Project ID: 731591, Horizon 2020
http://www.reassure.eu

REASSURE

Deliverable D3.7

Final report on tools

Editor:	E.Oswald (UNI-KLU)
Deliverable nature:	R
Dissemination level: (Confidentiality)	PU
Delivery date:	March 31st, 2020
Version:	1.0
Total number of pages:	14
Keywords:	Software, Data Sets, Simulator

Horizon 2020
European Union funding
for Research & Innovation

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 consists of a list of tools that were developed during the course of the REASSURE project by project partners. Tools are categorised as either software, data sets, or simulation tools; they can be internal tools strictly under the control of a single partner, or joint ventures and openly released.

We provide an overview of our collective achievements, followed by a list of tools, organised by REASSURE partners. We indicate if a tools is available for public use. Wherever possible we give figures for performance improvements as a result of a tool.

List of authors

Company	Author
UNI-KLU	E. Oswald

Other contributions gratefully received from

Company	Author
NXP	V. Verneuil
UCL	D. Bellizia
SGDSN	E. Jaulmes
Riscure	I. Buhan
IDM	N. Debande

Overview

This deliverable summarizes the tools that REASSURE partners created over the duration of the project. Some of these tools have been publicly released as open source, for some of the tools there is a free or open access version, and some tools are strictly internal to the respective partner. We indicate if a free or open source version is available.

Tools are categorised in three groups:

- software
- data sets
- simulator

"Software" refers to anything that would be part of an evaluation, ranging from measurements, leakage detection and mapping, to exploitation and characterisation.

Data sets are collections of traces, with appropriate documentation (including information about the target device and measurement setup).

A simulator is a specific software tool that faithfully predicts some behaviour of a target device.

Achievements

The consortium delivered on the data sets and simulator as described in the description of action. There are a number of data sets for both AES and ECC available on Zenodo as well as Github.

The consortium also delivered a stable version of the anticipated simulator (ELMO). The simulator is suitable for IoT devices based on the ARM Cortex M family (in particular the M0), and has a built-in leakage detection facility to help improving code quality with respect to leakage. An addition to another simulator (GILES) was developed as well, which provides fault simulation capability.

Two focus areas for REASSURE tools emerged over the project: deep learning and leakage detection. Deep learning as a novel approach for leakage modelling, leakage detection and exploitation has captured the imagination of industry and academia. All major evaluation schemes now demand that some form of deep learning is at least attempted in the course of an evaluation. Therefore industrial partners have developed or added to a variety of deep learning tools, typically as part of their existing analysis frameworks.

One of the reasons for the inception of REASSURE was that of the difficulty of an ever increasing number of methods for attacks, and deep learning in some sense contributes to this problem ("yet another method") but it perhaps (initially) came with the promise to alleviate this problem (via an increased promise for automation via "learning/attacks with less user input"). REASSURE partners' research has been two fold: firstly, SGDSN released (open source) a set of training traces to facility a fair comparison of newly developed deep learning methods. This trace set is now widely used for reporting deep learning experiments. Secondly, partners considered the potential for automation of deep learning approaches. Our findings here mainly indicated that deep learning "shifts expertise" away from statistics and signal processing towards configuring deep networks.

The second major focus area of REASSURE partners was leakage detection. Its significance impacted on tools both w.r.t. efficiency and effectiveness, as well as robustness and automation. NXP and IDM significantly increased their in-house capabilities in terms of efficient leakage detection (via parallelising their implementations). IDM increased their range of techniques and this enables them now to detect more leaks in their evaluations. UCL and Bristol both contributed to several academic advancements in the theory of detection, and together with Riscure they developed a tutorial and advanced training for leakage detection. The scripts for this tutorial are available as open source.

Tool descriptions

We now provide a concise overview of the tools that were developed by REASSURE partners.

