RSM: a Small and Fast Countermeasure for AES, Secure against 1st and 2nd-order Zero-Offset SCAs

Maxime Nassar*[†], Youssef Souissi*, Sylvain Guilley*[‡] and Jean-Luc Danger*[‡]

*Institut TELECOM / TELECOM ParisTech,
CNRS LTCI (UMR 5141), 46 rue Barrault
75 634 Paris Cedex, France.

[†]Bull TrustWay,
Rue Jean Jaurès, B.P. 68
78 340 Les Clayes-sous-Bois, France.

[‡]Secure-IC S.A.S.,
80 avenue des Buttes de Coësmes,
35 700 Rennes, France.

Abstract—Amongst the many existing countermeasures against Side Channel Attacks (SCA) on symmetrical cryptographic algorithms, masking is one of the most widespread, thanks to its relatively low overhead, its low performance loss and its robustness against first-order attacks. However, several articles have recently pinpointed the limitations of this countermeasure when matched with variance-based and other high-order analyses. In this article, we present a new form of Boolean masking for the Advanced Encryption Standard (AES) called "RSM", which shows the same level in performances as the state-of-the-art, while being less area consuming, and secure against Variance-based Power Analysis (VPA) and second-order zero-offset CPA. Our theoretical security evaluation is then validated with simulations as well as real-life CPA and VPA on an AES 256 implemented on FPGA.

Keywords: Side-Channel Attacks (SCA), Variance-based Power Analysis (VPA), zero-offset DPA, Mutual Information Analysis (MIA), substitution boxes (S-Boxes), Advanced Encryption Standard (AES), Boolean masking, Rotating S-boxes Masking (RSM).

I. INTRODUCTION

The Differential Power Analysis (DPA) [1], [2] takes advantage of the fact that the power consumption of a cryptographic device depends on the internally used secret key. Since this property can be exploited with relatively cheap equipment, DPA attacks pose a serious practical threat to cryptographic devices, like smart cards (ASICs) or embedded systems (DSPs, CPUs and FPGAs).

During the last ten years, there have been many endeavors to develop effective countermeasures against DPA attacks. Amongst the two major countermeasures [2] against DPA, namely *hiding* and *masking*, the latter is certainly the least complex to implement as it can be applied at the algorithmic level, in a software or hardware implementation. The idea of masking the intermediate values inside a cryptographic algorithm has been suggested in several papers [3], [4], [5], [6] as a possible countermeasure to power analysis attacks. Masking ensures that every single variable is masked with at least one random value so that a classical (first-order) DPA attack cannot be successfully carried out anymore. Nonetheless, straightforward implementations of this "first-order" countermeasure happened to be vulnerable to zero-offset "second-order" attacks [7], [8].

We call a "first-order" countermeasure an implementation where one single mask protects the sensitive data. Zero-offset attacks use one sample of side-channel trace, and are thus mono-variate. They apply when the masked variable and the mask are consumed simultaneously by the implementation, which is commonplace in hardware. Indeed, this architectural strategy allows to keep the throughput unchanged. Zero-offset second-order attacks consider not the plain observations themselves, but their variance instead [7], [8]. The variance of the leakage function, that involves its squaring (second-order moment), does depend strongly on the sensitive data. More sophisticated methods like the generic multi-variate attack called MMIA have been introduced in [9] to attack high-order countermeasures. Consequently, a branch of the research on masking countermeasures has evolved towards masking schemes with multiple masks. Another drawback of masking is the increase of complexity. In hardware implementation, the first-order masking countermeasure can be at least twice as much complex as the unprotected implementation; multiple masking is even worse. This significant increase comes from the mask path implementation necessary to operate the masking/unmasking operation, and specially the non-linear part [10]. Therefore, it is worthwhile to study masking countermeasures thwarting second-order DPA (2O-DPA) in particular, and high-order DPA (HO-DPA) in general, with a complexity increase almost negligible, or much less than a factor two. We describe in the present paper a masking method called RSM (short for "Rotating Sboxes Masking") which removes the masking path. Therefore this solution is low-cost, moreover it is robust against first-order and zero-offset 2O-DPA attacks.

The paper is organized as follows. Section II presents the principle of the RSM method. The RSM implementation is provided with in Sec. III. The section IV presents the theoretical security analysis of RSM. Then section V reports simulation and experimental results. Optimizations in terms both of security and of resources usage are given in Sec. VI. Finally, section VII concludes the paper and opens some perspectives.

