LEIA: the Lab Embedded ISO7816 Analyzer A Custom Smartcard Reader for the ChipWhisperer

Ryad Benadjila, Mathieu Renard, David Elbaze, and Philippe Trébuchet
firstname.lastname@ssi.gouv.fr

ANSSI

Abstract. Among the hardware security evaluation toolkits that are publicly available, the ChipWhisperer has become very popular. This framework based on low-cost open hardware platform and open source software, allows performing Side-Channel Analysis (SCA) like Simple Power Analysis (SPA), Differential Power Analysis (DPA) or Correlation Power Analysis (CPA) on Integrated Circuits (ICs). Unfortunately, previous attempts to integrate a smart card reader compatible with the ChipWhisperer are limited. In order to allow the community to study such targets which are ubiquitous in the world of security, but have a specific form factor, we present through this work LEIA, an open source and open hardware victim board for the ChipWhisperer allowing the evaluation of targets in the smart card's format.

1 Introduction

The ChipWhisperer [13, 36] is a project with the ambition of providing an affordable open source toolchain for side-channel power analysis and glitching attacks. Many academic papers present it as a reference evaluation platform [14, 32, 35]. ChipWhisperer is intended to be an improvement over older boards like for instance, SASEBO [23] and SAKURA [22]. Due to its relatively low price (from 250 $ to 3,800 $ depending of the kit) and good capture performances, it has become very popular in the recent years. Moreover, this framework is an open hardware platform and comes with an open source SDK.

The ChipWhisperer is very versatile: out of the box, it offers various targets (MCUs or CPUs) to put under scrutiny. However, no smart cards are currently publicly available as targets, at least with a plug-and-play integration. Even though some attempts have been made to integrate a preliminary support in old versions of the ChipWhisperer, they failed to make it to the current stable and upstream version. In order to allow the community to easily study such ubiquitous (but with a specific form factor) targets, we present the design of LEIA, a CW308 UFO daughter

board for smart card security assessment that is compatible with the ChipWhisperer.

This paper is organized as follows. First, the ChipWhisperer ecosystem is depicted in section 2. Then, the specificities of smart cards and the ISO7816 standards are summarized in section 3. The challenges faced during the hardware design of LEIA are discussed in section 4. The software architecture and the additional functionalities added to the ChipWhisperer SDK are detailed in section 5. Finally, the ASCAD [1,41] use case (an open side-channel attacks database using an open source smart card platform) is introduced in section 6 as a testing and validation vehicle.

2 The ChipWhisperer framework

The ChipWhisperer project has both academic foundations [37] and industrial ambitions with efficient manufacturing process. It aims at targeting various actors of hardware security:

— The students, by providing an affordable framework to learn the basics of side-channel analysis (SCA) and fault attacks (FA). It helps teachers offer practical exercises built on a complete ecosystem (all the hardware and software are provided and *plug-and-play*).
— The researchers, by helping them to reproduce experiments made by others, and by making their experiments reproducible by other researchers too.
— The embedded systems makers, by offering them a fast prototyping development platform.

By allowing to play most of the hardware security tests on Integrated Circuits (IC), it gives the end user an easy way to apply the recent research publications and validate the published results. This evaluation step is essential in order to improve the security of embedded systems.

Among the proposed set of features, the ChipWhisperer allows users to measure the *power consumption* of ICs [33]. The captured traces can be then used to realize various power analyses such as Simple Power Analysis (SPA) [30], Differential Power Analysis (DPA) [30] or even Correlation Power Analysis (CPA) [18].

It can also be used in order to inject faults by generating voltage glitches [15–17, 19, 20, 28, 29].

The project is composed of two parts, detailed in the next sections :

— The hardware platform: a control board and the targets.
— The software framework, which consists of all the scripts allowing to control the hardware and run the SCA and FA algorithms.

2.1 The ChipWhisperer hardware

Figure 1 shows examples of hardware kits as they are sold. They are detailed in the sequel.

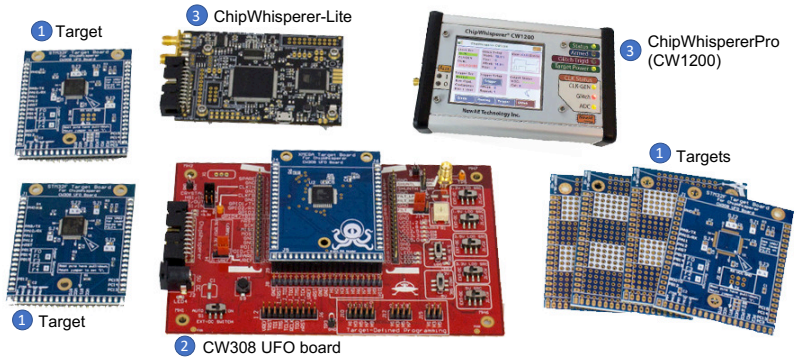

Fig. 1. ChipWhisperer kit composition.

Control and measurement device. The ChipWhisperer-Lite (figure 1.3) is a multipurpose board, providing a power analysis capture instrument, and a power supply glitching module. It is also supplied with an embedded target (XMEGA microcontroller). Both are on the same PCB but can easily be splitted in two different parts. Then, the user can connect the control part of the PCB to analyse another target. The board consists in an FPGA for generating signals (clock, data, trigger), a micro USB port enabling the configuration of the board with a computer, a 20-pin connector which allows to connect other target that the one provided (see the Targets paragraph).

The ChipWhisperer-Pro (figure 1.3) is an upgraded version of the ChipWhisperer-Lite. It includes a larger sample buffer, streaming-mode captures and a touchscreen interface, all of that in a fancy box. These features make it a device targetting laboratory use.

Targets. Both controller boards can be connected to what ChipWhisperer calls a *target*. The target is the component being studied. It can be a microcontroller, an FPGA, or any other IC. The connection between the controller board and the target is normalized and is a 20 pins connector simply named 20-Pin ChipWhisperer Connector.

NewAE, the company that sells the ChipWhisperer, provides various targets on its website, XMEGA (CW303, the one shipped with the ChipWhisperer-Lite), ATMEGA328P (CW304), FPGA (CW305), etc.

CW308 UFO board. The CW308 UFO board can be seen as a generic backplane allowing to evaluate multiple targets. It does not embed any FPGA neither is supposed to perform any computations on itself. It offers multiple power supplies (From 1.2V to 5V), drives a clock signal, provides an easy access to JTAG signals (if they are routed through the target board) and a SMA connector to measure with the ChipWhisperer (or an oscilloscope) the power consumption. As every ChipWhisperer Target, it also has the 20-Pin ChipWhisperer Connector.

The CW308 UFO board expects its daughter boards, called *victim boards*, to be plugged in a dedicated connector, the `U` connector, that can provide many things to the target:

Power supply: the board can deliver multiple power supply levels; 1.2 V, 1.8 V, 3.3 V and 5 V. Each can be selectively activated by positioning the corresponding jumper. A power filter functionality is available on the board, to improve power supply quality.

JTAG: As many victim boards are microcontrollers that can be reprogrammed, a JTAG connector is available on the CW308 UFO board and afferent pins are reserved on the connector.

GPIO: The board exposes different GPIO of the victim. 10 GPIOs are available through some pin headers and three others are connected to different LEDs.

Icc measurement: One can find on the board a current sensing resistance connected to a SMA. It can be used with the ChipWhisperer Capture module or an oscilloscope to measure the power consumption of the victim.

Clock generation: The board contains a crystal oscillator driver. This allows to use a standard crystal to drive either the victim board or the connected ChipWhisperer. Furthermore, this also allows the use of the CW308 stand-alone, as it is possible to have almost any frequency by connecting the appropriate crystal onto the board.

