View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by HAL-EMSE

SMART SECURITY MANAGEMENT IN SECURE DEVICES

Bruno Robisson, Michel Agoyan, Patrick Soquet, Sébastien Le Henaff, Franck Wajsbürt, Pirouz Bazargan-Sabet, Guillaume Phan

To cite this version:

Bruno Robisson, Michel Agoyan, Patrick Soquet, Sébastien Le Henaff, Franck Wajsbürt, et al.. SMART SECURITY MANAGEMENT IN SECURE DEVICES. PROOFS: Security Proofs for Embedded Systems, Sep 2015, Saint-Malo, France. <emse-01232670>

HAL Id: emse-01232670

https://hal-emse.ccsd.cnrs.fr/emse-01232670

Submitted on 23 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Smart Security Management in Secure Devices

Bruno Robisson[1], Michel Agoyan[1], Patrick Soquet[2], Sébastien Le Henaff[2], Franck Wajsbürt[3], Pirouz Bazargan-Sabet[3], and Guillaume Phan[4]

[1] CEA/EMSE, Secure Architectures and Systems Laboratory, Centre de Microélectronique de Provence, 880 Route de Mimet, 13541 Gardanne, France
bruno.robisson@cea.fr
[2] Viaccess-Orca, Les Collines de l'Arche, 92057 La Défense, France
[3] LIP6, 4 Place Jussieu, 75252 Paris, France
[4] Trusted Logic, 6 rue de la Verrerie, 92190 Meudon, France

Abstract. Among other threats, secure components are subjected to physical attacks whose aim is to recover the secret information they store. Most of the work carried out to protect these components generally consists in developing protections (or countermeasures) taken one by one. But this "countermeasure-centered" approach drastically decreases the performance of the chip in terms of power, speed and availability. In order to overcome this limitation, we propose a complementary approach: smart dynamic management of the whole set of countermeasures embedded in the component. Three main specifications for such management are required in a real world application (for example, a conditional access system for Pay-TV): it has to provide capabilities for the chip to distinguish between attacks and normal use cases (without the help of a human being and in a robust but versatile way); it also has to be based on mechanisms which dynamically find a trade-off between security and performance; all these mecanisms have to formalized in a way which is clearly understandable by the designer. In this article, a prototype which enables such security management is described. The solution is based on a double-processor architecture: one processor embeds a representative set of countermeasures (and mechanisms to define their parameters) and executes the application code. The second processor, on the same chip, applies a given security strategy, but without requesting sensitive data from the first processor. The chosen strategy is based on fuzzy logic reasoning to enable the designer to describe, using a fairly simple formalism, both the attack paths and the normal use cases. A proof of concept has been proposed for the smart card part of a conditional access for Pay-TV, but it could easily be fine-tuned for other applications.

Keywords: Hardware tamper resistance, Formalism to detect and counter fault and side-channel attacks, Architectures for trusted computing, Application of fuzzy logic

1 Introduction

Security is a key component for information technologies and communication. Among the security threats, a very important one is certainly due to vulnerabili-

ties of the integrated circuits that implement cryptographic algorithms to ensure confidentiality, authentication or data integrity (such as smartcards). With the access to one of these circuits, the attacker tries either to reconstruct the functionality of the circuit (*reverse engineering*) or to recover cryptographic materials when the cryptographic algorithm is known (*physical or hardware cryptanalysis*). Both threats share a set of techniques. The first one, called *side channel attacks*, consists in observing some physical characteristics which are modified during the circuit's computation [23, 5]. The second technique, called *fault attacks*, consists in disrupting the circuit's behavior [1, 4]. The third one consists in getting information about the chip design by direct inspection of its structure [24]. This inspection may be performed by using any kind of imaging techniques or by using destructive means such as abrasion, chemical etching or focused ion beam. Combinaison of those three techniques have also been proposed [21]. Many protections have therefore been proposed to counter such attacks. Some protections (hereafter referred to as "sensors") give information about the state of the system either by measuring the light, the voltage, the frequency or the temperature of the chip or by detecting errors during computations. This detection is generally based on spatial redundancy (i.e. performing the same computation several times simultaneously), temporal redundancy (i.e. performing the same computation several times) or information redundancy (i.e. performing a computation with more bits than required) [6] or ad-hoc sensors [10, 14]. Several mechanisms are also proposed to detect a modification in the software execution flow. In some cases, an additional hardware block is dedicated to this task [17]. To render physical attacks more difficult, "noise" has been added, for example, by using an internal clock, by randomizing the order of the instructions, by adding dummy operations or by masking the internal computations that can be predicted by the attacker [20],[8]. Another way to reduce sensitivity to side channel attacks consists in reducing the correlation between physical values (such as power consumption or electromagnetic radiation) and the data processed, for example, by using balanced data encoding and balanced place and route [13], by using power filters or electromagnetic shields. Finally, some countermeasures modify the functional behavior of the circuit in case of attacks. Such "reactions" may consist, for example, in temporarily stopping the communication with the reader (the card "mutes") and/or resetting parts of the running software. The ultimate reaction consists in permanently destroying (i.e. killing) all the data (including sensitive information) stored in the chip.

In practice, the chip security is achieved by a combination of such countermeasures. Nowadays, state-of-the-art circuits can protect data for weeks, months and even years. But the implementation of these countermeasures not only drastically decreases the performance of the chip in terms of power and speed but also decreases its availability. In order to overcome this limitation, we propose a complementary approach: implementing complex management (through the application of a "strategy of security") of the whole set of countermeasures embedded on a chip. Such a "system-level" approach has already been addressed in [12]. The authors proposed mechanisms which reconfigure the architecture

of cryptographic hardware blocks embedded in an FPGA in accordance with performance and energy consumption criteria. Our work addresses this trade-off but also the trade-off between security and availability. Besides, we consider that these trade-offs have to be adjusted only with technologies which are regularly available in smart cards (i.e. without the use of hardware reconfiguration capabilities provided by FPGAs).

In section 2, a representative case study (a conditional access system for Pay-TV) and the associated expected properties for complex strategy of security are detailed. In section 3, an example of such a strategy, based on fuzzy logic, is proposed. In section 4, an innovative hardware/software prototype enabling the execution of this strategy is described. Finally, in section 5, the results obtained with this prototype are presented.

2 Case study

In the followin g sections, an implementation of the chosen representative system, the smart card part of a conditional access system (hereafter referred to as the "host system"), is described. Its functional parts and some of its protections are presented. Finally, by expressing the difficulties of designing such a system, the specifications for complex strategies of security are highlighted.

2.1 Conditional Access System: Functional Part

The functional part of the conditional access system (or CAS) consists of a JavaCard application which is interpreted by a virtual machine (VM) which runs on a micro-controller.

Application The Conditional Access Systems (CAS) protect content (such as radio, TV, data stream) by requiring certain criteria to be met before granting access to this content. One of the main criteria is to own a smart card that stores sensitive information. This information is used by the receiver to decipher the content. Security of this content is achieved by combining protections at different levels. A management key protects the transfer of access rights (subscriptions, the related period of validity, geographical localization of the receiver, etc.) and the transfert of the exploitation key. This key, which is changed approximately once every month, protects the access criteria associated with the content and the value of the control word. This control word, which is changed approximately once every 5 to 10s, protects the content.

Virtual Machine The chosen virtual machine (VM) complies with the Java Card 2.2.2 specifications [15], and provides card content management capabilities based on the GlobalPlatform Standard [11]. Hence, it is possible to download and install on-card applications (or applets) running on top of the VM and to use standard Java Card and GlobalPlatform Application Programming Interfaces (APIs).

Host's micro-controller The host sub-system is built on a 5-stage pipelined 32-bit Harvard RISC microcontroller able to execute one instruction per clock cycle. The instructions are stored in 640kB of ROM and the data in 256kB of

RAM and in 128kB of EEPROM. The peripherals include two Universal Asynchronous Receiver Transmitters (or UARTs) compliant with the ISO7816 and RS232 standards and an Advanced Encryption Standard (or AES) crypto-engine (to accelerate the ciphering and deciphering of data) [19].