II. PRINCIPLE OF RSM

In this section a detailed description of the proposed countermeasure, namely RSM, is given in terms of rationale and architecture. RSM aims at keeping performances and complexity close to an unprotected AES design, while being

as robust against first-order SCAs (DPA, CPA) as the state-of-the-art masking in hardware (*e.g.* [11], [12]). To our best knowledge, these countermeasures are based on the *Global Look-up Table* scheme described in [13]: the S-Boxes are addressed by the masked data and the mask, thereby leaking at second-order in the same of zero-offset attacks [7, §4.1]. Instead, the RSM countermeasure adheres to the *Re-computation Method* described in [13]. The S-Boxes are addressed only by the masked data: we say RSM has a mono-path structure. This feature grants to RSM an immunity to variance-based attacks (VPA [14, §4.3]) and makes it considerably resistant to MIA [15]. Nonetheless, RSM is based on using precomputed sets of constant masks rather than random ones, and specific customized S-Boxes with built-in input and output unmasking/masking operations.

In the remainder of this article, our study is based on a straightforward implementation of *AES 256* on FPGA, without pipelining and with 16 S-Boxes implemented in ROM. We also use the following notations: *S* for S-Box, *SB* for the whole SubBytes operation, *SR* for ShiftRows, *MC* for MixColumns.

A. Rotating S-Boxes

As stated in Sec. I, for most countermeasures on symmetrical cryptoprocessors, like AES or DES, the critical part in terms of area and computation time is the non-linear operation (*i.e.* the S-Boxes). Therefore, the main improvement brought by our design lies within the definition of low-cost, high performance S-Boxes.

RSM uses the same number of S-Boxes (16) as an unprotected implementation for the entire computation of the AES algorithm. But unlike any previous masking scheme, all those S-Boxes are different. They all contain a mechanism to unmask the input data, perform the basic $S(x)$ (where x is an 8-bit unmasked data) and re-mask it with another constant. However, these new S-Boxes would clearly be a source of first-order information leakage if implemented in logic gates, as the unmasked variable would be associated to an actual net. Therefore those S-Boxes shall be stored in RAM/ROM for either FPGAs or ASICs [16], after being precomputed as follows:

- Before programming the device, sixteen 8-bit constants m_{0-15} are randomly chosen once and for all. Those will be the base masks for the rest of this counter-measure.
- The 16 rotating S-Boxes $S'_{0-15}(x')$ (x' being an 8-bit masked data) are then designed to verify: $S'_j(x') = S(m_j \oplus x') \oplus m_{j+1 \pmod{16}}$, with $j \in \{0-15\}$.
- At each round of the AES algorithm, the S-Boxes are rotated by one position in direction $\mathcal{D}$, in order to successively compute all 16 possible SB'_j such as:

$$SB'_j = SB(M_j \oplus X') \oplus M_{j+1 \pmod{16}}, \forall j \in \{0-15\}, \tag{1}$$

where *SB* denotes the whole operation on 128-bit data, X' is the 128-bit masked state, and $M_j = \{m_j, m_{j+1 \pmod{16}}, \ldots, m_{j+15 \pmod{16}}\}$.
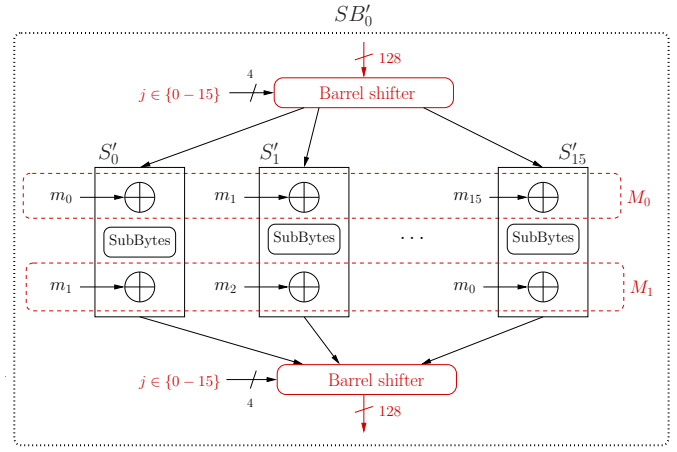

Figure 1.   Revolving S-Boxes.

