Management of the security in smart secure devices

B. Robisson, M. Agoyan, S. Bouquet, M.-H. Nguyen, CEA-LETI, Gardanne, France
S. Le Henaff, P. Soquet, Viaccess, La Défense, France
G. Phan, Trusted Logic, Versailles, France
F. Wajsbürt, P. Bazargan-Sabet, N. Drach-Temam, LIP6, Paris, France

1 Introduction

Security is a key component for information technologies and communication. It undoubtedly represents one of the main tools of its rise because it introduces confidence which is necessary for users. Among the security threats, vulnerability of electronic material that implements cryptography (such as smart card), for services of confidentiality, authentication and data integrity, is perhaps the most important. Indeed, some unfaithful means, or 'attacks', on this material make it possible to extract confidential information like encoding keys and thus to lower the security of all the protected transmission chain of information.

There are three main kinds of such ``physical'' attacks. The first kind, called ``side channel attacks'', exploits the fact that some physical values such as the power consumption, the electromagnetic radiation or the duration of computation of the chip depend on its internal computations [KJ99,MP07]. It is of particular concern, since it does not destroy the physical integrity of smart cards and it can be quickly mounted with cheap instrumentation equipments. The second type, called ``fault attacks'', consists in modifying the circuit's behavior in order to bypass hardware or software protections or to exploit computational errors [BS97,PQ03]. The faults are injected into the device by various means as laser, glitches on clock [AD10], spikes on voltage supply or electromagnetic perturbations. The third type, called ``invasive attacks'' [KK99], consists in analyzing the design of the chip by using destructive means such as abrasion, chemical etching and then probing the most informative signals with, for example, focused ion beam.

A lot of protections have been proposed to counter those attacks. The first kind (further called “sensors”) gives 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 error during computations. This detection is generally based on spatial redundancy (ie. realizing the same computation several times in parallel) or temporal redundancy (i.e. realizing the same computation several times) [MR10,NR10]. The second kind (further called “actuators”) does not modify the functional behavior of the circuit but only secures it. To reduce sensibility to side channel attacks, noise has been added to the power consumption, 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 [RP09,CK09]. Another way to reduce sensibility to side channel attacks consists in reducing the correlation between the physical values and the data processed, for example, by using balanced data encoding and balanced place and route [GS10], by using power filters or electromagnetic shields. The last kind of countermeasures (further called “reactions”) modifies the functional behavior of the circuit. A reaction consists in stopping temporary the communication with the reader (the card “mutes”) and/or

resetting parts of the running software. The ultimate reaction consists in definitively destroying (ie. killing) all the data (including the sensitive information) stored in the chip.
In practice, the security of the chip is achieved by a combination of such countermeasures. Nowadays, state-of-the-art circuits can protect data during weeks, months and even years. But the implementation of these countermeasures not only drastically decreases the performances of the chip in terms of power and speed but also its availability. That's why, to trade-off the security and performances some protections should be parameterized or activated/deactivated. We propose hardware and software mechanisms which enable the circuit to choose the best parameters of the actuators or the most appropriated reactions according to given values of the sensors. This mechanism of decision is further called "strategy of security".
In a first step, the expected properties for a complex strategy of security are detailed. In a second step, a representative case study (a conditional access for mobile pay-TV) and an example of complex strategy of security based on fuzzy logic are described. At last, an innovative hardware/software architecture enabling the execution of such a strategy is described.

