A Design Flow and Evaluation Framework for DPA-resistant Instruction Set Extensions

Francesco Regazzoni[1,4], Alessandro Cevrero[2,3], François-Xavier Standaert[1], Stephane Badel[3], Theo Kluter[2], Philip Brisk[2], Yusuf Leblebici[3], and Paolo Ienne[2]

[1]UCL Crypto Group, Université catholique de Louvain, Louvain-la-Neuve, Belgium.
{francesco.regazzoni,fstandae}@uclouvain.be
[2]School of Computer and Communication Sciences - EPFL, Lausanne, Switzerland.
{first_name.last_name}@epfl.ch
[3]School of Engineering - EPFL, Lausanne, Switzerland.
{first_name.last_name}@epfl.ch
[4]ALaRI - University of Lugano, Lugano, Switzerland. regazzoni@alari.ch

Abstract. Power-based side channel attacks are a significant security risk, especially for embedded applications. To improve the security of such devices, protected logic styles have been proposed as an alternative to CMOS. However, they should only be used sparingly, since their area and power consumption are both significantly larger than for CMOS. We propose to augment a processor, realized in CMOS, with custom instruction set extensions, designed with security and performance as the primary objectives, that are realized in a protected logic. We have developed a design flow based on standard CAD tools that can automatically synthesize and place-and-route such hybrid designs. The flow is integrated into a simulation and evaluation environment to quantify the security achieved on a sound basis. Using MCML logic as a case study, we have explored different partitions of the PRESENT block cipher between protected and unprotected logic. This experiment illustrates the tradeoff between the type and amount of application-level functionality implemented in protected logic and the level of security achieved by the design. Our design approach and evaluation tools are generic and could be used to partition any algorithm using any protected logic style.

1 Introduction

Security is a fundamental requirement for modern embedded systems. Mathematically strong cryptographic algorithms are insufficient due to the advent of side channel attacks, which exploit weaknesses in the underlying hardware platform rather than directly attacking the algorithm itself. At present, there is no perfect protection against side channel attacks. Hence, combining countermeasures implemented at different abstraction levels is necessary to reach a significant security level. In this context, solutions exploiting a dedicated technology such as protected logic styles are interesting because they directly tackle the problem of information leakage at their source. These logic styles can also be

combined with software countermeasures to increase the difficulty of performing an attack.

The main drawback of protected logic styles proposed so far is that their area and power consumption are both significantly greater than that of traditional CMOS. They can also significantly increase the design time compared to circuits built from standard cells. Hence, complete processors and ASICs implemented in protected logic styles are generally too costly for practical applications, and would likely have low production volumes.

To overcome this issue without compromising security, protected logic styles must be used sparingly. With this respect, an interesting alternative is to build processors and ASICs that are realized primarily in CMOS logic, with a small and security-critical portion of the design realized in a protected logic. This creates a new and challenging partitioning problem that designers must be able to solve. But doing so will be quite difficult unless there is a suitable metric to evaluate and compare the security of a computation performed in either CMOS or a protected logic. Additionally, CAD tools must be able to support complex design flows that mix different logic styles. Finally, there is a distinct need for a comprehensive evaluation framework that combines a simulation environment with suitable metrics and provides a fair and accurate comparison of designs in respect to different criteria (e.g., power consumption, area, throughput, security).

To this end, this paper proposes a complete design flow for implementing and verifying circuits realized combining protected and non protected logic styles. Our design flow is built from standard CAD tools and is integrated with a methodology to evaluate the security of the designs that have been partitioned, following a theoretical framework for analyzing the information leakage provided by side-channel attacks. Focusing on a processors augmented with custom Instruction Set Extensions (ISEs) realized in protected logic styles, we explore the tradeoffs between the type and amount of application-level functionality implemented in protected logic and the level of security that can be achieved by the design. In our experiments, in particular, we vary the portions of the cryptographic algorithm that are realized in protected logic and CMOS, which gives us a better understanding of the tradeoffs between the usage of protected logic and security.

Starting from an RTL description of the target processor and a software implementation of a cryptographic algorithm, our tool allows the user to manually select the sensitive parts of the algorithm to be protected. Our design flow then automatically generates the new software, the ISEs and their interface to the processor, synthesizing a complete system as described above. The power consumption of the full system is simulated at the SPICE level while running the application. These power traces are then used to compute an information theoretic metric in order to evaluate the information leakage of the protected core. We have selected MOS Current Mode Logic (MCML) as the protected logic for use in this study. However, the ideas presented in this paper are generally amenable to any type of protected logic that is compatible with a CMOS process.

The remainder of the paper is organized as follows. Section 2 summarizes previous works in the area of side channel attacks. Section 3 recalls the metric used to evaluate the side-channel leakage and presents several extensions that were necessary to make it usable in the context of our design flow. Section 4 describes our hybrid design and evaluation methodology. Section 5 presents the results we obtained applying our methodology to the PRESENT block cipher and discusses the security vs. cost tradeoffs. Section 6 concludes the paper.

