

Chapter 9

Side Channel Assessment Platforms and Tools for Ubiquitous Systems



Apostolos P. Fournaris, Athanassios Moschos, and Nicolas Sklavos

Abstract Side Channel Attacks are nowadays considered a serious risk for many security products and ubiquitous devices. Strong security solution providers need to evaluate their implementations against such attacks before publishing them on the market, thus performing a thorough assessment. However, this procedure is not straightforward and even with the appropriate equipment, it may require considerable time to provide results due to the slow process of collecting measurements (traces) and the inflexible way of controlling the tested implementation. In this chapter, we explore and overview the trace collection landscape for generic devices under test (including ubiquitous systems) highlighting and overviewing the latest trace collection toolsets and their shortcomings, but also proposing a trace collection approach that can be applied on the most recent, open source toolsets. We showcase our proposed approach on the FlexLeco project architecture, which we have developed in our lab, and manage to practically describe how an evaluator using the proposed methodology can collect traces easily and quickly without the need to completely redesign a control mechanism for the implementation under test.

9.1 Introduction

The transition of computing devices to the ubiquitous era, where the cyber world is merging with the physical world to bring an ever present layer of computer intelligence to everyday life objects, is bringing, among other things, cybersecurity and privacy into the physical world as an issue to be constantly considered. Due to IP interconnected cyberphysical systems, attackers can gain access not only to a victim's data but also to the victim's life itself, by controlling "smart"

A. P. Fournaris (✉)
Industrial Systems Institute/R.C. ATHENA, Patras, Greece
e-mail: fournaris@isi.gr

A. Moschos · N. Sklavos
Georgia Institute of Technology, Atlanta, USA

University of Patras, Patras, Greece

devices of urban and industrial systems, critical infrastructures and individual households [540]. This highlights the need for strong security features that must be installed on embedded systems. However, ubiquitous devices have several non-functional constraints like small processing power, low power consumption or small memory footprint, that prohibit the use of several traditional security and cryptography schemes (e.g., asymmetric cryptography). This has led to the redesign and simplification of existing security solutions (like TLS schemes and their cipher modes) or the development of cryptography algorithms especially designed for low performance devices (lightweight cryptography schemes [197]).

The design and evaluation of security schemes and cryptographic algorithms must take into account several parameters that are related to a scheme's cryptographic strength (addressed by cryptanalysis, formal security verification methods etc.), to its algorithmic performance (addressed by collecting and comparing performance and resource measurements) and to its resistance against implementation attacks like side channel and fault injection attacks [337]. Side channel attacks (SCAs) have a leading role in the design of modern cryptosystems, since they are the weakest point and they have been exploited by many well-known attacks in order to break otherwise unbreakable security/cryptography algorithms. SCAs can be easily applicable to ubiquitous, cyberphysical systems, where devices are left unattended in potentially security "hostile" environments (in remote, secluded areas, inside malicious user premises, etc.)

There is a broad research field related to various SCAs, aiming to exploit various physical characteristics of a device including timing, power consumption, electromagnetic emission, etc. The flagship SCA analysis methods are of statistical nature and can be categorized into horizontal attacks (using one or a few collected inputs) or vertical attacks (using many collected inputs). Among the most potent and successful such attacks are Differential Power Analysis (DPA) or Correlation DPA [390] as well as template, online template [58] and Mutual Information Attack (MIA) [229]. A simple review of the above advanced SCAs reveals that all of them require a considerable amount of collected leaked physical characteristic inputs in order to be effective.

Assessing if a security/cryptography scheme on a ubiquitous device is SCA resistant, is not a straightforward process. It usually follows two directions. In the first direction, a security scheme is evaluated against specific SCAs while the second direction is based on performing a generic information leakage assessment based on some statistical test. Student's or Welch t-test are two such tests that use statistical hypothesis testing in order to detect if one of a series of sensitive intermediate processes during a security procedure significantly influences the measurement data or (more often in a non-specific test) detect how different is a collected trace from random noise (indicating a bias addressed to exploitable information leakage).

Collecting inputs for SCA analysis is done using specialized equipment in a fairly cheap and easy way. However, when a huge number of inputs need to be acquired then the processing of an input (or trace, as they are usually denoted in the literature) becomes a very slow and cumbersome process. A restricted number of tools that help the acquisition of the needed traces for advanced SCA attacks exist. Most of

them still require a considerable amount of time for trace collection and employ custom (to the cryptosystem at hand) control mechanisms. There are open source SCA trace collection setups widely used by the research community [451] but have either very primitive support or are built on low-cost equipment that cannot endure very sophisticated attacks without having software developed by an attacker [427]. Also, adjustments for different cryptographic devices under test are always needed, sometimes to a great extent. There exist a few commercial companies in this research field (Riscure [501], CRI[495]) that offer their own proprietary equipment to the prospective crypto-analysts at a prohibitively high cost, affordable only by high budget labs.