NXP

Distributed leakage detection

This tool has been developed to serve as a proof-of-concept of distributed leakage detection and to study implementation optimisations. Both the central moments and the histograms methods described in deliverable 1.2 have been implemented using the Message Passing Interface (MPI) standard ; more details about the implementation can be found in deliverable 1.2. Furthermore, as presented in deliverable 1.4, the implementation was ported to a low-cost ARM platform in the aim to compare the cost-efficiency of this platform to a classical x86 server-grade platform.

The tool successfully demonstrates the feasibility of the distribution of the leakage detection calculations. The efficiency of the central moments-based implementation meets industrial requirements in terms of performance, storage, and usage. The implementation based on the histograms is deemed to have too high storage requirements to be usable in practice. A comparison of x86 and ARM platform has therefore been performed using the central moments implementation (see D1.4).

This development demonstrates that leakage detection can be computed efficiently using High Performance Computing (HPC) principles and served as a prototype for the implementation of the distributed central moments based method in the side-channel framework used at NXP for vulnerability analysis of internal products. This allowed to simultaneously cut the leakage detection computation time by 50% and increase the leakage detection coverage, by leveraging an existing HPC infrastructure.

ML-based SCA tool for the multi-channel experiments

This tool has been developed to support the research on the multi-channel approach for Deep Learning (DL) based side-channel analysis detailed in deliverable 1.2. It consists of a Python package implementing training and inference for multi-channel Neural Networks (NN) based on the TensorFlow[1] library. One main technical challenge was to implement an efficient input data pipeline for the attack, because of the necessity to reorder traces depending on the key guesses, as detailed in deliverable 1.4.

[1] `https://www.tensorflow.org`

The tool successfully demonstrates the interest of the multi-channel approach for side-channel analysis. In cases where wrong leakage model assumptions are made, which is unavoidable in a black-box evaluation context for instance, we found that the multi-channel approach requires up to 10 times less traces to perform a successful attack. This implementation served as a prototype for the integration of the multi-channel approach into the side-channel framework used at NXP for vulnerability analysis of internal products.

Key-less rank estimation

Key enumeration and key rank estimation methods are necessary ingredients for sound and pragmatic side-channel evaluations since they allow to trade measurements for computational power. This tool was motivated by the fact that, since key enumeration is a computationally expensive task, it is worth to estimate the effort it requires (i.e. approximate the number of candidates to enumerate) prior to starting it.

For this purpose this tool was developed by NXP to investigate a new question relating to the possibility of a key agnostic rank estimation method. This Python tool makes use of the histogram-based method for key rank estimation by UCL as a basis, to estimate the full key rank without any knowledge of the key and allowed us to compare this method, classical key rank estimation methods, and other key-agnostic rank estimation methods proposed in the side-channel literature.

Effectively, this tool leads to a better understanding of rank estimation methods and tools used during product evaluations and improves the overall confidence in side-channel evaluations.

The research involved in developing this tool was performed in collaboration with UCL. The tool has been released as open source: `https://github.com/AZOUAOUI/KeylessRankEstimation`.

ECC library and shortcut formulas for horizontal attacks

Motivated by the difficulty of mounting worst-case horizontal attacks on ECC implementations and the efficiency provided by classical shortcut formulas for symmetric cryptography described in deliverable 2.1, this tool integrates shortcut formulas to efficiently estimate the success rate of horizontal attacks. This tool is developed in Python and additionally includes an Elliptic Curve Cryptography (ECC) library to simulate side-channel leakages of an ECC implementation investigated by the REASSURE project.

Concretely, this tool allows us to estimate the required effort in leakage detection, mapping and modelling phases before mounting a full attack. As a result a preliminary analysis can be performed very efficiently. For instance, instead of detecting and modelling all required leakages for the success of the attack (which can be a tedious task for protected implementations), the shortcut formula tool relies on only identifying and modelling a few leakage points. This leads to very fast side-channel characterisation and estimates of the resistance of ECC implementations against adversaries with varying capabilities (including knowledge of the implementation and profiling abilities) as mentioned in deliverable 1.3.

