Practical Evaluation of DPA Countermeasures on Reconfigurable Hardware

Amir Moradi, Oliver Mischke, and Christof Paar *Fellow Member, IEEE*
Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{moradi, mischke, cpaar}@crypto.rub.de

***Abstract*—In CHES 2010 a correlation-based power analysis collision attack has been introduced which is supposed to exploit any first-order leakage of cryptographic devices. This work examines the effectiveness of the well-known DPA countermeasures versus the correlation collision attack. The considered countermeasures include masking, shuffling, and noise addition, when applied in hardware. Practical evaluations, which all have been performed using power traces measured from an FPGA board, show an increase in the number of required traces, e.g. from 10,000 to 1,500,000, when combining different countermeasures. This study allows for a fair comparison between the hardware countermeasures and helps identifying an appropriate key lifetime.**

I. Introduction

Physical attacks have become one of the major challenges that designers of cryptographic hardware are concerned about. For example, power analysis attacks, namely DPA [10], can easily break security features of unprotected implementations [7]. During the last years several countermeasure schemes, e.g., [9], [13], [17], in different levels of abstraction have been proposed. On the other hand, the research community has put considerable efforts into either improving attack methods or defeating the countermeasures, e.g., [8], [11], [16]. One of the attack schemes is known as side-channel collision attack [15] which mainly aims at detecting two identical intermediate values by means of comparing their corresponding side-channel leakages. This family of attacks which mostly focused on software implementations, e.g., [3], seem inefficient in the presence of randomizing-like countermeasures. However, a recent work [12] (so-called correlation collision attack) gets the benefit of correlation attacks to efficiently detect the collisions and overcome countermeasures in which even a small first-order leakage still remains.

The target device of [12] was an 8-bit serialized masked implementation of the AES computing one S-box per clock cycle which provides a suitable situation for a collision attack. A question may arise on the efficiency of such an attack having different architectures in addition to different countermeasures. Although some notes about a 32-bit architecture have been given by [12], it seems necessary to have a comprehensive study on the influence of different randomizing and noise-additive countermeasures. Two different architectures are considered in this paper. The first one, which has a 32-bit data path, in contrary to the architecture of [12] does not employ any shift register and implements four S-box instances. Both masking and shuffling schemes are the options which can be enabled in this architecture. The second architecture is based on the unrolling scheme, proposed in [2], as a DPA countermeasure. Using this approach, we are able to execute a whole AES encryption in 10, 5, 4, and 3 clock cycles.

We investigate the efficiency of the correlation collision attack when masking, shuffling, unrolling, and their possible combinations are enabled in our target architectures. The practical evaluations are performed using power consumption traces measured from the same platform as in [12], i.e., a Virtex-II Pro FPGA. The result of these investigations can be summarized as: non of the previously mentioned countermeasures can perfectly provide resistance against the considered attack. The reasons, which are well-known to the community, are that *(i)* implementing masking in hardware still leads to a kind of first-order leakage caused by glitches in the circuit that is detectable by our considered attack, *(ii)* shuffling which does randomization in the time domain is also defeated by increasing the number of traces or using a "windowing" scheme, and *(iii)* unrolling, which seems to have the most effect on collision-like attacks, adds noise to the measurements and is overcome by averaging, which is done inherently by the correlation collision attack. However, enabling each (or a combination) of these countermeasures leads to an increase in the number of required traces. Depending on the target application this can be considered as an important parameter helping to define the key life time of the device under evaluation.

The remainder of this article is organized as follows: Section II provides a short description of the correlation collision attack. The different implemented designs and countermeasures are presented in Section III. The evaluation results of the attack on the 32-bit architecture employing masking and shuffling are shown in Section IV, while the results of the attack on the unrolled architectures are depicted in Section V. A conclusion is finally given in Section VI.

II. Correlation Collision Power Analysis

The correlation collision attack has been introduced in [12]. In contrary to classical power analysis attacks which need to employ a hypothetical power model, e.g., in CPA, or a distinguisher, e.g., in MIA, the correlation collision attack needs neither a hypothesis for the power model nor an offline profiling phase. Also, unlike other collision attacks, it works well against certain masked implementations which still have some kind of first order leakage.

Similar to other collision attacks, it recovers the differences between the parts of the secrets, e.g., the xor between the key bytes in the case of AES, which finally allows an easy key recovery. Since the big combinational circuits, e.g., AES S-boxes, are usually shared in the computation of a cipher round because of area constrains, the collision attacks can compare the side-channel leakage of the same instance of the circuit in two different instances of time. Due to the bijectivity of the AES S-box, output collisions, i.e., two different S-box computations in time t_1 and t_2 taking the same value, require input collisions. Means, $\Delta = i_1 \oplus i_2 = k_1 \oplus k_2$ when $\mathrm{Sbox}(i_1 \oplus k_1) = \mathrm{Sbox}(i_2 \oplus k_2)$, which is known as linear collision attack on AES [3]. The advantage of the correlation collision attack in comparison to the other collision attacks is that it tries to detect the case when maximum collisions occur selecting the correct Δ.