2.2 ChipWhisperer software stack

The software stack of ChipWhisperer is split into two main parts: the firmware of the victims, which are mainly C or C++ embedded code, and the scripts that run on the PC and drive the measurements (these are mainly Python scripts).

ChipWhisperer victims's firmwares. They can be found on the github repository of ChipWhisperer. Firmwares are split into several parts: the Hardware Abstraction Layers (HALs) [4], which are libraries that provide a generic interface above specific hardware and the actual algorithm under scrutiny.

Using the HAL concept, the same code can run on multiple targets without much effort from the developer side.

The HAL abstraction layers for the CW308 UFO are located in the `chipwhisperer/hardware/victims/firmware/hal` and the actual code in the `chipwhisperer/hardware/victims/firmware` directory.

As presented in Section 4, LEIA is designed as a CW308 daugther board. All development including the LEIA firmware code will be pushed to the official ChipWhisperer repository.

ChipWhisperer scripting SDK. The SDK on the PC host side has probably helped to democratize the use of ChipWhisperer: it offers a simple Python library that can be used to script signal acquisition or glitch attacks. This SDK allows to:

— drive the control module (ChipWhisperer-Lite and Pro);
— communicate with the targets (most often by using an UART and a simple dedicated protocol named "SimpleSerial");
— realize simple attacks like SPA, DPA or DFA.

Script examples can be found on a dedicated part of the repository [3].

2.3 ChipWhisperer attempts to support smart cards

There have been previous attempts to support smart card targets with the ChipWhisperer. However, such attempts are either obsolete and not compatible with upstream ChipWhisperer, or are limited in some ways. We explain hereafter such limitations.

CW301. the CW301 Multi-Target board [7] is an old version of the ChipWhisperer board that is obsolete since 2016. It offered an embedded smart card connector present on the board, whose I/O is driven by an FPGA [2] and the clock is produced by a fixed oscillator at 3.579 MHz. This setup is not very flexible: the clock is not configurable, and the VHDL stack is inherently not flexible (it only supports a small subset of smart cards). For all these reasons, over and above its deprecation, the CW301 solution seems to be out of scope when considering a modern alternative for the ChipWhisperer project.

CWLite (CW1173). the more recent ChipWhisperer-Lite does not embed a physical smart card connector, but pins are present to handle the communication bus on headers J6 and J7 [6]. J6 is connected to the FPGA and uses the VHDL smart card stack described with CW301 [2]: it inherits from its multiple limitations. J7 is connected to the Atmel SAM3U MCU of the board (the one handling the USB communication with the host) : it uses a more flexible and versatile software stack [5]. Although, this last alternative appears to be a good building block for a smart card SCA test bench, we stress out that it suffers from some limitations. First of all, on the hardware part: measuring power consumption can be tricky, as explained in section 4. Isolating the consumption of the smart card chip from the one of the MCU driving the communication with it is challenging (but necessary when clean and non-noisy measurements are needed). Additionally, the SAM3U software driver [5] suffers from some limitations, and does not cover all the quirks of the ISO standard covering smart cards.

For all these reasons and limitations, we have decided to *develop our own solution LEIA*: a victim board compatible with the ChipWhisperer ecosystem. In order to have as much control as possible on the software and hardware aspects of the board, we have decided to build this solution from scratch with the ChipWhisperer compatibility constraints as input hypothesis.

3 Smart card and ISO7816

Smart card is a generic term usually used for embedded small electronic devices with a standardized form factor. The embedded ICs (usually secure ICs) in such devices are very compact, which allows embedding them in a plastic shell with reasonable dimensions. The rationale is to have a small yet secure electronic device in users' wallets and pockets.

This compact and portable form allows very versatile usages: banking for payments, authentication and identification (ID cards), telecommunications (SIM cards), healthcare (French "Vitale" cards), etc.

They usually exist in two flavors: contact and contactless cards. Contact cards use physical connectors to communicate with a so-called reader, exchanging data through a physical layer detailed in the sequel. The ISO7816 set of standards provides all the necessary specifications.

Contactless smart cards communicate with a reader using NFC (Near Field Communication), meaning an over the air (but at a small distance) communication channel. Although in their logical layers, such cards share

similarities with contact cards, the physical layer (described in ISO14443 standards [9]) is very different. The current article and the LEIA framework *only focuses on contact cards.*

3.1 Smart cards: electrical specifications (ISO7816-1/2)

The IC embedded in the card must ensure tight constraints: regarding its size, of course, but also regarding power consumption and Input/Output characteristics.

In order to standardize such constraints, the ISO/IEC standardization body has developed a series of international standards around contact smart cards. ISO7816-1 [24] presents the physical characteristics, while ISO7816-2 [25] addresses the dimensions and location of the contacts. Even though they have been amended, first versions of the documents date back to the end of the eighties decade, which proves how much this technology is time-tested.

The electrical constraints of the smart cards capture the need to have low energy systems: the chip communicates with the "outside world" using at most eight pins as presented in figure 2. These metallic pins are exposed over the plastic with quite large connectors, and are connected to the underlying metallic layer with the bonding wires rounding the internal chip. Here is a brief description of the pins as per ISO7816-2 specifications:

- **VCC**/Pin C1: this pin is an *input* of the chip and handles the power line feeding the IC. The ISO7816 standard provides three fixed voltage levels: 1.8 V (class C), 3.3 V (class B) and 5 V (class A). Legacy smart cards are usually supplied in 5 V, and newer chips tend to accept the three voltages. Specifically, for critical power consumption reasons, SIM cards are designed to be powered by at 1.8 V power supply.
- **GND**/Pin C5: this pin is an *input* connected to the ground.
- **RST**/Pin C2: this pin is an *input* handling the reset signal. Even though such a signal can be directly connected to the hardware reset signal of the embedded chip, modern ICs actually perform a polling of the signal and implement a software reset.
- **CLK**/Pin C3: this pin is an *input* handling the ISO7816 clock. More information is provided in the next sections about the specific properties of the clock. As for the RST signal, CLK used to directly clock the IC on legacy chips without an internal clock, but this is no more the case: this signal is sampled and handled in software by CPUs running at a higher frequency.

— **I/O**/Pin C7: this pin is an *input/output* line for a *bidirectional* communication between a host (the smart card reader) and the smart card itself. The communication is based on a half-duplex protocol with byte-based transmission. The I/O line is in a high state when not driven (i.e. pulled up). We provide more details on this in the section dedicated to ISO7816-3.
— **VPP**/Pin C6: this pin is a special bit, since it can both be used for standard or for proprietary nonstandard usages. Therefore, it can be both *input* and/or *output*. Usually, such a pin has been used for chip erasure and reprogramming: applying a voltage on C6 would allow reprogramming the embedded EEPROM, acting as a charge pump.
— Pins C4 and C8: these pins are marked RFU for future use.

Fig. 2. ISO7816 pins.

3.2 Smart cards: ISO7816 protocol (ISO7816-3/4)

The ISO7816-3 [26] specification defines the communication layer (both physical and logical) over the connectors described in section 3.1. It can be seen as the OSI physical and link layers). The ISO7816-4 [27] standard is built upon it to deal with the application level (OSI application layer).

The current section briefly presents the key concepts necessary to understand the challenges of building an ISO7816 compatible software stack.

The standard defines two communicating entities: the "card" (i.e., the smart card), and the "interface device" (i.e., the reader). In the sequel, we use these terms interchangeably. The protocol is asymmetrical, the reader is *master* whereas the card is *slave*. This means that all communications must be initiated by the reader.

ISO7816-4 APDU and RESP. At the logical level, the communication unit that the reader can send is called an *APDU* (for Application Protocol

Data Unit). Each APDU sent by the reader expects a response APDU, that we denote *RESP*, from the card. The ISO7816-4 standard provides the exact format of APDU and RESP (see figure 3):

— Command APDU, or APDU: it is composed of a mandatory header of four bytes `CLA`, `INS`, `P1`, `P2`, and a conditional body made of an optional `Lc` field, optional `Data` payload, and an optional `Le` field.
— Response APDU, or RESP: it is composed of a conditional body of `Data`, followed by two mandatory status bytes `SW1` and `SW2`.

