FPGA Implementations of the AES Masked Against Power Analysis Attacks

Francesco Regazzoni[1,3], Yi Wang[2], François-Xavier Standaert[1]

[1]UCL Crypto Group, Université catholique de Louvain, Louvain-la-Neuve, Belgium.
{francesco.regazzoni,fstandae}@uclouvain.be
[2]Embedded System and Networking Laboratory, HuNan University, Changsha, China.
yiwang@hnu.edu.cn
[3]ALaRI, University of Lugano, Lugano, Switzerland. regazzoni@alari.ch

Abstract. Power analysis attacks are a serious treat for implementations of modern cryptographic algorithms. Masking is a particularly appealing countermeasure against such attacks since it increases the security to a well quantifiable level and can be implemented without modifying the underlying technology. Its main drawback is the performance overhead it implies. For example, due to prohibitive memory costs, the straightforward application of masking to the AES algorithm, with precomputed tables, is hardly practical. In this paper, we exploit both the increased size of state-of-the-art reconfigurable hardware devices and previous optimization techniques to minimize the memory occupation of software S-boxes, in order to provide an efficient FPGA implementation of the AES algorithm, masked against side-channel attacks. We describe two high throughput architectures, based on 32-bit and 128-bit datapaths that are suitable for Xilinx Virtex-5 devices. In this way, we demonstrate the possibility to efficiently combine technological advances with algorithmic optimizations in this context.

1 Introduction

Physical Attacks are a relatively recent and very powerful form of attack against cryptographic devices. They do not target the mathematical structure of the algorithm, which is typically sound and robust, but attempt to gain information about the secret key by attacking the physical implementation of the algorithm itself. Side Channel Attacks are Physical Attacks based on the exploitation of the "side channel information" (typically time, power consumption, or electromagnetic emission), which can be measured while the cryptographic algorithm is being computed on the device. Among them, the attack based on power consumption, called Power Analysis Attack [11], has received significant attention, since it is very powerful and does not usually require detailed knowledge of the target device to be successfully implemented.

In order to counteract power analysis attacks, one solution is to remove the correlation between the power consumed and the secret key, e.g. by altering the power characteristic of the device. This task is particularly complex, especially for devices already manufactured. Among the countermeasures proposed in the past, the Boolean masking [2, 7, 15] is particularly appealing: it is rather simple to implement, does not require any novel and specific hardware, and leads to well quantifiable security improvements.

In a nutshell, masking attempts to randomize the power consumption by adding a random number, called mask, to all the intermediate values which can be exploited by the attacker. The mask is finally removed at the end of the computation to guarantee the correctness of the results. However, when non linear operations are involved, tracking of the masks is an expensive operation, both in terms of area/memory consumption and speed. Previous works [1, 29] attempted to mitigate such high requirements by proposing masked implementations of the cryptographic algorithms optimized for hardware resources. However, if the considered algorithm uses large S-boxes (as in the case of AES), realizing an efficient implementation of Boolean masking remains a challenging task.

In this paper, we present two masked implementations of the AES algorithm suitable for reconfigurable devices. Rather than to propose a completely new architecture, we attempt to maximize the exploitation of the FPGAs technology improvements by combining them with algorithmic optimizations previously introduced for software. Our target device, the Virtex-5 FPGA from Xilinx, features a larger number of slices as well as a novel slice structure (6 input Look-Up-Tables combined with multiplexers) which allows efficiently mapping 8-bit input tables. Hence, the pre-computed tables used in the masked S-box proposed by Oswald and Schramm [21] for 8-bit microcontrollers are good candidates to be implemented in such devices. We thus adapted that implementation to the characteristics of modern reconfigurable devices, hence achieving a full exploitation of their potential. Then, we show how such an S-box can be integrated into a complete AES coprocessor which operates on 128-bit plaintext and 128-bit mask, without imposing specific conditions to the user. Finally, we analyze in detail two masked AES designs, characterized by a datapath of 32 and 128 bits respectively, and we show that our architectures fulfill the area and performance requirements of most applications. We thus prove that a careful combination of algorithmic optimizations and technological advances allows us to efficiently implement AES masked against side channel attacks also on reconfigurable devices.

