On the Power of Fault Sensitivity Analysis and Collision Side-Channel Attacks in a Combined Setting

Amir Moradi[1], Oliver Mischke[1], Christof Paar[1],
Yang Li[2], Kazuo Ohta[2], and Kazuo Sakiyama[2]

[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{moradi,mischke,cpaar}@crypto.rub.de
[2] Department of Informatics, The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
{liyang,ota,saki}@ice.uec.ac.jp

Abstract. At CHES 2010 two powerful new attacks were presented, namely the Fault Sensitivity Analysis and the Correlation Collision Attack. This paper shows how these ideas can be combined to create even stronger attacks. Two solutions are presented; both extract leakage information by the fault sensitivity analysis method while each one applies a slightly different collision attack to deduce the secret information without the need of any hypothetical leakage model. Having a similar fault injection method, one attack utilizes the non-uniform distribution of faulty ciphertext bytes while the other one exploits the data-dependent timing characteristics of the target combination circuit. The results when attacking several AES ASIC cores of the SASEBO LSI chips in different process technologies are presented. Successfully breaking the cores protected against DPA attacks using either gate-level countermeasures or logic styles indicates the strength of the attacks.

1 Introduction

Since the last decade designers of cryptographic devices have to deal with the problem that embedded secret information, e.g., the used encryption key in a symmetric cipher, is leaking through physical side channels. The side-channel leakage can be the timing [11], the power consumption [12], the electro-magnetic radiation [8, 21] and so on. Also fault analysis attacks such as differential fault analysis (DFA) was demonstrated to be effective against implementations of block ciphers such as DES [5] and AES [20].

At CHES 2010 a new fault attack called Fault Sensitivity Analysis (FSA) [14] was proposed. The authors used fault injections by means of clock glitches to measure the calculation time of the S-boxes as side-channel leakage. This information, whose data dependency is caused by the underlying gates, is then used to recover the secret information. Using this attack the secret key of an AES ASIC implementation could be completely extracted, but it was stated

that masking might be a countermeasure against such a kind of attack since the randomization of the S-box input makes it difficult to repeat fault injections and to measure the timings of specific calculations.

Another contribution to CHES 2010 was a collision attack enhanced by correlation [15]. Compared to classical power analysis attacks, its main feature is that it does not rely on the knowledge of an underlying (hypothetical) power model. Instead, it directly correlates power traces to each other and – by finding colliding S-box computations – is able to recover the relation between key parts. Using such an attack a complete break of a masked FPGA implementation of AES has been demonstrated.

The combination and improvement of these two ideas is the main contribution of this article. From [14] we will use the fault injection method and utilize the fact that the timing characteristics of each S-box can be independently observed, while the correlation-collision approach from [15] is used to find the relations between key bytes without the need to have any knowledge how the observed characteristics relate to the inputs. Two options to combine these two schemes are presented:

- First we present an attack which exploits the finding that given a fixed clock glitch period and a fixed unmasked S-box input byte the distribution of the resulting faulty ciphertext byte will be data dependent. This is achieved by setting the period of the clock glitch so that about 50% of the executions lead to faulty values. Repeating this measurements for all possible differences between two targeted ciphertext bytes, one can identify the correct key difference if the distributions of two faulty ciphertext bytes collide.
- In the second attack we take a closer look at the fault rate of each S-box instance over a large range of clock glitch periods, thereby getting very detailed information about the timing characteristics of the targets. Again using the concept of the correlation collision attack, several collisions between these timing sets can be detected at the same time. This allows us to fully recover all relations between the key bytes, i.e., shrinking the key space to 2^8.

Similarly to [14], we have chosen the SASEBO-R [2] board as the evaluation platform. The board can hold different ASICs, and we have analyzed the SASEBO LSI2 [3], both in 130nm and 90nm technology, as well as the SASEBO LSI3 [4] in 65nm technology. Each of the LSIs contain the same 14 different implementations of AES, therefore the only difference is the process technology, which has a big influence on the timing characteristics. The implementations themselves differ in the style of the S-box realization and in side-channel countermeasures. Using the attacks presented in this paper we will provide detailed results showing the successful key recovery for a large number of cores including the ones applying gate-level DPA countermeasures and DPA-resistant logic styles.

In the later parts of this article the prerequisites, including a review of fault sensitivity analysis and the correlation collision attack, are given in Section 2. Our first proposed attack using collisions of faulty ciphertext distributions and its results on two SASEBO LSI2 cores are presented in Section 3. The second

proposed attack, namely the collision timing attack, is expressed in Section 4, which also contains the practical evaluation results on several of the 130nm, 90nm, and 65nm SASEBO LSI2 and LSI3 cores. Finally, Section 5 concludes the paper.

2 Preliminaries

This section summarizes the underlying attacks, namely the fault sensitivity analysis [14] and correlation collision attack [15], which are the basis to the attacks presented here. We also address how these two methods can be combined to develop more sophisticated attacks improving their efficiency and relaxing the requirements. The experimental setup used in all the practical results shown in this work are also introduced in this section.

2.1 Fault Sensitivity Analysis

A new type of fault attack called Fault Sensitivity Analysis (FSA) was proposed in [14]. Unlike some of the previous fault attacks, e.g., DFA [5], the FSA attack does not require the value of the faulty outputs in the key recovery process. Instead, the attack works by increasing the fault intensity until a distinguishable characteristic can be observed, e.g., the first appearance of a faulty output. The concept of the FSA attack was verified by attacking the unprotected AES_PPRM1 core of the SASEBO LSI2 [3] using only 50 different plaintexts[3]. By successfully revealing three key bytes of the AES_WDDL implementation of the same ASIC using 1200 plaintexts it was also shown that the FSA attack can bypass some known countermeasures against DFA attacks.