2.2 Protections

The different components of the host described above embed the protections described below. Some of them, such as the security sensors, may have outputs. Other countermeasures may be configured. Configuration parameters are either integer numbers (for redundancy level, insertion of dummy instructions and random power generator) or boolean values (which activate or not a security reaction). As will be explained in section 4.1, even though these protections are physically embedded in the host sub-system, these parameters are not driven by the host.

Security sensors Several sensors have been implemented or emulated. Some of them measure physical characteristics such as voltage, light intensity or chip temperature. Others detect errors during the execution flow of a piece of code, during I/O data transmissions or during computations. One way of detecting errors consists in performing the same computation several times and then comparing the results. If they are identical, the computation is considered error-free. If not, a counter of corrupted executions, noted CE, is increased. The number of times that the same computation is performed is called the "redundancy level" (noted RL). A redundancy level of "i" is noted "×i". It is equal to 1 when the redundancy countermeasure is not activated but when $RL \leq 3$, the host has error-correction capabilities.

The various sensors which are taken into account in our study and their different possible outputs are described in Table 1. The hardware (HW) or software components ("VM" denotes the virtual machine and "App" the application) which update these outputs are listed in the last column of this table. For example, the number of times that the check (performed by the application) of a Message Authentication Code (MAC) is false is noted ME. It takes its value between 0 and 10^4. Another sensor is the number of times that a cryptographic operation is performed with the same key. This number is noted CO.

Insertion of Dummy Instructions (IDI) Another protection consists in increasing the level of noise by inserting dummy instructions (i.e. that are not useful) during computations. The execution of a program is thus made up of several sequences of valid instructions followed by dummy instructions. In our framework, which has been drawn from [2], two parameters D and N configure the countermeasure. D and N denote the maximum number of consecutive useful and dummy instructions respectively in a sequence. Note that N is equal to 0 when the countermeasure is not activated.

Name	Values	Description	Updated by
LS	$\{0, 1, \ldots, 5\}$	# of triggers of the light sensor	HW
VS	$\{0, 1, \ldots, 10\}$	# of triggers of the voltage sensor	HW
EFE	$\{0, 1, \ldots, 10\}$	# of corrupted execution flow	VM
CE	$\{0, 1, \ldots, 10\}$	# of corrupted execution	VM
PE	$\{0, 1, \ldots, 10\}$	# of wrong PIN	App
NE	$\{0, 1, \ldots, 10^3\}$	# of methods processed without error	VM
ME	$\{0, 1, \ldots, 10^4\}$	# of MAC errors	App
CO	$\{0, 1, \ldots, 10^7\}$	# of cryptographic execution	App

Table 1. Outputs of the CMs

Random Power Generator (RPG) In order to blur the power consumption of the circuit, several Random Number Generators (RNG) may be activated. Assuming that each step t of this power consumption, noted $x(t)$, is drawn from a Gaussian shaped probability distribution (with a mean $\mu_c(t)$ and a constant standard deviation σ_c). Let us also suppose that the power consumption added by one random generator also follows a Gaussian distribution with a constant mean μ_R and with a standard deviation which is equal to σ_c. We also suppose that the different RNG are identical and that their power consumptions are statistically independent. The probability density function of the total power consumption, noted $x_{tot}(t)$, of the circuit when R RNG are activated is thus the following:

$$pdf(x_{tot}(t)) = \frac{e^{-\frac{(x_{tot}(t)-\mu_c(t)-R\cdot\mu_R)^2}{2\cdot\sigma_c^2\cdot(1+R^2)}}}{\sigma_c \cdot \sqrt{1+R^2} \cdot \sqrt{2\cdot\pi}} \quad (1)$$

Note that R is equal to 0 when the RPG countermeasure is not activated.

Security reactions Two reactions have been taken into account: stopping the communication and resetting the software (hereafter noted "Mute/Reset"), or destroying all the data (hereafter noted "kill"). But several reactions could also be considered at the application level: forbidding the access to one or several TV channel packages, suppressing services such as video on demand, restricting the access to only free TV or program contents, etc.

Other protections In the case of a commercial device, other security features (such as power filters, shields, balanced logic, etc.) which can never be deactivated, are also supposed to protect the circuit, independently of all kinds of strategy of security.

2.3 Impact of protections

An estimation of the evaluation of performances in terms of speed, energy consumption and security for different values of the parameters of the countermeasures is proposed in Annexe 1. To estimate the security level, we have only considered the threat related to side channel attacks (SCA) and differential fault attacks (DFA).

– The gain in terms of SCA, noted $FSCA$, is the ratio between the number of curves needed for the attacker to recover the key when the countermeasure is activated and the number of curves without countermeasure (the greater the better).
– The gain in terms of DFA, noted $FDFA$, is the ratio between the number of experiments needed to recover the key when the countermeasure is activated and the number of experimentations without countermeasure (the greater the better).
– The loss in terms of speed, noted $FTime$, is the ratio between the duration of a computation with the countermeasure and the duration of the same computation without countermeasure (the smaller the better).
– The loss in terms of energy, noted $FNRJ$, is the ratio between the energy consumption with countermeasure and the energy consumption without countermeasure (the smaller the better).

Figure 1 displays the values of these factors for different values of D and N (with $RL = 2$ and $R = 3$). Note that the values do not share the same scale. This figure clearly shows that performance decreases as the security level increases. Note also that the theoretical considerations described above are only approximations and have to be validated with experimental results.

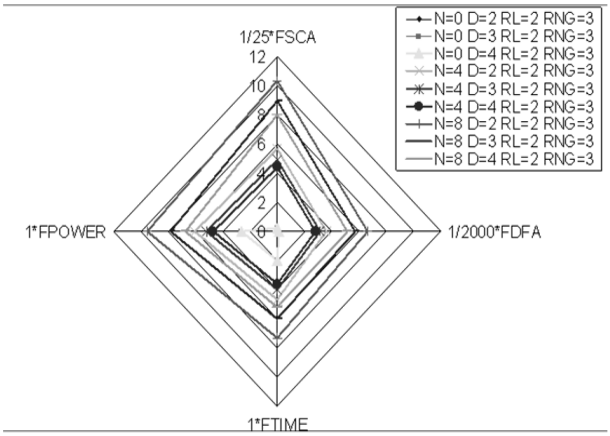

Fig. 1. Decrease of the performances with the increase of the security

2.4 Difficulties

The design of the smart card part of a CAS, as described in section 2.1, is a very challenging task because of the following specifications:

– Its security level has to be very high: the security of a smart card is expected to hold for 2-5 years. But it is generally used for 8 years. In this condition, a

card which resists one additional year represents large savings for the content and CAS providers.

- Its performance has to be high: For Pay-TV applications, the switch from channel to channel has to be as fast as possible.
- The availability has to be high: A malfunction of the card causes a prejudice for the user (who is no longer able to watch his favorite TV program) but also for the CAS provider who has to deal with expensive field feedback. A malfunction may be due to abnormal system usage (if the card, for example, is put in a very hot place) but also to incorrect system integration (if the card, for example, is inserted in a poor quality reader). Such cases will further be called "anomalies", in the sense that the component operates in abnormal conditions but is not subjected to attacks.
- It processes data with various sensitivity levels. For example, the retrieval of the exploitation key (EK) would enable the pirate to decipher the control word (CW) during the period of validity of the key EK. With this CW, he gains access to the content. So, the corruption of the key EK is a very serious security threat. On the contrary, knowing the value of the word CM at a given moment is of little value because it is only valid for a very short period of time. Let us call DS the sensitivity level of data which is handled by the application.
- Its power consumption must be low if it is embedded in a mobile phone or in a multimedia tablet.
- It has to be inexpensive.