The remainder of the paper is as follows: Section 2 summarizes the structure of the AES algorithm, revises the basic concepts of masking, and discusses the related works. In Section 3 we discuss the main challenges which characterize the design of a masked non linear transformation, we discuss how we tailored the selected S-box implementation to the needs of the target FPGA, we present our two designs and we report the experimental results.

2 Background

In this section we recall the AES algorithm, and we revise the basic concepts of masking and related works.

2.1 The AES Rijndael

Rijndael algorithm was selected as Advanced Encryption Standard in 2001 [18]. The standard supports block sizes of 128 bits and key sizes of 128, 192 and 256 bits. The encryption process starts with the first key addition, followed by a number of round functions which depends on the key size. In the encryption, the round function is composed of four transformations: *ShiftRows*, which cyclically shifts to the left the bytes in the last three rows of the state with different offsets; *SubBytes*, which is the non-linear byte substitution and operates independently on each byte of the state; the *Mix-Columns* which multiplies modulo $x^4 + 1$ the columns of the state by the polynomial $\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$; and, finally, the *AddRoundKey*, which adds a round key to the state. All the needed round keys are generated by a *key schedule* routine, which takes the secret key as input and expands it as specified in the standard.

2.2 The masking countermeasure

Masking is a countermeasure against power analysis attacks based on secret sharing [2] firstly proposed by Chari et al, and used by Messerges [15] to secure the five AES finalists. It decreases the correlation between the power consumed by a device and the data being processed by applying a random mask to the intermediate values. More formally, prior to the execution of the algorithm, the secret key value (or the input data value, or both of them) x is obscured using a random value m, called mask, to generate a masked value x', as follows: $x' = x * m$, where $*$ indicates a specific mask operation. The algorithm is then executed using x', the intermediate results are thus masked also. Masking leads to a well defined level of security: by adding one mask it is possible to prevent the so-called first order DPA, in which only one leakage sample is used by the attacker [9, 16, 19, 31, 27].

Different approaches to implement masking have been introduced so far, including multiplicative, Boolean and affine masking [2, 4, 7, 15, 30]. Each of them is characterized by good and bad points. The multiplicative masking for instance, is efficient for masking the non-linear functions over $GF(2)$. However its major problem lies in the fact that not all the intermediate values can be masked (for instance, the multiplication does not allow masking the intermediate value 0). This situation is particularly dangerous since an adversary may exploit the fact that specific intermediate masked values are not statistically independent from their unmasked counterpart [6].

The Boolean masking is the most common scheme used for masking. We thus consider it for our FPGA implementations. Contrary to the multiplicative masking, the Boolean one is efficient when applied to linear functions of the cryptographic algorithms. However, as drawback, the implementation of Boolean masking incurs significant overhead when applied to non linear transformations. The overhead can affect either the memory required, as in the implementation proposed by Piret and Standaert [22], or the computational time, as in the implementation of Prouff and Rivain [24]. In particular, for first order masking and for an S-box of size n bits, the first implementation requires a look-up table of size 2^{2n}, with no computational overhead, while the second implementation has a computation overhead of $2^n * 2$ XOR operations and $2^n * 2 + 1$ memory transfers, but the size of the look-up table is limited to 2^n.

Such an area or computational overhead makes the task of implementing Boolean masking on hardware devices particularly expensive. Masked implementations of the AES algorithm specifically designed for ASIC were discussed in the past by Canright and Batina [1] who proposed a compact design of masked AES S-box, and by Trichina et al. [29] who presented an implementation of a masked AES coprocessor tailored on the needs of GSM and ad-hoc networks applications. Masking was also explored for algorithms with smaller S-boxes: the work of Standaert et al. [28], for instance, explores the feasibility of using Boolean masking to protect hardware implementation of DES and Triple-DES.