The presented method of increasing the fault sensitivity in [14] is the shortening of the clock glitch, whereby the glitch period can be gradually decreased until a faulty output occurs or the fault becomes stable. Since the critical path of some gates, e.g., AND and OR gates, is data dependent, knowing the underlying model for this data dependency helps revealing the secret. For example, the simulation results ascertained that the timing delay of a PPRM S-box correlates to the Hamming weight (HW) of its input. For AES_WDDL, which in theory should be immune against set-up time violation attacks, by profiling with a known key it was shown that at least some bits correlate to the timing delay which lead to the aforementioned recovery of three key bytes. Moreover, this issue has been carefully studied later in [13].

In addition to the fact that FSA does not require faulty ciphertexts, another difference to DFA attacks is that the fault does not need to be restricted to a small sub-space. In contrary, by for example attacking the last round of the AES_PPRM1 implementation, each faulty output byte can be independently observed and therefore the same complete faulty output can be used to attack

[3] The fault sensitivity leakage for each plaintext has been recovered using around 200 faulty executions.

all key bytes simultaneously. On the other hand, as stated in [14], while countermeasures like masking are only of limited use against DFA attacks, they may have a great impact on FSA attacks since the critical path is affected by the random mask bits. Indeed, this is an issue which we demonstrate in this paper to be incorrect by providing the result of successful attacks on the masked implementations.

2.2 Correlation Collision Attack

The correlation collision attack was introduced in [15]. Its major advantage compared to classical power analysis attacks is that it neither relies on a hypothetical power model nor requires a profiling phase. Enhancing linear collision attacks [6] by the methods of correlation-based DPAs, it is able to overcome side-channel countermeasures as long as a minimal first order leakage remains.

A linear collision occurs if two instances of combinational circuits or one instance at two different points in time process(es) the same value, i.e., for AES two 8-bit outputs and thereby the inputs of the S-boxes must be the same. It is therefore possible to recover the key relation between the attacked bytes since rearranging of $\mathrm{Sbox}(i_1 \oplus k_1) = \mathrm{Sbox}(i_2 \oplus k_2)$ leads to $\Delta = i_1 \oplus i_2 = k_1 \oplus k_2$, where both i_1 and i_2 are known.

The correlation collision attack on AES works similarly, but starts by computing sets of mean traces for each possible input byte in case the attack is performed on the first round. To do this for two input bytes, namely i_1 and i_2, all traces are sorted based on the corresponding input byte value, and traces with the same value are averaged, thereby creating 256 different mean traces $M_1(i_1)$ and $M_2(i_2)$ for each of the two input bytes. Computing the variances for each set of mean traces will reveal the point in time where the corresponding bytes are processed by the S-box, which is necessary to align the mean traces for the attack.

If the power consumption of two S-box computations are highly similar, comparing pairs of mean sets also shows a high similarity between certain mean traces. Therefore, when attacking the input bytes i_1 and i_2 and $\Delta = k_1 \oplus k_2$, then $M_1(i_1) \approx M_2(i_2 = i_1 \oplus \Delta)$. The correct Δ can be found by computing the correlation between the two sets of mean traces for each of the 256 candidates of Δ. This yields a very high correlation coefficient since no hypothetical model is applied but instead the averaged real power consumptions are used and only a low number of points contributes to the estimation of the correlation coefficient.

2.3 Combinations

In order to avoid the need of a hypothetical model matching the fault sensitivity leakages, the two above attacks can be combined in several different ways. Two options for such a combination are expressed in this article. It should be noted that each of these two options has been independently developed by each group of the authors, and both have been submitted in parallel to CHES 2011. These two works have been merged as requested by the program committee.

The first option, which is developed by the team of the University of Electro-Communications (Japan), is expressed in Section 3. It extracts the distribution of the faulty ciphertext bytes and tries to find the collision within the distributions to recover the linear difference between the corresponding key bytes. The feasibility of this attack is practically confirmed by breaking two masked AES cores. The second option is developed by the team of the Ruhr University of Bochum (Germany) and is illustrated in Section 4. It extracts the precise timing characteristics of combinational circuits, e.g., S-boxes, and applies the correlation collision attack on timings to detect the colliding cases which reveal, similar to the first option, the linear difference between the corresponding key bytes. The shown practical results of this attack on several different AES cores in different process technologies highlight the strength of this attack.

2.4 Experimental Setup

All the practical results shown in this work are based on the AES implementations of three ASIC chips built for the SASEBO-R board, namely the SASEBO LSI2 (130nm), LSI2 (90nm), and LSI3 (65nm). Each chip contains the same 14 different AES cores including unprotected, DPA protected, and fault attack protected ones. The similar approach for fault injection as in [14] is used to inject the faults or extract the timing characteristics of the target circuit. An additional external clock, generated by an programmable digital function generator, is fed into the SASEBO-R control FPGA where it is multiplied using a Digital Clock Management (DCM) unit. This fast clock signal is then used together with some logic to shape the glitchy clock signal. An internal circuit controls the clock signal of the LSI to infer the glitchy clock at the preferred instance of time synchronized to the AES computation of the target core.