2 Background and Related Work

Side channel attacks are a powerful and easy to perform class of physical attacks. Consequently they have received much attention from the scientific community. The most frequently considered sources of side-channel information are power consumption and timing characteristics [13, 12]. To perform Differential Power Analysis (DPA), the attacker executes the cryptographic algorithm on a target device multiple times, and then uses statistical methods to evaluate the information observed from the executions. Countermeasures, such as algorithmic techniques [7, 23], architectural approaches [11, 18, 19], and hardware-related methods [20, 27] can help to protect against DPA. Even if none of them are perfect, these countermeasures increase the efforts required to mount a successful attack. In this paper, we are mainly concerned with technological solutions, usually denoted as *DPA-resistant logic styles* in the literature.

Many DPA-resistant logic styles have been proposed in the past. *Sense Amplified Base Logic (SABL)* [27], for example, combines dual-rail and pre-charged logic [17]. SABL cells consume constant power, provided that they are designed and implemented in a carefully balanced way. Other proposed DPA-resistant logic styles include: *Wave Dynamic Differential Logic (WDDL)* [28], which balances circuit activity with complementary logic gates; *Dynamic Current Mode Logic (DyCML)* [2, 15], a dual-rail pre-charge logic, similar to SABL, but with a reduced power-delay product; *Low-swing current mode logic (LSCML)* [9], which is similar to DyCML, but is independent of transistor sizes and load capacitances; *Masked Dual-Rail Pre-charge Logic (MDPL)* [21], which attempts to eliminate routing constraints that plague other dual-rail style gates; *Gammel-Fischer Logic (GF)* [8], a form of masked logic that protects against information leakage in the presence of glitches; finally, MCML that will be our running example [22], a MOS transistor-based current mode logic.

One of the key challenges when implementing protected logic styles is to analyze the DPA-resistance of the different operators in an application. This task is even more critical when partitioning a design between CMOS and a protected logic. To address possible shortcomings, Standaert et al. [24] introduced a combination of metrics that can be used to describe the amount of information leaked by a cryptographic device and the effectiveness of a side-channel adversary to exploit this information. When analyzing new countermeasures, it is primarily the information theoretic metric that is most useful, since it quantifies the reduction in information leakage resulting from the countermeasure using a sound criteria.

In theory, this metric yields an adversary-independent image of the asymptotic security of the device. This metric was first applied to DPA-resistant logic styles by Mace et al. [16], who describe in detail how to compute the entropy of a secret key conditionally with respect to the physical leakage in different scenarios. In this paper, we follow and extend this application of the metric.

Various studies on partitioning designs between CMOS and protected logic have been published in the literature. The most relevant work related to our concerns is probably the one of Tillich and Großschädl [26]. These authors analyze the resistance against side channel attacks of a processor extended with custom ISE for AES. They consider the possibility of implementing the most security-critical portions of the processor datapath in a DPA-resistant logic style. Our paper extends these initial ideas, providing different novel contributions. We present a fully automated design flow that allows realizing and simulating a complete environment (core + protected ISE). This proves the feasibility of combining CMOS and protected logic styles on the same chip and provides realistic measurements for area and power consumption. We also provide a more precise evaluation of the resistance against power analysis attacks for each design, due to the integration of an objective metric that quantifies the information leaked by different protected implementation. Lastly, our quantitative analysis applies jointly to security and performance issues and drives the process of ISE identification and synthesis; to the best of our knowledge, prior ISE identification methods have been driven primarily by performance [10, 4, 25].

3 Security Evaluation

The evaluation of the power consumption leakage provided by our simulation environment follows the principles of [24]. The goal of this methodology (that we don't detail here) is to provide fair evaluation metrics for side-channel leakage and attacks. In particular and as far as evaluating countermeasures or protected logic styles is concerned, the information theoretic metric that we now summarize allows being independent of a particular DPA attack scenario. It intuitively measures the resistance against the strongest possible type of side-channel attacks. In summary, let K be a random variable representing the key that is to be recovered in the side-channel attack; let X be a random variable representing the known plaintexts entering the target (leaking) operations; and let L be a random variable representing the power consumption traces generated by a computation with input x and key k. In our design environment, L is the output of a SPICE simulation trace T to which we add a certain amount of normally distributed random noise R, i.e. $L = T + R$. We compute the conditional entropy (or mutual information) between the key K and its corresponding leakage L, that is,

$$\mathrm{H}[K|L] = -\sum_{k} \Pr[k] \cdot \sum_{x} \Pr[x] \int \Pr[l|k,x] \cdot \log_2 \Pr[k|l,x] \; dl.$$

There are different factors that influence this conditional entropy. The first is the shape of the simulated traces T. The second is the standard deviation of

the noise in the leakage L. The number of dimensions in the traces is also important. Simulated traces contain several thousands of samples. Hence, directly applying multivariate statistics on these large dimensionality variables is hardly tractable. Mace et al. [16] reduce the dimensionality using Principal Component Analysis (PCA), and then evaluate the conditional entropy. Thus, the number of dimensions remaining after PCA is a third parameter that we consider.