In this chapter we review the latest commercial, open source and experimental trace acquisition tools and focus on their benefits and drawbacks. We also highlight the need for a cheap, fast and automated process for sending and receiving data between a Device under Test (DUT) and a controller. We comment on the importance of efficient control loops of DUT in order to speed-up the trace acquisition process. Finally, we propose and describe a three step approach for controlling, functionally validating and SCA assessing a DUT that goes beyond the traditional control model that other solutions follow. Finally, we describe the use of our solution using an open source hardware/software trace acquisition platform that we have developed in our lab (FlexLeco project [427]) that consolidates the latest trends in trace collection tools and platforms.

The rest of the chapter is organized as follows. In Sect. 9.2, the major SCA categories and leakage assessment methodologies are described and evaluated according to the number of traces needed. In Sect. 9.3 we overview existing open source and commercial trace collection platforms and toolsets, and the proposed trace collection approach is described. In Sect. 9.4, we describe the use of the proposed approach on the FlexLeco project architecture, and discuss the benefits of the solution. In Sect. 9.4 conclusions are provided.

9.2 Side Channel Attacks, Leakage Assessment Methods and Problems

A typical side-channel attack (SCA) measurement and analysis setup consists of several components, including measuring equipment (e.g., a digital signal oscilloscope), a DUT controller component that handles attacker-to-DUT communication as well as a Personal Computer (PC). The PC is used by an attacker/evaluator for providing input to the controller and for analyzing, with the help of programming tools, the collected leakage trace measurements by applying signal processing techniques on them [390]. Possible additions to this classic SCA measurement collection and analysis setup could be some kind of pre-amplifier to boost the signals acquired by the measuring equipment (e.g., an oscilloscope), a differential cable to help reduce acquired measurements noise, electromagnetic probes for acquiring

electromagnetic emanation signals, and some kind of tool that will speed up leakage trace capturing.

There are a large variety of DUTs that can be assessed for their SCA resistance using the above setup. Under the ubiquitous computing framework, small embedded system devices (e.g., RFID tags, wireless sensor network nodes, smart card devices), having lightweight versions of cryptographic algorithms, are the most prominent candidates to have SCA vulnerabilities. Assessing the SCA resistance of such devices can be achieved by evaluating the device when it is deployed and is fully operational (in that case the DUT is the whole device) or can be achieved by evaluating individually, in a controlled environment, a specific cryptography or security implementation that is meant to be deployed on the ubiquitous device. In the first case, SCA assessment is very hard to perform since, apart from security functions, on the DUT there are many operations, unrelated to security, that are being executed in parallel [219]. Such operations can be considered as hard-to-remove noise inside the collected traces [219]. In the second case, the DUT is not the full ubiquitous device but a specific security hardware or software implementation deployed on this device. No other operations are executed in the second case control environment, thus noise is minimized, enabling the evaluator to test many SCA resistance scenarios in depth.

When trace collection is needed for SCA analysis or assessment, the attacker/evaluator needs to pass the cryptography/security algorithm's expected input data each time from his control point (usually a personal computer) to the control component, which is responsible for sending the data to the DUT for one security/cryptography process to begin. After having set the right settings in the oscilloscope (e.g., sampling rate, time window capture, resolution capture) the attacker arms it. The attacker then sends a command to the DUT, for it to start performing the evaluated security process. Before the start of the process the DUT sends a triggering signal to the oscilloscope, warning it that the process is about to start and a trace capture must be performed. As soon as the trigger signal reaches the oscilloscope, it captures a leakage trace measurement (e.g., power consumption or the electromagnetic emanation, depending on the used probes) of the DUT. The attacker/evaluator then requests from the oscilloscope the captured trace for analysis at his PC. When the captured trace reaches the PC, various signal processing techniques are applied on it and it is used for side-channel analysis of the DUT. The above procedure (called loop round) is repeated for each new security/cryptography process we want the DUT to perform.

9.2.1 Side Channel Attack Categories

Adopting the formulation approach described in [60, 61, 220] we can model each security/cryptography computation C as a series of n different O_i operations (for $i \in \{0, 1, \dots, n-1\}$) that each require inputs X_i (thus $O_i(X_i)$). We can also assume that each operation output can be considered as input to another operation during

the C computation. Each operation O_i is linked to an information leakage variable L_i . A side channel attack is possible if there is some secret information s that is shared between O_i and its leakage L_i . The ultimate goal of a side channel analysis is, by using some strategy, to deduce s from a series of information leakage L_i values [220]. To achieve that we collect leakage traces L which are sequences (in time) $L = \{L_0, L_1, \dots, L_{n-1}\}$ and try to associate them with the computation C as a sequence of operations $C = \{O_0, O_1, \dots, O_{n-1}\}$.

SCAs follow either the vertical or horizontal leakage collection and analysis strategy, as originally described in [60]. In the vertical approach, the implementation is used N times using either the same or different inputs each time in order to collect leakage traces. Each trace is associated with the j -th execution of the computation. In the horizontal approach, leakage traces and related analysis is collected/performed from a single execution of the computation and each trace corresponds to a different time period within the time frame of this execution. As expected, in horizontal attacks the implementation input is always the same.