The tool simulates leakages of an ECC implementation investigated by REASSURE provided to NXP by UCL. Additionally, the shortcut formula is based on research performed in collaboration with UCL. The tool has been released as open source: `https://github.com/AZOUAOUI/ShortcutFormulasECC`.

Soft-analytical side-channel attacks

Soft-Analytical Side-Channel Attacks (SASCA) are extremely powerful attacks that allow to exploit most of the side-channel leakage of a cryptographic implementation. Due to this quality, SASCA is a fitting tool to analyse the worst-case security of new products. Accordingly, NXP has developed a Python tool to run the belief propagation algorithm to perform SASCA. The tool additionally includes the efficient alternative of running the corresponding model to SASCA, the Local Random Probing Model (LRPM).

The substantial research involved in developing this two-part tool, has lead to a better understanding and the application of advanced side-channel techniques such as SASCA, and the assessment of their limits with respect to realistic targets, as described in deliverable 1.3 for single-trace attacks. Furthermore, this REASSURE tool is the starting point of a subsequent tool to help designers at NXP evaluate different AES implementations and countermeasures.

Thanks to collaborative research performed in REASSURE, NXP has developed this tool and extended through it the application of SASCA to ECC implementations and the use of the LRPM to different cryptographic primitives. This tool has been released as open source: `https://github.com/AZOUAOUI/LRPM`.

Worst-case horizontal attack on the point randomization countermeasure

Point randomisation is an important countermeasure to protect ECC implementations against side-channel attacks, however its security was never analysed before. For that reason we developed a toolbox to perform divide-and-conquer horizontal attacks against common point randomisation procedures.

This tool consists of Python scripts to perform the different steps of a 'close to worst-case' horizontal attack on real measurements and corresponding simulations: leakage detection, mapping, modelling, and information extraction and combination. Among other steps, it includes target calculations, the points-of-interest selection method based on UCL's ρ test, Principal Component Analysis (PCA), multivariate gaussian template modelling, support for rank estimation tools, etc.

While this tool is specifically designed for the target point randomisation under investigation, it serves as an example and practical framework to analyse the worst-case security of other point randomisation options or general horizontal analysis of implementations and countermeasures.

UCL

Python scripts and notebooks for Understanding Leakage Detection tutorial

Leakage Detection is the first step in the evaluation process of a secure product. In the context of the REASSURE project, great emphasis has been put in investigating ways to speed up and improve this important step.

To deliver this knowledge to non-experts, and make them confident with widely used techniques for leakage detection, UCL has contributed in writing Python scripts in forms of pure scripts and Jupyter notebooks that have been used for the "Understanding Leakage Detection" tutorial presented at CARDIS 2018 in Montpellier, France. These notebooks are freely available on REASSURE's website.

These tools contain code useful to perform TVLA and correlation-based test on simulated and real traces (that UCL produced and made free available on Zenodo), as well as code to make non-experts familiar with the interpretation of such tests. This work was done in collaboration with the University of Bristol and Riscure.

The datasets used on Part2 have been downloaded 210 times (in the time frame from Feb. 22, 2019 to Mar. 30, 2020), as reported on Zenodo.

Two AES128 cores for FPGA

In the context of hardware implementations, it is critical how the measurement of the setup is configured, since the level of the side-channel signal can be orders of magnitude smaller than in software implementations, especially in the presence of countermeasures.

The tuning of a new measurement setup can be made simple if a solid and known implementation is used, similarly to what is used in the context of calibrating RF system with known load conditions (e.g. 50 Ω, short and open loads). The two cores developed within the REASSURE project have been used intensively for the calibration of measurement setup, since they offer a moderately trivial target suitable for ASIC and FPGA application.

Specifically, the SNR obtained on an FPGA for one of the AES128 cores for a specific intermediate variable (e.g. the output value of the sbox) is stable after 10k measurements with a low-noise setup, that makes the calibration and the verification of the setup very fast and easily portable within different methodologies and equipment.