In order to perform the correlation collision attack the power values corresponding to time t_1 are sorted based on the input byte value i_1 such that all traces where $i_1 = \alpha$ are summed and averaged to an average power consumption $M_1(\alpha)$. Hence, due to the 256 possible values that α can take for AES, we get 256 different $M_1(\alpha)$. Repeating the same procedure for input byte value i_2 on power values at time t_2 leads to 256 different $M_2(\alpha)$. The attack now assumes that the power consumption of the S-box computation for two different bytes at t_1 and t_2 has the same leakage if the same values are processed. For a known key difference $\Delta = k_1 \oplus k_2$, the S-box inputs are the same if $i_1 = i_2 \oplus \Delta$, and hence the average power consumptions $M_1(\alpha) \approx M_2(\alpha \oplus \Delta)$ should be highly similar. Such a similarity can be detected computing the correlation between all possible (M_1, M_2)-pairs for all possible key differences Δ. The correlation for a correct key difference Δ is very high, as each (M_1, M_2)-pair is a direct collision, while for false Δ's the correlation is low as unrelated computations are correlated. Repeating the same scheme for different S-box computations corresponding to different input byte values recovers sufficient Δ's to reveal the complete secret.

Since this attack compares the power consumption characteristics of two combinational circuits, as illustrated in [12], the best result is achieved by comparing the power consumption of one instance of the target combinational circuit, e.g., an S-box, in different clock cycles. Therefore, the best target for this attack is when only one instance of the S-box is implemented and shared in all S-box computations. If there are more instances of the S-box, e.g., a 32-bit architecture where four instances of the S-box are implemented and four S-box computations are performed in a clock cycle, the attack should compare the power consumption of each instance of the S-box in different clock cycles. If the architecture does not share any S-box for a round computation, and comparing the leakages of the same instance of the circuit in different clock cycles is not possible any more, the effectiveness of such an attack depends strongly on the similarity of power consumption characteristics of different instances of the S-box whose leakages are compared.

III. Architectures and Measurement Setup

This section gives an overview of the architectures used to evaluate the countermeasures, and provides characteristics of our implementation platform and the setup used for side-channel measurements.

A. 32-bit Architecture

The objective by choosing a 32-bit architecture was to use a real-world scenario during our measurements. While choosing an 8-bit architecture would be the best choice from an attackers point-of-view because of the reduced amount of noise and the option to observe each processed byte independently, selecting a 32-bit architecture provides a good compromise between size and throughput while still enabling us to use countermeasures like shuffling.

The 32-bit architecture in this context means that all module interconnections are using a 32-bit datapath including the outside ports. Module internals are not bound to this restriction, so the ShiftRows transformation and the key scheduling are performed in a single clock cycle using an internal 128-bit datapath.

The complete datapath of the AES engine excluding the key schedule and key registers are masked. Similarly to the scheme used in [12], we apply the additively masked AES S-box by Canright and Batina [5] that uses a tower-field approach [14] to implement the inversion in $GF(2^2)$. Each S-box operation needs two mask bytes, one for the input masking and one to mask the output byte. These mask values are independent of each other, and are generated by a PRNG with uniform distribution. The general order of mask switches is as follows: in the beginning of an encryption each input byte is masked by the corresponding input mask (let us call it m for one input byte)[1]. While passing trough the inversion part of the S-box the input mask is replaced by the output mask n. This process is reversed by the masked MixColumns module while in the last round, where no MixColumns operation is performed, the S-box output mask n is removed after the final AddRoundkey operation, and the final result of the AES operation is stored unmasked in the state register. An overview of this architecture is depicted in Fig. 1

Since the S-boxes are shared between the normal data operation and the key schedule, and four instances of the S-box are implemented in our target architecture, each round of the AES needs five clock cycles. During the first clock cycle the S-boxes are used by the key schedule unit. Since the key schedule is not masked, the input and output masks of the S-boxes are set to all zero by means of AND gates. Simultaneously, while the S-boxes are utilized by the key schedule, the ShiftRows transformation is performed both on the data state and the mask m registers. This is necessary because the data state is masked by the m masks which therefore have to be transformed as well to keep the accordance between the mask bytes and the data state. In

[1]No mask reuse is applied in a computation of a cipher round; two 128-bit masks are required for each encryption or decryption run.