By associating and distinguishing specific patterns of O_i operations (consisting of a single or multiple operations) in a leakage trace L , an attacker can recover the secret s . Such an attack approach is typically followed in simple SCAs (SSCAs) that are mostly horizontal attacks meaning that they are mounted using a single leakage trace that is processed in time. Vertical SSCAs that need more than one leakage trace rely on comparative analysis for pattern matching between computations with different inputs. Vertical SSCAs require a few hundred leakage traces to be collected considering that averaging technique for noise reduction is applied between leakage traces of the same computation.

There exist several countermeasures that can thwart SSCAs, thus, more advanced attacks have been devised (i.e., Advanced SCAs (ASCAs)). ASCAs do not focus only on single operations or series of operations (e.g., O_i) but also on the computation operands (i.e., the operation inputs X_i) [61, 220]. ASCAs are focused on a specific subset of the calculation C (e.g., some O_i operations that are strongly associated with s) and examine how this subset behaves over a large collection of (e.g., N) leakage traces L_j associated with computations C_j with different inputs (where $j \in \{1, \dots, N\}$). ASCAs on this subset of operations and associated leakage traces, exploit the statistical dependency between the calculation C for all X_j and the secret s . ASCAs follow the hypothesis test principle [60, 335] where a series of hypothesis \hat{s} on s (usually on some bit j of s i.e., $\hat{s}_j=0$ or 1) are made and a series of leakage prediction values are found based on each of these hypotheses using an appropriate prediction model. The values of each hypothesis are evaluated against all actual leakage traces using an appropriate distinguisher δ for all inputs X_i so as to decide which hypothesis is correct.

Most ASCAs are of vertical nature and their success is highly related to the number of processed leakage traces. The most widely used ASCA vertical attack is Differential Attack (DSCA) originally proposed by Kocher in [336], which was later expanded into the more sophisticated Correlation SCA (requiring fewer traces to reveal the secret than DSCA) [21] and the collision correlation attack [99, 210, 425], which can be mounted even if the attacker does not have full control of the

implementation inputs. There are, however, also horizontal ASCAs that apply differential or correlation statistics on a single leakage trace assuming that a subset of operations associated with the secret s appear many times in this trace. Finally, ASCAs can bypass strong SCA countermeasures (e.g., randomization/blinding) by combining horizontal and vertical approaches [59, 210, 220].

Following the above categorization (Vertical vs. Horizontal, SSCA vs. ASCA), we can include profiling SCAs like Template Attacks or Machine Learning Attacks [367] among the vertical ASCAs. Profiling attacks operate in two phases: Initially, they need to collect a series of leakage traces from a device under the full control of the attacker (with known inputs and secrets) so as to create a leakage model. In the second phase, the leakage model is used as a template or as the training set of a machine learning algorithm in order to recover a secret from a series of traces collected from a device (similar to the one used for profiling) not under attacker control.

9.2.2 Leakage Assessment Using t -Test

In addition to SCA resistance assessment based on the success of various SCAs, a generic methodology for finding information leakage from a DUT has been gaining ground. The dominant, generic, leakage assessment methodologies are based on Student's t -distribution following specific and non-specific t -tests [67, 516]. The goal is to detect any type of information leakage that occurs during the computation of security/cryptography functions in the DUT, at a certain n -th SCA order. [An SCA attack of order n appears when there exists an n set of intermediate variables that appear during the computation of the algorithm, such that knowing a few key bits (in practice fewer than 32 bits) allows us to decide whether two inputs (respectively two outputs) give the same value for a known function of these n variables.] Any sensitive computational intermediate operation O_i series that appears on the side channel as significantly different from random noise can potentially be detected using the above leakage assessment approach without conducting any specific SCA. This significant difference is enough to mark a DUT implementation as leaky and SCA insecure.

Test Vector Leakage Assessment (TVLA) is one of the most promising, generic, non-specific leakage detection techniques, initially proposed by Cryptography Research (CRI) [67]. The method is practically used as the first action towards assessing a system's SCA leakage. It consists of a univariate test that is performed on a series of traces obtained from a DUT. The DUT implementation is evaluated as non-leaky if the test throughout the duration of the DUT trace remains below a certain threshold, independently of the leakage model that might be used. More precisely, we test the case where there is no leakage (null hypothesis) versus the case where there is some leakage at a certain intermediate point $L(t)$ at time t . Let n_{tr} be the number of traces that the evaluator collects and n_s the number of samples in each trace. Following the notation of [597], and assuming that we have

N traces of n_s samples each, let $\mathbf{L} = \{L_1, \dots, L_{n_s}\}$ be the leakage traces of a security/cryptography implementation, having mean values \bar{L}_i . For the traces null hypothesis to be true the expected leakage value \bar{L}_{exp} should be the same as the measured value, if there is no leakage, i.e., $\bar{L}_{exp} = \bar{L}_i \quad \forall i \in \{1, \dots, n_s\}$. In practice, to perform the TVLA we conduct two experiments. In both experiments, we collect $N/2$ traces where the variable to be leakage assessed has a known, fixed, value and $N/2$ traces where this variable has a random value. This trace collection is done in an interleaved way by randomly choosing to acquire either a fixed trace or a random trace each time. After collecting the necessary traces, we perform a Welch's t-test as described in [516] and check the outcome against a threshold.