Thus, considering that the 128-bit masked state X' is such as $X'_i = X_i \oplus M_j$ at round i, SB'_{0-15} will unmask it using the first *Xor* with M_j, perform the usual $SB(X)$ and remask it with $M_{j+1 \pmod{16}}$. This way, during the next round, thanks to the rotation, the same process will take place, using the next constant: unmasking with $M_{j+1 \pmod{16}}$ and re-masking with $M_{j+2 \pmod{16}}$. The order in which the constants are used is always the same, as it is fixed by the rotation direction $\mathcal{D}$, but depending on the one chosen for the first round: thus 16 different scenarios are possible, which induces a masking entropy of 4 bits.

The rotating S-Boxes can be implemented in hardware by adding barrel shifters on both sides of the SB operation, as shown in Fig. 1. This way, at each round before the S-Boxes, the state register can be shifted in direction $\mathcal{D}$, by an amount of bytes equal to the number j of Eqn. (1). Afterwards, the inverse process is performed, in order to rotate the state back to its original position.

Hence our new optimized SB'_{0-15} operator is only composed of 16 customized S-boxes S'_{0-15} (the same size as a basic one), and two barrel shifters, which induce but a small increase in terms of complexity and computation time, with regards to an unprotected AES implementation.

B. Masking the linear operations

From the 16 8-bit base masks, m_{0-15}, chosen to create the rotating S-Boxes (see Sec. II-A), 5 sets of 16 128-bit constants are deduced, and will be used to mask the linear part of the algorithm, while matching the required inputs and outputs of the SB'_{0-15} operator:

1) The first set consists of basic constants denoted by M_{0-15}, with $M_0 = \{m_0, m_1, \ldots, m_{15}\}$, and M_{1-15} are the successive rotations of one byte of M_0 in direction $\mathcal{D}$, such that:

$$M_j = \{m_j, m_{j+1 \pmod{16}}, \ldots, m_{j+15 \pmod{16}}\}$$
$$\forall j \in \{1-15\}.$$

2) The second set, MMS_{0-15} is defined as:

$$MMS_j = MC \circ SR(M_j) \oplus M_j, \quad \forall j \in \{1 - 15\}.$$

3) Constants of the third set, denoted by MS_{0-15}, verify:

$$MS_j = SR(M_j), \quad \forall j \in \{1 - 15\}.$$

4) Finally the last two sets, namely $IMMS_{0-15}$ and IMS_{0-15} are respectively identical to MMS_{0-15} and MS_{0-15} but with the inverse functions.

These constants are precomputed and stored in RAM/ROM, for a total of 1280 bytes, as depicted in Fig. 2.
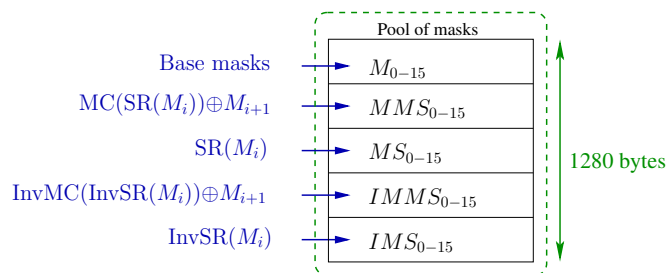

Figure 2. Storing masks in ROM/RAM.

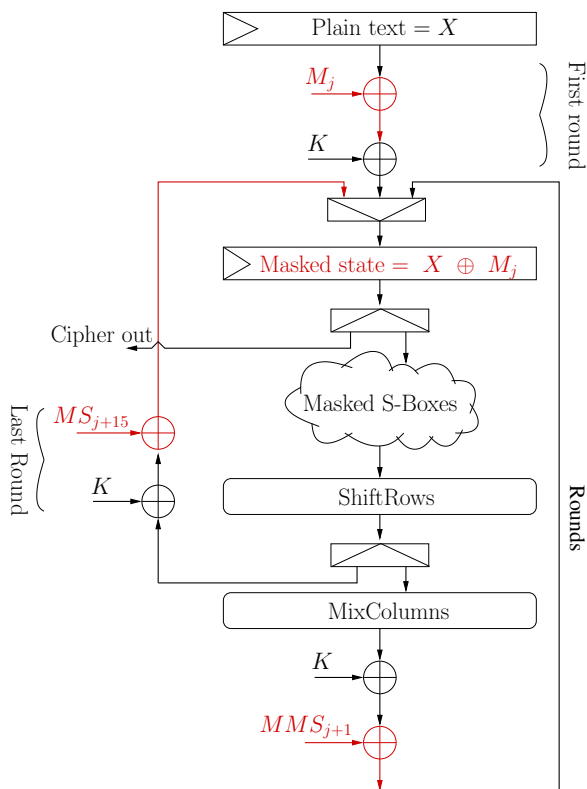Fig. 3 depicts the linear part of the datapath.