`Lc` encodes the number of bytes sent to the card (i.e. the size of `Data` payload). `Le` encodes the expected number of bytes to get back from the card, with 0 encoding 256 bytes (maximum size).

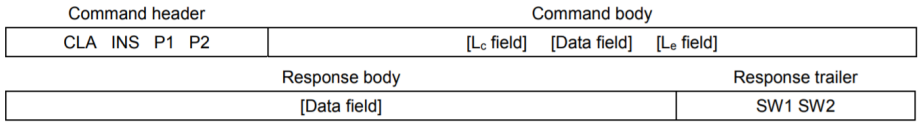

Fig. 3. APDU and RESP structures.

The ISO7816 standard distinguishes four types of APDU/RESP cases:

— Case 1 APDU: the command does not send data and does not expect data back from the card. `Lc` and `Le` are then absent.
— Case 2 APDU: the command does not send data, but expects data back from the card. `Lc` is absent, `Le` is present and encodes the data size, data payload is present in RESP.
— Case 3 APDU: the command sends data, but does not expect data back from the card. `Lc` is present and encodes data sent size, `Le` is absent.
— Case 4 APDU: the command sends data and expects data back from the card. `Lc` and `Le` are both present.

The formats of APDU presented here are so-called *short ones*. The data payloads are at most 255 bytes for the APDU, and 256 bytes for the RESP. A standard amendment has introduced *extended APDUs* where the `Data` payload is extended to 65,535 bytes for the APDU and 65,536 bytes for the RESP. In this case, `Lc` and `Le` are encoded on 0 to 3 bytes.

As already stated, APDUs and RESPs are logical views that the application level sends and receives on the line. Such commands and responses must be sent over a physical line using a dedicated transmission protocol. In this article, we focus on the ISO7816-3 way of dealing with the physical and link layers, as it is the most common way for contact

smart cards. Other ways (out of scope here) to handle this layer are NFC for contactless cards, USB bus provided by the USB specifications, etc.

The data packets transported by the transmission protocol to send an APDU are called **TPDU** for Transmission Protocol Data Units.

ISO7816-3: driving the I/O line. As already stated, the I/O line serves as a half-duplex channel where both the reader and the card send/receive data. The line is naturally driven to a high state (denoted H), and the sender must pull it down explicitly to a low state (denoted L). The receiver then samples this line state during a fixed number of clock cycles, and deduces a value.

The communication is character oriented, meaning that the character is the atomic element exchanged between the two parties.

Exchanging one character is performed in 10 "moments" as shown on figure 4, encapsulating one byte (8 bits) and one parity bit of actual data.

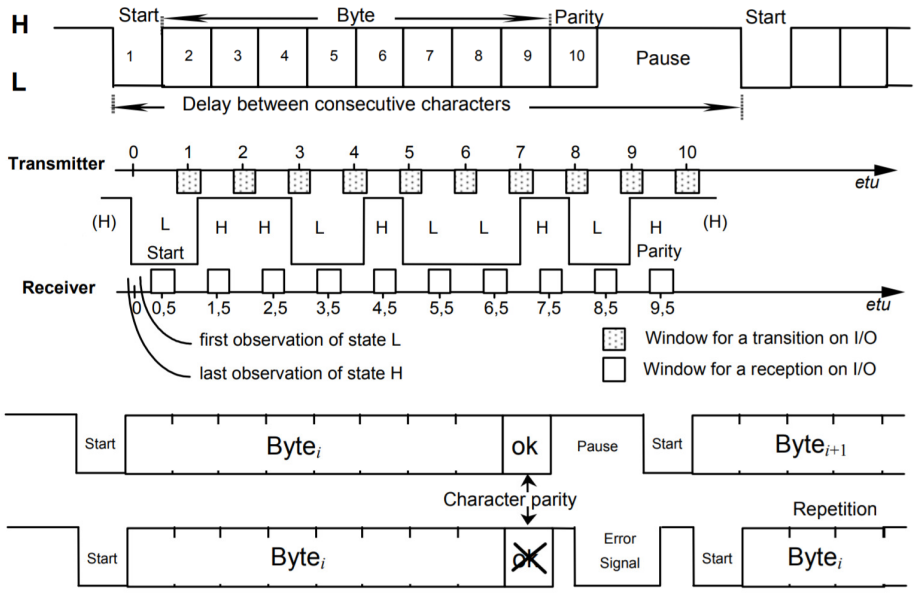

Fig. 4. The I/O line and characters transmission.

A character transmission is composed of four main phases:

1. A *start* bit in moment 1. The start bit is detected (by the receiver) when the line is forced to a low state during one moment (by the sender).

2. The eight bits of the *data byte* are sent during moments 2 to 9 (the interpretation of these bits depending on the state is discussed later in this section).
3. The parity bit of the data byte is sent during moment 10.
4. Finally, the sender releases the I/O line to the high level, and waits for the receiver to signal an error if necessary (usually in case of reception error detected with a bad parity bit). The "pause" time should be at least one moment.

In case of a signaled error, the sender is expected to send the same byte again. The way the bits are interpreted when received depends on a convention set by the card itself during the ATR (more on this later). Two conventions exist for the bit order and the polarity:

— Direct convention: a high state encodes a 1 bit, a low state encodes a 0 bit, the bits in the data byte are LSB (Least Significant Bit) first. For instance, the sequence from moments 2 to 10 `HHLH HHLL H` encodes the value `0x3B` with a parity bit set to 1.
— Inverse convention: a high state encodes a 0 bit, a low state encodes a 1 bit, the bits in the data byte are MSB (Most Significant Bit) first. For instance, the sequence from moments 2 to 10 `HHLL LLLL H` encodes the value `0x3F` with a parity bit set to 0.

We have so far talked about moments, without providing any time-related definition. The ISO7816-3 standard provides a specific definition named the **ETU** (Elementary Time Unit). Hence, one moment is exactly one ETU. In order to compute the ETU, the standard uses three physical quantities:

— f: the clock signal `CLK` frequency.
— F: the clock rate conversion integer.
— D: the baud rate adjustment integer.

At any time, the ETU is computed with the formula 1 ETU $= \frac{F}{D} \times \frac{1}{f}$ in seconds, or simply put one ETU is $\frac{F}{D}$ clock cycles of `CLK`. An interesting thing to notice here is that the clock frequency could be variable, the only value meaningful for the protocol being the ETU sampling timings.

The ATR (Answer To Reset). The possible f, F and D values fixing the ETU are described in the standard and are set up, and optionally negotiated, during a step called the *Answer To Reset*, or ATR. Other important timing-related values, such as *guard times* between characters, *waiting times* to be able to detect a timeout condition, and so on, are also fixed during this phase.

The ATR is happening just after a reset performed through the `RST` pin. Since no negotiation between the reader and the card is performed yet, ISO7816-3 fixes standard values to be used: $F = 372$ and $D = 1$, meaning a default ETU of 372 clock cycles. Typical guard times must also be respected by the sender and the receiver.

The ATR consists of a bunch of at most 33 characters sent by the card to the reader, and its variable length depends on the options the card proposes to the reader. The reader has to dynamically *parse* the bytes received during the ATR to decide if more bytes follow or not.

The first byte of the ATR is `TS`, it can be either `0x3F` or `0x3B` and encodes the convention (inverse or direct). Then, the `T0` (the format character) is conveyed; it encodes the presence or not of following optional interface characters. These interface characters are the ones encoding possible negotiable elements between the reader and the card (more on this in the PPS paragraph). After the interface characters the so-called historical characters (between 0 and 15 bytes) are optionally sent. Finally, a "check byte" is optionally sent as a checksum of the whole ATR.