TVLA and all similar leakage assessment tests can provide an initial indication regarding SCA leakage but require huge amounts of collected traces in order to provide an accurate assessment result. In the case of symmetric key cryptography algorithms (e.g., in AES implementations) this number is in the order of millions (in [516] $N = 100M$ traces for an AES evaluation). This number is reduced in Public Key algorithms (in the order of thousands of traces) yet still is hard to collect since each public key algorithm implementation trace consists of a very large number of samples n_s . From a trace collection perspective, TVLA is a very slow assessment method due to this high N number, and can become very frustrating for an SCA evaluator.

9.2.3 Practical Considerations in SCA Trace Collection

When, in practice, the above described SCA attacks are applied to actual, raw collected traces using real hardware or software implementations, their success rate is very low. This happens since in a trace, leakage information is mingled with a considerable amount of SCA-useless signals that we can consider here as "noise". Noise, as very accurately described in [411], can be external, intrinsic, quantization or algorithmic. External noise that is picked up from external to the DUT sources as well as intrinsic noise, due to the DUT's physical characteristics (capacitance, conduction, transistor, non-linear behavior, etc.), are not under the control of the attacker and the DUT SCA resistance designer, and must be removed or reduced by some appropriate trace preprocessing technique. On the other hand, quantization noise can be considerably reduced by using better trace collection equipment (with small A/D quantization for example). Algorithmic noise is usually designer infused on a DUT in an effort to increase randomness in data processing but also infused by computation functionalities unrelated to security, such as interrupts, pipelining, OS system calls, etc., these can very often appear in ubiquitous devices.

In order to practically make a successful SCA, noise cancellation techniques must be applied during or after trace collection in order to have clear and useful leakage traces. Traditional noise reduction techniques can be applied on traces after collection based on low-, band- or high-pass filtering after finding dominant frequencies using, for example, Fast Fourier Transform analysis or based on trace

resampling where sample windows are specified and they are replaced by their average value [501]. However, many researchers apply the averaging technique to increase the signal-to-noise ratio and mount a successful attack. Using averaging, we collect T leakage traces with the same inputs and secret, then average them sample-per-sample to come up with a single, averaged leakage trace that contains a lot less noise than each individual trace itself. The technique is very popular (almost mandatory) in embedded system SCAs to remove noise, but it increases the needed number of collected traces by a factor of T since we need T trace collections for each SCA useful, averaged, leakage trace.

Another practical factor to be considered during trace collection, which also has an impact on the number of collected traces, is trace misalignment. This phenomenon happens when, after triggering a security/cryptography computation, the various O_i operations do not always appear at the same point in time in all leakage traces of this computation, even if the same inputs and secret are used. Misalignment can appear frequently in software based security computations (a common case in ubiquitous devices) since such computations run in parallel to other security-unrelated processes, or random events may happen that influence the computation execution sequence. Misalignment can also appear in hardware-implemented security computations when related SCA countermeasures are introduced in the computation flow. To solve the problem, SCA theory states that traces should be realigned using postcollection trace processing techniques [390]. However, in practice, for several cases of ubiquitous devices, traces are so misaligned that they become SCA useless and cannot be effectively realigned. Thus, there is a percentage of collected traces that due to misalignment should be discarded. This makes it imperative to collect more traces than those needed, having in mind that some of them will be useless due to misalignment. A rule of thumb in such cases is to collect 20% more traces than needed. This percentage can increase to 50% in highly misaligned traces (usually on ubiquitous software implementations).

9.3 Side Channel Attack Trace Collection Platforms

There exist several Security Test Labs and individual researchers who have proposed and manufactured their own ad hoc hardware boards [399, 532, 565] for trace collection. In the years following the discovery of SCAs, several Side-Channel Analysis measurement boards and evaluation setups emerged in the security community. The purpose of such boards is to provide a common platform for the aspiring attackers to mount their attacks and help them get low noise measurements in an easy way. Typically, they accommodate a general purpose device (a microprocessor, an ASIC, or an FPGA) serving as the DUT, connected with a controlling device (control component) on the same board (mainly some sort of microcontroller). There were also boards that accommodated signal-enhancing mechanisms on the same board to ease the oscilloscope's work [117]. Gradually the quality of the boards improved to such a degree that several of them found their way to the market for commercial use, with significant success.

There is limited variety of commercial SCA boards, each with its pros and cons. The Cryptographic Engineering Research Group (CERG) presented the FOBOS board [565, 566], which consists of two FPGA's, one for handling control tasks and one for the implementation of the cryptographic algorithm. The FOBOS board contains a Digital Clock Manager that produces frequencies in the range of 31.25 kHz up to 50 MHz for the Victim Board. The communication with the Control FPGA is performed through a software control loop that is responsible for transmitting the appropriate commands and the necessary data to the victim FPGA. The software loop is also responsible for the communication with the oscilloscope (currently supporting only Agilent oscilloscopes). Unfortunately, the FOBOS approach relies on old FPGA boards, is not capable of automated multiple trace collection through the provided control loop, is only applicable to a specific type of oscilloscope, and involves the PC in each trace collection (part of the control loop) which considerably burdens the collection speed.