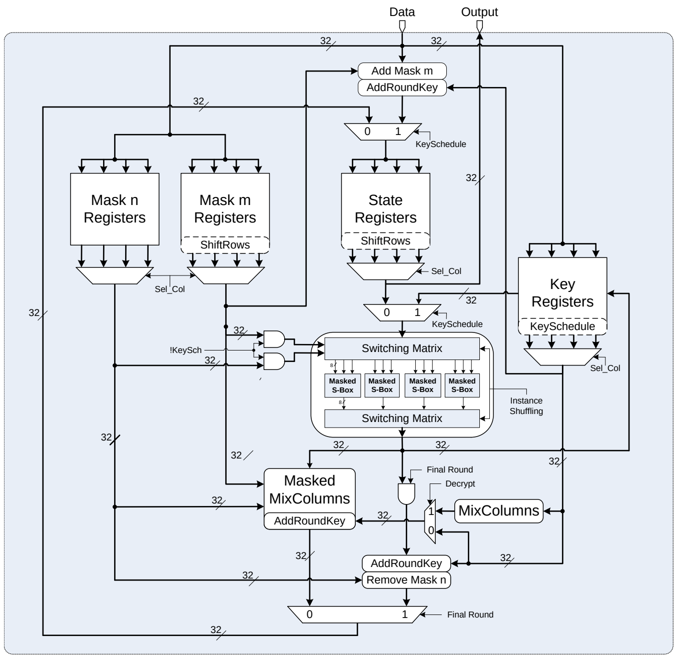

Fig. 1. Architecture of the 32bit Implementation, allowing masking as well as column-wise and S-box instance shuffling

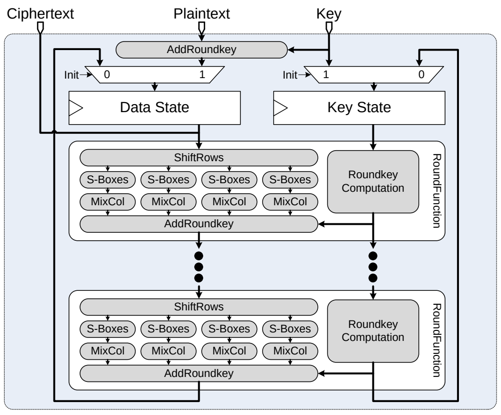

Fig. 2. Architecture of the unrolled designs

the next four clock cycles the SubBytes, MixColumns, and AddRoundkey transformations are applied on one column at a time. This allows implementing the second countermeasure, i.e., shuffling, which needs each column of the data, mask, or key registers to be selected and stored independently. It is therefore possible to switch the processing order of the columns during each encryption (we call this option column-wise shuffling). We also implemented another option to switch which byte of each column is processed by which S-box instance (we call this option instance shuffling). It should be noted that the same procedure and options have been considered for the decryption operation which shares some building blocks with the encryption unit.

Our final architecture has different options, i.e., masking, column-wise shuffling, and instance shuffling, which can be selected during the operation of the target device. Based on these options we define five different profiles, and later investigate the efficiency of each to a correlation collision attack. These profiles are as follows:

- Profile A: no countermeasure, using always zero for all the masks and turning off both shuffling options
- Profile B: column-wise shuffling only
- Profile C: masking only
- Profile D: masking and column-wise shuffling
- Profile E: masking, column-wise shuffling and instance shuffling

B. Unrolled Architecture

In addition to the masking and shuffling schemes we have tried to examine the effectiveness of unrolling, which has been explained in [2], counteracting correlation collision attacks. An overview of an unrolled design is shown by Fig. 2. In order to reduce the required area of our unrolled architecture, we chose the very compact unmasked S-box by Canright [4] in an encryption only scenario. Since the key scheduling is unrolled as well, twenty S-boxes are needed to implement each round function.

We implemented four different designs, varying the number of rounds which are computed per clock cycle. In the smallest design only one complete round is computed at each clock cycle. The second design features two complete rounds, the third computes three and the last design computes four complete rounds of the AES at each clock cycle creating a highly glitching circuit.

C. Target Platform and Measurement Setup

All designs have been implemented on a Xilinx Virtex-II Pro FPGA (xc2vp7) of a SASEBO circuit board which is particularly designed for side-channel attack experiments [1]. All experiments are performed on the power consumption of the Virtex-FPGA containing our implementation. Measurements are performed using a LeCroy WP715Zi 1.5GHz oscilloscope at a sampling rate of 1GS/s and by means of a differential probe which captures the voltage drop over an 1Ω resistor in the VDD (1.6V) supply path of the FPGA. In all the experiments the clock signal of our cryptographic engine is supplied by a stable oscillator at a frequency of 3MHz.

IV. Evaluation of the 32-bit Architecture

The later parts of this section deal with evaluating the resistance/vulnerability of different profiles of the 32-bit architecture addressed in Section III-A to correlation collision attacks.