In this paper we aim instead at demonstrating that, by exploiting the new potentialities of state-of-the-art reconfigurable hardware, and combining them with algorithm optimization, it is possible to successfully map masked implementations of the AES algorithm on FPGAs and achieve a high throughput. To achieve our goal, we looked at the many optimizations previously proposed, focusing in particular on the ones for 8-bit microcontrollers [13, 21, 25]. The most relevant work related to our concern is the one of Oswald and Schramm [21], which presents an efficient implementation of masked S-box. The authors concentrate on the inversion over $GF(2^8)$, since the affine mapping can be easily masked, and they show how the computation of the masked inversion can be mapped to 6 look up tables characterized by reduced memory requirements, assuming that the input and output mask of the S-box look-up table are identical. Since the size of these tables nicely fit the slice structure of Virtex-5 FPGA, we use this proposal as a starting point to design the masked S-box of our AES implementation.

3 FPGA implementation

In this section we discuss the design choices we made while implementing the AES S-box. We present the two AES cores we designed and report the area and timing performances.

3.1 The masked S-box design

When masking is applied, in order to be effective, it is of crucial importance that all the transformations of the cipher are performed on masked data. This statement is valid in particular for the non linear transformation, since it is a very suitable attack point. For an adversary, it is convenient to make his hypothesis on the 8 output bits of the S-box. Furthermore, the non-linear structure of the S-boxes highlights the differences between the correct and the wrong guesses and increases the possibility of a successful attack [23]. However, when the used masking scheme is Boolean (as in our case), the implementation of the non linear transformation is particularly challenging; therefore, we firstly concentrate on the design of the S-box.

The S-box of the AES algorithm operates independently on each byte. It is composed of two transformations: the calculation of the multiplicative inverse in the finite field $GF(2^8)$ and the application of the affine transformation which is specified by the standard. A possible way to implement the whole non linear transformation is by means of a look-up table which we call S. When this approach is followed, the S-box of the value x is stored in a corresponding index in the table S. When masking is applied, it is necessary to mask also the table S. This operation can be carried out in two ways. Depending on the approach used, it causes either performance or memory overhead.

The first approach requires the recomputation of the table which stores all the results of the non linear transformation. This operation must be performed every time the mask is changed. This causes a too high penalty for the performances of hardware designs. However, it is not feasible to implement all the possible look-up tables in hardware, especially reconfigurable. In fact, to cover all the possible input masks, we would need a table which stores $2^8 \times 2^8 \times 8$ bits. Nowadays, to use such a large portion of memory only to implement cryptography is not possible: in fact modern FPGAs typically store a complex System on Chip of which the AES coprocessor only represents a limited portion. Thus, the second approach is not practical either.

However, compared to early reconfigurable devices, state of the art FPGAs are larger and more complex devices which, togheter with the programmable logic blocks, embed multipliers, RAM memories, and sometimes also complete processors. Also the slice structure has been improved. Our target device, the Xilinx Virtex-5, exhibits two types of slices: sliceL, which is the basic slice, and sliceM, which can be also configured as shift register or distributed RAM. Each slice contains four flip-flops, four 6-inputs Look-Up-Tables (LUTs) and multiplexers. Thanks to this new structure, a 265×1 bit table can be efficiently packed into a single slice: the correct output is selected from four 6-to-1 LUTs by the two multiplexers F7MUX and F8MUX.

Since the 8-bit input tables nicely fit this slice structure, a straigthforward way to minimize the area required by masked AES S-box on FPGA is to realize an architecture which largely use them. An implementation of AES suitable to our needs is the one proposed for software by Oswald and Schramm [21]. The authors concentrate on the inversion in $GF(2^8)$, since the affine mapping is easy to mask. Adapting the representation presented by Wolkerstorfer et al [33], it is possible to transform a masked input to the composite field $GF(2^4) \times GF(2^4)$, where it can be efficiently inverted, and finally transformed back to the $GF(2^8)$. Rather than solving the needed equations [20], Oswald and Schramm perform the inversion in $GF(2^4)$ combining XOR operations with four pre-computed tables: T_{d_1}, T_{d_2}, T_m and T'_{inv}. To transform the result of the inversion back to $GF(2^8)$, two additional tables are required: T'_{map}, which performs the masked isomorphic mapping from $GF(2^8)$ to $GF(2^4) \times GF(2^4)$ and $T'_{map^{-1}}$, which performs masked isomorphic mapping back from $GF(2^4) \times GF(2^4)$ to $GF(2^8)$. The affine transformation needed to complete the calculation of the AES S-box is integrated with the isomorphic mapping, in order to use only one table for the two transformations.