In addition, the two cores have been used as baselines for developing and investigating new countermeasures. We also utilised them to gather a reference data set for the REASSURE project. The FPGA AES dataset provided by UCL as part of Deliverable 3.1 has been collected on one of the developed cores. This dataset has been downloaded 1221 times (from Oct. 8,2018 to Mar. 30, 2020), as reported on Zenodo.

Flexible, fast and low-noise SCA-setup (also with dual channel option)

Building the side-channel measurement setup is critical in the evaluation of a secure product, since mistakes and errors in the design impact dramatically on the Signal to Noise Ratio and therefore the quality of the measurements.

We aimed to design a measurement setup that has to be easily portable, reliable and low-noise, with the additional feature of allowing several researchers to use it for their own purposes. New sets of SCA-setups have been built based on the adoption of Picoscope oscilloscopes as digitizer interfaces and a set of probes for measuring power and electromagnetic emissions of devices, deploying very low noise setups.

In addition to that, UCL has written novel and reusable Python (and Matlab) code to perform automated and remote acquisition leveraging on these new setups, also being able to capture two side-channel observations at the same time (e.g. power + EM).

Prior to REASSURE, UCL was mainly using a Lecroy oscilloscope, and previous scripts used to gather measurements were running at a maximum of 30 traces/s. In the slowest setting, new setups can gather traces from 60traces/s up to 6000traces/s, depending on the target and the setting, which in turn means an improvement up to a factor 200 compared to previous used tools.

Control firmware and scripts for testing hardware designs on Sakura G board

Testing hardware designs may require long measurement campaigns, thus, implying a large overhead in time to deploy final versions. To speed up this process, UCL has developed a new framework for the Sakura-G board, where the measurement setup is extended to the Xilinx Spartan-6 LX9 FPGA, where it acts as a controller. A novel firmware has been developed in Verilog hardware description language, that has the role of receiving commands from the master PC through SPI messages (also the SPI slave side has been developed within REASSURE project), generating inputs and randomness (leveraging upon one of the developed hardware AES128 core) and collecting outputs.

The new controller has been designed with an APB-like interface to communicate with the target FPGA (Xilinx Spartan-6 LX75), through I/O pins. Of course, new designs for target FPGA are (and will be) designed as component of a top-level that is compatible with our novel controller and its APB-like interface, making the overall system flexible and portable. A new Python API has been developed to set and control easily the controller and the target. Since the new controller is able to generate locally inputs and randomness in a controlled way, we have been able to fully use the bandwidth and memory resource that the oscilloscope can reach.

Collecting 15kS per trace on a 6MHz design, we have been able to capture up to 6000 traces/s with our low-noise setups, compared to the 60 traces/s we have without a controller interface, obtaining an improvement by a factor 100. Adopting this new controller, we have been able to speed up the design as well as the evaluation of hardware designs on FPGA, and it could serve as well as a generic framework to test and evaluate new experimental ASICs.

SCA-toolbox

Over the last few years, we have been developing a tool-box containing many side-channel tools. Its ultimate goal was to be shared by many researchers, the latest being welcome to add features to it. The main motivation behind this is twofold. First, it allows to rapidly try a large range of techniques. The tool-box makes it possible without going necessarily to a tedious development/deep understanding time which is error prone and time consuming. Second, the quality of the methods is expected to be high since developed by a researcher active in that exact area.

REASSURE started at the beginning of this effort, and allowed adding many features. These are mainly in three categories. First it contains various leakage detection methods among others Welch's t-test, Hotelling's test and chi-square test. Second are information theoretic metrics. On simulations, it allows computing information on various scenarios (higher order masking) in order to gain intuition about implementations weaknesses (e.g. are masking proofs tight?). On practical settings, it allows bounding the data complexity of an attack. Third are state-of-the-art attack tools. This includes dimensional reduction techniques and automated template attacks.