We have first tried to generate the glitchy clock inside the control FPGA without using an external function generator, but the width of the glitchy clock could only be adjusted in large steps (e.g., of around 170ps [7]), which were not small enough to reach the desired results. Therefore, we had to use a function generator to externally provide the precise clock frequencies. As it is represented in the following, we change the width of the glitchy clock in steps of 25ps to 5ps. Also, the multiplication of the clock frequency is necessary because of the limitation (maximum frequency of 15 MHz) of the function generator we have used, while the frequencies necessary to inject a fault in the combinational circuit are up to the range of 300MHz. Also, the DCMs inside the Virtex-II control FPGA of the SASEBO-R can, when fed with a low frequency input signal, only generate output frequency up to 210 MHz. Since some of the cores, especially of the 65nm LSI3, require a higher frequency for fault injection, for these cores it was necessary to daisy chain two DCMs, one for generating a high frequency signal out of the function generator output and another one to reach the maximum supported output frequency which can only be generated by the DCM using a high frequency input [26].

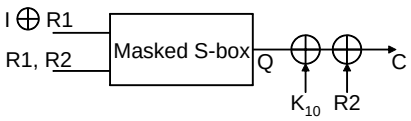

Fig. 1. A combinational circuit in the final round of a masked AES

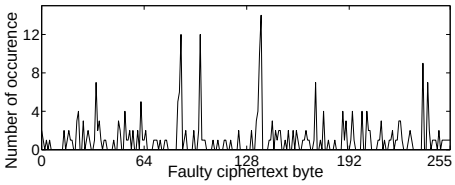

Fig. 2. A faulty ciphertext byte distribution

3 Option 1: Colliding Faulty Ciphertext Distributions

For the implementation with masking countermeasures, attackers cannot keep the device repeating the same calculation due to the randomization of the internal calculations in each trial. Thus, the fault sensitivity is difficult to be measured for a specific calculation, e.g., an S-box calculation with a fixed unmasked input and masks.

As shown in [15], the statistical observation of the side-channel leakage of a masked implementation may recover some sensitive information, e.g., the unmasked inputs, when there is still a first order leakage. We note that when faults are injected, the faulty ciphertexts can be used as an information source in the context of attacking the masked implementation. This section shows that the faulty ciphertext distribution is data dependent and can be used to detect the collision between unmasked intermediate values. This attack has been verified by successfully attacking two AES implementations masked using the Masked-AND gates and a form of threshold implementation.

3.1 Model and Attack Concept

As shown in Fig. 1, we make a simple model of a combinational circuit in the final AES encryption round of a masked implementation. A masked intermediate result $I \oplus R1$ goes through the substitution in a masked S-box calculation, the addition with the final round key K_{10} and the unmasking procedure (XOR with $R2$) to become a ciphertext byte C. In Fig. 1, Q denotes the output of the masked S-box. Hereafter, we use Q' and C' to denote the faulty masked S-box output and the faulty ciphertext byte, respectively. The attack procedure can be divided in two steps consisting of i) classifying the random numbers, and ii) detecting the colliding unmasked S-box inputs.

Classifying the random numbers A general security requirement for a masking countermeasure is the uniform distribution of the used random numbers, while we use a variation of fault sensitivity to classify the used random numbers.

Given a plaintext, the value of I in Fig. 1 is fixed. For all the possible random numbers, the calculations in the S-box circuit are different, and more importantly the critical delay timings are different. Attackers can focus on a specific S-box

calculation and trigger the setup-time violation in the final AES round. The fault injection intensity can be adjusted by modifying the period of the clock glitch.

In our attack, the clock glitch is set to a level where about 50% of the executions generate a faulty output. In this case, the executions are divided into two groups according to whether or not the output is faulty. Furthermore, these two groups of executions are corresponding to two groups of random numbers whose corresponding intermediate values, as we see later, are not uniformly distributed.

Detecting the colliding unmasked S-box inputs After using the fault sensitivity as a leakage to classify the random intermediate values which are non-uniformly distributed, the next step is to find another information source to effectively identify the sensitive intermediate value. According to Fig. 1, since both K_{10} and $R2$ are the inputs of the XOR gates, these part of the circuit can be seen as a set of fixed (per clock cycle) Buffers and Inverters. According to the architecture of our experimental setup, the round key K_{10} and $R2$ get available at the start of the corresponding clock cycle (last encryption round). Therefore, the computation of the masked S-box (Q), which is definitely longer than two following XORs, is interrupted by the clock glitch. So, we can assume that the faulty ciphertext C' is calculated as

$$C' = Q' \oplus K_{10} \oplus R2. \tag{1}$$

If we suppose that the faulty S-box output Q' has a fixed non-uniform distribution when I is fixed and the fault injection intensity is fixed at 50% success rate, those values of $R2$ which are corresponding to the 50% faulty executions should follow a fixed non-uniform distribution. On the other hand, the values of Q' and $R2$ are not independent of each other. As a result, we expect that the value of $Q' \oplus R2$ follows a fixed non-uniform distribution corresponding to the value of the fixed I. At last, the distribution of C' is permuted based on the value of K_{10}. An example of the distribution of C' is shown in Fig. 2.

The main idea of the attack is to check the similarity between two faulty ciphertext distributions, e.g., of two ciphertext bytes, each of which corresponds to a masked S-box followed by a fixed key addition, i.e., in the last round of the AES encryption. According to the linear collision in AES [6], the difference between key bytes equals to the difference between the corresponding fault-free ciphertext bytes when such a collision occurs.