2 Specifications of the strategy of security

2.1 Misuse/Anomaly detection

The availability is a key feature of a component. It has to be operational during the main part of its life time because its untimely replacement is costly for the circuit's provider. But security and availability are often conflicting features. For example, let's consider the hardware voltage sensor implemented to detect voltage glitch attacks. Its detection's level may be set at design time to a particular value. If this value is too low, the sensor may trigger when the chip is connected to a non malicious but low quality card reader. It may be interpreted as a fault attack at the application level, causing the self-destruction of the card. Such a normal functioning considered as an attack will further be called "false positive" case. On the contrary, if the value of the detection's level is too high, real glitch attacks may not be detected. Such an attack considered as a normal functioning will further be called "false negative" case.
The aim of a complex strategy of security is to drastically reduce both the rate of false positive and false negative cases. In the first case the availability is increased. In the second case, the security is increased. One way to reduce these rates consists in applying methods proposed in Intrusion Detection Systems (or IDS) [B00]. The first method consists in determining the whole set of states of the system reached by a user having a ``normal'' behavior. Each time the state of the system goes out of this set, an "anomaly" is detected. The second method is complementary. It consists in determining the whole set of states which have to be reached by an evil-minded user to perform his attack. When the system enters this set, a misuse (considered as 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 a robust and a secure behavior, mechanisms have to detect both anomaly and misuse.

2.2 Dynamic reconfiguration of countermeasures

The performances of the component in terms of speed and power consumption are also two key features. The component has to be as fast as possible, for example for pay-TV applications where the switch from channel to channel has to be quite immediate. It also has to consume as few as energy as possible, for example for mobile applications where the energy is a critical resource. But security and performances are also often conflicting features. For example, let's consider the countermeasure which consists in inserting random instructions between genuine ones in order to add "noise". The more numerous are the random instructions, the stronger is the security but both the speed and power consumption are impacted.

The aim of the strategy of security is to provide mechanisms which modify, if necessary, dynamically the trade-off between performances and security, i.e switching the circuit from a low-performance but secured state to a high-performance but poorly secured one.

3 Case study

In the following sections, the major components of the chosen representative system (further called the "host system") are described. Its software parts are the application (the smart card part of a conditional access system) and the virtual machine. Its hardware parts are the micro-controller and the countermeasures. Then, a complex strategy of security tailored for this application is described.

3.1 Conditional access for mobile pay-TV

Conditional Access Systems (CAS) protect a content (such as radio, TV, data stream) by requiring certain criteria to be met before granting access to this content. The security of the content is generally achieved by combining protections at different levels. The following description of the functioning of a CAS (represented on Figure 1) is a very simplified one but aims to give its general philosophy.

When a subscriber pays for the access to a certain content, he receives a smart card in which is stored a management key (MK) and inserts the smart card in his receiver (for example embedded in a mobile phone).

The access provider generates an entitlement management message (EMM). This EMM contains access rights (AR) that are parameters such as subscriptions, the period of validity of these subscriptions, the geographical location of the receiver, etc. It also contains exploitation keys (EK). Then, the EMM is ciphered and signed with MK. The obtained cryptogram is transmitted into the receiver (typically at a rate of a month), for example via SMS. The smart card deciphers and authenticates the cryptogram by using MK and stores EK and AR in its persistent memory.

The content provider scrambles its content with an 8-byte secret key, called a control word (CW) and generates a message, called the entitlement control message (ECM). This ECM contains the access criteria associated within the content (CA) and the value of the CW. The ECM is then ciphered and signed by using EK. Under normal conditions, content providers change the control word every 5 to 10s. Then, the scrambled content and the cryptogram are broadcasted to the receiver. The smart card deciphers and checks the integrity of the cryptogram by using EK. Then, it compares the access rights (AR) to the access criteria (AC). If they match, the CW is used by the receiver to de-scramble the content.