2.5 Need for complex strategies of security

Section 2.3 shows that the setup of the countermeasures defines the performances of the circuit in terms of security, speed and power consumption (but also availability) and that those performances are closely linked. For example, the increase in security causes the decrease in speed. Thus, the set of performances which may be reached, despite the wide range of setups of the countermeasures, is restricted. This restriction may be such that none of the performances of this set fulfills the specifications described above. In such a case, the specifications are antagonistic and the designer has to make a trade-off. He has, for example, to choose between availability and security.

We propose to add mechanisms which enable to dynamically modify the setups of the countermeasures, i.e. which enable to switch from a high performance but "poorly" secured state to a low performance but secured one. To draw an analogy, these mechanisms provide a new "degree of freedom" which make it possible to reach new sets of performances. In other words, these mechanisms release the links that exist between the different performances. We consider that a smart strategy of security should be able to control these mechanisms to fulfill all the specifications (or at least as many as possible).

3 Example of strategy of security

In this section, a complex strategy of security tailored for the application described in section 2.1 is proposed. The chosen strategy is inspired from methods and concepts developped in Intrusion Detection Systems (or IDS) [3]. In this framework, two methods are proposed. The first method consists in determining the whole set of states of the system reached by a user with "normal" behavior. Each time the state of the system leaves this set, an "anomaly" is detected. The second method is complementary. It consists in determining the whole set of states which would have to be reached by an evil-minded user to perform an attack. When the system enters this set, misuse (considered an attack) is detected. Contrary to classical IDS, the component is off-line most of the time. So, it has to react autonomously without any outside help. In order to obtain, in this context, both robust and secure behavior, mechanisms must detect both anomalies and misuses.

According to principles developed for IDS, the chosen strategy of security is decomposed into three different processes: The first consists in collecting information about the state of the host system, the second, called analysis, consists in computing (from this information) the anomaly and misuse levels (called resp. ML and AL) and the third in configuring (according to ML and AL) the parameters of the countermeasures.

3.1 Information sources

We consider that the sensitivity level of data DS (provided by the application and defined in 2.4) and the outputs $\{LS, VS, EFE, CE, PE, NE, ME, CO\}$ (provided by the sensors and defined in section 2.2) are the inputs of the analysis algorithm.

3.2 Analysis

As the secure circuit designer generally expresses the anomaly and misuse cases with words which are very often vague and inaccurate, fuzzy logic has been used to formalize the analysis mechanism [9]. The details of such an analysis is out of scope of this article but it is sketched here. Within this framework, the analysis process is mainly described by a set of "IF-THEN" rules and an inference process.

The "IF-THEN" fuzzy rules are expressed with words or "linguistic variables" such as LOW, $HIGH$ and adverbs such that $RATHER$, $VERY$ whose meanings are precisely defined with "fuzzy sets". For example, the chosen strategy of security is expressed in our system by using a dozen of rules such as :

- R_0: "IF the number of methods that have processed without error (NE) is $VERY\ HIGH$ THEN the misuse is LOW "
- R_1: "IF the voltage (VS) is $RATHER\ HIGH$ and the light (LS) is $HIGH$ THEN the misuse is $HIGH$ "
- R_2: "IF the number of cryptographic errors (CE) is $RATHER\ HIGH$ THEN the misuse is $HIGH$ "

– R_3: "IF the number of PIN code error (PE) is $RATHER\ HIGH$ OR the number of triggers of the voltage sensor (VS) is $HIGH$ THEN the misuse is $HIGH$ "
– ...

The inference process chosen for computing the misuse and anomaly level from fuzzy rules and inputs was first proposed by Mamdani in [16]. This process consists in fuzzifying the inputs, then computing the degree of truth of each rule, then aggregating the results of all the rules and lastly "defuzzyfying" them to obtain the misuse and anomaly levels (which are scalar values between 0 and 1). This process can be activated, for example, each time an input value is modified.

3.3 Configuration of countermeasures

The configuration of the countermeasures has been chosen from the analysis of their impact, described in section 2.3. We have evaluated the impact of the RPG for $R \in \{0; 3; 10\}$ (the value for α is chosen equal to 10%). As the cost of the redundancy technique is very high, we have only considered $RL \in \{1; 2; 3\}$ (the default value for q is 8) and the impact of the IDI for $D \in \{0; 4; 8\}$ and for $N \in \{2; 3; 4\}$ (the value for m is chosen equal to 150). Among all those possibilities, only four configurations of the different countermeasures are taken into account. These configurations are chosen since they cover a large range of security/performance trade-offs. They are further called $Safe$, $Unsafe$, $Critical$ and $Fatal$ and are reported in Table 2. It is important to note that whatever the configurations, the different physical sensors are always activated. It is also important to note that, in practice, the security level is determined as the minimum value between $FDFA$ and $FSCA$ (in bold in the table).

$Config.$	$Safe$	$Unsafe$	$Critical$	$Fatal$
$Sensors$	ON	ON	ON	–
RL	×1	×2	×3	–
RPG	0	3	10	–
IDI	$(D = 2; N = 0)$	$(D = 3; N = 4)$	$(D = 4; N = 8)$	–
$Mute/Reset$	No	No	Yes	–
$Kill$	No	No	No	Yes
$FSCA$	1.0	122.5	1346.7	–
$FDSA$	1.0	6270.7	1.0E+08	–
$Time$	1.0	4.0	7.8	–
$Energy$	1.0	5.2	15.6	–

Fig. 2. Countermeasure configurations

The function which assigns the configuration to each value of the anomaly (AL) and misuse (ML) levels is described in Table 3. The different thresholds have been determined according to the parameters chosen for the analysis process.

	ML					
		$[0; 0.2]$	$]0.2; 0.4]$	$]0.4; 0.6]$	$]0.6; 0.8]$	$]0.8; 1]$
$1 - AL$	$[1; 0.8]$	$Safe$	$Safe$	$Unsafe$	$Critical$	$Fatal$
	$]0.8; 0.6]$	$Safe$	$Unsafe$	$Unsafe$	$Critical$	$Fatal$
	$]0.6; 0.4]$	$Unsafe$	$Unsafe$	$Unsafe$	$Critical$	$Fatal$
	$]0.4; 0.2]$	$Unsafe$	$Unsafe$	$Critical$	$Critical$	$Fatal$
	$]0.2; 0]$	$Unsafe$	$Critical$	$Critical$	$Fatal$	$Fatal$

Fig. 3. Configurations of the countermeasures according to the Misuse (ML) and Anomaly (AL) Levels

4 Prototyping

This section describes a hardware implementation which has been defined for our case study but which could also be used for different strategies of security.

4.1 Architecture

We choose an architecture, represented in Figure 4, which separates the calculations related to the execution of the application software and those related to the management of the security. This leads to the addition of a system called "monitor" and strictly dedicated to the management of the security. In the left part of the figure, the host system, described in section 2.1, is represented with its software part (the CAS application running above the JavaCard virtual machine) and its hardware part (the 32-bit RISC microcontroller). Each of these parts implements its own set of countermeasures described in section 2.2. These countermeasures may be configured via signals noted "CMs config." and may give information about the state of the host via signals noted "CMs outputs". The signal which indicates the sensitivity of the data handled by the application is noted DS. All these signals are exchanged between the host and the monitor through a communication channel. The monitor is represented in the right part of the figure. The software parts of the monitor are the strategy of security and the component-based operating system. Its hardware part is a micro-controller.

The software and hardware parts of the monitor and an example of communication between the host and the monitor are described above.

Operating system and strategy of security The monitor uses a specific component-based operating system. It needs a small memory footprint (less than 4kB) and it responds in less than a hundred clock cycles to the host's requests. An application is a set of components linked together through exchange memory zones. A component is a set of ports towards these exchange zones, of private attributes and of methods (which describe its behavior). The strategy of the security formalized by using fuzzy logic and described in section 3 is embedded in such a component. It is important to note that, thanks to a particular choice of fuzzification functions and off-line precomputations, the fuzzy logic mechanisms can easily be implemented in the very light-weight hardware of the monitor.