Our usage of the metric builds on Mace et al.'s in two respects. First, we move from a 1-dimensional analysis to a multi-dimensional analysis, and we discuss the extent to which more samples increase the estimated value of $\mathrm{H}[K|L]$. Additionally, we analyze complete designs, rather than 2-input logic gates, thereby establishing the scalability of the aforementioned metric and evaluation tools.

4 The Proposed Hybrid Design Flow

This section describes the entire design flow, from RTL to the integration with the information theoretic metric discussed in the preceding section. For any application, there exists a range of possible architectural and electrical implementations, such as ASIC vs. processor, or standard cell vs. full custom design. The choice of the platform has historically been dictated by performance, area, and/or power consumption, each of which can be measured accurately. Once the initial design point is fixed, designers consider a fine-grained space of possible solutions, and only at this point is security typically considered, often based mainly on empirical evaluation. The aim of this work is to bring security to the forefront of design variables for embedded systems by associating it with a clear quantitative metric. To achieve our goal, we propose a flexible and fully automated design flow based on standard CAD tools. The flow supports partitioning of a design between CMOS and protected logic, and includes a sound metric to measure resistance against side-channel attacks.

Figure 1 depicts the design flow, which is an extension of prior simulation-based methodologies to evaluate resistance to power analysis attacks [6, 22]. The key idea is to use commodity from EDA tools and to leverage on transistor level simulation to provide a high degree of accuracy. The design flow that is used in this study targets an embedded processor that is augmented with instruction set extensions. The ISE are designed using information leakage as part of the objectives to optimize, and are implemented in a protected logic style. This point represents a major innovation, since in recent years, ISEs have been used primarily to improve performance, rather than to enhance security.

The flow has two inputs: the RTL description of an embedded processor that supports ISEs, and a software implementation of a cryptographic algorithm. The software code is passed to a tool that automatically extracts its data flow graph. The user manually select from the aforementioned graph the portion of the algorithm to be implemented in protected ISEs. Once this selection is done, the remainder of the flow is fully automated: the rewritten software, including an explicit call to the ISE is generated, along with an RTL description of the ISE and its interface to the processor. The output of the flow is a place-and-routed

hardware design that is evaluated using a sound metric that measures information leakage. By iterating through different ISE implementations, an architect could generate a set of a design among which select the most suitable one.

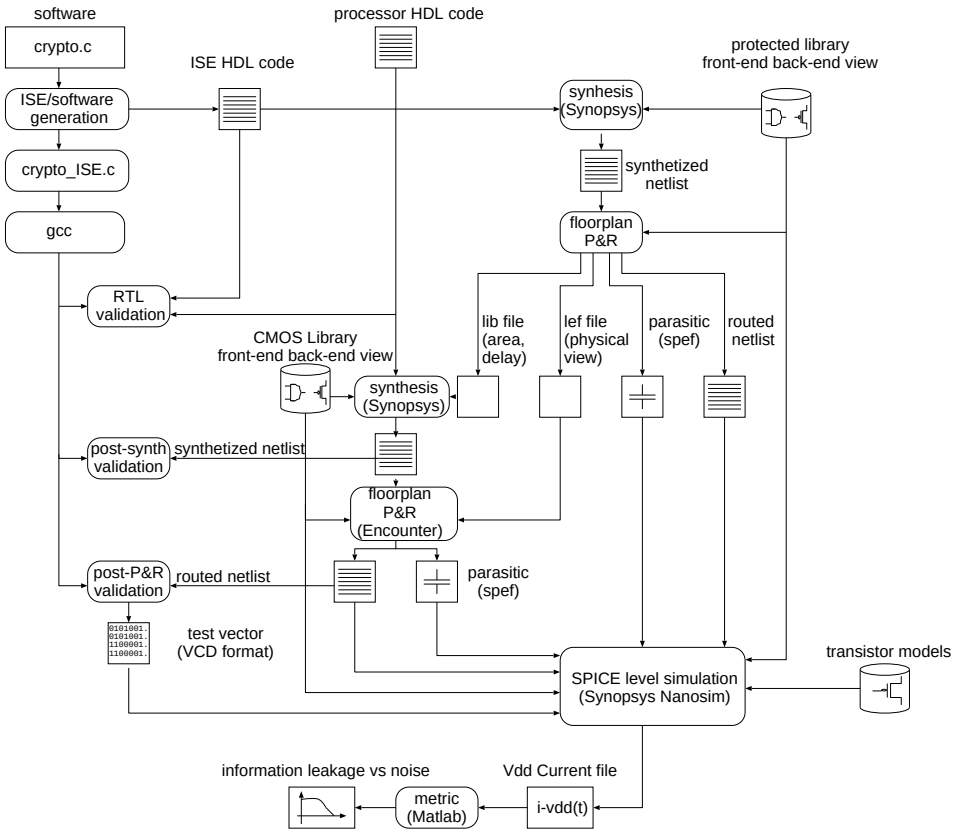


Fig. 1. Full view of the design flow.