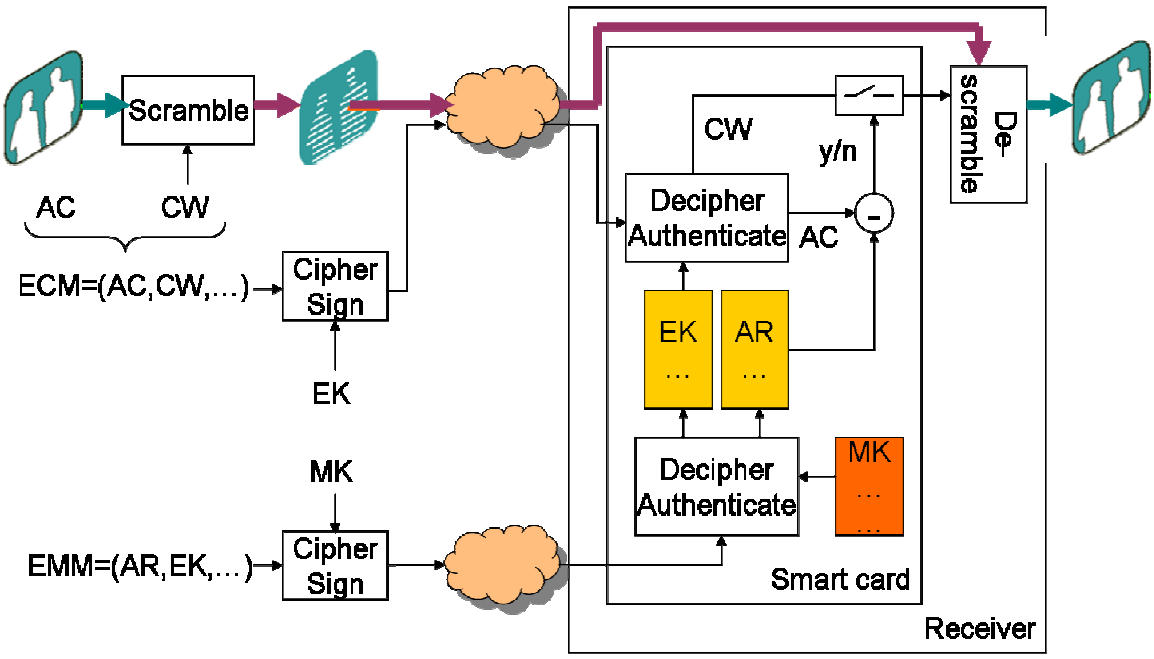

Figure 1: Simplified view of a Conditional Access System

It is important to note that the corruption of the exploitation key (EK) enables the hacker to decipher the CW during the period of validity of EK. With CW, he gains access to the content and is able to change the access criteria. So, the corruption of EK is a very serious security threat. On the contrary, knowing the value of CM at a given moment is of relatively little value because it is only valid for a very short period of time. This highlights that some data are more sensitive (EK or MK) than others (CW, Personal Identification Number or parental control password). In this example, it is very interesting to be able to dynamically have a trade-off between performance and security in order to enhance the autonomy of the mobile phone.

3.2 Virtual Machine

In order to be as close as possible to a real product and in order to benefit of efficient software security features, a state-of-the-art virtual machine (VM) has been implemented in the host system. The chosen VM complies with the Java Card 2.2.2 specifications [JC], and provides card content management capabilities based on the GlobalPlatform standard [GP]. 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).

3.3 Host's micro-controller

The host sub-system is built on a 5-stage pipelined 32-bit Harvard RISC micro-controller able to execute an instruction at each clock cycle. The instructions are stored into 640kB of ROM and the data into 256kB of RAM and into 128kB of EEPROM. The peripherals comprise two Universal Asynchronous Receiver Transmitter (or UARTs) and an Advanced Encryption Standard crypto-engine (to accelerate the ciphering and deciphering of data). Note that even though the hard wired countermeasures described below are physically embedded in the host sub-system, they cannot be directly controlled by it.

3.4 Countermeasures

3.4.1 Security sensors

Several sensors have been implemented or emulated. Some of them are ``physical'' sensors which measure, for example, the voltage, light intensity or the temperature of the chip. Others are ``logical'' sensors which detect errors during I/O data transmissions or during computations (with the aid, for example, of Cyclic Redundancy Check). Software components (in the virtual machine or in the application) may also detect errors during the execution of a piece of code (for example, by dynamically verifying the execution flows and precomputed ones). At last and for the uniqueness of the notation, the kind of data (sensitive or non sensitive ones) which are manipulated by the application is considered as a particular sensor. The different sensors taken into account in our study and their different possible output values are described in Table 1.