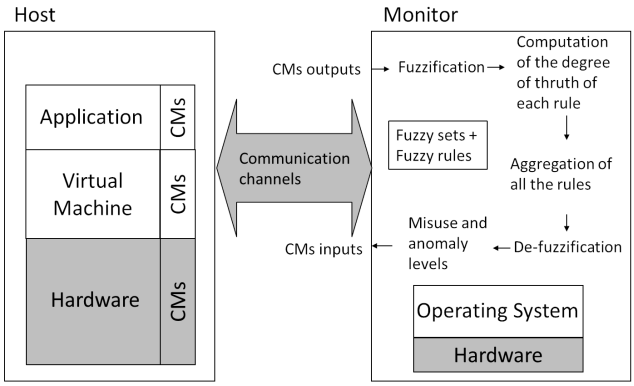

Fig. 4. Architecture of the prototype (hardware in dark grey, software in grey)

Monitor's micro-controller The micro-controller of the monitor is built around a 5-stage pipelined 32-bit Harvard RISC processor. To increase the security of the system, this processor does not share any piece of hardware with the host system and in particular it has its own addressing space: the instructions are stored in 32kB of ROM and data in 4kB of RAM. The peripherals comprise one UART, two FIFOs and an Interrupt Controller Unit (ICU). These ICU and FIFOs provide two communication channels: The monitor is informed of the events occurring in the host (without having access to the sensitive data themselves) and in return, the monitor is able to change the configuration of the countermeasures embedded into the host.

Communication between the host and the monitor The communication between the host and the monitor is based on a request/acknowledge protocol. The request is always initiated by the host. The host waits after its request until the monitor responds. The following example describes such a communication initiated by the application.

1. The application indicates that the sensitivity of the data that it manipulates (DS) is changing.
2. The virtual machine sends this information to the monitor via the FIFOs.
3. The host stops its current execution.
4. From the fuzzy sets and the "IF-THEN" fuzzy rules defined by the user, the monitor processes the CM's outputs and the DS's value by using the fuzzy reasoning described in section 3.2. The outputs of this reasoning are the misuse and the anomaly levels. These levels are used to select the configuration of the countermeasures thanks to Tables 3 and 2 reported in section 3.3.
5. The monitor asks to the host system to configure software (via FIFOs) and hard wired countermeasures (via the ICU).
6. The monitor waits until all the countermeasures are configured and are ready.
7. The monitor asks to the host system to resume its execution.

4.2 FPGA synthesis results

The whole system has been implemented in the Xilinx Virtex-5 FPGA of an ML501 evaluation platform. This platform has been extended with an ISO7816 interface adapter (for an easy connection with off-the-shelf smart card readers) and with an additional RS232 serial interface (for trace or debug purposes). The details of the hardware implementation of the host, of the monitor and of the communication channels are described in Figure 9 of Annexe 3. The combinatorial part of the system (i.e. excluded the memory areas) uses 48% (3490 slices out of 28800) of the total FPGA's resources in term of slices. As a reminder, a slice is the smallest hardware element used in a Xilinx FPGA. A slice contains four D flip-flops associated to four look-up tables with six independent inputs. Excluding the monitor, the resources usage falls to 34% (2462 slices out of 7200). So, the combinatorial part of the monitor represents around 40% of the combinatorial part of the whole system. As the resources associated with the memory parts are far larger than the combinatorial ones, the estimated total overhead (i.e. including the memory areas) decreases to about 7%. But it is important to note that this total overhead greatly depends on the size of the host's memories.

5 Security analysis

5.1 Side channel analysis

For this discussion, we distinguish two kinds of information leakage. The first kind is "directly" related to the value of sensitive data (such as cryptographic keys) and is exploited with classic side channel techniques. The second kind of leakage is not directly linked to the sensitive data but only helps the attacker to understand the functioning of the system and so, to focus his attacks. For example, if the circuit blurs the manipulation of sensitive data with the activation of RPG, the attacker could easily detect, by measuring the power consumption, the location in time of this sensitive computation. This information will help him perform his side channel attacks on this location.

The system is protected against direct information leakage by the host's countermeasures. The theoretical efficiency of some countermeasures is estimated in Section 2.3. Several methods have also been proposed to quantify in practice the direct information leakage of circuits in terms of "number of traces to recover the key". One of these methods, described in [22] is based on a profiled information theoretic analysis. If the security of the host has been first estimated with such a worst-case security evaluation, the user is able to know how many bits per trace the host leaks. From this data, from the number of cryptographic executions which has be done with the same key (sensor CO), he is able to estimate the number of bits of the key which could have been retrieved by the attacker with side channel analysis. The parameters of the strategy of security are to be chosen according to these estimations. Besides, as the host sends to the monitor only the values of the sensors and the sensitivity level of the data handled by the application, there is no additional direct information leakage due to the communication channels or due to the monitor.

The indirect information leakage is linked to the "discretion" of the host's countermeasures. We consider that the computation of the configuration of the countermeasures executed in the monitor is very small compared to the signature of the countermeasure itself and so does not represent an additional indirect leakage of information. Besides, a more complex strategy of security such as the one described in Section 3, could be defined in order to reduce the indirect information leakage (for example, by creating a randomly high level of noise even when no sensitive data is manipulated).

5.2 Fault attacks

As for protections against side channel attacks, the system is mainly protected by the host's countermeasures. In particular, the host has to embed highly effective protections in order to be protected against highly effective attacks such as those described in [18] (which enable the pirate to retrieve the key with only one faulted execution). In the following discussion, we consider that it is the case, at least when sensitive data is handled by the application. During fault attacks, several cases have been distinguished:

- If the functioning of the monitor and of the communication channels are not corrupted during a fault attack, two sub-cases are possible:
  - The functioning of the host is not corrupted during the fault attack. In this case, the system functions normally.
  - The functioning of the host is corrupted during the fault attack. In this case, as the host is protected by using error detection or correction, the host stops its computation, the counter of corrupted execution is incremented and this information is processed by the monitor. The monitor will decide to increase the redundancy level until providing error correction capabilities (rendering DFA ineffective). The only possibility for the pirate in such a case consists in bypassing the HW/SW error detection or correction capabilities of the host to avoid the increase of the redundancy level.
- If the functioning of the monitor (or of the communication channels) are corrupted during a fault attack, this incorrect functioning does not directly give information about sensitive data because the monitor never handles such data but two sub-cases are possible:
  - The monitor is unable to compute a configuration. Two sub-sub-cases have to be taken into account:
    - The functioning of the host is not corrupted during the fault attack. In this case, as the host waits after any request until the monitor responds, there is no risk of external communications of data.
    - The functioning of the host is also corrupted during the fault attack. In this case, as the host is protected by using error detection or correction, the host stops its computation and a request towards the monitor is triggered through the increase of the counter of corrupted execution. As the host waits after this request until the monitor responds, there is no risk of external communications of data. The

only possibility for the pirate in such a case consists in bypassing the HW/SW error detection or correction capabilities of the host to avoid any request towards the monitor.

- The monitor computes an incorrect configuration of the countermeasures. In this case, the sensitive data is still protected by the countermeasures that can never be deactivated and by this incorrect set of countermeasures. In the worst case, the attacker will have to face a lower level of security. In the best case, he will have to face a higher level of security. But in both cases, the host has to be attacked after the monitor.

This discussion shows that in order to take advantage of the proposed architecture, the pirate has to attack first the monitor and, in a second step, to attack the host. Such a scenario is unfortunately possible with state-of-the-art attack equipment but we consider that, in practice, it leads to more difficult attacks.

6 Discussion

As explained in Section 4.2, the proposed architecture leads to a penalty of 5-20% (depending of the size of the host's memories) in terms of area (and to some extent, of power consumption), but we underline that these figures have been obtained with no design optimization. The penalty in terms of performances of the adjunction of the monitor greatly depends on several parameters such as, for example, the rate of communication of data between the host and the monitor, the rate of change of configurations of the countermeasures or the time necessary for the host to reconfigure its countermeasures. But it is important to note that these penalties have to be compared with the penalties due to the countermeasures embedded in the host.