The PPS (Protocol and Parameters Selection). After the ATR phase where the card is the sender and the reader the receiver, the reader is left the opportunity to potentially negotiate various elements with the card. Such a negotiation, called the PPS (for Protocol and Parameters Selection), mainly allows to negotiate the protocol (T=0, T=1 and so on), the baudrate and maximal frequency (F, D and f_{max} parameters, only a fixed and limited number of $\{F, D, f_{max}\}$ triplets are allowed by the standard), various timings (guard times, timeouts, etc.) as well as parameters that are specific to the chosen protocol.

Whenever the ATR and the PPS steps are over with both parties agreeing on the exchanged parameters, the "nominal" phase begins and the reader can send to the card APDU commands while expecting RESP responses back.

The TPDUs are the ways APDUs and RESPs are encoded over the physical I/O line, i.e., what are the bytes exchanged to encode such APDUs and RESPs. The ATR and PPS encoding allows to support up to 16 transmission protocols T=0 to T=15, but only two of them (T=0 and T=1) are standardized in ISO7816-3. T=2 is in the process of being standardized by ISO, but is not yet out (it is based on T=1 with a full-duplex flavor). Other values are either reserved for future use by the ISO standard, or dedicated to proprietary/national usage on specific cards [1].

1. For instance, T=14 is used by Deutsche Telekom for the card-phone system.

The T=0 protocol. The T=0 transmission protocol is the original one specified by ISO7816-3 in the eighties, which explains why T=0 is implicit during the ATR phase.

It is half-duplex and *byte oriented*, meaning that the smallest unit exchanged between the reader and the card is a byte. In order to send an APDU on the line, the protocol splits it in two parts as show in figure 5:

— The header: it is composer of five bytes. Four byte are the `CLA`, `INS`, `P1` and `P2` from the APDU. The fifth byte, named `P3`, is context dependent and encodes either the number of outgoing bytes `Lc` from the reader to the card, or the expected data back from the card in the response `Le`.

— The data part: this represents the actual data payload of size `Lc`.

Sending an APDU is performed in at least three steps: first sending the header, getting an acknowledgment from the card (one byte), and then sending the rest of the data. When handling the response, the card sends various so-called *procedure bytes* (the acknowledgment byte being one of them). Using such interleaved procedure bytes allows to handle various cases such as time extensions requests (when the card is performing an long computation and wants to extend the ISO7816 standard timeouts).

At its basic level, T=0 only handles Cases 1, 2 and 3 APDUs since `P3` encodes *either* incoming or outgoing data size. The way Case 4 APDUs are encoded actually involves the application layer described in ISO7816-4 using specific application level commands `GET_RESPONSE`. Such an OSI layers mixing, mainly inherited from the byte oriented protocol limitation, induces many restrictions: T=0 is hardly robust against errors, and does not have any "session" notion. This is why the standard has introduced a newer T=1 transmission protocol during the nineties.

The T=1 protocol. It is half-duplex and *block oriented*, meaning that the smallest unit exchanged between the reader and the card is a block, conveying both payload data and protocol control messages.

All the exchanged blocks in this protocol have a common structure presented in figure 6:

— A prologue field: 1 byte `NAD` (node address in order to support multiple slaves/cards), 1 byte `PCB` for the block type (more on this below), 1 byte `LEN` encoding the length of the embedded data (the ISO7816-3 specifies lengths ≤ 254 bytes).

— An information field: it contains the encapsulated APDU data.

— An epilogue field: it contains a checksum on optionally one or two bytes (the checksum type is fixed by the ATR and PPS).

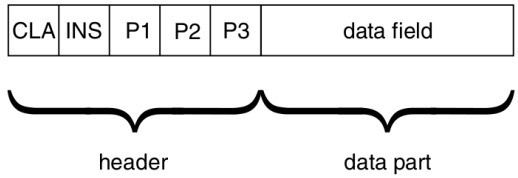

Fig. 5. T=0 header and data.

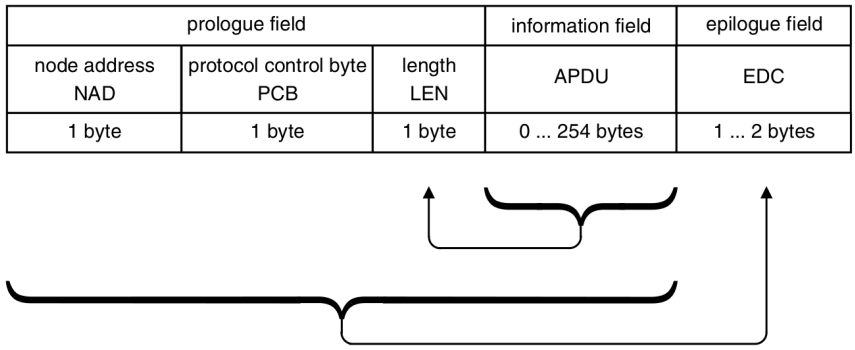

Fig. 6. T=1 basic block structure.

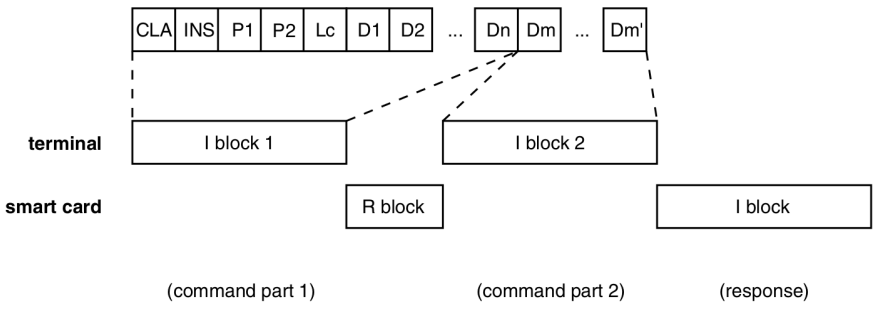

Fig. 7. T=1 block chaining.

There exist three types of blocks in T=1: I-Blocks, R-Blocks and S-Blocks. I-Blocks are transporting data information, i.e. the APDU payload. R-Blocks are control messages dedicated to positive or negative acknowledgments (for instance in case of bad checksum in the epilogue field). S-Blocks are control messages related to the protocol itself such as resynchronization, wait time extensions, etc. As we can see, error detection and synchronization have been part of the design of the protocol, which makes it much more reliable than T=0.

When sending APDUs or receiving RESPs that exceed 254 bytes, they must be split across multiple blocks. The T=1 protocol has a builtin feature called *chaining*; it uses metadata in the `PCB` byte to allow it. Figure 7 provides an example of such chaining: the reader splits an APDU in two I-Blocks. The first I-Block is sent, the card responds with an R-Block for positive acknowledgment, and then the reader sends the second I-Block to complete the transfer. Finally, the card sends its RESP in an I-Block.

3.3 Smart cards: ISO7816-4 and above

As we have already explained, ISO7816-4 handles the application layer of the protocol. However, because the OSI layer separation is not clear for T=0, ISO7816-4 and ISO7816-3 are intermixed. As discussed in the software section of the document, this means that in order to implement a proper ISO7816 stack to communicate with various smart cards, ISO7816-3 and *parts of* ISO7816-4 must be implemented. Specifically, only the parts related to APDUs (Case 1 to 4) of ISO7816-4 are of interest.

This standard also specifies a *file system* and binary storage of objects on the card as well as access rights. Secure messaging is also a part of the specification. However, these elements are not necessary to implement a core ISO7816 driver whose sole purpose is to send and receive APDUs. Furthermore, they can be implemented in a much easier fashion **host PC side** by formatting the proper APDUs and sending them to the reader that handles the low-level communication. By extension, this is also the case for (most of) other ISO7816-x standards above 4 since they take place in the application layer.