The Tamper-resistance Standardization Research Committee (TSRC) released two trace collection hardware boards [399]. The first is INSTAC-8 which uses an 8-bit microcontroller and the second is INSTAC-32 with a 32-bit microcontroller as the control component and an FPGA chip as the DUT. Unfortunately, very limited information is provided on the clock frequencies and the communication methods that those two boards employed. Similarly to FOBOS, they also featured some custom-made software loops for the communication between the users and the victim chips. An evolution of the above approaches is the Sakura-Sasebo Project [354], which provides several choices regarding measurement boards. Initially, low-end-based chip boards were implemented, leading to the Sasebo-G, Sasebo-GII and Sasebo-W solutions, which for several years constituted the most widely used platforms for trace collection. Later, more sophisticated versions of those boards were launched, the Sakura-G, Sakura-X and Sakura-W. Both Sakura-G and Sakura-X contain two FPGAs each (a Control FPGA acting as the control component and a cryptography FPGA acting as the DUT), making them perfect for evaluation of hardware-implemented cryptographic algorithms, while Sakura-W was suitable for evaluation of smartcard security. Unfortunately, the boards are still supported by a primitive interface on the Control FPGA that enables the interfacing of a particular cryptographic algorithm with limited key-length. Also, the provided software loop in charge of data and command transmission to the Cryptographic FPGA is slow, oriented towards a specific algorithm with fixed key-length, and offers through a PC program very basic functionality only for a single trace capture per loop round.

An attempt to remedy this issue was made in the IAMeter project, which is focused exclusively on developing an efficient control loop for commercial and custom FPGA board platforms (including the Sasebo-G and GII boards) [307]. However, even this attempt can provide only a single trace collection per loop round, it is not capable of adjusting/controlling the DUT clock, and it relies heavily on PC-based configurations (including Python scripts along with MySQL databases queries) which slow the trace collection as a whole.

Recently, in an effort to provide a cheap SCA setup, the ChipWhisperer project [450] managed to provide a diverse tool-chain for embedded hardware security research. The ChipWhisperer featuring a CW1173-Lite board is suitable for capturing traces and attacking software cryptographic algorithms implemented in its Atmel XMEGA microcontroller, and it was recently upgraded to the CW1200-Pro version that offers additional ease-of-use on the process. ChipWhisperer though also provides the CW305-Artix FPGA board that specifically targets attacks on hardware cryptographic algorithms implemented in its ARTIX 7 FPGA as well as the OpenADC dongle that enables trace capturing directly from hardware devices (without the need for an oscilloscope). Both CW1173 and CW305 allow the modification of Victim's Chip frequency (e.g., the frequency range for the CW305-Artix board starts from 5 MHz and goes up to 160 MHz due to the onboard PLL (Phase Locked Loop)). The communication between the user and the Victim Chip in both boards relies on a software control loop called Chipwhisperer-Capture. This is a PC-based loop responsible for sending the appropriate data and commands to the Victim board. In the case of the CW305-Artix FPGA board the PC software loop is assisted by an onboard AVR microprocessor (partially acting as a control component).

Complete measurement and evaluation setups are provided by experienced security test laboratories with longtime presence in the hardware security community. BrightSight delivers the Sideways acquisition center. CRI-Rambus offers the DPA Workstation [495] which is a complete platform that enables the testing and analyzing of cryptographic chips and systems vulnerabilities, to power and electromagnetic (EM) side-channel attacks. Finally, Riscure launched the Inspector [501] which is an integrated test tool for side channel analysis and fault injection attacks on security devices. The above commercial setups offer SCA resistance evaluation/assessment on individual security/cryptography hardware and software implementations as well as on fully working DUT ubiquitous devices (e.g., embedded systems).

The gap between the measurement boards and the complete evaluation setups is huge when it comes to cost. Trace collection and SCA evaluation setups providing a complete package that includes measurement collection, side-channel attacks and DUT security analysis, have a considerable cost. On the other side, the cost of the measurement boards is much more affordable, nonetheless exploiting their capabilities to the full extent is not always supported by their manufacturers in terms of software or hardware tools.

9.3.1 Proposing a Fast Trace Collection Approach Beyond the Traditional Model

As the need for a huge number of traces for SCA evaluation and leakage assessment becomes considerable (in the presence of DUT with SCA countermeasures) the

traditional SCA trace collection control loop model is not fast enough for efficient, practical, trace acquisition. The main bottleneck in such control loops is the presence of a PC device for providing inputs, collecting outputs and controlling each execution of the DUT security/cryptography implementation [427]. The solutions presented in the previous subsection, although they do not manage to eliminate the need for a PC inside the control loop, clearly indicate a tendency to migrate traditional PC-related functionality to other hardware or software entities that are closely associated with the DUT. In some solutions, the control loop operations are implemented in hardware and are downloaded on a dedicated control FPGA that is physically connected to the DUT. This is done in the Sasebo/Sakura project and in some ChipWhisperer (NewAE) technologies, just to name some examples. Hardware, however, is not flexible and thus cannot be easily adapted to different algorithms or assessment techniques. Similarly, control loop functionality is partially migrated on dedicated control loop ASIC microcontrollers or microprocessors that operate alongside a PC in order to implement the DUT control in software. Such solutions lack speed when transmitting DUT test vectors to the DUT itself since PC usage is still needed.