Name	Values	Description	Triggered by
LS	{0,…,5}	Light value	Hardware Sensor
VS	{0,…,10}	Voltage value	Hardware Sensor
EE	{0,…,10}	# of corrupted execution flow	Virtual Machine
CE	{0,…,10}	# of corrupted cryptographic execution	Virtual Machine
PE	{0,…,10}	# of wrong PIN	Application
UE	{0,…,50}	# of error during the test of integrity of the user data	Application
NE	{0,…,1000}	# of methods that have processed without error	Virtual Machine
DS	{0,…,5}	Sensitivity of the data manipulated by the application	Application

Table 1: Description of security sensors

3.4.2 Security actuators

The first actuator is dedicated to counteract side channel attacks. It consists in increasing the level of noise by inserting dummy instructions (ie., that are not useful) during the computations. The number of useful instructions divided by the total number of instructions is called the ``ratio of useful instructions'' (noted ``RUI''). It is noted *“1/N”*, for 1 useful instruction out of N instructions. It is equal to 1 when the countermeasure is not activated. The second mechanism is dedicated to counteract fault attacks. It consists in detecting errors by performing several times the same computation and then comparing the results. If they are identical, the computation is considered as error-free. 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 countermeasure is not activated.

3.4.3 Security reaction

The first reaction consists in stopping temporary the communication with the reader (the card mutes) and/or resetting part or the whole running software. Another reaction consists in definitively destroying (ie. killing) all the data (including the sensitive information) stored in the chip.

3.5 Proposed strategy of security

According to principles developed for IDS, the chosen strategy of security is decomposed into 3 different processes: The first consists in collecting information about the state of the host system, the second in computing the anomaly and misuse levels (to the aim described in 2.1) and the third in activating the security actuators or in triggering a reaction (to the aim described in 2.2).

3.5.1 Collecting information

We consider that the set of all the possible outputs for all the security sensors described in 3.4.1 constitutes the state of the host system.

3.5.2 Analysis: Computation of anomaly and misuse level

As the secure circuit designer generally expresses the anomaly and misuse cases with words which are very often vague and imprecise, fuzzy logic has been used [DP00]. In this framework, the anomaly and misuse are mainly described by a set of ``IF-THEN'' rules expressed with words such as LOW, HIGH and adverbs such that RATHER, VERY whose meanings are precisely defined with ``fuzzy sets''. For our system, a dozen of rules such as the following are used:

- 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 errors (PE), the voltage (VS) and the light (LS) are VERY LOW THEN the anomaly is LOW''
- Etc.

Next, given the values of the sensors, the inference process (not detailed in this article but which follows the method proposed by Mamdani [M77]) computes two real numbers between 0 and 1, respectively denoted anomaly level (or AL) and misuse level (or ML). The closer the number to 1 is, the higher is the degree of truth of the respective assertions ``it is an anomaly'' or a ``it is a misuse''.

3.5.3 Configuration of countermeasures

For the sake of simplicity, only five possible configurations of those countermeasures are taken into account. The chosen configurations are reported in Table 2

Name	RL	RUI	Reset	Kill
Safe	0	No	No	No
Anomaly	$\times 2$	1/10	No	No
Unsafe	$\times 3$	1/100	No	No
Critical	--	--	Yes	No
Final	--	--	--	Yes

Table 2: Configuration of countermeasures

The function which assigns the configuration to each value of the anomaly and misuse levels is depicted in Figure 2. The different thresholds have been determined according to the parameters chosen for the analysis process.