This tool-box reached its main goals. First, it has been used by around 10 researchers on a daily basis. Depending on the function they used, this saved between 2 weeks and 4 months in development. Second, at this time, the tool-box is mainly implemented in Python. However, some widely used methods are implemented in low level languages such as Rust or C while keeping an easy to use Python interface. This allows to gain a factor larger than 100 on the computation time of metrics such as t-test or SNR. By using the tools available, we were for example able to evaluate the AES implementation released by SGDSN. This allowed performing the first successful attack against this implementation. Namely, we were able to recover a full key in less than 2,000 measurements while no attack succeeded with 100,000 traces.

In the long term we plan to release this tool-box under an open-source license. However, before this, some features still have to be added as well as extensive code documentation. Indeed, the first stage of development mostly focused on performances and interfaces. The REASSURE project will of course be credited when the tool is released.

UNIVBRIS/UNI-KLU

The Universities of Bristol and Klagenfurt concentrated on the development of the leakage emulator ELMO and the framework GILES. ELMO and GILES are open source and available via `github.com/sca-research`.

ELMO

UNIVBRIS initially developed the foundations to ELMO as part of a project which sought to explore trade-offs between side channel resilience and energy consumption of implementations. A first prototype of ELMO, which included not just energy modelling but also the modelling of instantaneous power, was then the starting point for all further work within REASSURE. The clear intent of the emulator that is one of the declared goals of WP3 is to support developers of software implementations on processors which can be typically found in Internet of Things devices.

ELMO simulates instantaneous power for implementations of the ARM Cortex M0 architecture. The ARM Cortex M0 is a typical low end microprocessor, which is often found in Internet of Things devices. ELMO consists of two essential components: a customised ARM M0 instruction set emulator (we utilise the Thumbulator) and a set of power models (which we derived experimentally from a concrete instance of an M0 core). The emulator takes in Thumb assembly code and emulates the workings of an M0 core. Thereby it enables us to derive the inputs and outputs of instructions in their correct sequence. Based on triplets of instructions (in sequence) and their corresponding data, the pre-determined models then enable prediction of the instantaneous power consumption.

The approach that UNIVBRIS took for modelling differs substantially from previous work: rather than determining coefficients to a priori fixed models, we use statistics to determine which model coefficients contribute significantly to the data dependent part of the instantaneous power consumption. Because of that, different instructions (in sequence) are described by different analytical expressions (aka models). This means that ELMO is highly accurate to the true power consumption of the modelled implementation of an M0 core.

Within REASSURE UNIVBRIS were able to challenge their novel methodology by extending the number of cores that were modelled. Having applied this methodology multiple times on cores from different vendors successfully contributes to the confidence the robustness and practicality of the approach. UNIVBRIS have utilised ELMO for other research in the meantime: e.g. with ELMO we were able to develop attack techniques for post-quantum crypto implementations, which would have been too time consuming to test otherwise. ELMO also facilitated the discover of bit wise dependencies that exist within words/variables, which ultimately lead to the demonstration of the impossibility of secure share slicing on M0 cores. ELMO has been used within NCSC to demonstrate leakage attacks, within NXP in the context of some joint work on post quantum crypto, and it is the basis of ROSITA (`https://eprint.iacr.org/2019/1445.pdf`). According to Github statistics, ELMO has been cloned 7 times, and there were 29 unique visitors to the site (with 127 visits in total).

GILES

The REASSURE consortium committed to ELMO early on as emulator. The main advantage of building on the existing code of ELMO was that we were able to challenge our modelling methodology and thereby come to a point where we can consider it as robust enough for practical use. The disadvantage of ELMO is that the two parts (the emulator and the models) are tightly connected in the code. Therefore it is possible to emulate cores which are similar to an M0 (such as the M4) but it is not possible to use the code framework to work with entirely different processor families. Therefore in the context of work outside of REASSURE, UNIVBRIS developed a novel framework to integrate a variety of emulators and leakage models.