4 LEIA hardware design

In order to allow the community to compare results of attacks on smart card platforms, LEIA has been designed to be compatible with the CW308 UFO board. The CW308 UFO board is a generic main board

for attacking all sorts of embedded targets. It can be used as a stand-alone bench (with an external oscilloscope for the measures) or with the ChipWhisperer-Capture hardware provided with the ecosystem. Moreover, it is annouced by NewAE [2] as the new standard form factor for the targets supported by this device [8].

4.1 CW308 UFO target board

The CW308 UFO daughter boards (commonly named Targets, or Victims) are usually designed for attacking a target with a *single chip*. However the LEIA board differs from this classical model: it embeds both an interface to the ISO7816 standard which is used to communicate with the target, and the target itself. The LEIA board is presented on figure 8. For simplicity and clarity, we use the term **Reader** to designate the ISO7816 interface *smart card reader* and the tested smart card is denoted the **Target of Evaluation (TOE)**.

The ISO7816 protocol is complex, as shown in section 3, and the above architecture allows abstracting this protocol out of the controlling logic: the controlling hardware (would it be the ChipWhisperer or not) does not drive directly the TOE but communicates with it through the Reader which implements the ISO7816 stack and hides its complexity.

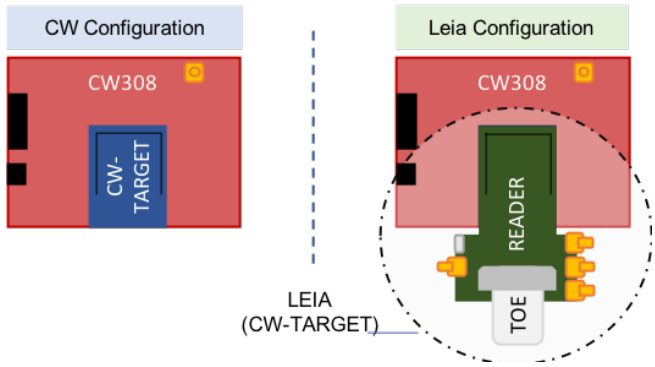

Fig. 8. Comparison between standard targets and the LEIA target.

The desired compatibility with the CW308 UFO board comes at the expense of some restrictions regarding the architecture, functionalities, communication protocol and hardware design:

2. The company branding ChipWhisperer.

— The first limitation is that the form factor of all new targets must fit with the CW308 UFO connector. As a result the geometry and the distribution of the signal and power lines are fixed. Notice that the ChipWhisperer device has been designed for providing both powering and measurement and so the connector imposes both power and I/O line distribution. This is all the more problematic as, in order to reduce crosstalk [3] between signals [4], both the TOE and the Reader signals shall be well distributed across the board and the respective power lines shall have to be properly split.
— Voltage selection provided by the CW308 UFO board is useful if one wants to study different targets requiring different power supply levels. The good point is that the CW308 UFO board provides the 1.8 V (class C), 3.3 V (class B) and 5 V (class C) levels which permit to work with all the standard smart cards.
Nevertheless, the nowadays design can be improved by providing a galvanic isolation between the TOE and the capture hardware (the ChipWhisperer-Capture or an external oscilloscope) on one side, and the control workstation on the other side. This will help to filter out unwanted signals (eg. ground loop [5]) and will limit the probability of a hardware breakdown.
— Concerning the JTAG port provided by the CW308 UFO board, within the context of smart card, it is left unconnected as smart cards do not have JTAG compatibility [6].
— The same reasoning can be applied to the GPIO. As smart cards usually do not have any GPIO beyond the ISO7816 pins, those available upon the CW308 are not used.
— Due to LEIA complexity, and in order to provide the best measurement quality, we choose to embed on LEIA the shunt resistor for power consumption measurement. More details are provided in the next sections.
— Given the fact that the clock used with smart card can change over the time, we can not use the one provided by the board. As detailed in the next sections, the STM32 of LEIA's reader uses its

3. Crosstalk is any phenomenon by which a signal transmitted on one channel creates an undesired effect in another channel.

4. To provide a clean acquisition chain and reduce post capture cleaning operation.

5. A ground loop occurs when two points of a circuit both intended to be at ground reference potential have a different potential.

6. The SWD protocole is used in the context of LEIA to reprogram the smart card *Reader* firmware, but through a dedicated connector for the sake of galvanic isolation between the capture and the control domains.

own internal clock, and generates the ISO7816 clock used by the smart card.

— Regarding the communication between the ChipWhisperer and the TOE, the SDK imposes to drive the daughter board through the UART interface using a specific protocol. This is why we have added to LEIA a dedicated MCU as a proxy translating commands from the UART to ISO7816 packets to and from the TOE.

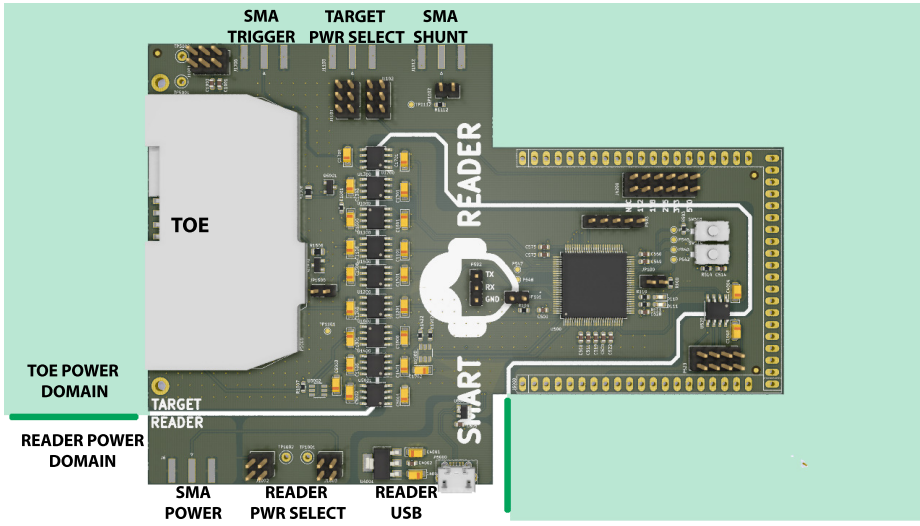

Fig. 9. Top view of smart reader target: Power Domains.

4.2 Hardware specifications

The functional specifications lead to natural design choices and/or requirements when it comes to the hardware platform.

The microcontroller at the heart of the design must be able to communicate with the TOE. The communication must be based on the ISO7816 protocol either using an internal hardware unit or using a software implementation driving GPIOs (so-called *bit banging*). It must have an UART interface available in order to communicate with the ChipWhisperer measurement device. Since the platform design will be open sourced, all components and their data-sheets must be publicly available.

We detail in the next sections the rationale behind our specific choices for the hardware components.

4.3 LEIA microcontroller choice

The Reader interface is designed around a microcontroller of the `STM32F4xx` family: the `STM32F439`. These Systems On Chip (SoC) are based on 32-bit ARM Cortex-M cores: their main advantage is to embed a plethora of versatile hardware modules in a compact form factor.

Using a microcontroller with an embedded firmware appeared as the optimal choice to implement the ISO7816 specification with respect to the flexibility, possible reuse, and the convenience for the community to modify/improve the software stack. This is to be compared with using a Field-Programmable Gate Array (FPGA) whose disadvantage is less flexibility in software development and contribution.