3.2 Attack Scheme

The attack target is the linear difference ΔK between two bytes of K_{10}, i.e., $\Delta K = K_{10}^i \oplus K_{10}^j$, where $i, j = 1, 2, \cdots, 16$. After guessing the value of ΔK as ΔK_g, the attacker provides one plaintext so that the corresponding ciphertext satisfies $C^i \oplus C^j = \Delta K_g$. Therefore, if the current key difference guess is correct, the unmasked input I of the corresponding masked S-boxes collide. Such a case can be detected by examining the similarity of the distributions of the

Algorithm 1 Collecting the distribution of the faulty ciphertext byte

Inputs: A plaintext P, number of executions N, position of the target: j
Outputs: The count of all the faulty ciphertext byte: $Cnt(i), i = 0, 1, 2 \dots 255$.
Set the plaintext as P, $Cnt(i) \leftarrow 0$ for $i = 0, 1, 2 \dots 255$
Obtain the fault-free ciphertext byte C^j by running the fault-free encryption on P
for $i = 1$ to N **do**
    Obtain the j-th byte of the output C'^j by running the faulty encryption on P
    if $C'^j \neq C^j$ **then**
        $Cnt(C'^j) \leftarrow Cnt(C'^j) + 1$
    end if
end for

corresponding faulty ciphertext bytes. The distributions of the faulty ciphertext byte for the targeted S-boxes can be collected using Algorithm 1.

Given two distributions of the faulty ciphertext bytes Cnt^i and Cnt^j, one can use the current $\Delta K_g = C^i \oplus C^j$ to rearrange Cnt^j as

$$Cnt'^j(i \oplus \Delta K_g) = Cnt^j(i),\ i = 0, 1, 2 \dots 255,$$

and check the similarity using the correlation coefficient as $\rho(Cnt^i, Cnt'^j)$, where ρ denotes the calculation of the Pearson product-moment correlation coefficient. Repeating this procedure for all possible key differences, the ΔK_g corresponding to the largest correlation coefficient is expected to be the correct key difference. For clarification we have provided a pseudo code of the attack shown by Algorithm 2.

3.3 Practical Results

Results on AES_MAO The first attack target is the AES core protected against power analysis attacks using the masked-AND gates [24] in 130nm technology. In our experiments, the total number of executions to obtain the faulty ciphertext byte distribution is set to $N = 400$.[4] In order to cover all possi-

[4] We should mention that N is the total number of executions including faulty and fault-free ones.

Algorithm 2 Attack algorithm (colliding faulty ciphertext distributions)

Inputs: Position of the target key bytes: i and j
Output: Most probable key difference $\Delta K = K^i \oplus K^j$
for $\Delta K = 0$ to 255 **do**
    Select randomly plaintext P so that ciphertext bytes $C^i \oplus C^j = \Delta K$
    Obtain faulty ciphertext distributions Cnt^i and Cnt^j using Algorithm 1
    Set Cnt'^j as the rearranged form of Cnt^j based on ΔK
    $Cor(\Delta K) = \rho(Cnt^i, Cnt'^j)$
end for
return $\arg\max_{\Delta K} Cor(\Delta K)$

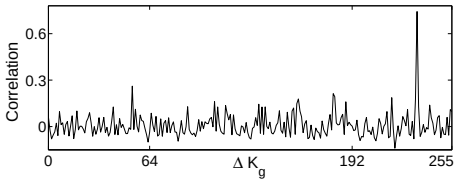

Fig. 3. Correlation vs. Key byte difference for AES_MAO (130nm).

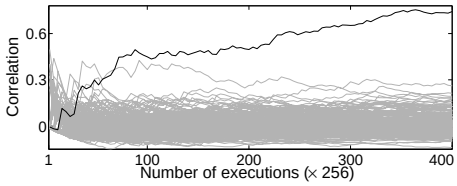

Fig. 4. Correlation evolution vs. Number of executions for AES_MAO (130nm).

ble key byte differences, we collected the distributions for 256 plaintexts which correspond to 256 differences between the first two ciphertext bytes. Running the attack algorithm, which is given by Algorithm 2, led to the results shown in Fig. 3 where the correct key byte difference can be clearly identified. Furthermore, Fig. 4 shows that 150×256 executions of AES_MAO is sufficient to identify the correct key byte difference. The successful attack experiments have been also confirmed recovering the difference between other key bytes.

Results on AES_TI The next target is the AES core in 130nm technology realized using the threshold implementation scheme which is a high-order masking countermeasure based on secret sharing [18]. Even in the presence of signal glitches, its resistance against power analysis attacks has been theoretically proven [19]. We should emphasize that the threshold implementation is an algorithmic-level countermeasure which needs to fulfill certain properties including correctness, non-completeness, and uniformity. In contrary to [17], our targeted AES_TI core has been made without considering the later two properties. This core has been realized by modifying a plain AES core at the gate level. The non-linear gates are provided by only 2-input AND gates, every signal is represented by four shares, and finally the AND gates are replaced with the 4-shared threshold implementation of a 2-input AND gate which is available in [18]. This can be verified by examining the source code of this core available at [1].

The attack procedure is the same as the one applied to the AES_MAO core even with the same number of executions, i.e., 400, to obtain the distribution of the faulty ciphertext bytes. The attack result on the key byte difference between the first two key bytes is shown in Fig. 5. The peak corresponding to the correct key byte difference can be clearly identified. Figure 6 also shows that at around 200×256 executions are required to identify the correct key byte difference.

3.4 Observations

Relaxing fault requirements One important observation from experiments is that the setting of the fault injection success rate is not as strict as we expected. In our experiments, we could collect the distributions of faulty ciphertext bytes simultaneously for parallel S-boxes. Due to the difference between the inherent

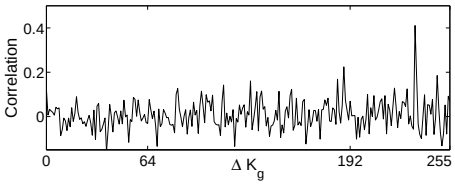

Fig. 5. Correlation vs. Key byte difference for AES_TI (130nm).