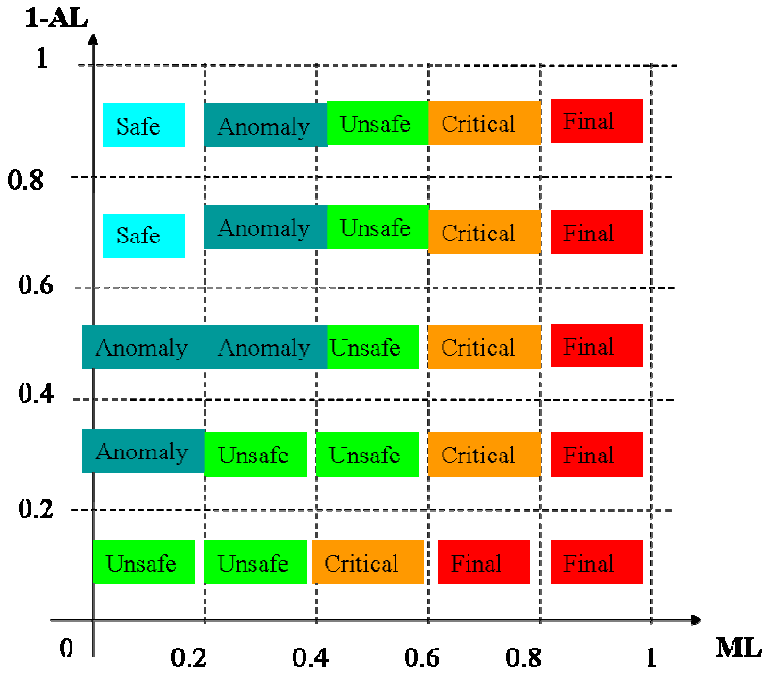

Figure 2: Relation between the anomaly and misuse levels and the configurations

4 Prototype

4.1 Architecture

In order to avoid additional information leakage, we choose an architecture which separates the treatments related to application data and those related to the strategy of security. This leads to the adjunction of a small system called “monitor”, represented on **Figure 3** and strictly dedicated to the execution of the strategy. Its software parts are the strategy of security and the component-based OS. Its hardware part consists of the micro-controller and channels to communicate with the host system.

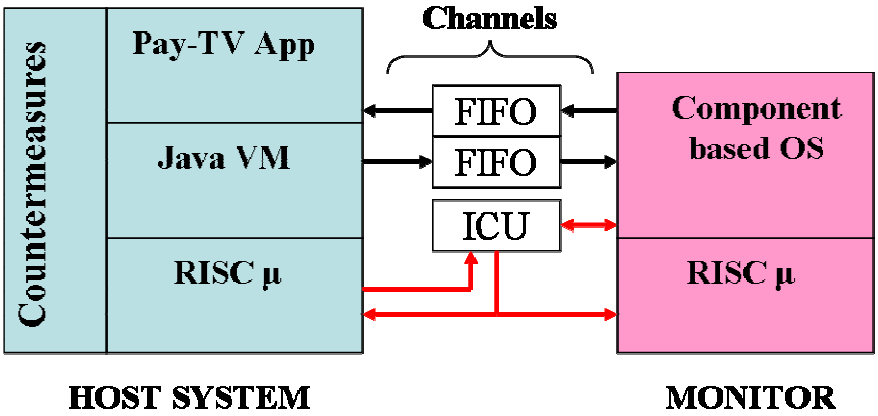

Figure 3: Architecture of the system

4.1.1 Component based OS and strategy of security

The monitor system uses a specific component-based operating system called “Sloop”. It needs a small memory footprint (less than 4kB) and it responds as fast as possible (typically a hundred of cycles) to the host requests. A Sloop application is a set of components linked together through exchange memory zones. A Sloop component is a set of ports towards theses exchange zones, of private attributes and of methods (which describe its behavior). The strategy of security in included in one of the Sloop’s components.

4.1.2 Monitor's micro-controller

The monitor's micro-controller 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 4kB 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) and in return, the monitor is able to change the configuration of the countermeasures embedded into the host.

4.2 Example of communication between the host and the monitor

The following example describes one communication between the host and the monitor:

1. The application indicates that the sensitivity of the data that it manipulates (DS) changes.
2. The virtual machine sends this information to the monitor via the FIFOs and waits.
3. The monitor analyses the values of the security sensors: the anomaly and misuse levels are computed (according to 3.5.2) and a configuration of the countermeasures (according to 3.5.3) is selected.
4. The monitor asks to the host system to configure software (via FIFOs) and hard wired countermeasures (via the ICU).
5. The monitor waits until all the countermeasures are configured and are ready.
6. The monitor asks to the host system to resume its execution.