A. Profile A: No Countermeasures

Performing the correlation collision attack described in Section II, we start by creating two sets of 256 mean traces according to the plaintext byte values corresponding to the target S-box instances. As explained in [12], the best situation for a collision attack is when the side-channel leakages of an S-box instance in two different clock cycles are compared.

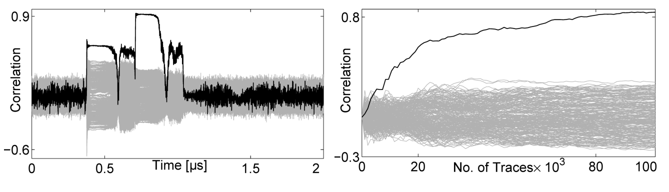

Fig. 3. Profile A: the result of an attack (a) using $1,000,000$ traces and (b) over the number of traces

We therefore have selected two plaintext bytes which are processed by the same S-box instance. Looking at the variance traces computed over a set of mean traces, e.g., Fig. 12(b), clarifies in which clock cycle the corresponding plaintext byte is processed.

In order to perform the attack, aiming at recovering the relation between two key bytes, the mean traces must first be aligned to have both S-box executions – virtually – at the same instance of time. Though $1,000,000$ traces have been used for the attack result depicted in Fig. 3(a), plotting the result over the number of traces, i.e., Fig. 3(b), shows that for this case even $10,000$ traces are sufficient to mount a successful attack.

B. Profile B: Column-wise Shuffling Only

When the column-wise shuffling is enabled the target S-box computation does not take place at a specific clock cycle while always the same S-box instance performs the computations. Therefore, as an example is given by Fig. 12(d), the variance over the mean traces shows high values in all four clock cycles when the S-boxes are computed. This can be in fact considered as an evidence of the existing time-randomization countermeasure. Without taking this countermeasure into account and just performing the last attack[2], as depicted in Fig. 4(a), the correct difference between the target key bytes is still detectable and appears in all four mentioned clock cycles. It however requires a higher number of traces here, i.e., $50,000$, since on average only one fourth of the mean traces are aligned to each other.

As mentioned in [12], one can divide a trace into clock cycles and sum them up to have a sum trace with a length of one clock cycle, which is known as "windowing" (integration over a sliding comb) [6]. Doing so on the traces of this profile considering those four clock cycles where the SubBytes transformations of the first round are performed, guarantees that the mean traces, which now are as long as a clock cycle, are aligned and contain the desired information. Performing the same attack on the combined traces reveals the correct secret and decreases the required number of traces to $20,000$, as depicted in Fig. 4(c) and Fig. 4(d).

C. Profile C: Masking Only

While column-wise shuffling had low area and power overheads, implementing the masked S-boxes (as explained in Section III-A) needs significantly more area and leads to a high power consumption. This can be seen when comparing

[2]It is not needed to shift the mean traces and align them in this case.

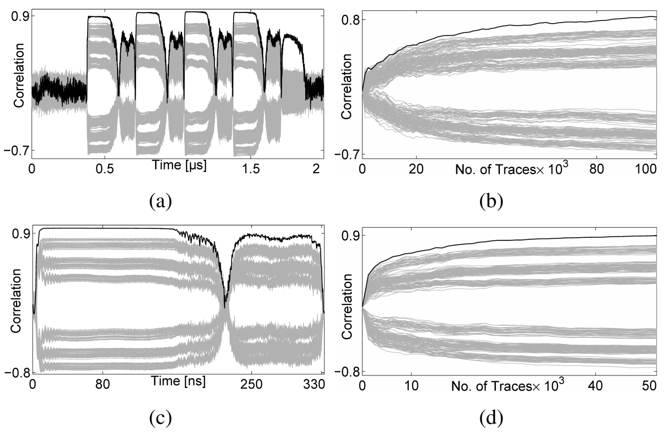

Fig. 4. Profile B: the result of an attack (a) using $1,000,000$ traces, (b) over the number of traces, (c) using windowing, and (d) over the number of traces using windowing

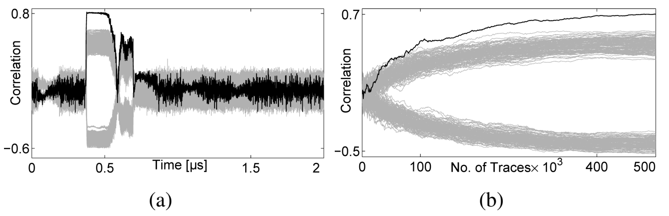

Fig. 5. Profile C: the result of an attack (a) using $5,000,000$ traces and (b) over the number of traces

the sample power traces of these two architectures in Fig. 12(c) and Fig. 12(e).