Another difficulty concerns the design of the best strategy of security for a given application. The theoretical considerations of Section 2.3 help the user, first, to evaluate *a priori*, that is according to a given set of hypotheses, the performances and the security levels of the host for several sets of parameters and, second, to choose an *a priori* strategy of security for the given application. As the set of hypotheses may be false (and it is unfortunately often the case in the security domain), the performances have to be measured in practice and security evaluations have to be performed with state-of-the-art attack equipment. As most secure designs, these results have to be used to improve the a priori strategy of security. At the end of this refinement process, the strategy should not only increase the availability but also trade-off the performances and the security according to the application's constraints.

Finally, the testing and debugging of the whole prototype is also more complex than for a single component. Each hardware and software component has been debugged one by one. After these unitary tests, the monitor and the host have been tested independently. Finally, the host and the monitor have been tested together.

7 Conclusion

This article has completed the first step towards "system level" management of the security dedicated to the improvement of the availability and the performance of a circuit without reducing its security. The three main contributions of this article are the following. First, the impact in terms of security and performances of different combinations of well-known countermeasures has been quantified. This study has shown, without surprise, that security and performances are antagonistic. But this study has also shown that, by modifying only a few parameters of these countermeasures, states with very distinct performance and security levels can be reached. Second, we have proposed a strategy of security designed to minimize both the rate of anomalies considered to be attacks (to increase the availability) and the rate of attacks considered to be normal functioning (to increase the security). The strategy is based on mechanisms which dynamically modify the parameters of the countermeasures. Third, we have proposed an HW/SW architecture which implements these mechanisms. The proposed solution is based on a double-processor architecture: one processor embeds a representative set of countermeasures (and mechanisms to define their parameters) and executes the application code. The second processor, on the same chip, applies the strategy, but without requesting sensitive data from the first processor.

Future work will consist in determining the theoretical performances and security levels for combinations of other countermeasures. It will also consist in refining the strategy by performing real attacks on the prototype and in measuring its real performance (preliminary simulation results are described in Annexe 4). We also plan to develop mechanisms to enable over the air upgrades of fuzzy rules according to the evolution of the threat. It could also be interesting to investigate, in parallel, more expressive formalisms such as, for example, those used for the detection of intrusion in complex computer systems.

References

1. Ali, S., Mukhopadhyay, D., Tunstall, M.: Differential fault analysis of aes: towards reaching its limits. Journal of Cryptographic Engineering 3(2), 73–97 (2013)
2. Ambrose, J.A., Ragel, R.G., Parameswaran, S.: RIJID: Random Code Injection to Mask Power Analysis based Side Channel Attacks. In: Proc. Design Automation Conference − DAC, ACM. pp. 489–492 (2007)
3. Bace, R.G.: Intrusion detection. Macmillan Publishing Co., Inc., Indianapolis, IN, USA (2000)
4. Barenghi, A., Breveglieri, L., Koren, I., Naccache, D.: Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. Proceedings of the IEEE 100(11), 3056–3076 (2012)
5. Bauer, A., Jaulmes, É., Prouff, E., Reinhard, J., Wild, J.: Horizontal collision correlation attack on elliptic curves - - extended version -. Cryptography and Communications 7(1), 91–119 (2015)
6. Carlet, C., Freibert, F., Guilley, S., Kiermaier, M., Kim, J., Solé, P.: Higher-order CIS codes. IEEE Transactions on Information Theory 60(9), 5283–5295 (2014)

7. Clavier, C., Coron, J.S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems. pp. 252–263. CHES '00, Springer-Verlag, London, UK (2000)
8. Coron, J.S., Kizhvatov, I.: Analysis and improvement of the random delay countermeasure of CHES 2009. In: Mangard, S., Standaert, F.X. (eds.) CHES. Lecture Notes in Computer Science, vol. 6225, pp. 95–109. Springer (2010)
9. Dubois, D., Prade, H.: Fundamentals of Fuzzy Sets Series: The Handbooks of Fuzzy Sets, Vol 7. Kluwer Academic (2000)
10. Dutertre, J., Bastos, R.P., Potin, O., Flottes, M., Rouzeyre, B., Natale, G.D., Sarafianos, A.: Improving the ability of bulk built-in current sensors to detect single event effects by using triple-well CMOS. Microelectronics Reliability 54(9-10), 2289–2294 (2014)
11. Global platform specifications website, http://www.globalplatform.org
12. Gogniat, G., Wolf, T., Burleson, W., Diguet, J.P., Bossuet, L., Vaslin, R.: Reconfigurable hardware for high-security/ high-performance embedded systems: The SAFES perspective. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 16(2), 144 –155 (feb 2008)
13. Guilley, S., Sauvage, L., Flament, F., Vong, V.N., Hoogvorst, P., Pacalet, R.: Evaluation of power constant dual-rail logics countermeasures against DPA with design time security metrics. IEEE Trans. Computers 59(9), 1250–1263 (2010)
14. Homma, N., Hayashi, Y.i., Miura, N., Fujimoto, D., Tanaka, D., Nagata, M., Aoki, T.: Em attack is non-invasive? - design methodology and validity verification of em attack sensor. In: Batina, L., Robshaw, M. (eds.) Cryptographic Hardware and Embedded Systems CHES 2014, Lecture Notes in Computer Science, vol. 8731, pp. 1–16. Springer Berlin Heidelberg (2014)
15. Java card technology website, http://java.sun.com/javacard
16. Mamdani, E.: Application of fuzzy logic to approximate reasoning using linguistic synthesis. Computers, IEEE Transactions on C-26(12), 1182 –1191 (dec 1977)
17. Mao, S., Wolf, T.: Hardware support for secure processing in embedded systems. Computers, IEEE Transactions on 59(6), 847 –854 (june 2010)
18. Moradi, A., Shalmani, M.T.M., Salmasizadeh, M.: A generalized method of differential fault attack against AES cryptosystem. In: CHES. pp. 91–100 (2006)
19. NIST: Announcing the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication, n. 197 (Nov 26, 2001)
20. Rivain, M., Prouff, E., Doget, J.: Higher-order masking and shuffling for software implementations of block ciphers. In: Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems. pp. 171–188. CHES '09, Springer-Verlag, Berlin, Heidelberg (2009)
21. Roche, T., Lomné, V., Khalfallah, K.: Combined fault and side-channel attack on protected implementations of AES (2011)
22. Standaert, F.X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 5479, pp. 443–461. Springer (2009)
23. Stefan Mangard, E.O., Popp, T.: Power Analysis Attacks Revealing the Secrets of Smart Cards. Springer Verlag (2007)
24. Torrance, R., James, D.: The state-of-the-art in ic reverse engineering. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2009, Lecture Notes in Computer Science, vol. 5747, pp. 363–381. Springer Berlin Heidelberg (2009)

Annexe 1: Impact of protections

As explained in 2.3, we consider only the threat related to side channel attacks (SCA) and differential fault attacks (DFA). In order to successfully mount such attacks, the attacker proceeds in a divide and conquer manner (i.e. he attacks small pieces of the key one by one). On each iteration of these attacks, he targets the result of one particular piece of computation, hereafter referred to the "targeted result". It is, for example for SCA, the computation of a SBox during the first round. To mount a DFA, the attacker also needs the cipher text resulting from one or several faulty computations.

Impact of Redundancy

FSCA We consider that the redundant computations generate identical power traces which could easily be added in the time domain by the attacker, decreasing the number of curves needed to recover the key by a factor RL.