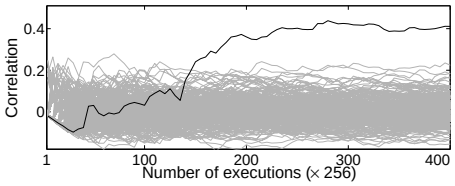

Fig. 6. Correlation evolution vs. Number of executions for AES_TI (130nm).

delay of parallel S-boxes, the fault injection success rates were different from 40% to 60%. Surprisingly, the key difference can still be clearly recovered. In other words, we found that the distribution of the faulty ciphertext byte is not very sensitive to the fault injection intensity.

As a result, compared to the one-byte fault injection in the DFA attacks or the accurate intensity management in the FSA attack, our proposed attack has the fewest requirements about the fault injection.

Reducing the number of executions In our experiments, we have collected the distributions for 256 ciphertexts to identify a linear difference between two key bytes. The attack efficiency regarding to the number of executions of AES can be easily improved by specifically selecting ciphertexts (plaintexts). For example, one can collect 16 distributions for the first fault-free ciphertext byte as 0x00, 0x01 ... 0x0F and 16 distributions for the second fault-free ciphertext byte as 0x00, 0x10 ... 0xF0, respectively. The combinations of these two groups of 16 distributions already cover all the possible linear key differences. Therefore, with a delicate selection of the ciphertexts (plaintexts), collecting the distributions for the 16 selected ciphertexts are enough to identify a key byte difference.

4 Option 2: Colliding Timing Characteristics

This section expresses the second combination of fault sensitivity analysis and correlation collision attack where the timing characteristics of combinational circuits like S-boxes are analyzed. In the following the fundamental concepts which are essential for the attack are explained, and later practical results of the attack breaking a couple of ASIC AES cores are presented.

4.1 How to Measure the Timing

As explained in [14], when the input of a combinational circuit changes, its output stops toggling after a certain time (so-called Δt). The maximum value of Δt for different inputs is known as the longest critical path of the circuit, and defines the maximum frequency of the clock signal which triggers the flip-flops providing the input and storing the output of the considered combinational

circuit. Timing characteristics of a circuit are therefore defined as a set of Δt ($\{\Delta t^1, \Delta t^2, \ldots, \Delta t^n\}$), where Δt^i is the minimum Δt for the given input i.

Let us suppose that the target combinational circuit is a part of a bigger circuit, e.g., a co-processor, which provides some I/O signals for communication. If the output of the target combinational circuit is stored into registers which are triggered by a clock signal that can be controlled from the outside, as shown in [14], one can steadily shorten the time interval between the input transition and the output storage (known as *setup time*) till an incorrect value is stored into the registers while input i is given to the combinational circuit. The minimum time interval when the considered register stores the correct value can be concluded to Δt^i. Note that this procedure is similar to the scheme explained in Section 3. However, measuring Δt in this case does not deal with the faulty outputs; once a faulty output is detected, Δt^i can be concluded.

It should be noted that, because of the environmental noise, it might be required to repeat the same procedure and shorten the clock glitch period until the probability of detecting faulty output gets higher than a threshold. Also, if the target combinational circuit is not a single-bit function and it is possible to detect which output bit is faulty, one can measure Δt^i for each output bit independently.

Therefore, we define the adversary model and define his capabilities in order to be able to measure Δt^i of the target combinational circuit for the given input i:

- The adversary has access to and can control the clock signal which triggers the registers providing the input and saving the output of the target combinational circuit.
- He knows in which clock cycle the target combinational circuit processes the desired data, e.g., known or guessed input or output.
- He can control the target device in a way that the same input value i is repeatedly processed by the target combinational circuit during shortening the time interval of the clock glitch.
- He is equipped with appropriate instruments to shorten the duration of the clock glitch with suitable accuracy.

4.2 Definitions

Bitwise Capture: $\mathrm{BitCap}^i_{b,\Delta t}$ is the result of a *Bernoulli trial* whether the output of the target combinational circuit at bit b is faulty while processing the input i and when Δt is the time interval of the clock glitch. Correspondingly, $p^i_{b,\Delta t}$ is defined as the probability of "success" in independently repeated $\mathrm{BitCap}^i_{b,\Delta t}$ trials.

Capture: $\mathrm{Cap}^i_{\Delta t} = \bigvee_b \mathrm{BitCap}^i_{b,\Delta t}$. In other words, $\mathrm{Cap}^i_{\Delta t}$ is the same as the above defined trial regardless of a certain output bit, and is meaningful when differentiating between different faulty output bits is not possible, e.g., if a circuit is equipped with a fault detection scheme and prevents the propagation of faulty results. $p^i_{\Delta t}$ is also the probability of "success" in independently repeated

$\text{Cap}^i_{\Delta t}$ trials.
Time: To represent the timing characteristics of the target combinational circuit, we define $T^i_b = \Delta t;\ p^i_{b,\Delta t} \approx p_{TH}$ as the time required to compute the corresponding output bit b when input i is given, where p_{TH} is a threshold for the probability and is defined based on physical characteristics of the target circuit and is also based on the maximum probability achieved by shortening Δt. Accordingly, the time required to complete the computation of all bits when processing input i is defined as $T^i = \Delta t;\ p^i_{\Delta t} \approx p_{TH}$.
Remark: Depending on the target device, its architecture, and the role of the target combinational circuit inside the target device, it might not be possible to know the input i processed. However, if the output of the target combinational circuit is accessible, one can make all the above defined terms based on the fault-free output o, i.e., $\text{BitCap}^o_{b,\Delta t}$, $p^o_{b,\Delta t}$, $\text{Cap}^o_{\Delta t}$, $p^o_{\Delta t}$, T^o_b, and T^o.