The `STM32F439` is able to run up to 196 MHz. It has a built-in USART mode for accelerating *ISO7816* (more on this in the software section 5), *UART* and many other communication modules. It also features a cryptographic coprocessor (the CRYP engine) as well as a TRNG (True Random Number Generator) that might prove useful for testing. The high frequency of the embedded Cortex-M4 and its numerous provided I/O lines guarantee that it will not hinder further evolution both on the software and hardware sides; and that the platform is future proof for potential extensions and new revisions. Moreover, it also has an integrated Full Speed USB PHY (12 Mb/s capable) and can achieve High Speed (480 Mb/s) using an external PHY. This last feature is not necessary in order to interface the device with the ChipWhisperer but can be useful for the development phase, for a standalone mode of the Reader, as well as for future improvements.

4.4 Designing a measurement device

Designing a measurement device implies preserving signal integrity, which is why the loss associated with PCB transmission lines constitutes an important topic [21, 31, 38, 40].

Low frequency signals are almost unaffected with parasitic responses, unless the transmission medium is particularly long. However, many parasitic effects become visible at high frequency, and even short lines can suffer from problems such as crosstalk, ground loop, etc. This seriously hampers the response of the signal and damages its integrity. These problems can be overcome by good design practices and by following simple layout guidelines. Here are the PCB layout rules that have been applied to the whole design:

— Current-carrying traces should be as thick and short as possible.

— The board should have a low impedance ground.
— The TOE should have its own power circuit including ground (GND) and power (VDD) plans.
— Sensitive signals should be shielded from noisy traces.

PCB layout. We designed the LEIA board on a classical 4-layer stack-up: 2 signals layers, 1 ground plane (GND) and 1 power plan (VDD). As both VDD and GND are planes, this will provide a very low ground impedance path to the microcontroller and help to reduce Electromagnetic Interference (EMI) [39, 43] that could interfere with the signal under scrutiny. In addition, we flooded other layers with closely spaced vias (VIA stitching) in order to keep the impedance low. PCB vias become inductive at high frequencies and will therefore increase the ground impedance. Having multiple closely spaced vias in a plane will reduce this effect as the parasitic inductances are in parallel [43].

The PCB material is a classical FR-4 PCB laminate material which is commonplace in the electronic industry. It is a cost effective solution for most digital designs (allowing to convey signals up to $2.5 - 3$ GHz range). The maximum frequency of our design is far from the 2.5 GHz limit, even if we update our design to use the USB High Speed interface (480 MB/s). The full description of the PCB material and stack-up specification can be found on OshPark [7] website [11].

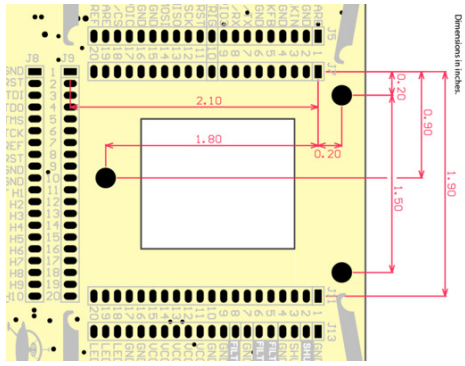

Fig. 10. CW308 UFO connector specification.

The geometry of the PCB is mostly imposed by the CW308 UFO motherboard since LEIA will be interfaced through the CW308 UFO

7. Oshpark is a company that produces bare printed circuit boards. Their commercial model is focused on the needs of prototyping, hobby design, and light production.

connector (see figure 10). Nevertheless, due to the galvanic separation between the TOE (power and signals lines) and the Reader we have diverged from the standard target rectangle board shape. The resulting shape of our study is a `T` shaped PCB as presented on figure 9.

Isolated power domain and signal transmission. Power domain isolation (uncorrelated VDD *and* GND for the TOE and the Reader) helps to prevent EMI propagation between different parts of a circuit. It also prevents ground loops. This is why the LEIA PCB is split into two parts as shown on figure 9. However, isolating power domains adds complexity to the transmission chain between the TOE and the Reader. In addition, the load impedance of the signals and measurement lines is matched to 50 Ω [34, 42]. This guarantees to transfer the maximum amount of power and preserves the quality of the signals.

Signals isolation. In order to allow the communication between the TOE and the Reader which have separate power domains, we use galvanic isolation. We have based the isolation on an optocoupler-based setup. By definition optocouplers are mono-directional. While this not an issue for monodirectional lines (clock, reset, power, and ground), it becomes a problem for the I/O line that is bidirectional (for half-duplex transmission). The solution we adopted is to use two optocouplers arranged head-to-tail and two multiplexers. The first optocoupler allows transition from the Reader to the TOE and the second provides the reverse path. The signal is then routed to one or to the other path using two multiplexers both driven by the Reader. Of course, on the target side the multiplexer is not driven directly by the STM32, but through a dedicated optocoupler. The switching between the transition direction is handled by the Reader using a GPIO. After each smartcard command the IO line is reconfigured. This slows down the communication but it is not an issue since the Reader is *master* and is the one imposing the clock frequency to the smart card. Moreover, the ISO7816 f_{max} (20 Mhz) is only a *maximum* value that does not prevent using lower values. Nonetheless, it is important to keep in mind the ISO7816 f_{max} when choosing the optocouplers because most of them are designed for lower communication frequencies. It is also interesting to check the maximum propagation delay that will be added to the output signal. With all these elements we decided to build our galvanic isolation around ten `FOD8001` produced by ON Semiconductor / Fairchild. These optocouplers support up to 25 Mbit/s data rate and have a 40 ns maximum propagation delay with a 6 ns maximum pulse width distortion.

The UART signals (RX,TX) used for the communication between the *Reader* and the controlling device (in other words the ChipWhisperer capture) are also isolated using the `FOD8001` optocouplers.

4.5 Smart card interface and signal acquisition

Side-channel attacks are divided into two phases: signal acquisition and signal statistical analysis. The quality of the acquisition of side-channel signal is essential to get proper results during the analysis phase. This is why the acquisition circuitry must be designed with great care.

Over time, many different methods of acquisition and exploitation of side-channel signals have been experimented. Among them are the traditional power signals measurement, the electromagnetic signals (using dedicated probes), time-based information (to exploit timing attacks), etc. On the ChipWhisperer the leakage signal used for the side channel signals capture is the power consumption in the form of current sensing on the power lines.

Current sense. When sensing current, the designer can choose to place the sensors (usually a serial resistor on the power line) either between the supply voltage (VCC) and load, or between the load and ground. The former is called *high-side sensing* whereas the latter is called *low-side sensing*.

High-side sensing has the advantage that the load is directly connected to the ground GND. In other words, there is no change on the load side except a small power drop due to the current sensor on the VCC line.

Nevertheless the main disadvantage is that we have to use a differential probe to measure the current. In low-side sensing the current is sensed in the ground return path (GND) of the power line to the monitored load. This has the advantage to produce a ground referenced measured signal but the load is no more directly connected to GND.

In our design the TOE, the Reader and the ChipWhisperer can have separate power domains, since we do not want the noise produced by the power switching supply of ChipWhisperer to be visible in the measurements. We choose to use a *low-side* measurement as it minimizes the amount of hardware to support all the power domains.

Shunt Resistor. The shunt resistor is the element that is used in a circuit to redirect currents around the measuring device. The addition of a shunt resistor induces a voltage drop at the maximum current rating.

This is why the value of the Shunt Resistor must be selected carefully. Important parameters include the resistance tolerance, the power rating and the temperature coefficient:

— The power rating indicates the amount of electrical power that the resistor can dissipate at a given ambient temperature without being damaged nor changing the resistor parameters.
— The temperature coefficient describes the relative change of resistor value according to the temperature.
— Resistance tolerance is the accuracy the constructor guarantees on the component's characteristics.