Since no shuffling is enabled in this profile, the mean traces must be aligned according to the clock cycles reported by the variance traces, e.g., Fig. 12(f). It should be noted that as expressed in [12] and can be seen in the variance trace, the masked S-box implementation still has a first order leakage which is due to the glitches that occur in the combinational functions. The fact that the variance is lower than that of the previous profiles implies that a higher number of traces is necessary to reveal the correct key relation. The result of the attack using $5,000,000$ traces is shown in Fig. 5(a), but based on Fig. 5(b) $150,000$ measurements are sufficient in our case to reveal the desired secret.

D. Profile D: Masking and Column-wise Shuffling

Attacking an implementation that combines both of the previously applied countermeasures proves to be highly resistant against the correlation collision attack. Observing the variance traces, e.g., Fig. 12(h), shows that the dependency of the mean traces on the plaintext byte values is decreased because of the used masking scheme, and is spread over four clock cycles because of the column-wise shuffling. Fig. 6(a) shows the result of the attack using $10,000,000$ traces. As expected after comparing the variance traces to those of the previous profiles, Fig. 6(b) reports around $4,500,000$ as the number of traces we required which is considerably higher than in the previous cases. If the same windowing scheme is used to overcome the time-randomization effect of the shuffling, the number of traces here decreases to $700,000$ as depicted in Fig. 6(c) and Fig. 6(d).

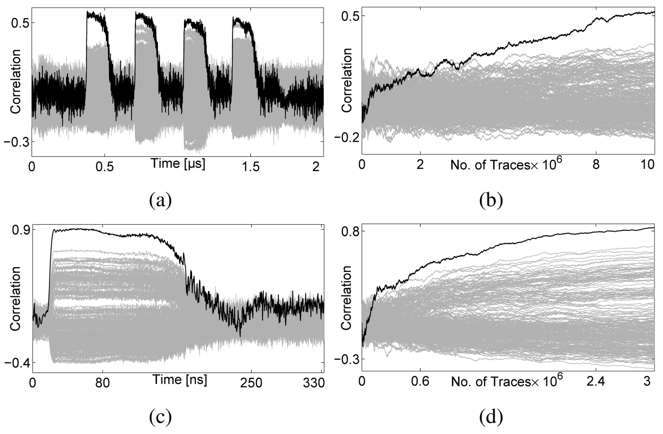

Fig. 6. Profile D: the result of an attack (a) using $10,000,000$ traces, (b) over the number of traces, (c) using windowing, and (d) over the number of traces using windowing

E. Profile E: Masking, Column-wise and Instance Shuffling

As stated before, the correlation collision attack works best if the target plaintext bytes are processed by the same S-box instance. Randomizing which of the four S-box instances compute which bytes of a column (called instance shuffling in Section III-A) should further increase the resistance of the implementation. This is confirmed comparing a variance trace over the mean traces of this profile (Fig. 12(j)) and that of profile D. The results of the attack using $10,000,000$ traces in both cases with and without windowing in addition to their required number of traces are shown by Fig. 7. While around $5,500,000$ traces are necessary to distinguish the correct guess when performing the considered attack in a straightforward way, employing windowing reduces this number to $1,500,000$ which is significantly higher than of previous profiles.

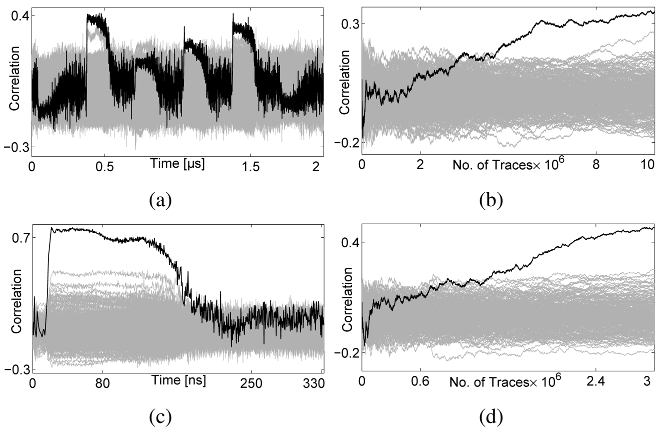

Fig. 7. Profile E: the result of an attack (a) using $10,000,000$ traces, (b) over the number of traces, (c) using windowing, and (d) over the number of traces using windowing

V. Evaluation of the Unrolled Architecture

The same attacks, which have been done on the 32-bit architecture, are performed on the traces measured from the unrolled implementations. Since in our smallest unrolled architecture one round of the cipher encryption is performed per clock cycle, there is no shared hardware unit during the computation of a round. Therefore, one cannot compare the side-channel leakage of an unit in different clock cycles, and needs to consider different S-box instances to perform a collision attack. This, of course, decreases the efficiency of the attack since two different circuits are compared which even with the same netlist have been differently placed and routed by the synthesizer. Moreover, the switching noise level generated by the other parts of the circuits, e.g., S-boxes, which are not considered in the attack is considerably higher than the case of 32-bit architecture. We therefore have expected a higher number of required traces and have collected more traces compared to the 32-bit cases. The results of this attack on different unrolled implementations are given by this section.