4.3 Attack Scheme

For simplicity let us suppose that the target combinational circuit is an S-box of the first round of an AES encryption, i.e., Sbox(i $\oplus$ k), where i is the corresponding input plaintext byte and k the target key byte.

If (bitwise) timing characteristics of an S-box, i.e., T^i (T^i_b), show a diversity of Δt depending on input i, one can perform an attack and recover the secret knowing how the secret k contributes in T^i (T^i_b). In other words, if the timing characteristics of an S-box itself regardless of k and prior key addition ($\oplus$) are known as an extra information or are obtained by profiling using a circuit similar to the target, one can make a hypothetical leakage function and examine its similarity to T^i (T^i_b) for each key guess. A similar approach has been presented in [14], where the timing characteristics of an AES S-box implementation were profiled and an attack similar to a correlation power analysis using a HW model was successfully performed. In fact, a set of $\text{Cap}^o_{\Delta t}$ for a specific Δt is used in [14] to mount the attack at the last round of the AES encryption.

One may also try using information theoretic tools, e.g., mutual information analysis [9], to relax the leakage model. However, it is necessary to use a suitable leakage model that cannot be selected without extra knowledge about the (timing) characteristics of the target combinational function [25], or several different models must be examined to find a suitable one. It is noteworthy that the leakages ($\text{Cap}^i_{\Delta t}$) consist of only two values ("fail" and "success"). This causes probability distributions (used in e.g., mutual information analysis) to be represented by only two bins in a histogram, and using other schemes to estimate the probability distributions, e.g., kernel density estimation, in this case leads to increasing the noise. Here, when using histograms, mutual information will also be identical to the variance of means.

In contrast to a correlation attack or mutual information analysis using a leakage model, we apply a correlation collision attack [15] to avoid the necessity of considering any such model. Here the correlation collision attack compares the timing characteristics T^i (T^i_b) of two S-box instances running on two input sets, each of which is previously XORed by a secret key byte. Suppose that $T1^i$ and

Algorithm 3 Correlation Timing Attack (the last round of the AES encryption)

Input: $T1^o : \left(\Delta t^{o=0}, \Delta t^{o=1}, \ldots, \Delta t^{o=255}\right)$; $o = \text{Sbox}(i) \oplus k1$
Input: $T2^o : \left(\Delta t^{o=0}, \Delta t^{o=1}, \ldots, \Delta t^{o=255}\right)$; $o = \text{Sbox}(i) \oplus k2$
1: **for** $0 \leq \Delta \leq 255$ **do**
2: $\quad Cor(\Delta) = \text{Correlation}(T1^o, T2^{o \oplus \Delta})$
3: **end for**
4: **return** $\arg\max_{\Delta} Cor(\Delta)$

$T2^i$ (or their corresponding bitwise versions) are the timing characteristics of the S-box when processing $\text{Sbox}(i \oplus k1)$ and $\text{Sbox}(i \oplus k2)$ respectively. As stated in [15], the aim of a correlation collision attack is to find the linear difference between $k1$ and $k2$, i.e., $\Delta = k1 \oplus k2$.

This can be extended when attacking the last round of the AES encryption, thanks to the absence of the MixColumns in the last round. For example, suppose that $T1^o$ and $T2^o$ are the timing characteristics of the S-box followed by the key addition when calculating $o = \text{Sbox}(i) \oplus k1$ and $o = \text{Sbox}(i) \oplus k2$. Then, the correlation collision attack can – exactly as in the previous case – recover $\Delta = k1 \oplus k2$ comparing $T1^o$ and $T2^o$ for all possible guesses of Δ. For clarification of the attack scheme see Algorithm 3.

4.4 Practical Results

In all cores of the LSIs 16 instances of the S-box are implemented to perform the complete SubBytes operation in each clock cycle. According to [3, 4], all cores – except the one supporting a counter mode and the fault-protected one – realize a round based architecture, i.e., S-boxes and MixColumns are performed consecutively in each clock cycle except for the last round where MixColumns is absent. Therefore, extracting the timing characteristics of the S-boxes in the first 9 rounds is not easily possible. So, one needs to inject and play with the width of the clock glitches in the last round, when the target cores only compute the SubBytes operation followed by the final key addition and the result is stored in registers (similar scheme as used in [14]). In addition, one can see from the design architecture of the cores (see Fig. 5.1 of [3] and Fig. 13.1 of [4]) that the round key of the last round is already computed in the previous round and is stored into a register. The glitchy clock at the last round, hence, does not affect the key scheduling computations.

In the following the results of the attacks on the different cores and different LSIs are presented. Because of the high number of broken cores, only a subset of the performed attacks are presented in detail, giving additional information about the differences to the not mentioned cores as required.

Attacking the Unprotected Cores We start by showing the results of the attack on the first AES core of the $130n$m chip, namely AES_Comp, whose S-boxes have been made using a composite field approach. As stated before,

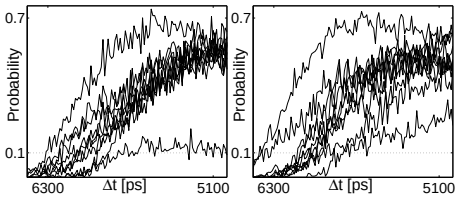

Fig. 7. First 10 $p^o_{b=0,\Delta t}$ curves for S-box instances no. (left) 0 and (right) 4 of AES_Comp (130nm)

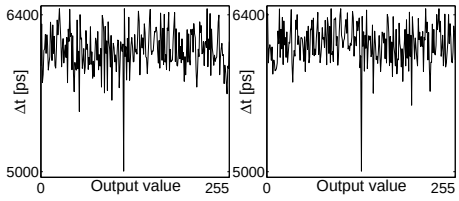

Fig. 8. Bitwise timing characteristics $T^o_{b=0}$ of S-box instances no. (left) 0 and (right) 4 of AES_Comp (130nm)

16 separate S-box instances have been implemented which are active at the same time. Therefore, it is not possible to compare the timing characteristics of one S-box instance when processing e.g., two values with different key bytes, that would be an ideal case for a collision timing attack. In contrast, the timing characteristics of different S-box instances must be compared, which may slightly vary because of different placement and routing even when being based on the same netlist.