The first four tables take as input two elements of $GF(2^4)$ and return as a result an element of $GF(2^4)$, T'_{map} takes as input an element of $GF(2^8)$ and returns as a result an element of $GF(2^4)$, and $T'_{map^{-1}}$ takes as input two elements of $GF(2^4)$ and returns as a result an element of $GF(2^4)$. It can be noticed that all these tables have input size of 8 bits, thus fit perfectly the slice structure of our target FPGA.

The overall design of the masked AES S-box is depicted in Figure 1. As can be seen, it requires two inputs of 8 bits each, the masked data and the current mask, and produces two outputs, also of 8 bits each: the masked result of the non linear transformation and the updated mask. The implementation proposed by Oswald and Schramm [21] requires 14 table look-up operations and 15 XORs to compute the masked non linear transformation. In order to perform all the required steps within one clock cycle, we replicated several tables. The final result is a design composed of 15 tables, eleven of which store 2^4 entries, while the other four are used for the isomorphic mapping, its inverse and the affine transformation.

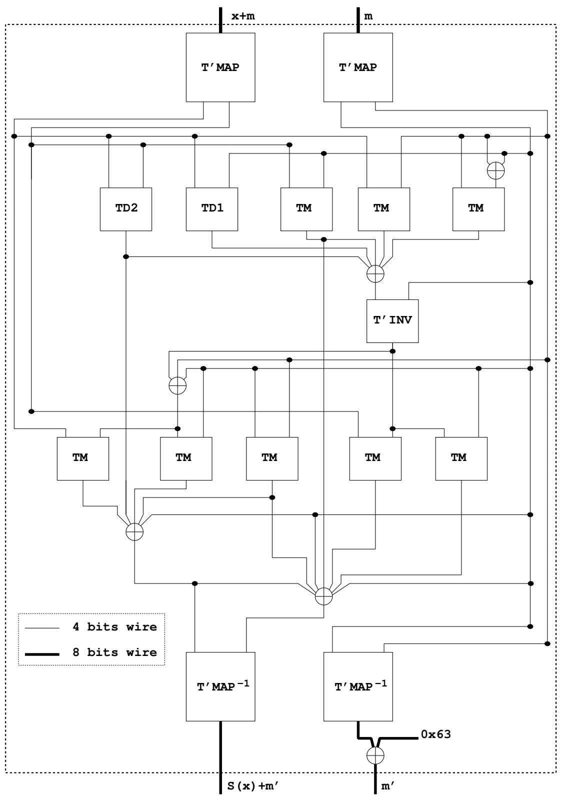

Fig. 1. The Masked S-box

Table 1 reports the area occupation of the proposed masked AES S-box compared to a reference unprotected S-box. The results are obtained using the Xilinx ISE version 12.2 tool for synthesis and place and route. The selected target device was the XC5vlx50 of the Virtex-5 family, the optimization speed was set to 3 and the synthesis tool was forced to use distributed RAM instead of block RAM. It is possible to notice that, even if the number of used LUTs and the number of used slices increases when moving from the unprotected implementation to the masked one, the resources required still meet the constraints of a wide range of applications.