A. One Round per Clock Cycle

Observing a sample power trace of a whole encryption run by our smallest unrolled implementation depicted in Fig. 13(a) verifies the execution of one round per clock cycle[3]. The same as before, two S-box instances and hence their corresponding plaintext bytes have been selected to make two sets of 256 mean traces. Since both the selected S-boxes are executed at the same clock cycle, their corresponding traces are already aligned and when performing the correlation collision attack we do not need to shift the mean traces. Fig. 8(a) shows the attack result using $1,000,000$ traces, and as depicted in Fig. 8(b) the attack is still successful using $100,000$ measurements. We should emphasize that the difference between the side-channel leakage of the implemented 16 S-box instances varies because of their different similarity in placement and routing. Therefore, the efficiency of the attack also varies selecting different S-box instances. The result shown in Fig. 8 is one of the best cases.

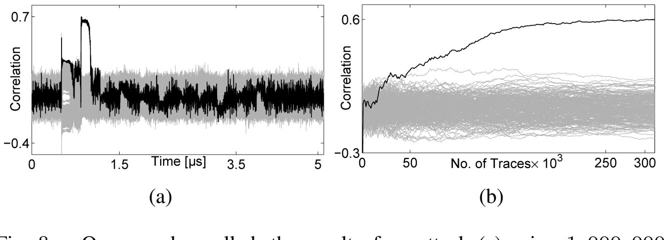

Fig. 8. One round unrolled: the result of an attack (a) using $1,000,000$ traces and (b) over the number of traces

[3]In Fig.13(a) 11 clock cycles with high power consumption are detectable. The last one is due to the case when the ciphertext is saved into the state register and appears at the input of the combinational circuit again.

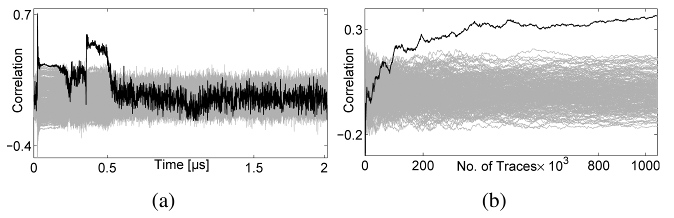

Fig. 9. Two rounds unrolled: the result of an attack (a) using $7,500,000$ traces and (b) over the number of traces

B. Two Rounds per Clock Cycle

When two rounds of the cipher encryption are unrolled, as can be seen in Fig. 13(c), the whole of an encryption is performed in 5 clock cycles. Comparing the variance traces shown in Fig. 13(b) and Fig. 13(d) of one and two unrolled rounds respectively, shows a significant increase of the noise level. Repeating the same attack procedure as before on $7,500,000$ traces collected from the two-round unrolled implementation led to the result shown by Fig. 9(a) as amongst the most successful cases. Also, Fig. 9(b) reports around $300,000$ as the number of traces we have required to recover the correct relation between the selected key bytes. Although evaluation of the later rounds knowing their inputs when more than one round is unrolled has been included in the original proposal of the unrolling countermeasure [2], we have not reported the results of the corresponding collision attacks because of their similarity to the case when attacking the first round. Moreover, knowing the input of the e.g., second round of the AES, in contrary to the DES case, reveals all the secrets used in the first round, and one does not need to perform the attack on the second round.

C. Three Rounds per Clock Cycle

Fig. 10 shows the results of a similar attack on the first round when three unrolled rounds are implemented, and the whole encryption process is completed in 4 clock cycles. As a reference, Fig. 13(e) and Fig. 13(f) respectively show a sample power trace of this implementation and a variance trace over a set of 256 mean traces. According to the low variance (Fig. 13(f)) and unclear distinguishability of the correct hypothesis amongst the others (Fig. 10(a)), a high number of required traces is expected, e.g., around $3,000,000$ as shown in Fig. 10(b).