Therefore, it has become apparent that a different, updated approach to realize DUT control loops for SCA evaluation and leakage assessment is needed. In this work, considering the previous paragraph's described hardware and software control loop limitations, we propose an approach that relies on a hardware/software co-design solution for controlling the trace acquisition process. Extending and generalizing the work in [427], we propose a three-step SCA trace collection process. For this process to be possible, the control loop is not executed on a PC but exclusively on a microcontroller that is directly connected to the DUT. The microcontroller can be an ASIC (hard-core) or FPGA based (softcore) depending on the SCA trace collection board at hand. Using software that is executable on the microcontroller we gain the flexibility of a software control loop solution. Using the microprocessor that is directly connected (through a bus interface) to the DUT we gain very high control loop speed, which is not achievable using PC-based control loops. More information on such an architecture can be found in the use case example presented below.

In the first step of the proposed control trace collection process, denoted as the design phase, the evaluator can describe in a programming or scripting language (e.g., C, Python, JavaScript), using some developed Application Program Interface (API), the SCA trace acquisition experiment that needs to be performed. The experiment includes inputs that need to be provided to the DUT, specification of the security/cryptography operations that need to be executed, the execution sequence, the delay between experiment executions (in case the experiment needs to be executed more than once) and DUT output storage. The goal of the design phase is to fully specify the inputs, parameters and execution sequences of the experiment. The outcome of this phase is an executable file that can be transmitted to the microcontroller bootloader for execution.

The second step of the proposed control trace collection process, denoted as the execution phase, is focused on the execution of the designed experiment. This

phase does not include any non-trivial transmission delays between the control loop entity (i.e., the microcontroller) and the DUT, since the bus connecting them is extremely fast, in contrast to the PC-based control loop where such transmission is done serially. During execution, the microcontroller control loop is responsible for transmitting the appropriate signals to the DUT so as to execute one or multiple times a security/cryptography operation as well as to generate appropriate trigger signals for trace collection by a DSO.

In the final step of the proposed control trace collection process, denoted as the trace processing phase, the execution of the experiment has been concluded and the experiment traces have been collected by the DSO. In this phase, post-collection operations are performed on the collected traces, like averaging and alignment but also operations related to the specificities of a particular experiment. This phase is performed on the DSO or on a PC with an appropriate digital signal processing toolbox, and it can be slow (depending on the post-collection operations that are executed). However, the performance delay is considerably smaller than when a PC is included in the control loop during an experiment.

9.4 A Use Case of a Flexible and Fast Platform for DUT SCA Evaluation

To showcase the applicability and effectiveness of the above-proposed three-step trace collection architecture and mechanism in action, we focus on the Flexible Leakage Collection (FlexLeco) project, which was recently published in [427]. The FlexLeco project was designed to match the latest trace collection challenges and to introduce a unified mechanism for applying various trace collection scenarios. It provides an architecture that tries to blend the reconfigurability of Software Control loops with the speed of Hardware Control loops. Taking advantage of the latest trace collection boards that utilize schemes with two FPGAs (one acting as the Control Unit and the other as the Device Under Test), the project created two generic hardware interfaces that enable fast communication between the two FPGAs, and managed to include an embedded softcore processor inside the Control FPGA, which is in charge of the Control Loop during the execution of trace collection scenarios. The project is currently instantiated on the boards of the Sakura/Sasebo project (Sakura-X and Sakura-G), but it can be modified to fit any board that adopts the approach of two distinct, hardware-isolated FPGA chips.

The FlexLeco architecture of Fig. 9.1, consists of two generic interfaces and an embedded softcore processor. Inside the Cryptographic FPGA a generic cryptographic interface is implemented for the communication of the Control Unit with the DUT. This interface contains two variable memory spaces (called “Hyperegisters”) that handle the inputs and outputs of the DUT.

Inside the control FPGA exists a generic control interface that is directly connected with the embedded microprocessor (a Xilinx Microblaze for the Spartan-