Some specific feedback from the Strategic Advisory Board was that REASSURE's focus on side channels perhaps overlooked interesting questions and opportunities w.r.t. fault analysis. Whilst our focus in WP1 was clearly on side channels, the consortium discussed the possibility of providing fault simulation capabilities. With the code base of ELMO this seemed infeasible, but because of GILES' modular design, integration into GILES was relatively cheap. Consequently, UNIVBRIS spent some time of defining and integrating fault simulation capability into GILES. GILES fault simulation capability consists of enabling developers to define registers and time points at which faults should occur.

GILES is now capable of simulating architectural faults alongside side channel leakage. Because of its modular design and versatility it will be utilised by UNI-KLU for any further developments regarding the simulation of leakages and potential automation of mitigation strategies.

For the architectural fault simulation we used inspiration from Riscure's device based fault simulator. We made GILES available to Riscure for potential integration into the Inspector Cloud framework. Some initial tests were performed but the question of how to safely run customer code in the cloud could not be answered satisfactorily within the duration of the project.

IDM

Idemia's specific interest was in tools that would benefit their internal product evaluation capabilities.

Shortcut formulas

Robust evaluations require to repeat attacks in order to get stable results for side channel experiments. In particular in a low signal/high noise setting, such repeating of experiments can be very expensive. Consequently, alternative options such as shortcut formulas are attractive to speed up evaluations, and therefore save cost on our side.

IDM performed research in WP2 that led to a better understanding of when and how shortcut formulas can be safely used in the context of evaluations. Such shortcut formulas essentially require a characterisation of the device (under evaluation), and then based on a number of assumptions one can project how attack success changes as a function of countermeasures.

IDM integrated shortcut formulas to estimate the success rate for first and second-order correlation based DP attacks into our internal tool chain. For certain implementations, where noise related parameters are changed via security patches, we can now utilise them to project attack results instead of rerunning attacks.

These formulas were also integrated into the Inspector Cloud tool in a collaboration with Riscure. An open source version is available: `https://zenodo.org/record/1445529#.XLX1jKTgqUk`.

Leakage detection toolbox

Leakage detection is the essential first step in Idemia's internal evaluation flow. It is absolutely essential as it can either convince an evaluator that there is no exploitable leakage, or else guide the evaluator to a concrete attack exploitation.

Owing to the research conducted within REASSURE (in particular the contents of the D2.7 and the D1.2 deliverables), IDM significantly improved their tool dedicated to leakage detection. Concretely, IDM added to their existing leakage detection tool the following statistical tests:

- Continuous Mutual Information
- Correlation Test
- Chi-Square Test

These tests have been implemented so that they can be applied in an arbitrary analysis way but also on a specific intermediate value.

With these three novel detection techniques IDM now have a better detection coverage, and this increases our trust about the conclusions we can make after the leakage detection step: IDM determined that in about 10% of evaluations, they have been able to detect leaks that they were not able to detect before.

Furthermore, as many statistical tests require similar base computations, IDM adapted the core of our leakage detection tool to a parallel architecture (thanks to D1.2, Section 4.). IDM observed up to 16 times faster calculations with our new architecture.

Deep learning tool

Quite recently, a new way to exploit side channel leakages has been introduced, deep learning-based analysis. The enthusiasm about these kind of attacks was so strong that today, in all evaluation laboratories, it is strongly recommended to try a deep learning analysis at least once. In order to maintain IDM's expertise at a high level, IDM constantly seeks to improve their deep learning platform.

A technique that IDM found particularly interesting, was presented by NXP during this project — it represents a novel design of a neural network; the so-called multi-channel neural network. This kind of network shows an improvement of attack success, in particular when the chosen leakage model does not match the true leakage model, which is particularly relevant for practice.

Thus, IDM integrated this analysis method in their platform, which is developed in Python, using Tensorflow.

IDM expects that this approach will help them to better highlight side channel leakages.

SGDSN

ASCAD Scripts and Database

SGDSN produced datasets under the name ASCAD providing common traces measured on a secure AES implementation, for the community to test and compare machine learning algorithms. In order to automatize these aquisitions and register the parameters used, we needed scripts to launch the cryptographic execution, setup the scope and register the data in a file. We also wanted the scientific community to be able to reproduce our analysis results on these data, for comparison with their own, so we developed and released scripts to that point.