Figure 3. Linear part of the RSM datapath.

During the first round, a constant (M_j) is randomly chosen from the first set and *Xor*-ed with the initial plain-text (X) (this

way, as in the state-of-the-art masking, the power consumption is decorrelated from the actual data). The resulting masked state, $X'_{state1} = X \oplus M_j$, is the input of SB'_j. Then, as described in Sec. II-A, its output is $X'_{sbox1} = SB(X) \oplus M_{j+1}$.

Thanks to their linear properties, masking the *SR*, *MC* and *AddRoundKey* functions only requires a simple *Xor* operation. Keeping that in mind, we use the second set to simultaneously unmask the data at the end of each round and re-mask it with the next constant. Hence, the result of the linear operations is:

$$X'_{state2} = MC \circ SR(SB(X) \oplus M_{j+1}) \oplus K_{round}$$
$$= MC \circ SR(SB(X)) \oplus MC \circ SR(M_{j+1}) \oplus K_{round}.$$

Thereby, *Xor*-ing this value with MMS_j, removes the current mask: $MC \circ SR(M_{j+1})$, and re-masks it with M_{j+1}. Thus ensuring that the state register of the next round is indeed the expected masked value $X \oplus M_{j+1}$.

Finally during the last round, due to the absence of *MC*, the masked ciphered value is:

$$SR(SB(X) \oplus M_{j+14 \pmod{16}}) \oplus K_{round}, \quad j \in \{0 - 15\}.$$

Therefore, it can directly be unmasked with the constants of the third set, *i.e.* MS_j.

III. Implementation on Altera

As shown in the previous section, our overall design differs, in term of complexity, from an unprotected AES by three *Xor* operators, two barrel shifters and 1280 bytes of ROM. We implemented a reference AES 256, as well as one protected with the RSM countermeasure on an Altera Stratix-II, soldered on a SASEBO-B board provided by the RCIS [17]. Both bitstreams were generated using version 11.0 of the QuartusII software, with default synthesis and fitter options. Area occupation and performance results are shown in Table I.

Table I
IMPLEMENTATION RESULTS FOR REFERENCE AND PROTECTED AES

	Unprotected	RSM	Overhead
Number of ALUTs (%)	2136 (8%)	2734 (10%)	28%
Number of M4K ROM Blocs (%)	20 (14%)	24 (17%)	20%
Frequency (MHz)	133	88	34%

The number of ALUTs and M4K are given both in absolute value and in percentage of the total FPGA resources. As we can see, the overheads in terms of logical cells, ROM blocks and clock frequency are all within reasonable ranges, even for real-life applications where several IPs are included in the same FPGA.

As of now, few papers have dealt with an actual implementation of a complete masked AES design on FPGA. In [18], Mentens *et al.* proposed such an implementation, combining Boolean and multiplicative masking. However this type of countermeasure has been shown to be susceptible to so-called zero-value attacks, that exploit the absence of masking on the `0x00` byte value. More recently, Regazzoni *et al.* [12] have developed a full Boolean masking scheme on a Xilinx Virtex5

FPGA, obtaining an area consumption of roughly three times the unprotected one, and a performance penalty of 50%.

In this context, our implementation seems to bring a significant improvement in terms of area overhead, while keeping a reasonable performance degradation. It is however noteworthy that a precise and fair comparison between two FPGA designs is quite difficult, and depends on many factors [19] such as technology, vendor and synthesis options.

IV. THEORETICAL SECURITY EVALUATION

First of all, we notice that the initial offset j cannot be guessed by SPA, since irrespective of $j \in \{0 - 15\}$, all the masks are accessed in parallel. Thus, we consider differential SCA. For a theoretical security analysis, we place ourselves in the case where the last round of AES is attacked. The attacker is thus able to guess the transition on a sensitive variable x (one byte), that is nonetheless masked with an unknown byte m. Thus, the leakage function, in the Hamming distance model, is equal to $HW(x \oplus m)$. Nonetheless, contrary to usual masking schemes, m is not fully entropic: it does not take all the possible values in $\mathbb{F}_2^8$, but only a strict subset of them.