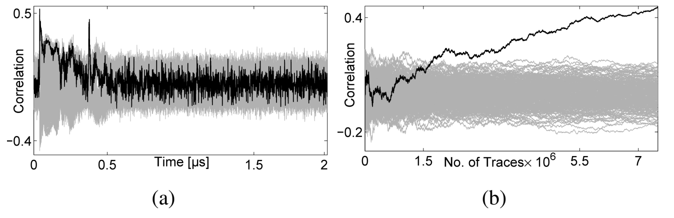

Fig. 10. Three rounds unrolled: the result of an attack (a) using $7,500,000$ traces and (b) over the number of traces

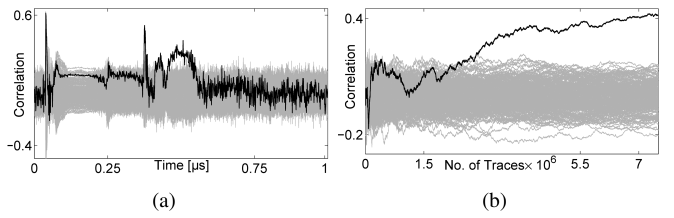

Fig. 11. Four rounds unrolled: the result of an attack (a) using $30,000,000$ traces and (b) over the number of traces

D. Four Rounds per Clock Cycle

In our last unrolled architecture, where four unrolled encryption rounds are implemented, every encryption run needs 3 clock cycles (see Fig. 13(g) as an example), and the switching noise has the highest level compared to all previous examined architectures (see Fig. 13(h) as a variance trace over the mean traces). The results of the attack, which are shown in Fig. 11, are practical evidence of the success of the correlation collision attack using around $3,500,000$ traces countering unrolling as a countermeasure. Although the required number of traces is comparably higher than in previous cases, since the correlation collision attack employs the mean traces, increasing the number of traces helps removing the switching noise effect and finally recovers the relation between the key bytes.

VI. Conclusions

The results of correlation collision attacks on different hardware countermeasures have been presented. We have chosen this attack scheme for our investigations since no hypothetical power model is required and its efficiency does not rely on the leakage model of the target device which allows for a fair comparison. It is not a surprise that each countermeasure alone is not able to overcome the vulnerability against the attacks since even (theoretically) perfectly masked implementations still contain a slight first order leakage in practice due to glitches in the circuit. Similarly time randomization or noise addition countermeasures, which diminish the SNR, can be overcome by increasing the number of measurements.

However, when different countermeasures are combined, it is possible to significantly strengthen the resistance against DPA attacks. Applying all implemented countermeasures of the 32-bit architecture, the number of necessary traces for a successful correlation collision attack can be increased from $10,000$ to $1,500,000$. If the area constrain is not an issue (e.g. unused FPGA resources) unrolling can further increase the resistance. Increasing the number of rounds per clock cycle from one to four increases the number of required traces from $100,000$ to $3,500,000$. The combination of unrolling and masking has not been considered because of the rather large and impractical area requirements.

Since our implementation platform has been specifically designed for side-channel purposes considering an appropriate measurement setup and a well-defined trigger point, in real-world scenarios especially when the crypto cores are not the only circuits computing at one instance of time even more measurements will expectedly be required. However, knowing the number of required traces for an attack on an implementation in a low-noise environment helps choosing appropriate key lifetimes to further protect the secrets.

Acknowledgment

The authors would like to thank Akashi Satoh and RCIS for the prompt and kind help in obtaining SASEBOs.

References

[1] Side-channel Attack Standard Evaluation Board (SASEBO). Further information are available via http://www.rcis.aist.go.jp/special/SASEBO/index-en.html.

[2] S. Bhasin, S. Guilley, L. Sauvage, and J.-L. Danger. Unrolling Cryptographic Circuits: A Simple Countermeasure Against Side-Channel Attacks. In *CT-RSA 2010*, volume 5985 of *LNCS*, pages 195–207. Springer, 2010.

[3] A. Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In *CHES 2008*, volume 5154 of *LNCS*, pages 30–44. Springer, 2008.

[4] D. Canright. A Very Compact S-Box for AES. In *CHES 2005*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005.

[5] D. Canright and L. Batina. A Very Compact "Perfectly Masked" S-Box for AES. In *ACNS 2008*, volume 5037 of *LNCS*, pages 446–459. Springer, 2008. the corrected version is available at Cryptology ePrint Archive, Report 2009/011 http://eprint.iacr.org/2009/011.

[6] C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In *CHES 2000*, volume 1965 of *LNCS*, pages 13–48. Springer, 2000.

[7] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In *CRYPTO 2008*, volume 5157 of *LNCS*, pages 203–220. Springer, 2008.

[8] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In *CHES 2008*, volume 5154 of *LNCS*, pages 426–442. Springer, 2008.

[9] C. Herbst, E. Oswald, and S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In *ACNS 2006*, volume 3989 of *LNCS*, pages 239–252. Springer, 2006.

[10] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[11] S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In *CHES 2005*, volume 3659 of *LNCS*, pages 157–171. Springer, 2005.