The scripts launching the aquisitions are written in Python. We chose Python because it is widely compatible and versatile. Researchers, developers and analysts can then easily adapt our scripts to fit their own uses. They communicate with the ARM implementation to trigger the AES execution and with the scope to launch the trace registration. The communication scripts allowed us to automatize the aquisitions and easily register the parameters used. We were able to launch campaigns during the night or over the week end and save time. It also assured the reproducibility of our experiments. The communication tools were developped for our internal use only since they correspond to our material.

The analysis scripts allows anyone to reproduce our results on their own and compare easily with their own networks or modify some parameters to do their own tests. The analysis scripts, also in Python, are based on Keras and Tensorflow libraries. They are divided in three parts : the first one generates the ASCAD databases from any of the available raw traces database. By tuning the parameters, one is able to generate multiple ASCAD databases specialized for various values of interest, with customized desynchronization as well as customized profiling and attacking traces.The second set of scripts train models with the parameters explored in the artifact for CNN and MLP networks.The third one computes the real key byte ranking among the 256 possibilities, depending on the number of traces the trained model takes as input for prediction and plot the rank evolution.

Partners benefited, along with the whole scientific community, of our databases and the analysis scripts

SGDSN Internal Library

SGDSN develops for its internal use a library of analysis tools implementing state-of-the-art algorithms. This tool evolves with the publications of the whole side-channel community. It allows to test new ideas and explore promising trends of research. This tool evolved during the course of the project, as we experimented on neural networks.

During the course of this project, the developments of deep learning analysis both from SGDSN's work and partner's feedback were integrated in our library as new functions. The whole code is developed in C (for performance reasons) and C++ (for versatility and object handling). The new added functions were developed in the same languages.

The update on the library code allows us to stay at the state of the art of side-channel analysis and communicates our expertise to the evaluation labs. The implementation is the result of experimentation on neural network for side-channel analysis during the course of the project. It benefited from the discussion and work with all partners.

AES implementations

Recent works have demonstrated that deep learning algorithms were efficient to conduct security evaluations of embedded systems and had many advantages compared to the other methods. Unfortunately, their hyper-parametrization has often been kept secret by the authors who only discussed on the main design principles. This is clearly an important limitation since (1) the latter parametrization is known to be a challenging question in Machine Learning and (2) it does not allows for the reproducibilty of the presented results and (3) it does not allow to to draw general conclusions.

To address these points, SGDSN developed two challenging masked implementations of the AES algorithms to be used as benchmarks for the ASCAD databases. The first one is made for an ATMega8515 device. This component is not a secure one ; in particular, it works with an external clock and contains no hardware random generator. The information leakage is consequently high and there is almost no jittering. Thus, it is a perfect target for an educational approach. This implementation has been released as open source: `https://github.com/ANSSI-FR/secAES-ATmega8515`.
The second one is a C library for 32-bit Cortex-M architecture and was tested on STM32F3 and STM32F4, which are not hardened against side-channel attacks either. To secure these implementations against first order side-channel attacks, it has been chosen to apply state of the art techniques. A signal-to-noise characterization has been done to validate that there is no first order leakage. The more hardened version works as follows. The internal state of the processing is secured with the so-called affine masking countermeasure. The AES key-scheduling is protected similarly as the encryption/decryption with its own random material. The elements of the state are always manipulated in random order. For security purpose, different random permutations are used for the masked states and the masks states. Also, dedicated permutations are used for the MixColumns executions. This implementation has also been released as open source: `https://github.com/ANSSI-FR/SecAESSTM32`.

These developments enhanced our own understanding of secure AES implementations and provided prefect targets for the ASCAD databases. We were able to provide reproducible results on Machine Learning approaches together with the corresponding implementations and traces, for the sake of comparison. This represents a concrete step to address the three previously identified problems in Machine Learning based security analysis.