FDFA As explained in section 2.2, a redundant computation is associated with a comparison of the results in order to increment the counter CE if the results are different. To bypass the redundancy protection, the attacker will have to both avoid the update of the counter in case of error detection and realize several faults of the same value, noted e_0, during the RL successive computations of the targeted result. The probability of realizing such a set of faults determines the number of realizations which are required to mount the attack. We shall call q the number of bits of the targeted result. If we consider that all the faults on these bits are equally probable, then the probability of realizing the same fault e_0 (whose value does not matter) during the RL successive execution of the targeted instruction is equal to $(1/2^q)^{RL-1}$. In classical DFA schemes, a fault generally has to affect 1 byte. So, the default value is chosen equal to $q = 8$.

FTime In our framework, the redundant computations are not performed in parallel. We also assume that the comparison of the results is negligible in terms of computation time. Thus, the redundancy countermeasure increases the computation time by a factor RL.

FNRJ We assume that the comparison of the results is negligible in terms of energy consumption. So, the redundancy countermeasure increases the energy consumption by a factor RL.

Impact of Insertion of Dummy Instructions Let us call $\mathcal{D}$ the random variable equal to the number of instructions in the sequences of useful instructions. The domain of $\mathcal{D}$ is chosen equal to $\{1; \ldots; D\}$. Let us call $\mathcal{N}$ the random variable equal to the number of instructions in the sequences of useless instructions. The domain of this random variable is set equal to $\{0; \ldots; N\}$. We consider that the random variables $\mathcal{N}$ and $\mathcal{D}$ follow uniform distributions.

$$FSCA_{RL} = \frac{1}{RL}$$
$$FDFA_{RL} = (1/2^q)^{RL-1}$$
$$FTime_{RL} = RL$$
$$FNRJ_{RL} = RL$$

Fig. 5. Performances for the redundancy level countermeasure

Let us also suppose that the m^{th} valid instruction should be the instruction which computes the targeted result. For the sake of simplicity, we consider that each instruction (useless or not) is executed in one clock cycle. The typical value for m is equal to 100-200 which corresponds to a software implementation on a 32-bit processor of a round of an AES [19]. Let us define the random variable $\mathcal{X}$ equal to the number of the clock cycle associated with the execution of the instruction m.

Each realization x of this random variable, is equal to $x = \sum_{i=1}^{k}(d_i) + \sum_{i=1}^{k}(n_i)$, with k such that $\sum_{i=1}^{k}(d_i) = m$, with i being the i^{th} sequence of useless/useful instructions and with d_i and n_i the i^{th} realizations respectively of the random variables $\mathcal{D}$ and $\mathcal{N}$. We have $x = m + \sum_{i=1}^{k}(n_i)$. We consider that, because $m \gg D$, x could be approximated by $x \sim m + \sum_{i=1}^{q}(n_i)$ with $q = 2 \cdot m/(D+1)$. In these conditions, the density of probability of $\mathcal{X}$ follows a normal distribution (μ_X,σ_X) with:

$$\mu_X = m + q \cdot \mu_N = m \cdot (1 + N/2) \tag{2}$$

$$\sigma_X^2 = q \cdot \sigma_N^2 = m \cdot \frac{N \cdot (N+2)}{6 \cdot (D+1)} \tag{3}$$

For the sake of simplicity, we consider, as proposed in [7], that m is uniformly distributed (with the probability 1 out of $2 \cdot \sigma_X$) between $m - \sigma_X$ and $m + \sigma_X$.

FSCA In these conditions, the SCA peak is reduced by a factor $2 \cdot \sigma_X$ and the number of curves necessary to retrieve the key increases by a factor $4 \cdot \sigma_X^2$. But by using the sliding window method (with consists in reconstructing the peak by integrating the consumption curves on $2 \cdot \sigma_X$ samples) also described in [7], this saving in terms of number of power curves is only equal to $2 \cdot \sigma_X$.

FDFA In order to realize a DFA, we suppose that the attacker is able to target clock cycles comprising between $m - \sigma_X$ and $m + \sigma_X$. In these conditions, he has one chance out of $2 \cdot \sigma_X$ to modify the instruction m. The number of faulty realizations is thus increased by a factor $2 \cdot \sigma_X$.

FTime The formula 2 indicates that the computation time is increased by a factor $(1 + N/2)$.

FNRJ Because we consider that useful and dummy instructions consume the same energy, the consumption of the circuit is also increased by a factor of $(1 + N/2)$.

$$
\begin{aligned}
FSCA_{IDI} &= \begin{cases} 1 & \text{if N=0} \\ 2 \cdot \sqrt{m \cdot \frac{N \cdot (N+2)}{6 \cdot (D+1)}} & \text{otherwise} \end{cases} \\
FDFA_{IDI} &= FSCA_{IDI} \\
FTime_{IDI} &= 1 + N/2 \\
FNRJ_{IDI} &= FTime_{IDI}
\end{aligned}
$$

Fig. 6. Performances for the IDI countermeasure

Impact of Random Power Generator

FSCA According to [7], if we call δ the amplitude of the SCA peak (i.e. the difference of the power consumption or electromagnetic radiation between data) and σ_c the standard deviation of the power consumption curve, the number of power curves necessary to recover the key has to be higher than $(\sigma_c/\delta)^2$. The activation of the R random generators increases this number by a factor $(1+R^2)$.

FDFA We suppose that the RPG does not protect against differential fault attacks.

FTime As the RPG processes at the same time as the computation, computation time is not increased by its activation.

FNRJ We consider that the power consumption of an RNG is equal to α times the temporal mean of $\mu_c(t)$ (with $\alpha = 10\%$ because the random number generator is a small piece of hardware). In these conditions, the total energy consumption of the circuit is increased by a factor $(1 + \alpha \cdot R)$.

Combination of countermeasures When the different countermeasures are combined, factors computed above for the different countermeasures are simply multiplied for side-channel attacks, for the time of computation and the energy consumption. For the DFA, the chance of making the same fault (in order to bypass detection of RL) on the targeted result (whose position is blurred with IDI) is equal to the chance obtained in the first occurrence (i.e. $1/(2^q \cdot 2 \cdot \sigma_X)$) up to the redundancy level minus one. So we obtain:

$$
\begin{aligned}
FSCA_{RPG} &= 1 + R^2 \\
FDFA_{RPG} &= 1 \\
FTime_{RPG} &= 1 \\
FNRJ_{RPG} &= (1 + \alpha \cdot R)
\end{aligned}
$$

Fig. 7. Performances for the RPG countermeasures

$$
\begin{aligned}
FSCA &= FSCA_{RPG} \cdot FSCA_{IDI} \cdot FSCA_{RL} \\
FDFA &= FDFA_{RPG} \cdot FDFA_{IDI}^{RL-1} \cdot FDFA_{RL} \\
FTime &= FSpeed_{RPG} \cdot FSpeed_{IDI} \cdot FSpeed_{RL} \\
FNRJ &= FNRJ_{RPG} \cdot FNRJ_{IDI} \cdot FNRJ_{RL}
\end{aligned}
$$

Fig. 8. Performances for combination of the countermeasure

Annexe 2: Fuzzy logic analysis

The process chosen for inferring a decision from fuzzy rules and inputs was first proposed by Mamdani in [16]. As reported in the following pseudo-code, this process consists in fuzzifying the inputs (according to the fuzzy subsets described in section 7.1), then computing (according to 7.2) the degree of truth of the rules described in section 3.2, then aggregating the results of these rules (according to 7.3) and lastly "defuzzyfying" them to obtain the output values (according to 7.4). Only the computation of the misuse level is described in detail below.