Even in these conditions, it is explained in [20] how to best attack a masking countermeasure with first- and second-order CPA attacks. An optimal correlation coefficient ρ_{opt} is defined in Eqn. (15) at page 802. Using a SAT-solver, we have identified several sets of 16 masks that resist CPA and second-order zero-offset CPA [21]. We note the following results:

- Without any mask (the random variable m is deterministic), both $\rho_{\text{opt}}^{(1)}$ at first-order and $\rho_{\text{opt}}^{(2)}$ at second-order are nonzero, and the mutual information leaked $\mathsf{I}[HW(x \oplus m); x]$ is equal to 2.5442 bit;
- With two complementary masks (m is uniformly distributed in a pair $\{\tilde{m}, \neg\tilde{m}\}$ for a given byte $\tilde{m}$, e.g. $\{0x00, 0xff\}$), $\rho_{\text{opt}}^{(1)} = 0$ but $\rho_{\text{opt}}^{(2)} \neq 0$, and the mutual information leaked is equal to 1.8176 bit; Thus the significant progress is the cancellation of $\rho_{\text{opt}}^{(1)}$. Regarding the mutual information, it still remains quite large.
- With the 16 masks found by the SAT-solver, $\rho_{\text{opt}}^{(1)} = \rho_{\text{opt}}^{(2)} = 0$, and $\mathsf{I}[HW(x \oplus m); z]$ can be found as low as 0.2168 bit.

From a leakage analysis point of view, we have shown that the RSM countermeasure with the 16 identified masks is secure against zero-offset second-order attacks. We nonetheless precise that this is not equivalent to being secure against any second-order attack. Indeed, if the attacker is able to spy simultaneously the random number generator (RNG) that decides for the masks' initial phase j and the addressing of the S-Boxes, then a second-order SCA could reveal information about the unmasked address of the S-Box. This is why we assume the RNG is not observable, and that the leakage only comes from the calls to the S-Boxes. In this case, our scheme is indeed second-order zero-offset resistant.

V. SIMULATION AND EXPERIMENTAL RESULTS

In this section, in order to validate our theoretical approach, simulation is performed on both *state-of-the-art* masking and RSM, as well as real life attacks on our countermeasure.

A. Simulation

We simulated attacks on the last round of AES for both *state-of-the-art* and RSM, considering the attacker would use the Hamming Distance between one byte of the last state register (*ST*) and the corresponding 8-bit known ciphertext (*C*). As a matter of fact, the Hamming Distance is one of the most commonly used leakage model in SCAs [22], especially on FPGAs [23].

Let x and x' respectively be the values of *ST* and *C* for the unprotected AES, and $\Delta(x)$ such as: $\Delta(x) = x \oplus x'$. Leakage observations were simulated in a "perfect" scenario (without any noise), and as single values depending on the countermeasure:

- *State-of-the-art*: sum of Hamming Distance for the masked data part, and Hamming Distance for the mask part, such as: $L_{mask} = HW(\Delta(xm)) + HW(\Delta(m))$, where:
  - $\Delta(m) = m \oplus m'$, m and m' being respectively the values of two last mask registers,
  - $\Delta(xm) = xm \oplus xm' = (x \oplus m) \oplus (x' \oplus m')$, corresponding to *ST* and *C* of the masked data part.
- *RSM*: Hamming Distance between the last masked state and the unmasked ciphertext, such as:

$$L_{RSM} = HW((x \oplus m_{0-15}) \oplus x') = HW(\Delta(x) \oplus m_{0-15}),$$

where m_{0-15} denotes the 16 base masks described in Sec. II-A.

This model complies with the theoretic one used in Sec. IV. The following analyses are performed on both architectures:

- Differential Power Attack (DPA);
- Correlation Power Attack (CPA);
- Variance-based Power Attack (VPA);
- Mutual Information Analysis (MIA).

1) Results: We evaluated the simulation results using the first-order success rate and guessing entropy metrics, proposed by Standaert in [24]. An attack is said to be successful when a success rate of 90% is reached.

First-order analyses, namely DPA and CPA, were unsuccessful on both countermeasures, for 100 attacks of 200000 simulated observations. Moreover the resulting success rate, in both cases, is always 0%.

VPA was successful on the *state-of-the-art* in about 1200 observations, as depicted in Fig. 4, whereas the RSM showed a success rate of 0% for up to 200000 observations (see Fig. 5). Additionally, the guessing entropy of RSM is roughly stable at 175, which means that the attack is not likely to succeed even with an increasing number of observations.