Table 1. Implementation results of S-box on Virtex-5

XC5vlx50	Slices	LUTs	Registers
Reference S-Box	8	32	0
Masked S-box	61	208	7

3.2 The whole masked AES design

In this section we describe the design of two masked AES coprocessors which operate on 128-bit plaintext and 128-bit mask, whitout imposing to the user specific conditions regarding the mask. In particular, we discuss how we extended two reference implementaions characterized by different datapaths: one of 32 bits and one of 128 bits. In both cases, except for the non linear part, the mask update required by all the round transformations can be simply computed by applying to the mask the same transformation as the one applied to the state. This operation can be carried out in parallel with the cipher, and in particular is performed by duplicating the hardware.

The whole 128-bit coprocessor is depicted in Figure 2. As can be seen, the coprocessor requires three inputs of 128 bits: the secret key, the plaintext and a random mask. The random mask in particular is assumed to be generated within the FPGA in a secure way, possibly using a true random number generation such as the ones proposed by Güneysu and Paar [8], by Shackleford et al. [26], or by Kohlbrenner and Gaj [12], since the security of the whole design depends on the security of the mask. The 128-bit datapath is designed to compute a complete round in a single clock cycle. In order to do this, it uses 16 masked S-boxes, implemented as discussed in Section 3.1. Also, from the figure, it is possible to notice the two separated paths, one dashed for the mask, which is initially xored with the plaintext and one plain for the masked state. The two paths merge only during the S-box computation and in the final removal of the mask performed prior to output the cipher-text.

The 32-bit datapath design is similar to the 128-bit one, but operates only on portion of 32 bits of the state and completes a quarter of round operation at each clock cycle. For this reason, it requires only 4 masked S-boxes.

Table 2. Implementation results of masked AES on Virtex-5

	Reference 32 bit	Masked 32 bit	Reference 128 bit	Masked 128 bit
Number of Slices	290	637	478	1,462
Number of LUTs	595	1,429	1,557	4,772
Number of Registers	467	643	648	904
Clock Cycles core (+ inteface)	44 (+8)	44 (+8)	11 (+8)	11 (+8)
Clock (ns)	5	10	4	10
Frequency (MHz)	200	100	245	100
Throughput (Mbit/s) core	581	290	2909	1163
Throughput (Mbit/s) core + interface	492	246	1684	673

Table 2 reports the results obtained when mapping the 128-bit and the 32-bit datapaths on the XC5vlx50 of the Virtex-5 family, using the same optimization options and