ISO7816 Class A devices, which are the most power-consuming devices among smart cards, can draw at most 160 mA for 400 ns and continuously draw at most 60 mA. We want the voltage drop at maximum current to be at most 50 mV for not disturbing nominal working of the TOE whatever class tho TOE belongs to. Thus we decided to use a 0.1 Ω resistor with a tolerance of 1%, a temperature coefficient of 300 PPM/C and a power rating of 100 mW. It is a widespread, easily available component that meets our needs. This resistor, as it induces a maximum voltage drop far from the limit, allows us to get clean measurements.

Connectors and measurement. In order to provide high quality measurements, we use SMA End Launch Connectors since they offer reliable broadband performance from DC to 18 GHz with low reflection and constant 50 Ω impedance.

5 LEIA software design

5.1 Implementation rationale

The project aims at providing an implementation of the ISO7816 protocol on the STM32 microcontroller.

As described in section 3, the implementation of the protocol is mainly split in two complementary parts: the physical and transmission protocols layers as described in ISO7816-3; and the APU layer which is specified by some parts of the ISO7816-4 standard. Some specific cases mix those layers. For instance, Case 4 APDUs require `GET_RESPONSE` commands, and extended APDUs require more elaborate commands defined at the application layer.

These elements are implemented in the STM32 driver, and the *host PC* communicating with the custom smart card reader can implement all

the application-level logic to forge APDUs and send them to the STM32 that acts as a proxy formatting the necessary TPDUs. Actually, such a strategy is the same as the one adopted by industrial smart card readers that offer a PC/SC [12] interface: many high-level features such as Secure Messaging or on-card file system management are performed in a software stack on the PC (on Linux, the PC/SC daemon, the OpenSC middleware and so on forge APDUs that are sent to a reader through a standardized driver over USB or a serial link).

5.2 STM32F439 ISO7816 hardware: the physical layer

The STM32F439 line of products has USART hardware modules with a so-called *smart card mode*. The idea of such a feature is to provide to the developers some help on the physical layer side. Since the I/O line described in ISO7816-3 is very close to classical UART, using the same hardware IP for both seems natural.

When using the USART in smart card mode, the characters transmission on the I/O line is fully handled by the hardware. Hence, the start and stop bits, the parity bit and so on, become transparent to the software layer. Parity errors are signaled to the software stack by either polling a status register or by dedicated interrupts. Similarly, the clock is automatically generated by the hardware module once the proper baudrate is set in the dedicated registers.

This means that on the software side, the driver will only have to configure the clock frequency through its baudrate, and bytes send and receive primitives are almost “free”. Of course, all the remaining logic and automatons of the ISO7816 standard have to be implemented in software.

On a side note, it is also possible (and planned to be integrated to LEIA) to implement both the clock generation and I/O line handling in a *bit banging* fashion, using only GPIOs toggling. Thanks to the high frequency of the STM32F439 MCU, sampling the I/O line in software is not a problem (196 MHz allows sampling at around 40 MHz easily, which is far more than enough for ISO7816). This approach has the advantage of fully controlling the way bits are transferred on the line, as well as dynamically adapt the clock frequency if necessary (which might prove tedious to realize using the USART hardware module).

5.3 STM32 driver features

In order to have a clean separation between the physical layer and the ISO7816 logical automatons, we have decided to split the driver in two parts:

— A low-level driver implementing the driver that sends characters on the I/O line, handles the clock, and handles the time measurement primitives (for timeouts and so on). This can be either USART accelerated or bit banged.

— A high-level driver that implements ISO7816 ATR parsing, PPS negotiation, and the transmission protocols T=0, T=1. This layer is not adherent to the underlying hardware and is **portable** across various architectures. It only expects a specified API with the low-level driver.

ATR parsing as well as the T=0 and T=1 protocols are fully implemented (although some parts of T=1 for handling S-Blocks still require some work). An early version of the PPS protocol is also implemented. There is also a support for extended APDUs both in T=0 and T=1. Due to the open source aspect of the project, we hope that all these features will improve with time using the community feedbacks and tests with new and various smart cards.

5.4 Testing the stack

In order to validate our software stack in versatile conditions, we have tried to test it with different smart cards of the industry. Around 30 models from various manufacturers (NXP, Feitian, Gemalto, banking cards, etc.) have been successfully tested. Both T=0 and T=1 are represented in this panel, with some cards supporting both (and T=0 or T=1 is negotiated with PPS) and some only supporting exclusively T=0 or T=1.

Of course, *this does absolutely not mean that our software is bug-free*, and it is in constant improvement when testing new cards with new behaviors (possibly on the edge of the standard). We hope that its open source aspect will help this improvement with external contributions.

5.5 The compatibility with the ChipWhisperer SDK

The ChipWhisperer SDK comes with its own software ecosystem, and beyond the hardware compatibility described in the hardware design section 4, we must also ensure a software compatibility.

When adding a new daughter board in the ChipWhisperer project, the SDK must be adapted in order to integrate it both for the firmware and for the SDK Python code. This way, the host and the daughter board can exchange data according to a predefined protocol. This one is very specific to LEIA and allows to use all the features previously listed and can be easily extended. In this protocol, the host sends commands to the Reader which then answers. A command is a simple list of bytes sent through the UART line:

1. The host sends the byte command (1 byte).
2. The host sends the payload size (4 bytes).
3. The host sends the payload (up to $2^{32} - 1$ bytes).
4. The reader sends the response size (4 bytes).
5. The reader sends the response (up to $2^{32} - 1$ bytes).

The available commands are the following :

`n` With this command, the host can force different parameters to the reader during PPS negotiation:
— The frequency of the ISO7816 clock.
— The ETU.
— The preferred protocol `T=0` or `T=1`.

`a` This command does not take any argument, so the payload size should be set to `0x0000`. The reader answers with the ATR.

`s` This command sends an APDU to the reader. Due to the diversity of APDU/TPDU formats given by the ISO7816 standard (T=0, T=1, extended APDUs, etc.), we choose to encapsulate the APDU in a format that can be clearly interpreted by the reader and correctly formatted. This flexibility allows to run some ISO7816 compliance tests.

`t` This command allows to set the trigger strategy. Currently, the trigger handling in ChipWhisperer is quite straightforward: for the AES analysis, the trigger is set high just before calling the algorithm in the victim. For smart cards power consumption analysis things are a bit more complicated since the target is a “black box”, and different triggering strategies can be adopted. One would prefer to trigger at the end of the command APDU, or at the beginning of the RESP response from the card, or at any other interesting timing of the protocol. We extend the public ChipWhisperer SDK to handle such triggering strategies.

This protocol is implemented in both the firmware of the MCU of LEIA, and in the Python SDK of ChipWhisperer. Since the Firmware SDK of ChipWhisperer includes the official `STM32F4` HAL, it requires almost no effort to integrate the protocol and the ISO7816 stack.

Concerning the Python SDK, its flexibility makes adding a new protocol or a target quite easy. The main divergence with the upstream version of the SDK is on the PC host side. Because our STM32 on the daughter board is not the TOE target but only an APDU proxy, the ChipWhisperer framework is updated in order to handle this specific feature. This is possible since the SDK is open source and open to pull requests.

5.6 Standalone mode

The fact that the communication between the daughter board and the host is performed through an UART serial line makes it possible to use LEIA without the CW308. This specific use case can be useful when the user wants to make some tests on the smart card and the way it handles the ISO7816 protocol. The way the APDU commands are abstracted on the STM32 makes it easy to send various commands (even invalid ones according to the ISO7816 standard). Even if it is not the core contribution of LEIA, it helps to check the compliance of ISO7816 slave stacks, and allowed us to find small divergence and quirks with some cards.

6 Use case: ASCAD

In a purpose of testing LEIA, we reproduce the publicly available database on the public 8-bit ATMega8515 target with a software AES implementation described in [41]. Because of a lack of space in the article, the results are detailed in the extended LEIA article [10].