5 Conclusion

In this article, we have described an innovative hardware/software architecture aiming to improve the availability and the performances of a circuit without compromising its security. This architecture enables to easily implement complex strategies of security. We show that, for example in the context of conditional access for pay-TV, such strategies could be expressed as a set of intelligible rules. Future work will consist in refining this set of rules by performing real attacks on the prototype. It could also be interesting to investigate, in parallel, more expressive formalisms such as those used for the detection of intrusion in complex computer systems.

6 Acknowledgement

This work has been supported by the ``Agence National de la Recherche'' and by the ``Solutions Communicantes Sécurisées'' competitiveness cluster via the project

Smart On Smart (ANR-07-SESU-014-01). The authors are grateful to the anonymous reviewers and Jacques Fournier for their suggestions of improvements.

7 Bibliography

[AD10] M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria. « When clocks fail: On critical paths and clock faults ». In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, CARDIS, vol 6035 of LNCS, pages 182–193. Springer, 2010.

[B00] R. G. Bace, "Intrusion detection". Indianapolis, IN, USA: Macmillan Publishing Co., Inc., 2000.

[BS97] E. Biham and A. Shamir. « Differential fault analysis of secret key, cryptosystems ». In B.S. Kaliski Jr., editor, Advances in Cryptology −CRYPTO'97, vol 1294 of LNCS, pages 513–525. Springer, 1997.

[CK09] J.-S. Coron, I. Kizhvatov, « Analysis and Improvement of the Random Delay Countermeasure of CHES 2009 », in Cryptographic Hardware and Embedded Systems, vol 6225 of LNCS, 2010, pp 95-109.

[DP00] D. Dubois and H. Prade, Fundamentals of Fuzzy Sets Series: The Handbooks of Fuzzy Sets, Vol 7. Kluwer Academic, 2000.

[GP] "Global platform specifications website," http://www.globalplatform.org.

[GS10] S. Guilley, L. Sauvage, F. Flament, V.-N. Vong, P. Hoogvorst, R. Pacalet: "Evaluation of Power Constant Dual-Rail Logics Countermeasures against DPA with Design Time Security Metrics". IEEE Trans. Computers 59(9): 1250-1263 (2010)

[JC] "Java card technology website," http://java.sun.com/javacard.

[KK99] O. Kömmerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in Proceedings of the USENIX Workshop on Smartcard Technology, Chicago, 10–11 May, 1999, pp. 9–20.

[KJ99] P. C. Kocher, J. Jaffe, and B. Jun. « Differential power analysis ». In Michael J. Wiener, editor, CRYPTO, vol 1666 of LNCS, pages 388–397. Springer, 1999.

[MR10] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent Structure-Independent Fault Detection Schemes for the Advanced Encryption Standard," IEEE Transactions on Computers, pp. 608-622, Vol. 59, No. 5, May 2010.

[M77] E. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," Computers, IEEE Transactions on, vol. C-26, no. 12, pp. 1182 – 1191, dec. 1977.

[MP07] E. O. Stefan Mangard and T. Popp, "Power Analysis Attacks Revealing the Secrets of Smart Cards". Springer Verlag, 2007.

[NR10] M.-H. Nguyen, B. Robisson, M. Agoyan, and N. Drach. « Low-cost fault tolerance on the ALU in simple pipelined processors ». In Proceedings of IEEE conference Design and Diagnostics of Electronic Circuits and Systems, pages p28–31, Vienne Autriche, 2010.

[PQ03] G. Piret and J.-J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and Khazad," in Cryptographic Hardware and Embedded Systems, vol 2779 of LNCS. Springer, 2003, pp. 77–88.

[RP09] M. Rivain, E. Prouff, Julien Doget, « Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers », in Cryptographic Hardware and Embedded Systems, vol 5747 of LNCS, 2009, pp 171—188