In the newly generated software, calls to the ISE are automatically generated using the correct syntax, consistent with the RTL code, and thus supported by the compiler. The corresponding binary file is generated and simulated using an interpreter that mimics full system behavior, including the boot loader, and generates the corresponding values of the processor pins for each cycle of software execution. The pin values are then used in every validation step, including the generation of test vectors for use in SPICE level simulation. The SPICE level simulation of the full core (core + protected ISE) generates power traces which are used to measure the leakage of the processor.

To generate the customized processor, we begin with an HDL description of the processor core and a software implementation of a cryptographic algorithm.

The first step is to select from an automatically generated data flow graph one or more ISEs to realize in protected logic; at present, this selection is the only step still performed manually, although we intend to automate it in the future. Once the HDL description of the ISE(s) has been generated, the circuit is synthesized using protected logic based on a standard cell library using Synopsys Design Compiler. The circuit is then placed and routed with Cadence Design Systems SoC Encounter. A parasitics file (in spef format) is produced, along with the Verilog netlist of the circuit and an sdf file for back annotation of the delays. The library file describing the ISE (i.e., timing information, power and dimension) and the layout exchange format file (the abstract view for placement and routing) are generated to integrate the ISE as a black box during synthesis and placement and routing of the complete design. Next, the complete design (i.e., the processor augmented with ISEs as a black-box) is synthesized and placed and routed using a standard CMOS flow. For the front end, the ISE library file is loaded by Design Compiler; the unit is recognized by the synthesis engine and the ISE's timing information is used during the synthesis process. During the physical design phase, the ISE is treated as a macro just like typical IP blocks and is pre-placed into the core. The flow produces the spef and sdf files and the Verilog netlists of the whole design.

Post-place-and-route simulation is now performed using ModelSim, with the previously generated sdf files (CMOS and protected logic) under the considered cryptographic benchmark. This simulation verifies the functionality of the processor and generates the test vectors for transistor-level simulation that will be used to generate power traces that will be input to the security evaluation. Synposys Nanosim performs transistor-level simulation, using the spef files for the protected ISE and CMOS core, with the relative Verilog netlists, SPICE models of the technology cells and the transistor models. This simulation generates vector-based time-varying power profiles which are stored in a simple text format. This data typically corresponds to the simulated traces represented by the variable T in Section 3 which concludes the treatment of the flow.

5 Case Study and Results

In this section we present the results of the evaluation of our design flow evaluated with different metrics of performances and security.

5.1 PRESENT Algorithm and the Considered Versions Overview

PRESENT [5] is a block cipher based on an SP-network that consists of 31 rounds. The block length is 64 bits and two key lengths of 80 and 128 bits are supported. During the encryption process, three different transformations are iterated 31 times. The three basic transformations are: *addRoundKey*, *sBoxLayer*, and *pLayer*: the first is function of the state and the secret key, while the final two are only functions of the state. At the completion of the last round an extra addRoundKey transformation is performed. The added key is different in

each round and these round keys are generated by a key schedule routine that takes the secret key and executes an expansion as specified in algorithm description. The evaluation performed in this work, is done on a reduced version of the PRESENT algorithm, composed of just addRoundKey and sBoxLayer of 4 by 4 bit. We selected PRESENT as case of study for this work because the size of its S-box permits an exhaustive search of the design space without compromising the generality of the proposed methodology and results.

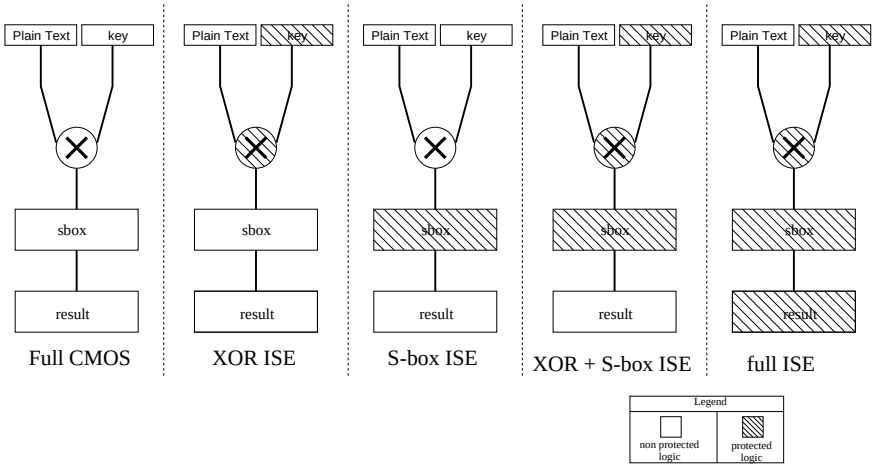


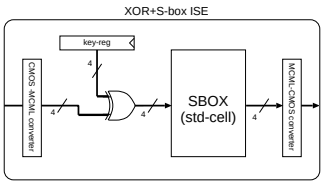
Fig. 2. Considered implementations for the algorithm.