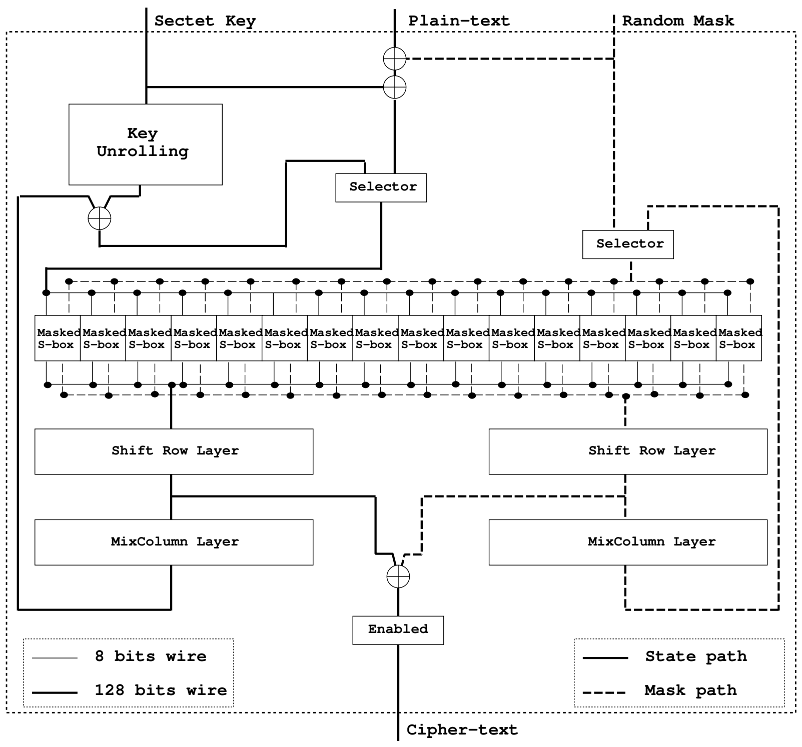

Fig. 2. The Masked 128-bit AES

the same tool used for mapping the masked S-box. Several synthesis were carried out imposing to the tool different clock frequencies, the ones reported in the table are the fastest which still meet the timing constraints. Both designs include an interface with a 32-bit data bus, since it is the most commonly used bus in embedded systems. The interface has only a negligible impact on the slice count, since it only slightly increases the control unit. However it affects the speed: the bus interface requires 8 clock cycles to completely load and offload the data from the coprocessor, therefore in Table 2 we separated the clock count and the throughput of the cores from the ones where the interface is included.

Both protected datapaths are compared with their unprotected counterparts, which have exactly the same architecture as the protected ones but without masking. The two masked implementations have the same clock frequency since in both cases the critical path is determined by the masked S-box. As can be noticed, the masked implementation requires a larger area (approximately up to 3 times the resources required by the unmasked counterpart) and has a reduced throughput (approximately half of the one achieved by the unmasked counterpart), but this does not represent a limitation. In fact, the throughput achieved by both masked implementations remains sufficient to fulfill the needs of most applications. Furthermore, the device occupation is still limited, hence the majority of the FPGA resources are still available to the designers.

Carrying out a fair and meaningful comparison of our designs with the ones reported in literature is difficult, since the results are affected by many factors [3], including not only the design itself, but also tool versions, device architecture and vendor, implementation options, and strategy used to achieve DPA resistance. Also the number of masked AES implementations for FPGAs available in open literature is limited. An implementation of AES which combines Boolean with multiplicative masking was proposed by Mentens et al. [14]. The area overhead of their secured core compared to the reference unsecured version is approximately 20%, while the speed is degradated by 30%. Kamoun et al. [10] implemented a masked AES S-box on Virtex-4 FPGA which incurring an area overhead of 44% and a frequency decrease of 31%. FPGA implementations of AES resistant to power analysis attacks were presented in the past by Nassar et al. [17] using a different countermeasure, a precharged logic, and a target device coming from a different vendor. Their result show that the protected version of the core is approximately 3 times bigger then its unprotected counterpart, while the speed was decreased of approximately one third. We can thus conclude that the penalty of our protected designs is in line with the one of previous works.

4 Conclusions

In this paper, we explored the use of Boolean masking to protect FPGA implementations of the AES algorithm. In particular we took advantage of slice structure of Xilinx Virtex-5 FPGA and we reduced the size of the masked S-box by exploiting algorithmic optimizations previously proposed for the software domain. Finally, we integrated our masked S-box into two masked AES coprocessors, characterized by 128-bit and 32-bit datapaths, respectively. Our results obtained when mapping the two designs on the Virtex-5 platform showed that our masked implementations allow sufficient performances for most applications while the overall device occupation is kept acceptable. This work showed that the AES offers excellent opportunities to efficiently combine technological advances with algorithmic optimizations in reconfigurable hardware.

5 Acknowledgments

This work was partially supported by the Walloon Region projects E-USER and S@T Skywin. François-Xavier Standaert is an associate researched of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

References

1. D. Canright and Lejla Batina. A very compact "perfectly masked" s-box for aes. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 5037 of *Lecture Notes in Computer Science*, pages 446–459, 2008.
2. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Wiener [32], pages 398–412.
3. Saar Drimer. Security for volatile fpgas (univertisy of cambridge technical report number 763). November 2009.

4. Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. Cryptology ePrint Archive, Report 2010/523, 2010. http://eprint.iacr.org/.
5. Henri Gilbert and Helena Handschuh, editors. *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*. Springer, 2005.
6. Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of aes. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002.
7. Louis Goubin and Jacques Patarin. Des and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.
8. Tim Güneysu and Christof Paar. Transforming Write Collisions in Block RAMs into Security Applications. In *IEEE International Conference on Field Programmable Technology (FPT 2009)*, pages 128–134, Sydney, Australia, December 2009. IEEE Society.
9. Marc Joye, Pascal Paillier, and Berry Schoenmakers. On second-order differential power analysis. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.
10. N. Kamoun, L. Bossuet, and A. Ghazel. SRAM-FPGA implementation of masked S-Box based DPA countermeasure for AES. In *Design and Test Workshop, 2008. IDT 2008. 3rd International*, pages 74–77. IEEE, 2009.
11. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [32], pages 388–397.
12. Paul Kohlbrenner and Kris Gaj. An embedded true random number generator for fpgas. In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, FPGA '04, pages 71–78, New York, NY, USA, 2004. ACM.
13. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Advances in Information Security. Springer, New York, 2007.
14. N. Mentens, L. Batina, B. Preneel, I. Verbauwhede, N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede. An FPGA Implementation of Rijndael: Trade-offs for side-channel security. In *IFAC Workshop-PDS*, pages 493–498. Citeseer, 2004.
15. Thomas S. Messerges. Securing the AES finalists against power analysis attacks. In Bruce Schneier, editor, *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.
16. Thomas S. Messerges. Using second-order power analysis to attack dpa resistant software. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.
17. Maxime Nassar, Shivam Bhasin, Jean-Luc Danger, Guillaume Duc, and Sylvain Guilley. Bcdl: A high speed balanced dpl for fpga with global precharge and no early evaluation. In *DATE*, pages 849–854. IEEE, 2010.
18. NIST. Announcing the advanced encryption standard aes. Technical report, Federal Information Processing Standards Publication 197, 2001.
19. Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical second-order dpa attacks for masked smart card implementations of block ciphers. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.
20. Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A side-channel analysis resistant description of the aes s-box. In Gilbert and Handschuh [5], pages 413–423.
21. Elisabeth Oswald and Kai Schramm. An efficient masking scheme for aes software implementations. In JooSeok Song, Taekyoung Kwon, and Moti Yung, editors, *WISA*, volume 3786 of *Lecture Notes in Computer Science*, pages 292–305. Springer, 2005.

22. G. Piret and F.-X. Standaert. Security analysis of higher-order boolean masking schemes for block ciphers (with conditions of perfect masking). *Information Security, IET*, 2(1):1 –11, 2008.
23. Emmanuel Prouff. Dpa attacks and s-boxes. In Gilbert and Handschuh [5], pages 424–441.
24. Emmanuel Prouff and Matthieu Rivain. A generic method for secure sbox implementation. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *WISA*, volume 4867 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2007.
25. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
26. Barry Shackleford, Motoo Tanaka, Richard J. Carter, and Greg Snider. Fpga implementation of neighborhood-of-four cellular automata random number generators. In *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, FPGA '02, pages 106–112, New York, NY, USA, 2002. ACM.
27. François-Xavier Standaert, Eric Peeters, and Jean-Jacques Quisquater. On the masking countermeasure and higher-order power analysis attacks. In *ITCC (1)*, pages 562–567. IEEE Computer Society, 2005.
28. François-Xavier Standaert, Gaël Rouvroy, and Jean-Jacques Quisquater. Fpga implementations of the des and triple-des masked against power analysis attacks. In *FPL*, pages 1–4. IEEE, 2006.
29. Elena Trichina and Tymur Korkishko. Secure aes hardware module for resource constrained devices. In Claude Castelluccia, Hannes Hartenstein, Christof Paar, and Dirk Westhoff, editors, *ESAS*, volume 3313 of *Lecture Notes in Computer Science*, pages 215–230. Springer, 2004.
30. Manfred von Willich. A technique with an information-theoretic basis for protecting secret data from differential power attacks. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 44–62. Springer, 2001.
31. Jason Waddle and David Wagner. Towards efficient second-order power analysis. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.
32. Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.
33. Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. An ASIC implementation of the AES sboxes. In Bart Preneel, editor, *CT-RSA*, volume 2271 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2002.