7 Conclusion

This article presents LEIA, a low-cost, highly modular smart card reader which is compatible with the ChipWhisperer-Pro (CW1200) and the ChipWhisperer-Lite. The combination of the ChipWhisperer and LEIA enables reliable Side-Channel Analysis (SCA) like Simple Power Analysis (SPA), Differential Power Analysis (DPA) or Correlation Power Analysis (CPA) on smart cards.

Beyond the mere signal acquisition platform, LEIA offers a custom ISO7816 platform that could be used in an independent fashion (specifically a versatile software stack).

We aim at sharing LEIA with the research community with the hope that it becomes an educational platform, and serves as a comparison basis for open research on smart cards.

Finally, we must emphasize on the fact that countermeasures exist to protect against side-channel analysis. Certified secure ICs are available, and we encourage industries to base their development on these platforms.

References

1. ASCAD Database. `https://github.com/ANSSI-FR/ASCAD`.
2. ChipWhisperer FPGA smartcard driver. `https://github.com/newaetech/chipwhisperer/blob/develop/hardware/common/hdl/smartcard/`.
3. ChipWhisperer SDK scripts. `https://github.com/newaetech/chipwhisperer/tree/develop/software/scripting-examples`.
4. ChipWhisperer victims firmwares HAL. `https://github.com/newaetech/chipwhisperer/tree/develop/hardware/victims/firmware/hal`.
5. CW1173 ChipWhisperer-Lite SAM3U ISO7816 stack. `https://github.com/newaetech/chipwhisperer/blob/develop/hardware/capture/chipwhisperer-lite/sam3u_fw/SAM3U_VendorExample/src/scard/iso7816.c`.
6. CW1173 ChipWhisperer-Lite/8-Pin SmartCard Connector. `https://wiki.newae.com/CW1173_ChipWhisperer-Lite/8-Pin_SmartCard_Connector`.
7. CW301 board. `https://wiki.newae.com/CW301_Multi-Target`.
8. CW308 UFO Target. `https://wiki.newae.com/CW308_UFO_Target`.
9. ISO14443. `http://nfc-tools.org/index.php/ISO14443`. Accesed: 2019-02-01.
10. LEIA: the Lab Embedded ISO7816 Analyzer, A Custom Smartcard Reader for the ChipWhisperer (extended paper). `https://www.sstic.org/2019/presentation/LEIA_the_lab_embedded_iso7816_analyzer/`.
11. OSHpark 4 Layer Prototype Service. `https://docs.oshpark.com/services/four-layer/`.
12. PS/SC Workgroup. `https://www.pcscworkgroup.com/`.
13. ChipWhisperer. `https://newae.com/tools/chipwhisperer/`.
14. Hyunjin Ahn and Dong-Guk Han. Multilateral White-Box Cryptanalysis: Case study on WB-AES of CHES Challenge 2016. *IACR Cryptology ePrint Archive*, 2016:807, 2016.
15. Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Annual international cryptology conference*, pages 513–525. Springer, 1997.
16. Johannes Blömer and Volker Krummel. Fault based collision attacks on AES. In *Fault Diagnosis and Tolerance in Cryptography*, pages 106–120. Springer, 2006.
17. Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of checking cryptographic protocols for faults. In *International conference on the theory and applications of cryptographic techniques*, pages 37–51. Springer, 1997.

18. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
19. Gian-Carlo Cardarilli, F Kaddour, A Leandri, Marco Ottavi, Salvatore Pontarelli, and Raoul Velazco. Bit flip injection in processor-based architectures: a case study. In *On-Line Testing Workshop, 2002. Proceedings of the Eighth IEEE International*, pages 117–127. IEEE, 2002.
20. Chien-Ning Chen and Sung-Ming Yen. Differential fault analysis on AES key schedule and some countermeasures. In *Australasian Conference on Information Security and Privacy*, pages 118–129. Springer, 2003.
21. Robin Getz and Bob Moeckel. Understanding and eliminating EMI in Microcontroller Applications. *National Semiconductor*, 1996.
22. Hendra Guntur, Jun Ishii, and Akashi Satoh. Side-channel attack user reference architecture board SAKURA-G. In *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, pages 271–274. IEEE, 2014.
23. Yohei Hori, Toshihiro Katashita, Akihiko Sasaki, and Akashi Satoh. SASEBO-GIII: A hardware security evaluation board equipped with a 28-nm FPGA. In *The 1st IEEE Global Conference on Consumer Electronics 2012*, pages 657–660. IEEE, 2012.
24. ISO Central Secretary. Identification cards – Integrated circuit(s) cards with contacts – Part 1: Physical characteristics. Standard ISO/IEC TR 7816-1:1987, International Organization for Standardization, Geneva, CH, 1987.
25. ISO Central Secretary. Identification cards – Integrated circuit cards – Part 2: Cards with contacts – Dimensions and location of the contacts. Standard ISO/IEC TR 7816-2:1999, International Organization for Standardization, Geneva, CH, 1999.
26. ISO Central Secretary. Identification cards – Integrated circuit cards – Part 3: Cards with contacts – Electrical interface and transmission protocols. Standard ISO/IEC TR 7816-3:2006, International Organization for Standardization, Geneva, CH, 2006.
27. ISO Central Secretary. Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. Standard ISO/IEC TR 7816-4:2013, International Organization for Standardization, Geneva, CH, 2013.
28. Chong Hee Kim. Differential Fault Analysis against AES-192 and AES-256 with Minimal Faults. In *FDTC*, pages 3–9, 2010.
29. Chong Hee Kim. Improved differential fault analysis on AES key schedule. *IEEE transactions on information forensics and security*, 7(1):41–50, 2012.
30. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, Berlin, Heidelberg, 1999. Springer-Verlag.
31. F. B. J. Leferink and M. J. C. M. van Doom. Inductance of printed circuit board ground planes. In *1993 International Symposium on Electromagnetic Compatibility*, pages 327–329, Aug 1993.
32. Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

33. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

34. Mehdi M Mechaik. An evaluation of single-ended and differential impedance in PCBs. In *Quality Electronic Design, 2001 International Symposium on*, pages 301–306. IEEE, 2001.

35. Erick Nascimento, Łukasz Chmielewski, David Oswald, and Peter Schwabe. Attacking embedded ECC implementations through cmov side channels. In *International Conference on Selected Areas in Cryptography*, pages 99–119. Springer, 2016.

36. Colin O'Flynn. A framework for embedded hardware security analysis. 2017.

37. Colin O'Flynn and Zhizhang (David) Chen. ChipWhisperer: An OpenSource Platform for Hardware Embedded Security Research. In *IN: CONSTRUCTIVE SIDE-CHANNEL ANALYSIS AND SECURE DESIGN - COSADE*, 2015.

38. Henry W Ott and Henry W Ott. *Noise reduction techniques in electronic systems*, volume 442. Wiley New York, 1988.

39. Tom Page, Paul Baguley, and Dirk Schaefer. Maintaining Electromagnetic Compatibility (EMC) Through Design for Fabrication and Assembly of Printed Circuit Boards (PCBs). In *ECAD/ECAE2004–1st International Conference on Electrical/Electro¬ mechanical Computer-Aided Design & Engineering*, pages 101–105. University of Bath, 2004.

40. Vishram S Pandit, Woong Hwan Ryu, and Myoung Joon Choi. *Power integrity for I/O interfaces: with signal integrity/power integrity co-design*. Pearson Education, 2010.

41. Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. *IACR Cryptology ePrint Archive*, 2018:53, 2018.

42. M. S. Sharawi. Practical issues in high speed PCB design. *IEEE Potentials*, 23:24–27, 2004.

43. Xiaoning Ye, DA Hockanson, Min Li, Yong Ren, Wei Cui, James L Drewniak, and Richard E DuBroff. EMI mitigation with multilayer power-bus stacks and via stitching of reference planes. *IEEE Transactions on Electromagnetic Compatibility*, 43(4):538–548, 2001.