MIA is used as a metric to evaluate the information leakage, as described by Veyrat-Charvillon and Standaert in [25]. In this context, simulation results in a leakage of ≈ 1.0 bit on the *state-of-the-art* masking, and only ≈ 0.015 bit for RSM. These figures are lower than those announced in Sec. IV because the entropy estimation is based on histograms.
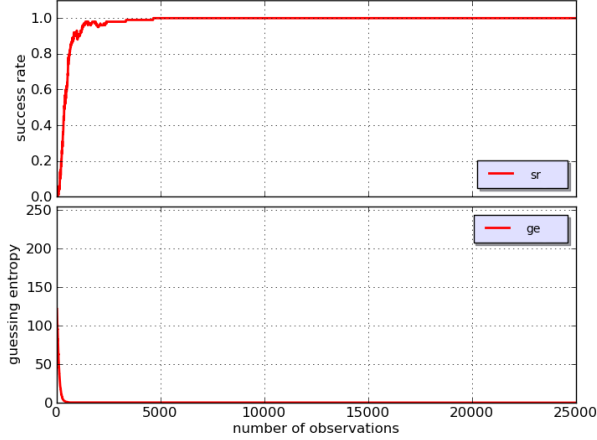
Figure 4. Success Rate and Guessing Entropy for 100 VPA Simulations on "State-of-the-Art" Masking.
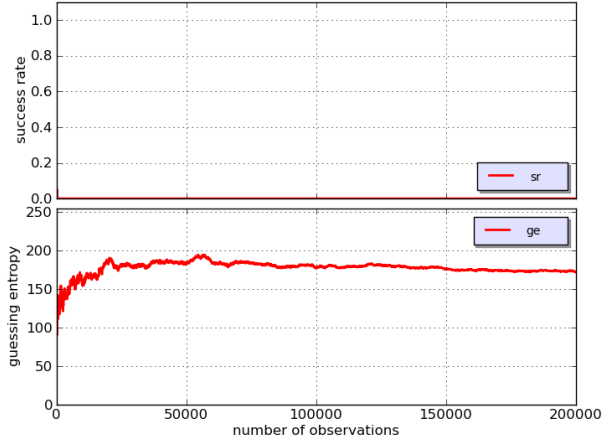

Figure 5. Success Rate and Guessing Entropy for 100 VPA Simulations on RSM.

B. Experimental setup

To corroborate our simulation results, we performed the same attacks on a real-life implementation of the RSM countermeasure. Power consumption measurements were acquired, using a *differential probe* plugged to the positive rail of the FPGA core power supply through a $1\,\Omega$ shunt resistor, coupled with a *54855 Infiniium oscilloscope* [26].

1) DPA, CPA, VPA: Those three attacks were all unsuccessful on 150000 power consumption measurements. Moreover they all displayed a success rate of roughly 0%.

2) MIA: In order to experimentally evaluate the information leakage of RSM, MIA as a metric [25] was performed on the same 150000 power traces. Results, displayed in Tab. II, show that, for all subkeys, the leakage is included between 0.001 and 0.012 bit, which corroborates the simulation performed in Sec. V-A, and should hardly be exploitable for a conclusive attack.

Table II
MUTUAL INFORMATION ON THE AES PROTECTED BY RSM.

Sub-key	0	1	2	3	4	5	6	7
MIA	0.012	0.006	0.008	0.006	0.010	0.007	0.006	0.005

Sub-key	8	9	10	11	12	13	14	15
MIA	0.004	0.011	0.001	0.008	0.004	0.012	0.009	0.002

VI. OPTIMISATIONS

As stated in Sec. II-B, although our implementation proves to be robust against CPA and DPA, the security evaluation shows a possible leakage due to the difference in entropy between the S-Boxes inputs and constants. Thus, in order to ensure an optimal security and remove all possible leakage, we introduce three new versions of this countermeasure: a time-security, a surface-security trade-off, and one using partial reconfiguration. In all cases, the idea is to regularly generate new pools of constants and customized S-Boxes, in order to approach the full 8-bit masking entropy.

A. Surface-security trade-off

The first idea is to compute new sets of constants and S-Boxes while using the old ones, and store them in additional memory, therefore not altering the performances. For this purpose, it is mandatory to use RAM for storing both S-Boxes and constants, and the required size is doubled. Some logic operators also need to be added:

- A 128-bit random number generator (RNG), to produce the 16 new base masks m_{0-15}.
- A barrel shifter.
- A *MC* and *InvMC* operator.
- A *SR* and *InvSR* operator (no actual logic gates).
- One basic AES S-Box and its inverse.
- 4/8 8-input *Xor* gates.