The partners benefited from the traces available in the ASCAD database for their own analysis tools. UCL also made a full security analysis of the ARM AES implementation using traces that we transmitted.

Riscure

Inspector Cloud

Inspector Cloud is an easy to use intuitive platform that is meant to let users experiment and learn about side channel analysis vulnerabilities based on power or EM traces. The platform includes several analysis (correlation) and trace processing (filters, alignment, resampling) features that users can take benefit of for their own analysis.

Inspector Cloud is written in Python programming language, the back-end engine where the calculations take place is written in Java and benefits from the years of development that Riscure has done on its Inspector product.

Inspector Cloud is based on the Django platform that allows us to take benefit of standard options in the platform for scaling, multi-language features, error handling, logging and security. The platform features an asynchronous module execution framework that allows a user to continue with other activities while the calculations on a certain traceset are executed in the background. The user is notified when the calculation or trace processing is finished. The framework is setup in such a way, that future modules can easily be added and/or proprietary module creation for users is possible.

Inspector Cloud is offered free of charge at first, for users to test their side channel analysis competencies. A paid version will become available for customers who would like to use the platform to analyze their own devices for vulnerabilities. The main benefit of Inspector Cloud for Riscure is reputation and get interest for the paid version. In this way, Riscure aims to get access to new market segments, especially markets with lower security requirements that are in a need for an easy accessible platform for this purpose. For example large parts of the IoT market.

Some of the shortcut formulas (researched by Idemia within the REASSURE project) have been implemented as a module in Inspector Cloud. This module is called the "signal-to-noise module" and allows to predict the success rate of a first order analysis. It comes with two modes of using it. The first mode is when it computes the Signal-To-Noise ratio per sample for AES-128 with the leakage model HW(SBoxOut). For this it uses the SNR definition in DPA book. The second mode uses uses the SNR and a confusion matrix based on the leakage model to predict the success rate of a first order CPA attack given a number of traces.

JuliaSCA

JuliaSCA runs standalone or inside Riscure's Inspector tool as a module offering a wide range of easy to use analysis methods. Therefore, it is ideal for non-expert users who would like to understand how DPA works (it allows them to run basic examples). But also within Inspector the tool adds value because it gives access to some additional features such as non-profiled linear regression analysis, conditional averaging in different adversarial contexts, fast parallel algorithms, incremental statistics, MIA, etc.

In more detail, JuliaSCA offers algorithms for performing the computational part (DPA) of a side channel attack. The JuliaSCA toolbox supports:

- Conditional averaging, for analog measurements
- Conditional bitwise sample reduction, for whiteboxes
- Threaded and tiled incremental correlation statistics
- Parallelization of all the above (multi-core / multi-machine)
- Correlation power analysis (CPA)
- non-profiled Linear regression analysis (LRA)
- Mutual Information Analysis (MIA)
- AES128/192/256 enc/dec, backward/forward S-box attacks, backward/forward S-box round attacks
- AES128 enc/dec chosen input MixColumn attack
- Some whitebox models for AES (INV MUL / Klemsa)
- DES/TDES1/TDES2/TDES3 enc/dec, backward/forward attack
- SHA1 backward/forward attack (for HMAC)
- Known key analysis + key rank evolution CSV output
- Inspector trace set input and output
- Split sample and data raw binary (Daredevil) input and output
- Trace alignment using FFT convolution
- Trace alignment using Dynamic Time Warping

JuliaSCA has been written in Julia programming language. The main benefit of this language is its speed. Further, various components (conditional averaging, conditional sample reduction, incremental correlation) in JuliaSCA can be split and run in parallel in different worker processes. Only one component in JuliaSCA is currently threaded, and that is the incremental correlation statistics.

JuliaSCA is offered free of charge as open source: `https://github.com/Riscure/Jlsca`. The main benefit of this tool for Riscure from this tool is reputation and access to a new market segment.