For our experiments, we considered the five possible implementations of the algorithm, that are depicted in Figure 2. Each implementation has a different section realized in protected logic. The first version, *Full CMOS*, is the reference version, in which the core is completely implemented in CMOS and the software does not leverage on any kind of ISE. In the second, *XOR ISE*, the full program is executed on the CMOS core, except for the secret key, that is stored into a protected register and the keyAddition, that is implemented using protected ISE. The third version, *S-box ISE*, implements only the sBoxLayer in a protected ISE, while the rest of the algorithm executes on the CMOS core. The fourth, *XOR + S-box ISE*, stores the secret key in a protected register and executes both addRoundKey and sBoxLayer using a protected ISE, but writes the result back to the processor register file, which is unprotected. Lastly, *full ISE* implements addRoundKey and sBoxLayer in protected logic, and stores the secret key and the result in a protected register that is part of the ISE as well.

5.2 Experimental setup

The processor used in this work is an OpenRISC 1000 [14], a five stage pipelined in-order embedded processor. The processor provides a 32-bit datapath and a 32-entry single write-port, dual read port register file. The processor includes extra opcode space to support ISEs and is provided with a gcc cross-compiler.

We have selected MOS Current-Mode Logic (MCML) as secure logic style to implement the protected ISEs. MCML cells are low-swing, fully-differential circuits built with networks of differential pairs biased with a constant current source [1]. The constant DC current and the differential nature of the cells provide an almost constant current consumption profile, which is independent of the switching activity. In theory, this results in a dramatic decrease of side-channel leakage and thus increased resistance against DPA attacks [22, 29]. The increased DPA-resistance in differential logic circuits is obtained by the simultaneous and opposite switching of differential signal pairs resulting in almost perfect cancellation of current transients. In order to obtain consistently robust circuits, it is therefore critical to match the time constants in the two wires of each pair. This implies that each pair of wires must be physically routed along the exact same path, in order to equalize the length and parasitics of individual routes. To achieve this, the design flow proposed in [3] is used in this work. This entirely automated methodology enables the implementation of standard cell based-differential circuits from RTL with true differential routing, using a classical timing-driven design flow without human intervention. With this approach, the sensitive parts of the processor are implemented with secure logic and converter circuits are inserted at the boundary to interface between the two different logic styles. This increases security in a transparent way, without additional effort from the designers.




```
// Calculate S-box (plaintext XOR key)
int PRESENT_XOR+S-box-ISE(int plaintex) {
1 int result = 0; // initialize the result
  // call the new instruction to calculate s-box(pt ^key)
2 Instr_1(plaintex, result);
3 return result; }; // return the result
```

Fig. 3. Example of an ISE and its source in C: the XOR + S-box .

One example of an automatically generated ISE, reported in Figure 3, depicts the hardware view corresponding to the XOR + S-box ISE. The figure includes the converters between CMOS and MCML, which are necessary to interface the protected logic with the processor. These conversion circuits are automatically added at the inputs and outputs of the ISE.

We performed our experiments using the following versions of the design tools: Mentor Graphics Modelsim 6.2d for logic simulation, Synopsys Design Compiler 2007.12 for synthesis, and Cadence Design System SoC Encounter 7.1, for placementing and routing. Our CMOS target library was a 0.18μm commercial standard cell library. SPICE level simulation was carried on using Synopsys Nanosim 2007.03, and the transistor model is BSIM 3.3.

5.3 Results

Each of the five implementations presented in this paper has been synthesized to run at a clock frequency of 100 MHz, under worst-case process conditions. The same clock frequency is used for each of the ISEs.

Table 1 reports the area and average power consumption of the base OpenRISC1000 processor, as well as all the four versions augmented with protected ISEs. The average power consumption is calculated for each core during the execution of the PRESENT algorithm, including calls to the ISEs. We report the silicon area occupation and the gate count. The absolute silicon area provides a clear measure for the physical cost of each implementation, while the number of equivalent gates highlights the complexity of the circuit. We calculated the number of equivalent gates for each implementation, with appropriate weights to account for the disparity in sizes between reference gates for CMOS and MCML. In our experiments, the difference in area penalty between the largest and smallest ISE is 0.2%. This is primarily due to the small size of the PRESENT algorithm, which tends to be overshadowed by the size of the conversion circuitry at the CMOS-MCML boundary.