Since changing the glitchy clock width in our setup requires reseting the DCM(s), we have collected $\text{BitCap}^o_{b,\Delta t}$ for a specific Δt while random plaintexts are given to the core. This was repeated shortening Δt by steps of 10ps and finally exploiting the bitwise timing characteristics T^o_b. Figure 7 shows $p^o_{b,\Delta t}$ of the LSB (i.e., $b = 0$) for some output byte values of two S-box instances[5] extracted from their corresponding bitwise captures (10 000 captures for each Δt). Also, Fig. 8 presents the bitwise timing characteristics $T^o_{b=0}$ of these two S-box instances obtained by defining $p_{TH} = 0.1$ (as can be seen in Fig. 7). The diversity of Δt for these two S-boxes shows the dependency between the timing characteristics and the output values. Performing the attack Algorithm 3 on $T^o_{b=0}$ of S-box instance number 0 and all other instances led to recovering all 15 independent relations between the 16 bytes of the last round key; part of the result is shown in Fig. 9. The attack works the same considering other output bits to derive T^o_b as well as on other LSI chips.

Carefully study of the timing characteristics shown in Fig. 8 revealed that Δt is much smaller than the other cases when the S-box input is zero, that is a known issue since the zero-input power model has been defined [10] to mount CPA attacks on AES S-box leakages. In fact, it is not needed to mount the collision timing attack in this case, and the key bytes can be recovered observing $T^o_{b=0}$ of each S-box instance separately. However, as it is shown later this property does not hold for the other cores realized by different S-boxes, and mounting our proposed attack is essential to reveal the secrets.

In order to perform the attack on the cores AES_PPRM1, AES_PPRM3, AES_Comp_ENC_top, and AES_PKG the same procedures as explained above

[5] S-box instance numbers start from 0 and are corresponding to ciphertext byte indexes.

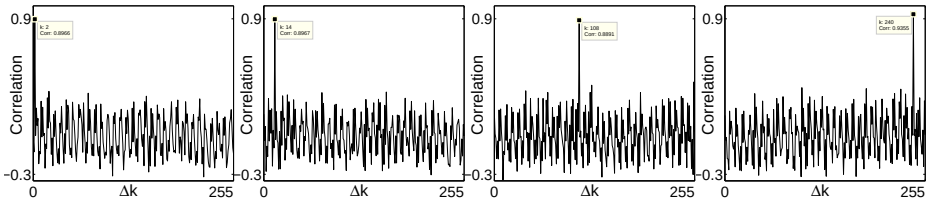

Fig. 9. Result of the attack on the last round of AES_Comp (130nm) recovering Δk between key bytes (from left to right) (0,1), (0,2), (0,3), and (0,4)

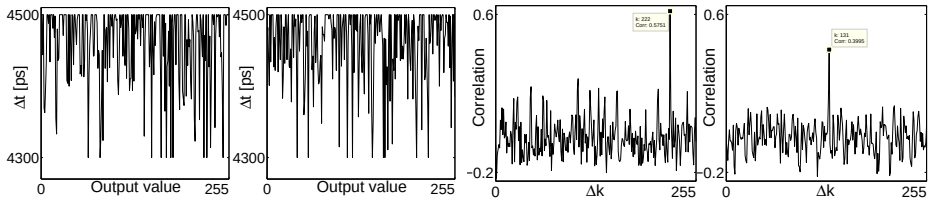

Fig. 10. Bitwise timing characteristics $T^o_{b=0}$ of S-box instances no. (left) 5 and (right) 6 of AES_MAO (65nm)

Fig. 11. Result of the attack on the last round of AES_MAO (65nm) recovering Δk between key bytes (left) (5,6), (right) (5,7)

have been repeated. As a reference for the timing characteristics and the number of captures collected to mount the attack on different cores in different LSIs, we have provided a list shown in Table 1. Attacking the AES_TBL core, where S-boxes have been realized by look-up tables (case statements), is different to the aforementioned cores. We illustrate this case when explaining how to mount the attack on the WDDL and MDPL cores.

Attacking the DPA-Protected Cores Most of the DPA-protected cores can be attacked in the same way as the unprotected ones. In Fig. 10 one can see that even when using the masked AND-gates of the AES_MAO (65nm) core, the timing characteristics for different outputs still differ. Consequently, it is possible to extract the relation between the key bytes, which is depicted in Fig. 11. Interestingly the randomness provided by the masked gates does not have much impact on the timing characteristics, as shown in Fig. 11, where the results after obtaining 10 000 captures (the same technique as used to attack the unprotected cores) while shortening Δt with steps of 25ps are presented.