This way, each time a given set of S-Boxes and constants is used for the countermeasure, another is being computed and stored in the additional memory. With the additional barrel shifter, *MC*, *SR* and their inverse operators, the 5 sets of 16 constants can easily be generated from a 128-bit random number in 80 clock cycles (one per constant).

As for the S-Boxes, if the target device includes Dual-Port RAM, one S-Box can be used for two parallel computations, hence 8 *Xor* gates instead of 4. As a matter of fact, they can be created within $256 \times 8 = 2048$ or $256 \times 16 = 4096$ clock cycles, for respectively 8 and 4 *Xor* gates, considering the basic S-Boxes and their inverse are generated in parallel.

Eventually the area consumption of this countermeasure would be roughly twice the regular one (for both logical gates and memory blocks), for the same performances, but with a masking entropy of almost 8 bits.

Moreover, considering that a straightforward implementation of AES 256 takes 14 clock cycles to process, it follows that a potential attacker would be able to take at most 150 to 300 side-channel measurements for a given S-Box/constants

set, which should hardly be sufficient to perform a meaningful analysis.

B. Time-security trade-off

The second idea is to pause the encryptions every once in a while, in order to compute a new sets of S-Boxes and constants, using the existing AES operators, as well as an additional basic AES S-Box/InvS-Box, $4/8$ 8-input *Xor* gates and a 128-bit RNG. As stated in the previous section, it would take between $4096 + 80 = 5076$ and 2538 clock cycles, that is respectively about $50\ \mu s$ or $25\ \mu s$ at 100 MHz.

C. Using partial reconfiguration

Finally it should be possible to take advantage of recent FPGA technologies, like the Xilinx Virtex5 family, which allows partial reconfiguration of the device. As a matter of fact, the mask recomputation could be performed by an embedded processor, which would regularly reconfigure the RAM blocks containing both constants and S-Boxes. This way, the countermeasure size and performances would be almost unchanged, except for the reconfiguration times, during which the computations must be paused. We insist that knowing the masks is of no use for an attacker (since only the direction $\mathcal{D}$ is sensitive): therefore, the mask fresh process needs not be protected against SCA.

VII. Conclusion and perspectives

In this paper we presented a new masking scheme for AES called RSM. A theoretical evaluation pointed out the security of RSM against first- and second-order zero-offset SCAs, and was corroborated by both simulations and real-life attacks. Moreover the implementation results on Altera StratixII FPGA shows that the performances in terms of speed and complexity is very near to unprotected implementation and far better than usual masking structures.

The RSM countermeasure discussed in this paper has been described as a Boolean masking scheme. However, it could also be implemented using other masking flavors. Therefore, as a perspective, we intend to study its overhead and security with affine masking [27]. On another hand, exploiting partial reconfiguration on recent FPGAs also seems to be an interesting lead to further improve the robustness of RSM by refreshing the mask regularly, in order to thwart dth-order attacks ($d > 2$).

References

[1] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *CRYPTO*, ser. LNCS, vol. 1666. Springer, 1999, pp. pp 388–397.

[2] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. http://www.springer.com/Springer, December 2006, ISBN 0-387-30857-1, http://www.dpabook.org/.

[3] M.-L. Akkar and C. Giraud, "An Implementation of DES and AES Secure against Some Attacks," in *Proceedings of CHES'01*, ser. LNCS, LNCS, Ed., vol. 2162. Springer, May 2001, pp. 309–318, Paris, France.

[4] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks," in *CRYPTO*, ser. LNCS, vol. 1666. Springer, August 15-19 1999, Santa Barbara, CA, USA. ISBN: 3-540-66347-9.

[5] L. Goubin and J. Patarin, "DES and Differential Power Analysis. The "Duplication" Method," in *CHES*, ser. LNCS. Springer, Aug 1999, pp. 158–172, Worcester, MA, USA.

[6] T. S. Messerges, "Securing the AES Finalists Against Power Analysis Attacks," in *Fast Software Encryption'00*. Springer-Verlag, April 2000, pp. 150–164, New York.

[7] J. Waddle and D. Wagner, "Towards Efficient Second-Order Power Analysis," in *CHES*, ser. LNCS, vol. 3156. Springer, 2004, pp. 1–15, Cambridge, MA, USA.

[8] É. Peeters, F.-X. Standaert, N. Donckers, and J.-J. Quisquater, "Improved Higher-Order Side-Channel Attacks With FPGA Experiments," in *CHES*, ser. LNCS, vol. 3659. Springer-Verlag, 2005, pp. 309–323, Edinburgh, UK.