Table 1. Area occupation and average power consumption of each implementation.

Version	**Power Consumption (mW)**	**Full Die Size (mm^2)**	**ISE Size (mm^2)**	**Gate Count (GE)**
full CMOS	87.77	1.8603	-	139071
XOR ISE	129.24	1.9810	0.1207	140787
S-box ISE	129.42	1.9838	0.1235	140843
XOR + S-box ISE	129.81	1.9844	0.1241	140853
full ISE	129.83	1.9849	0.1246	140865

The full ISE implementation, which is the most resistant to DPA, increases the power consumption by 47.9% with respect to the full CMOS design, while the area overhead is to 6.7%. A similar level of leakage using the same protected logic would be possible by implementing a full processor in MCML. Our results show that this would increase the power consumption by a factor of approximately 40 time higher compared to a CMOS implementation, while increasing the total area 2.65 times. The large power difference is due to the static current consumption of MCML gates, whose power consumption becomes close to CMOS gates for high switching activity and operating frequancy. The MCML library, which has been developed internally, has not been tuned for battery operated devices; an MCML library tagetting these devices would significantly improve the results.

Preliminary results of our security evaluations using the information theoretic metric of Section 3 are plotted in Figure 4 for different implementations. They show that increasing the number of dimensions to evaluate the mutual information $\mathrm{I}(K, L)$ improves the quality of the evaluations up to a certain extent, where the noise variance is sufficiently large for hiding the small information leakage in the higher dimensions of the transformed traces.

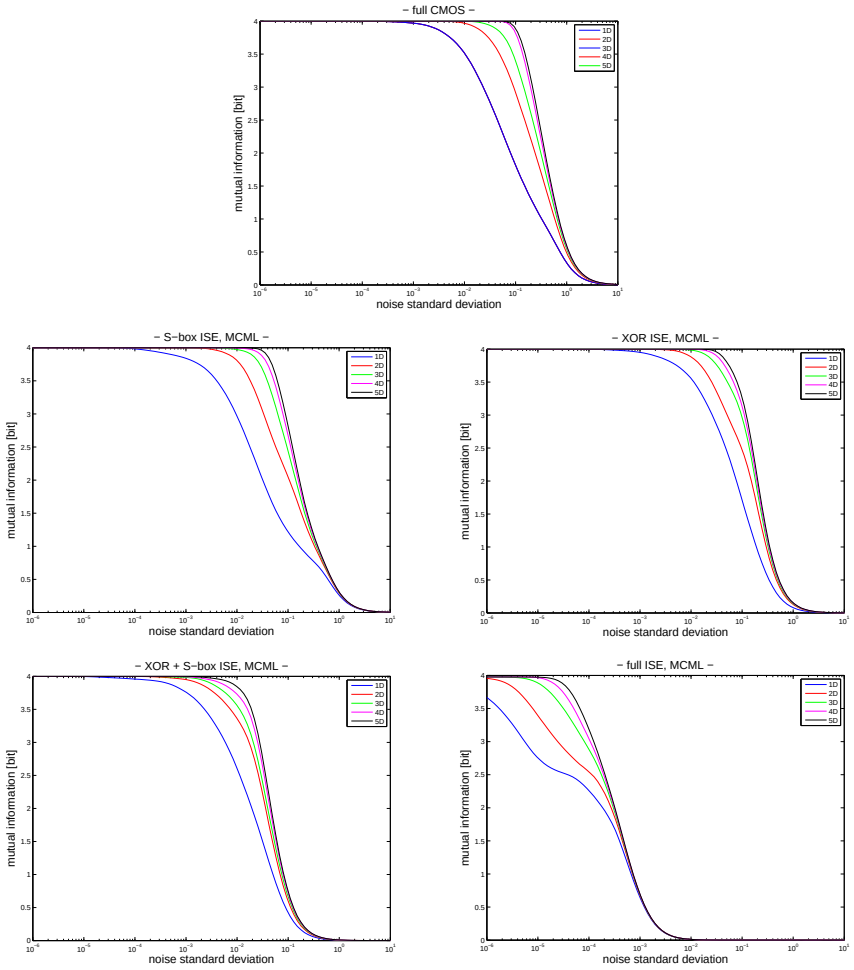


Fig. 4. Mutual information leaked by different implementations in function of a noise standard deviation, for different dimensions kept after application of the PCA.

Figures 5 and 6 compare the results for the five different processor and ISE combinations. Figure 5 plots the mutual information leaked by the different implementations. Figure 6 takes the opposite point of view, and illustrates the amount of noise that is required to reduce the leakage up to a threshold. The goal of a robust countermeasure is to reduce the information leakage.