[12] A. Moradi, O. Mischke, and T. Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *CHES 2010*, volume 6225 of *LNCS*, pages 125–139. Springer, 2010. The extended version is available on ePrint http://eprint.iacr.org/2010/297.

[13] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In *FSE 2005*, volume 3557 of *LNCS*, pages 413–423. Springer, 2005.

[14] C. Paar. *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*. PhD thesis, Institute for Experimental Mathematics, University of Essen, Germany, 1994.

[15] K. Schramm, T. Wollinger, and C. Paar. A New Class of Collision Attacks and Its Application to DES. In *FSE 2003*, volume 2887 of *LNCS*, pages 206 – 222. Springer, 2003.

[16] D. Suzuki, M. Saeki, and T. Ichikawa. DPA Leakage Models for CMOS Logic Circuits. In *CHES 2005*, volume 3659 of *LNCS*, pages 366–382. Springer, 2005.

[17] K. Tiri, M. Akmal, and I. Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *European Solid-State Circuits Conference - ESSCIRC 2002*, pages 403–406, 2002.

Appendix

Fig. 12 and Fig. 13 show sample power traces and variance traces computed over the mean traces of different profiles of the 32-bit and unrolled architectures respectively.

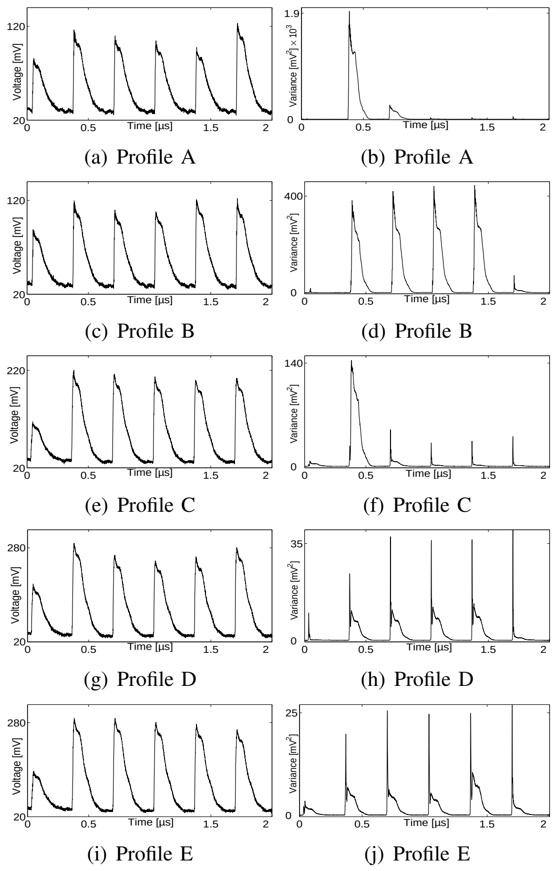

Fig. 12. 32-bit architecture: (left) sample power traces, (right) variance traces of different profiles

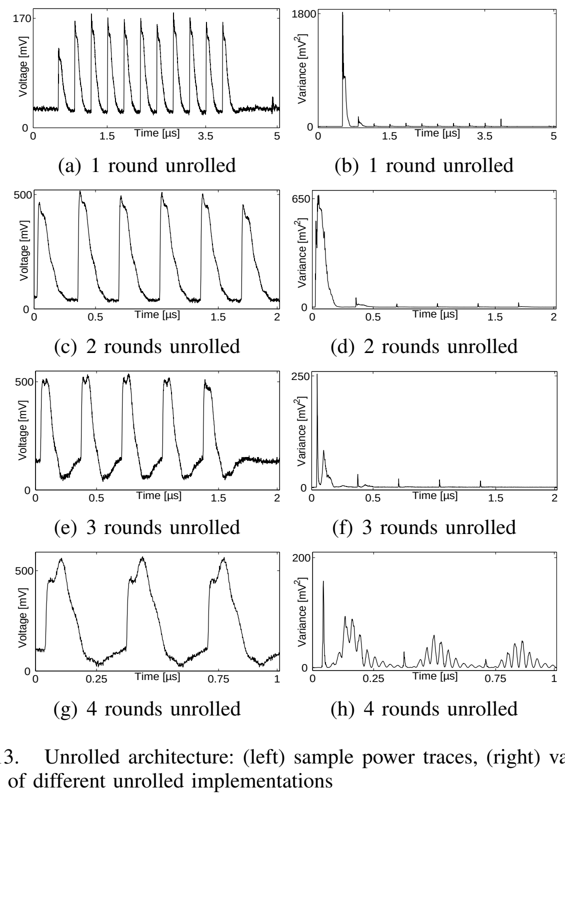

Fig. 13. Unrolled architecture: (left) sample power traces, (right) variance traces of different unrolled implementations