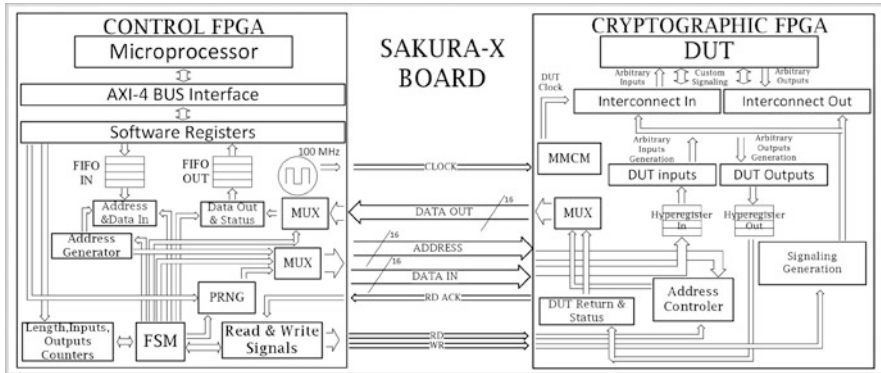


Fig. 9.1 General architecture

6 Control FPGA of the Sakura-X board), residing in the same FPGA. With this approach, the softcore microprocessor undertakes the duty of control and communication with the Cryptographic FPGA, through a hardware Finite State Machine implemented inside the control interface, which utilizes a custom hardware protocol between the two FPGAs. Data inputs to the DUT can be provided by the softcore processor, as well as from a hardware Pseudo Random Generator (PRNG) module (to support TVLA leakage assessment scenarios). The communication between the two FPGAs is performed using a 16-bit address bus, a 16-bit data bus for sending data to the crypto side and a 16-bit data bus for receiving data from the Cryptographic FPGA side.

The use of an embedded system design inside the control FPGA provides support for the proposed three-step trace collection approach thus allowing the execution of a software API on the microprocessor for the realization of the trace collection control loop. This API consists of reusable code functions that fit multiple trace collection scenarios and DUTs. Through these functions the control FPGA remains unchanged (no need for redesign or reprogram), regardless of the DUT inside the cryptographic FPGA, as it can be quickly reconfigured only by passing certain values to software registers inside the control interface. By setting code values to these registers, the control interface's Finite State Machine is ready to serve any updated cryptographic component inside the cryptographic FPGA. In this way, the control component can be permanently be downloaded inside the control FPGA's flash memory, thus negating the inflexibility issues that other hardware control loops present in their adaptability to different DUTs and scenarios.

The above-mentioned software API provides functions that, beside the initialization of the control loop, set up the leakage trace collection parameters (inputs/outputs number, bit-length, randomness), trigger encryption/decryption, send or receive plaintext/ciphertext values to FIFOs, and register and randomize

input values.¹ By doing so, the attacker/evaluator can use the API functions to create any SCA or leakage assessment scenario to be executed inside the softcore microprocessor, omitting the slow, PC-based software control loops. In that sense, the FlexLeco solution fully supports and favors the design phase of the proposed trace collection approach of this chapter and provides to an SCA evaluator all the tools required to design diverse and complex SCA evaluation experiments.

The cryptographic interface on the cryptography FPGA side is primarily designed for testing specific security/cryptography hardware implementations. It is an open source HDL (Hardware Design Language) design that can be adapted to the DUT's cryptographic algorithm specifications during design time. By simply assigning appropriate values for five HDL generic parameters, the number of inputs, outputs and their bit length is adjusted to that of the DUT's algorithm. The generic interface is synthesized and downloaded whenever a new hardware implemented cryptography algorithm is tested.

Inside the cryptographic interface, a Digital Clock Manager has been included that provides different clock frequencies to the DUT and the interface. The frequency of the interface is the same as that of the other one inside the control FPGA, while the frequency of the DUT can be clocked as high as the component's critical path and the FPGA chip's functional specifications allow (or as low as the attacker/evaluator desires). By raising the frequency of the DUT, the evaluator is now able to use DSOs with low memory size buffers, thus fitting more traces on the time interval the DSO offers (as long as the DSO's sampling frequency allows it). The DCM's output frequencies update is a straightforward process done by changing a single parameter during the cryptographic FPGA's synthesis phase.

Presenting such flexibility and scalability, the FlexLeCo mechanism allows the evaluator to perform various trace collection scenarios like a Single-Encryption, a Single-Encryption with Rapid Block Mode (if an oscilloscope with such a feature is available) and a Multi-encryption mode [427], for different DUTs and with minimum overhead between the mode updates.

During any trace-collection scenario (Fig. 9.1), at design phase, the softcore microprocessor is set up so as to initiate communication transactions with the cryptographic FPGA, in which it either reads and sends the contents of the corresponding test vector records (i.e., plaintexts) or signals a random value generation (using an API function or the hardware PRNG) and transmits it to the DUT. We can design an experiment where this procedure continues until all of the test vectors on the microprocessor's memory have been sent to the cryptographic device (DUT) or until the needed number of random inputs is reached. After the design phase, the actual experiment is executed in the softcore microprocessor and post-collection operations may be performed. As an example of such postcollection operations, we showcase the Multi-encryption scenario, detailed in [427], which is enabled in the FlexLeco project in case an RBM (Rapid Block Mode) Digital

¹Both software- and hardware-based randomization is supported through the PRNG module.