[9] B. Gierlichs, L. Batina, B. Preneel, and I. Verbauwhede, "Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis," in *CT-RSA*, ser. LNCS, vol. 5985. Springer, March 1-5 2010, pp. 221–234, San Francisco, CA, USA.

[10] G. Piret and F.-X. Standaert, "Security Analysis of Higher-Order Boolean Masking Schemes for Block Ciphers (with Conditions of Perfect Masking)," *IET Information Security*, vol. 2, no. 1, pp. 1–11, 2008, DOI: 10.1049/iet-ifs:20070066.

[11] F.-X. Standaert, G. Rouvroy, and J.-J. Quisquater, "FPGA Implementations of the DES and Triple-DES Masked Against Power Analysis Attacks," in *FPL*. IEEE, August 2006, Madrid, Spain.

[12] F. Regazzoni, Y. Wang, and F.-X. Standaert, "FPGA Implementations of the AES Masked Against Power Analysis Attacks," in *COSADE*, February 2011, pp. 56–66, Darmstadt, Germany.

[13] E. Prouff and M. Rivain, "A Generic Method for Secure SBox Implementation," in *WISA*, ser. Lecture Notes in Computer Science, S. Kim, M. Yung, and H.-W. Lee, Eds., vol. 4867. Springer, 2007, pp. 227–244.

[14] F.-X. Standaert, B. Gierlichs, and I. Verbauwhede, "Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices," in *ICISC*, ser. LNCS, vol. 5461. Springer, December 3-5 2008, pp. 253–267, Seoul, Korea.

[15] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon, "Mutual Information Analysis: a Comprehensive Study," *J. Cryptology*, vol. 24, no. 2, pp. 269–291, 2011.

[16] S. Shah, R. Velegalati, J.-P. Kaps, and D. Hwang, "Investigation of DPA Resistance of Block RAMs in Cryptographic Implementations on FPGAs," in *ReConFig*, V. K. Prasanna, J. Becker, and R. Cumplido, Eds. IEEE Computer Society, 2010, pp. 274–279.

[17] Japanese RCIS-AIST, SASEBO development board: http://www.rcis.aist.go.jp/special/SASEBO/index-en.html.

[18] N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "An fpga implementation of rijndael: Trade-offs for side-channel security," 2004.

[19] S. Drimer, "Security for Volatile FPGAs (university of Cambrige technical report number 763)," November 2009.

[20] E. Prouff, M. Rivain, and R. Bevan, "Statistical Analysis of Second Order Differential Power Analysis," *IEEE Trans. Computers*, vol. 58, no. 6, pp. 799–811, 2009.

[21] M. Nassar, S. Guilley, and J.-L. Danger, "Formal Analysis of the Entropy / Security Trade-off in First-Order Masking Countermeasures against Side-Channel Attacks," in *INDOCRYPT*, ser. LNCS, vol. 7107. Springer, December 11-14 2011, pp. 22–39, Chennai, India. DOI: 10.1007/978-3-642-25578-6_4.

[22] É. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," in *CHES*, ser. LNCS, vol. 3156. Springer, August 11–13 2004, pp. 16–29, Cambridge, MA, USA.

[23] F.-X. Standaert, É. Peeters, F. Macé, and J.-J. Quisquater, "Updates on the Security of FPGAs Against Power Analysis Attacks," in *ARC*, ser. LNCS, vol. 3985. Springer-Verlag, March 2006, pp. 335–346, delft, The Netherlands.

[24] F.-X. Standaert, T. Malkin, and M. Yung, "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks," in *EUROCRYPT*, ser. LNCS, vol. 5479. Springer, April 26-30 2009, pp. 443–461, Cologne, Germany.

[25] N. Veyrat-Charvillon and F.-X. Standaert, "Mutual Information Analysis: How, When and Why?" in *CHES*, ser. LNCS, vol. 5747. Springer, September 6-9 2009, pp. 429–443, Lausanne, Switzerland.

[26] Agilent Technologies: http://www.agilent.com/.

[27] G. Fumaroli, A. Martinelli, E. Prouff, and M. Rivain, "Affine Masking against Higher-Order Side Channel Analysis," in *Selected Areas in Cryptography*, ser. LNCS, A. Biryukov, G. Gong, and D. R. Stinson, Eds., vol. 6544. Springer, 2010, pp. 262–280.