These two figures concisely illustrate and confirm our intuition, namely, that protecting a part of the algorithm reduces the leakage; however, the overall security of a system depends on its weakest point. Consequently, there is a significant improvement when considering a fully protected ISE. The analysis shows that the fully protected ISE has no obvious logical weaknesses; however, it remains to be determined the extent to which a fabricated piece of silicon can

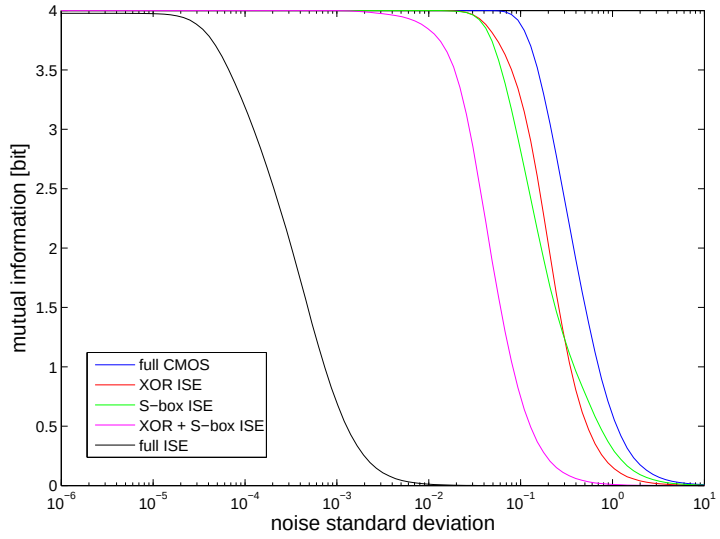


Fig. 5. Mutual information leaked by different implementations in function of a noise standard deviation, with 5 dimensions kept after application of the PCA.

be built to confirm the results of the simulations. Anyway, these results confirm the applicability of our proposed design flow up to the analysis of the side channel leakage. The computation of the evaluation metrics, including the selection of the points of interest in the leakage traces with a PCA, is fully automated.

6 Conclusions

With the increased use of embedded systems in security applications, protection against side channel attacks has become increasingly important. This paper summarizes the first attempt to integrate a meaningful information leakage metric into an industrial design flow for secure systems. Our results establish the feasibility of the proposed flow, and show that the use of ISEs in protected logic styles is a reasonable and low-cost method to provide the desired security. Future work will focus on manufacturing the most promising implementations. It is in fact well known that the decisive proof of robustness is obtained only when the actual fabricated microchip is attacked using high frequency probes and an oscilloscope. Nonetheless, our design flow is fundamental to perform a deeper design space exploration before the fabrication.

Acknowledgements

The authors would like to thank Federico Ferrari for fruitful discussions as well as the anonymous referees for their constructive comments. This work was partially

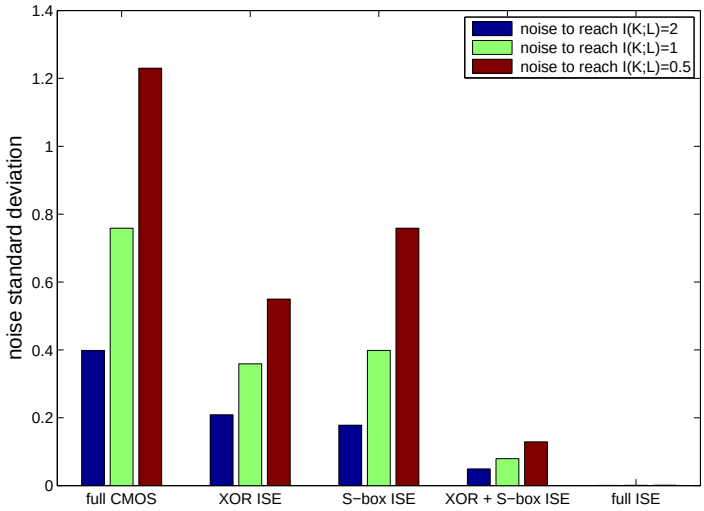


Fig. 6. Noise standard deviation required to reach a certain level of information leakage, with 5 dimensions kept after application of the PCA.

supported by HIPEAC Network of Excellence (collaboration grant) and by the Walloon Region project E-USER (WIST program). François-Xavier Standaert is an associate researched of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

References

1. Massimo Alioto and Gaetano Palumbo. *Model and Design of Bipolar and MOS Current-Mode Logic: CML, ECL and SCL Digital Circuits*. Springer, Dordrecht, The Netherlands, 2005.
2. Mohamed W. Allam and Mohamed I. Elmasry. Dynamic current mode logic (DyCML): A new low-power high-performance logic style. *IEEE Journal of Solid-State Circuits*, 36(3):550–58, March 2001.
3. Stéphane Badel, Erdem Guleyupoglu, Ozgur Inac, Anna Pena Martinez, Paolo Vietti, Frank K. Gürkaynak, and Yusuf Leblebici. A generic standard cell design methodology for differential circuit styles. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pages 843–48, Munich, March 2008.
4. Sandro Bartolini, Irina Branovic, Roberto Giorgi, and Enrico Martinelli. A performance evaluation of ARM ISA extension for elliptic curve cryptography over binary finite fields. In *Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing*, pages 238–45, Foz do Iguacu, Brazil, October 2004.
5. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe.

PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems—CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–66. Springer, Berlin, September 2007.

6. Marco Bucci, Michele Guglielmo, Raimondo Luzzi, and Alessandro Trifiletti. A power consumption randomization countermeasure for DPA-resistant cryptographic processors. In *Proceedings of the 14th IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation*, volume 3254 of *Lecture Notes in Computer Science*, pages 481–90. Springer, Berlin, September 2004.
7. Jean-Sébastien Coron and Louis Goubin. On boolean and arithmetic masking against differential power analysis. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems—CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 231–37. Springer, Berlin, August 2000.
8. Wieland Fischer and Berndt M. Gammel. Masking at gate level in the presence of glitches. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems—CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 187–200. Springer, Berlin, August 2005.
9. Ilham Hassoune, François Macé, Denis Flandre, and Jean-Didier Legat. Low-swing current mode logic (LSCML): A new logic style for secure and robust smart cards against power analysis attacks. *Microelectronics Journal*, 37(9):997–1006, September 2006.
10. Intel's advanced encryption standard (AES) instructions set (white paper), April 2009.
11. James Irwin, Dan Page, and Nigel P. Smart. Instruction Stream Mutation for Non-Deterministic Processors. In *Proceedings of the 13th International Conference on Application-specific Systems, Architectures and Processors*, pages 286–95, San Jose, Calif., July 2002.
12. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, Berlin, August 1999.
13. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal I. Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–13. Springer, Berlin, September 1996.
14. Damjan Lampret. *OpenRISC 1000 Architecture Manual*, April 2006.
15. François Macé, François-Xavier Standaert, Ilham Hassoune, Jean-Didier Legat, and Jean-Jacques Quisquater. A dynamic current mode logic to counteract power analysis attacks. In *Proceedings of the XIX Conference on Design of Circuits and Integrated Systems*, Bordeaux, France, November 2004.
16. François Macé, François-Xavier Standaert, and Jean-Jacques Quisquater. Information theoretic evaluation of side-channel resistant logic styles. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems—CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 427–42. Springer, Berlin, September 2007.
17. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Advances in Information Security. Springer, New York, 2007.
18. David May, Henk L. Muller, and Nigel P. Smart. Non-deterministic processors. In Vijay Varadharajan and Yi Mu, editors, *Information Security and Privacy—ACISP 2001*, volume 2119 of *Lecture Notes in Computer Science*, pages 115–29. Springer, Berlin, July 2001.

19. David May, Henk L. Muller, and Nigel P. Smart. Random register renaming to foil DPA. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems—CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 28–38. Springer, Berlin, May 2001.
20. Simon Moore, Ross Anderson, Paul Cunningham, Robert Mullins, and George Taylor. Improving Smart Card security using self-timed circuits. In *Proceedings of the 8th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 211–18, Manchester, April 2002.
21. Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems—CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–86. Springer, Berlin, August 2005.
22. Francesco Regazzoni, Thomas Eisenbarth, Axel Poschmann, Johann Großschädl, Frank Gurkaynak, Marco Macchetti, Zeynep Toprak, Laura Pozzi, Christof Paar, Yusuf Leblebici, and Paolo Ienne. Evaluating resistance of MCML technology to power analysis attacks using a simulation-based methodology. In Marina L. Gavrilova, Chih Jeng Kenneth Tan, and Edward D. Moreno, editors, *Transactions on Computational Science IV*, volume 5430 of *Lecture Notes in Computer Science*, pages 230–43. Springer, Heidelberg, 2009.
23. Alexander G. Rostovtsev and Olga V. Shemyakina. AES side channel attack protection using random isomorphisms. Cryptology e-print archive, March 2005. `http://eprint.iacr.org/`.
24. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology—EUROCRYPT '09*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, Berlin, April 2009.
25. Stefan Tillich and Johann Großschädl. Instruction set extensions for efficient AES implementation on 32-bit processors. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems—CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 270–84. Springer, Berlin, September 2006.
26. Stefan Tillich and Johann Großschädl. Power analysis resistant AES implementation with instruction set extensions. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems—CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 303–19. Springer, Berlin, September 2007.
27. Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on Smart Cards. In *Proceedings of the 28th European Solid-State Circuits Conference*, pages 403–6, Florence, September 2002.
28. Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pages 246–51, Paris, February 2004.
29. Zeynep Toprak and Yusuf Leblebici. Low-power current mode logic for improved DPA-resistance in embedded systems. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 1059–62, Kobe, Japan, May 2005.