Algorithm 1 Algorithm for calculating the Misuse Level (ML)

Require: Inputs: Scalar values of the inputs ($\boldsymbol{S}$)
Ensure: Output: Misuse Level (ML)
Require: Fuzzy Sets for inputs (see 7.1)
Require: Fuzzy Sets for the outputs (see 7.1)
Require: The set of rules (see 3.2)
  Fuzzify the values of the inputs
  Compute the degree of truth of each rule of the rule set (see 7.2)
  Aggregate all the rules to obtain the membership function of the ML (see 7.3)
  Defuzzify this membership function to obtain the scalar value of the ML (see 7.4)

7.1 Fuzzy subsets

Definition The significance of the words (or "linguistic variables") "low" and "high" for the inputs and outputs are defined with "fuzzy subsets". The *membership function*, denoted μ_A, of a fuzzy subset A is a generalization of the characteristic function in classical set theory. It is any function mapping the values s of the input S to the real unit interval $[0, 1]$. The value $\mu_A(s)$ is called the *membership degree* of s in the fuzzy subset A. This degree quantifies the grade of membership of the element s to the fuzzy subset A: the value 0 means that s is not a member of the fuzzy set; the value 1 means that s is a full member of the fuzzy set. The values between 0 and 1 characterize fuzzy members, which belong to the fuzzy set only partially. The degree of membership also quantifies the degree of truth of the assertion "s is A" (which is true with a degree $\mu_A(s)$).

Membership functions for inputs Firstly consider the fuzzy sets associated with the value of the input S^i. The 8 membership functions (or fuzzy subsets) selected for each input S^i are presented in Table 2. For example, let us consider the voltage sensor input whose maximum value is equal to $VS_{max} = 10$. If this input is equal to 3 (and so is comprised between $VS_{max}/5$ and $2 \cdot VS_{max}/5$) it is considered "rather high" with a grade of truth of 1/4 and "very low" with a grade of truth of 1/2. If this voltage sensor value is equal to 7 (and so is comprised between $3 \cdot VS_{max}/5$ and $4 \cdot VS_{max}/5$) it is considered "high" with a grade of truth of 2/3 and "very very low" with a grade of 0.

Name	Rather Low	Low	Very Low	Very Very Low	Very Very High	Very High	High	Rather High
Acronym	$L_{-}^{S^i_{max}}$	$L_{--}^{S^i_{max}}$	$L_{---}^{S^i_{max}}$	$L_{----}^{S^i_{max}}$	$H_{++++}^{S^i_{max}}$	$H_{+++}^{S^i_{max}}$	$H_{++}^{S^i_{max}}$	$H_{+}^{S^i_{max}}$
$s^i \in [0; S^i_{max}/5]$	1	1	1	1	0	0	0	0
$s^i \in]S^i_{max}/5; 2 \cdot S^i_{max}/5]$	3/4	2/3	1/2	0	0	0	0	1/4
$s^i \in]2 \cdot S^i_{max}/5; 3 \cdot S^i_{max}/5]$	1/2	1/3	0	0	0	0	1/3	1/2
$s^i \in]3 \cdot S^i_{max}/5; 4 \cdot S^i_{max}/5]$	1/4	0	0	0	0	1/2	2/3	3/4
$s^i \in]4 \cdot S^i_{max}/5; S^i_{max}$	0	0	0	0	1	1	1	1

Table 2. The 8 membership functions (or fuzzy subsets) for an input S^i (S^i_{max} is the maximum value of this input)

Membership functions for outputs The outputs of the analysis mechanisms are the levels of anomaly and misuse. Their values are real values in $[0, 1]$. The two different membership functions "LOW" and "HIGH", considered for these outputs are described in Table 3:

Name	LOW	$HIGH$
$o \in [0;0,2]$	1	0
$o \in]0,2;0,8]$	$-5/3 \cdot o + 2/3$	$5/3 \cdot o - 1/3$
$o \in]0,8;1]$	0	1

Table 3. Membership functions for outputs

Operations on fuzzy subsets The boolean operations such as AND, OR and NOT are also defined in fuzzy logic. In the following, we consider the standard Zadeh operators on fuzzy subsets A_0 and A_1 respectively defined on variable x and y such as:

$$\begin{aligned} Z\_NOT &: \mu_{NOT(A_0)}(x) = 1 - \mu_{A_0}(x) \\ Z\_AND &: \mu_{AND(A_0,A_1)}(x,y) = \min(\mu_{A_0}(x), \mu_{A_1}(y)) \\ Z\_OR &: \mu_{OR(A_0,A_1)}(x,y) = \max(\mu_{A_0}(x), \mu_{A_1}(y)) \end{aligned}$$

The extension of Z_AND and Z_OR to "n-ary" operators is trivially defined and denoted as:

$$\begin{aligned} \mu_{AND(A_0,\ldots,A_k)}(x_0,\ldots,x_k) &= \min(\mu_{A_0}(x_0),\ldots,\mu_{A_k}(x_k)) \\ &= \min_{j=0}^{k}(\mu_{A_j}(x_j)) \\ \mu_{OR(A_0,\ldots,A_k)}(x_0,\ldots,x_k) &= \max(\mu_{A_0}(x_0),\ldots,\mu_{A_k}(x_k)) \\ &= \max_{j=0}^{k}(\mu_{A_j}(x_j)) \end{aligned}$$

The degree of truth of the premise is a real number in $[0,1]$ which depends on the values $\boldsymbol{S}$ of the inputs. This degree of truth of the premise of the rule i is denoted $pre_i(\boldsymbol{S})$.

7.2 Compute the degree of truth of the rules

Computing the values of the premises The degree of truth of each premise is computed first by evaluating the degree of membership of the input S^i to fuzzy subsets. Then, if necessary, the Zadeh operators are applied according to the premise's formula.

It is important to note that because we have chosen input membership functions that are discontinuous and because we have chosen Zadeh operators, regardless of the values of the entries and regardless of the different formulæ used to express the premise i, the result $pre_i(\boldsymbol{S})$ is always an element of the set $P = \{0; 1/4; 1/3; 1/2; 2/3; 3/4; 1\}$;

Modification of the membership function of the conclusion of a rule In the method proposed by Mamdani, the degree of truth of the premise of a rule modifies the membership function of its conclusion. The modification consists in

truncating the membership function $A_k(y)$ of the conclusion with the value of the premise, that is:

$$\mu_{R_k}(y|\boldsymbol{S}) = \min(pre_k(\boldsymbol{S}), \mu_{A_k}(y))$$

In fuzzy logic inference mechanisms, all the rules are considered to fire in parallel. So, at first glance, the use of this logic could lead to inconsistency, that is, several rules could lead to different conclusions (for example, the misuse is both HIGH and LOW). The aim of aggregation of rules and defuzzification is to compute a unique value for the decision.

7.3 Aggregation of rules

The different rules are considered to be linked together with the operator OR. So, combining the rules consists in taking for all $y \in [0, 1]$, the maximum value of the conclusions of the different rules, according to the following formula:

$$\mu_{\mathcal{R}}(y|\boldsymbol{S}) = \max_{k=0}^{p}(\mu_{R_k}(y|\boldsymbol{S}))$$

7.4 Defuzzification

The operation of defuzzification consists in calculating a scalar value (also called "crisp" value) for the output membership function of the conclusion ($\mu_{\mathcal{R}}(y|\boldsymbol{S})$), obtained given a set of entries and a set of rules. Before explaining this process of defuzzification, this output membership function is rewritten by taking into account the properties of our set of rules. We distinguish the rules which conclude to a LOW value for misuse (noted "LOW-m" rules) and those which conclude to the HIGH values for misuse (noted "HIGH-m" rules). Without loss of generality, we reorder the rules so that those numbered from 0 to $q-1$ are "LOW-m" rules and those numbered from q to p are "HIGH-m" rules. The considered set of rules is noted $\mathcal{R}$.