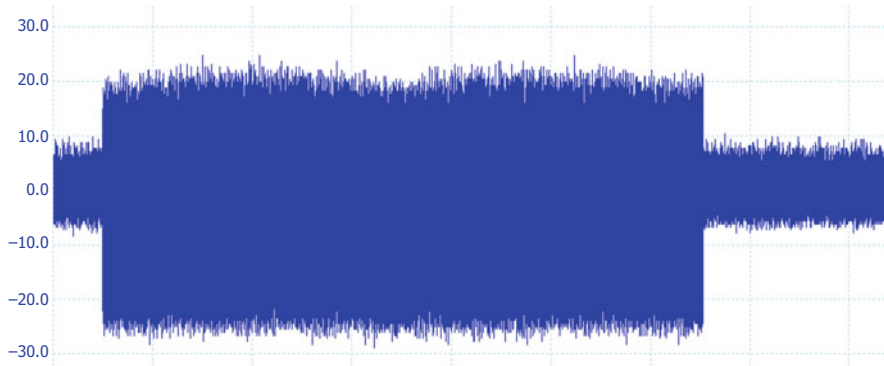


Fig. 9.2 Multi-encryption trace with 1000 AES encryptions

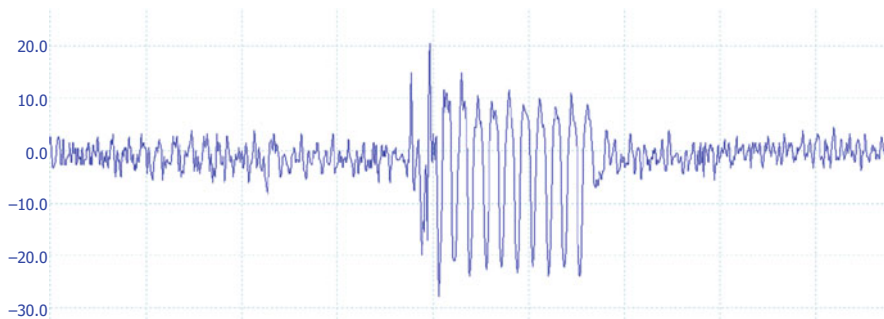


Fig. 9.3 Post-collection extraction outcome as a single AES leakage trace

Signal Oscilloscope is not available to the evaluator. During this scenario, the DSO starts capturing a continuous waveform (Fig. 9.2) of leakage traces starting from the first de/encryption and continuing for all de/encryptions until the end of the DSO's chosen time window. By setting up the appropriate time window, after execution phase, we capture a single continuous waveform that should contain the leakage traces of all the cryptographic processes we have instructed the DUT to execute. The split of this continuous multi-encryption waveform into individual single de/encryption traces is done during the trace processing phase. The outcome single encryption trace from this post-collection operation can be seen in Fig. 9.3. The whole process is considerably faster than if we tried to capture each de/encryption trace autonomously during the execution phase [427].

To qualitatively compare the proposed three-step trace collection approach of Sect. 9.3.1 as was realized using the FlexLeco project, we present Table 9.1 where our approach is compared with recent open source trace collection projects in terms of flexibility, usability and various post-collection feature supports. The presented results are collected from actual experimentation of the authors with the compared

Table 9.1 Leakage trace collection architectures qualitative features comparisons

Collection setup	Flexibility	Single encr.	Multi-encr.	Averaging	Ease-of use	Clock control
FlexLeco three step approach	High	Yes	Yes	Yes	High	5–230 MHz
Sakura/Sasebo [353]	No	Yes	No	No	None	No
ChipWhisperer [451]	Moderate ^b	Yes	Partial ^c	Yes	Moderate ^b	5–160 MHz
FOBOS [565]	No	Yes	No	No	Minimum ^a	Max. 50 MHz

^a Hardware interfaces are unique for each cryptography DUT

^bHard to implement software control loop and tedious, time-consuming interfacing of different cryptographic DUTs

^cMulti-encryption only with constant or random plaintexts

projects or from personal communication with the projects' developers. The table indicates that the three-step approach is flexible enough to rival existing and well-established solutions offered to the SCA community by Sakura and ChipWhisperer.

9.5 Conclusions

In this book chapter we focused on an important aspect of SCA analysis, evaluation and leakage assessment, which is the efficient and easy collection of needed SCA traces. We presented the traditional mechanism for collecting traces from DUT ubiquitous devices and commented on the drawbacks of this approach. After briefly describing dominant SCA attack categories and leakage assessment methodologies in view of their needed number of traces, considering also the high level of trace noise and possible misalignments that are ever present in ubiquitous devices, we concluded that the traditional model is not practically useful for ubiquitous systems SCA evaluation. To further explore the recent SCA trace collection and analysis landscape, we described the most prominent open source and commercial toolsets, both research and commercial. Most have shortcomings in terms of controlling in a flexible and easy manner the DUT to be SCA evaluated, thus giving us the motivation to propose a three-step trace collection methodology using a design, an execution, and a trace processing phase. To validate the applicability, efficiency, and ease-of-use of this proposed approach, we applied it to the FlexLeco project open source solution, which is highly compatible with our proposal. Using this use case we managed to easily design experiments and collect results even when we applied complex design scenarios, like the multi-encryption mode where multiple inputs are provided on the DUT, multiple traces are collected as one, and the actual single traces (that are usable for SCA evaluation or leakage assessment) are extracted after

postprocessing. To conclude, the evaluation of our proposal and the exploration of the recent toolset landscape indicate that there is a need for a different model for trace collection. In this model, the trace collection DUT control functionality is migrated close to the DUT (on a device physically connected to the DUT) and not on a remote control entity (like a PC).

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

