of size $d$. For each measured trace, convert the $d$ sample points to an $N_b$-bin histogram. For an 8-bit oscilloscope, the max $N_b$ is 256.
2. The “histogram traces” are then partitioned based on the key guess and hypothetical leakage model. In this work we consider the Hamming weight leakages of an AES S-box output. As a result, we obtain $N_b \times 9 \times 256$ partitioned histogram traces (i.e., 9 Hamming weights, 256 key candidates).

3. Compute the distributions $pdf_{g,h}[u]$ using the partitioned histogram traces, for each key guess $g$ and corresponding Hamming weight hypothetical leakage $h$, where $u$ denotes the histogram value:

$$pdf_{g,h}[u] = \frac{Acc_{g,h}[u]}{\sum_{u'=0}^{u'} Acc_{g,h}[u']}$$

in which $Acc_{g,h}[u]$ is the total number of occurrences of value $u$ for a key guess $g$ and its corresponding Hamming weight hypothetical leakage $h$.

4. The correct key guess $g_{correct}$ is distinguished by applying a generic-emulating side-channel distinguisher ($\chi^2$, MIA, . . . ) to the estimated distributions $pdf_{g,h}[u]$.

### 6.3 Setting #1: a simulated shuffled implementation

The authors made some preliminary comparison of Scatter with univariate CPA-based attacks. Our observation is that such attacks are not good points of comparison, since misalignment spreads the informative samples over multiple time dimensions (i.e., a multivariate distribution). We therefore compare Scatter with a more natural competitor for first-order attacks, namely the on-the-fly linear regression distinguisher from [14].

#### 6.3.1 Experiments and results

Shuffled implementations are the typical context in which Scatter’s multivariate transform was claimed to be a useful tool when the paper was presented at COSADE 2018.

**Setting.** The main parameter influencing the security of a shuffled implementation is the number of parallel operations $d$ which are randomized. We next consider a default size of 16 (corresponding to the AES case) and additionally experimented with a permutation of size 64, which could correspond to the execution of 48 dummy S-boxes. In our default setup, a $n_i = 1$ POI is leaking (corresponding to the target S-box execution) but we also considered a case with $n_i = 4$ POIs (which does not reflect a concrete AES implementation and was just aimed to understand the impact of a denser leakage in the Scatter window). Finally, we used a Signal-to-Noise Ratio (SNR) of 10, 1 and 0.1, reflecting low-noise, medium-noise and high-noise contexts [28]. The way we generated simulated traces is similar to the Scatter paper.

For each simulation setting, we estimated the Success Rate (SR) of the different attacks under investigation based on 100 independent experiments. The results are in Figure 22.

Based on these results, we can evaluate Scatter as follows:

- LRA always outperforms Scatter with both the $\chi^2$ and MIA distinguishers, no matter the permutation size, noise level and number of POIs;

- The performance gap between LRA and Scatter is getting bigger as the attacks become more difficult (i.e., when the permutation size increases, the noise level increases and the number of POIs decreases);

- Scatter with the MIA distinguisher performs slightly better than Scatter with the $\chi^2$ distinguisher (which is in line with the COSADE 2018 results).

- As for the impact of the number of bins for Scatter: more bins generally show better results with lower noise and less bins generally works better with higher noise. The latter is in line with the findings of [19].
Figure 23: Success rate on simulated shuffled implementations, with $d$ the window size, $n_i$ the number of POIs per window and various SNR values.

6.4 Setting #2: a concrete jittery implementation

We now experiment with a real device performing a software AES implementation on a Cortex-M4 chip. Due to the variable internal clock, the inserted random instructions during AES calculations, and the interrupts caused by the running Android-like operating system (OS), the measured traces are very jittery and we cannot really align the traces at the leaking time interval. We study how well Scatter can handle this challenging scenario.

Our comparisons are based on 99,902 aligned EM traces focusing on the leaking part, applying a Fast Fourier Transform (FFT). After this pre-processing, LRA was able to recover all 16 key bytes of an AES state, but Scatter was not (neither with the Chi$^2$ nor with the MIA distinguishers). These results are illustrated in Figure 24 where the success rate is estimated based on 100 independent experiments.

6.5 Conclusion

From this and some more advanced analysis given in the paper [59], we conclude that on-the-fly linear regression consistently outperforms the Scatter method for leakage exploitation. It is not impossible that Scatter has some specific use in a case that we missed, but as long as no such case is put forward, we advise evaluators to use on-the-fly linear regression in the case of misaligned multivariate traces.
7 Conclusion

7.1 Deep Learning

This work presented optimization techniques for deep learning applications to profiled side-channel analysis. The first contribution refers to the identification of the learned features from side-channel traces after a neural network is trained. We could observe that the proposed method, called salient features, helps in visualizing whether the neural network is learning correctly the leakage or simply overfitting. The second contribution refers to the improvement of generalization for side-channel attacks based on the usage of ensembles. We demonstrated that selecting best models based on final key ranking metric and building an ensemble from these best models provides higher success rates. The third contribution of this work is based on a new metric for early stopping where the mutual information between labels and output class probabilities indicates the epoch where the neural network generalizes the better. Empirical results show that using mutual information as a metric during training improves the success rates of profiled side-channel attacks.

7.2 Multi-Channel

A distributed memory parallel implementation of the leakage detection computation was presented using the central moments and the histograms methods. The implementation uses MPI for distributing the memory and OpenMP for shared memory systems. The histograms method is not deemed usable in the current state for realistic examples, as the storage of the histograms using dense arrays uses a huge amount of memory. A benchmark run on a real-world example shows that the scalability of the central moments method is still perfectible.

As future work, a better algorithm is needed to reduce the communication overhead from IO to calculation processes and improve the scalability. One major aim is to overlap the communication and computation to avoid the communication bottleneck. The current code is also able to be executed on small ARM processors. More benchmarks must be performed to study the efficiency conventional CPUs versus ARM CPUs in terms of initial costs, power consumption and speed of computation. Finally, it would be interesting to study if the use of sparse arrays for the computation of the histograms would be a practical solution to the memory consumption issue of this method.
References