Output membership function rewriting The fuzzy subset of $LOW-m$ is defined as:

$$\begin{aligned}\mu_{LOW-m}(y|\boldsymbol{S}) &= \max(\min(pre_0(\boldsymbol{S}), \mu_{LOW}(y)),\\ &\quad \ldots, \min(pre_{q-1}(\boldsymbol{S}), \mu_{LOW}(y))\\ &= \min(\max(pre_0(\boldsymbol{S}),\\ &\quad \ldots, pre_{q-1}(\boldsymbol{S})), \mu_{LOW}(y))\\ &= \min(\max_{k=0}^{q-1}(pre_k(\boldsymbol{S})), \mu_{LOW}(y))\end{aligned}$$

In the same way, the fuzzy set associated to the "HIGH-y" rules is defined as:

$$\mu_{HIGH-m}(y|\boldsymbol{S}) = \min(\max_{k=q-1}^{p}(pre_k(\boldsymbol{S})), \mu_{HIGH}(y))$$

As explained in section **??**, the different premises pre_k are in the set P. So, let us define p_l and p_h and the membership functions $\mu_{LOW-m}(y)^{p_l}$ and $\mu_{LOW-m}(y)^{p_h}$ such as:

$$\begin{aligned} p_l &= \max_{k=0}^{q-1}(pre_k(\boldsymbol{S})) \in P \\ p_h &= \max_{k=q}^{p}(pre_k(\boldsymbol{S})) \in P \\ \mu_{LOW-m}(y)^{p_l} &= \min(p_l, \mu_{LOW}(y)) \\ \mu_{HIGH-m}(y)^{p_h} &= \min(p_h, \mu_{HIGH}(y)) \end{aligned}$$

The output membership function for the output y can in these conditions always be written in the following form:

$$\mu_{\mathcal{R}}(y)^{p_l,p_h} = \max(\mu_{LOW-m}(y)^{p_l}, \mu_{HIGH-m}(y)^{p_h})$$

Defuzzification techniques There are different kinds of defuzzification techniques to compute the crisp output from the output membership function. We have considered four of the more popular ones, called "centroid" (CT), "mean of max" (MM), "first of max" (FOM) and "last of max" (LM). For example, the FOM crisp value is computed using the formula:

$$FOM(\mu(y)) = min\{v \in [0,1] | \mu(v) = SUP_{y\in[0,1]}\{\mu(y)\}\} \tag{4}$$

Then, we have computed their values for the whole set of possible values of premises, that is, for all of the 49 membership functions $\mu_{\mathcal{R}}(y)^{p_l,p_h}$ with $P_h \in P$ and $P_l \in P$. The results for First Of Max ("FOM") is reported in Table 4.

	P_h						
FOM	0.00	0.25	0.33	0.50	0.67	0.75	1.00
0	0	0.35	0.4	0.5	0.6	0.65	0.8
0.25	0	0	0.4	0.5	0.6	0.65	0.8
0.33	0	0	0	0.5	0.6	0.65	0.8
P_l 0.5	0	0	0	0	0.6	0.65	0.8
0.66	0	0	0	0	0	0	0.8
0.75	0	0	0	0	0	0	0.8
1	0	0	0	0	0	0	0

Table 4. First of Max results for $\mu_{\mathcal{R}}(y)^{p_l,p_h}$

Annexe 3: Hardware architecture details

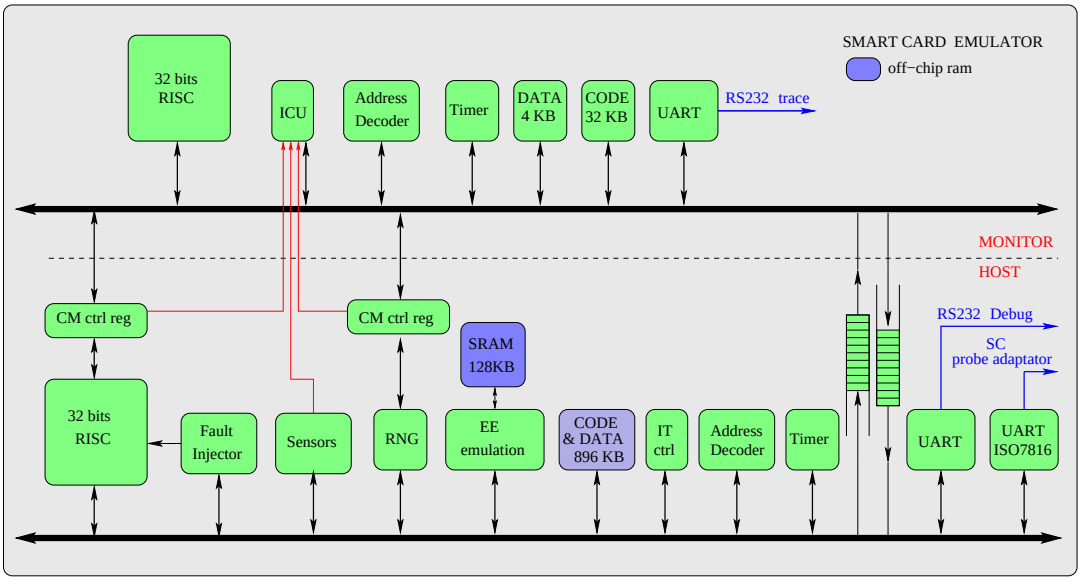

Fig. 9. Hardware part of the prototype

Annex 4: Simulations of scenarios

The second analysis consists in defining several attack and normal use scenarios and playing these scenarios on an algorithmic model of the prototype. For example, consider a scenario where the card is connected to a low quality card reader. In such an abnormal case, the electrical connection between the two devices regularly triggers the voltage sensor. In the first part of the scenario (called I), there is no error and the level of security remains low, as represented in Figure 10 (top). In this figure, the x-axis corresponds to the time and the y-axis to the levels of security. In the second part (called II) of the scenario, the voltage sensor alone is triggered. The security level increases slowly (because we consider that these mistakes are not important). In the third part (called III) of the scenario, the MAC error sensor ME is also triggered. In this part, the security level increases quickly. In the last part of the scenario (called IV), the sensors stop being triggered and the security level decreases quickly. On the contrary, let us consider a laser attack scenario. In the first part of this scenario, there are no errors and the level of security remains low, as represented in Figure

10 (bottom). In the second part, the light sensors are triggered because the attacker injects faults in the middle of a long sequence of correct commands. Even with this precaution, the level of security increases quickly (we consider that the triggers of the light sensor are important). We suppose that the attacker is able to detect this increase (the activation of the RPG when the system switches to the "unsafe" configuration is easy to detect) and that, in the third part of the scenario, he stops to inject faults. The level of security tends to decrease. But when the laser attack is re-started in the fourth part of the scenario, the monitor increases the security level very quickly until sensitive data is deleted.

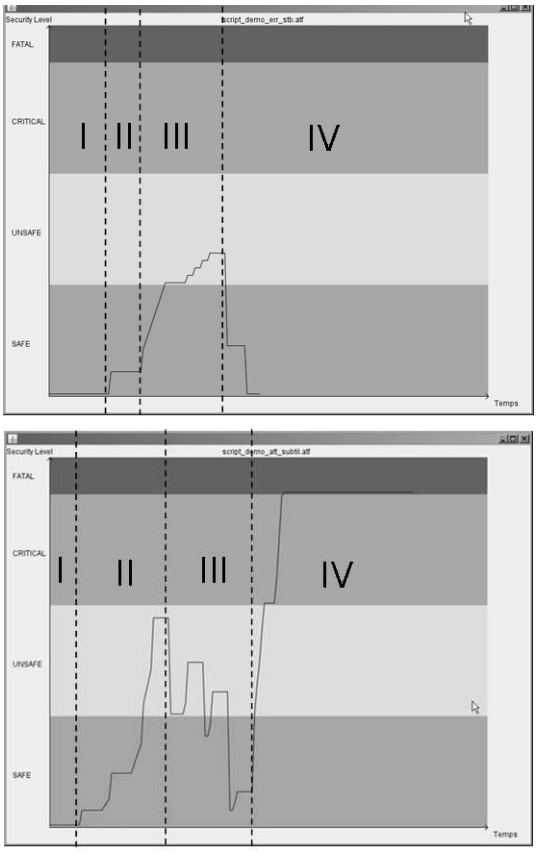

Fig. 10. Response of the system for an anomaly case (top) and for an attack case (bottom)