Attacking the other DPA-protected cores is the same except on those realizing WDDL and MDPL logic styles. The result of the attack on the AES_WO core, which is implemented using an Pseudo-RSL [22] logic style, is shown in Fig. 12 and Fig. 13. Although we have used 10 000 captures for each Δt in steps of 10ps to attack the AES_WO core, attacking the AES_PR core (which is another realization of Pseudo-RSL) and the AES_TI (which has been discussed in Section 3.3) required considerably more captures. As stated in Table 1 we have

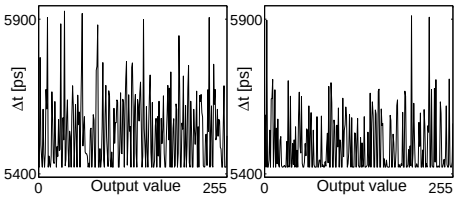

Fig. 12. Bitwise timing characteristics $T^o_{b=1}$ of S-box instances no. (left) 7 and (right) 8 of AES_WO (90nm)

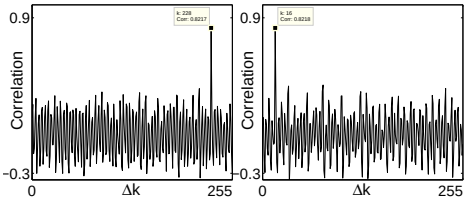

Fig. 13. Result of the attack on the last round of AES_WO (90nm) recovering Δk between key bytes (left) (7,8), (right) (7,9)

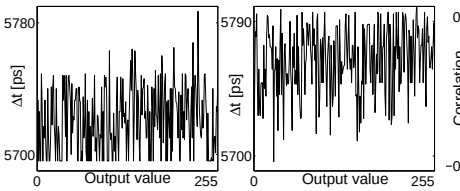

Fig. 14. Timing characteristics T^o of S-box instances no. (left) 2 and (right) 3 of AES_MDPL (65nm)

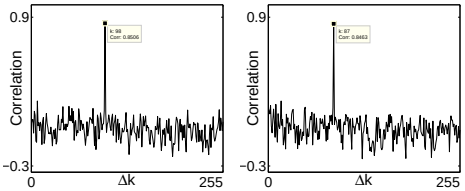

Fig. 15. Result of the attack on the last round of AES_MDPL (65nm) recovering Δk between key bytes (left) (2,3), (right) (2,6)

used 1 000 000 captures for each of these cores to successfully mount the attack. To the best of our knowledge it is due to the amount of randomness provided by the DPA countermeasures. For instance, third-order masking is used in the AES_TI core compared to first-order masking in AES_MAO.

The AES_WDDL and AES_MDPL cores require an slightly adjusted approach, since they need two clock cycles per round because of the used master-slave flip-flops, i.e., four flip-flops to store a single bit value. Also, an injected fault by a clock glitch at the evaluation phase can only lead to a bit flip from 1 to 0, not vice versa, because of the pre-discharge phase of both WDDL and MDPL styles. This issue has been also addressed in [13] where a successful attack is performed on an AES_WDDL core. Interestingly, we have seen – for reasons unknown to us – the same behavior when attacking the AES_TBL core. Therefore, bitwise timing characteristics T^o_b does not provide any information for those output values o in which bit b is zero. Our solution is to avoid using bitwise characteristics, and apply the attack on timing characteristics T^o, e.g., those shown in Fig. 14, which are of the AES_MDPL (65nm) core. Two attack results are also shown in Fig. 15. It should be noted that, to attack the AES_MDPL and AES_WDDL cores, we only used 10 000 captures for each Δt with steps of 5ps. On the other hand, a successful attack on the AES_TBL required around 1 000 000 captures, which might be because of marginal differences between the critical paths of the circuit realizing the look-up table.

We should emphasize that this attack has been successfully performed on all AES cores including one equipped by a fault attack countermeasure [23]. Because

of the page restriction the details of the attack on the other cores are left for the extended version which can be found in [16].

Difficulties In the following some of the difficulties that we experienced during our practical investigations are explained in detail.

- As stated in [3] and [4], each core has its own clock tree which had a strong impact on glitchy clocks. The capacitive and resistive features of the clock line of the LSIs, which is supplied by the control FPGA, changed the glitchy clock shape and modified the situation whether the registers are triggered two times, or if one of the positive edges is filtered by the clock tree elements. Therefore, we had to put different capacitors and resistors in the SASEBO-R board to change the rising and falling slopes of the clock signal to reach the desired situation.
- Since the temperature has an effect on the critical path and the speed of the ASICs, some values are given in Table 4.7 of [3], we not only kept the room temperature constant during capturing, but also kept the board and the LSIs in different temperatures while playing with capacitances and resistances to solve the problem mentioned above.
- Since DCM outputs have an increased jitter when they are used to multiply the clock inputs, the glitchy clock width also had significant jitter that made the capturing process noisy. The situation got worse when we had to cascade two DCMs for some cores, especially in $65n$m technology, to reach the desired Δt. In this case, the second DCM often could not get locked because of the high jitter of the first DCM. So, we had to provide another circuit controlled by the PC to automatically reset the control FPGA (in fact the DCMs) until the DCMs get locked and provide the requested high frequencies.
- Since we have required high amounts of captures, e.g., 1 000 000, for different Δt values to successfully mount the attacks, we have developed a special design for the control FPGA to speed up the capturing process. Our control FPGA communicates with the target LSI, makes the glitchy clock on the desired clock cycle, and finally after performing a couple of capturing process sends the result back to the PC. In this way we could efficiently increase the speed of the capturing up to couple of thousands per second.
- According to [3] and [4], the clock signal of the interface circuit of the LSIs is separated from the core clocks. So, the glitchy clock does not appear on the interface circuit which makes the attacks easier. It might be a challenging case when the interface circuit sees the glitches, and the control flow of the target core gets infected.

5 Conclusions

We have presented two collision attacks which utilize certain kinds of side-channel leakage which is made possible by the fault injection method of [14]. One is a major improvement of the attack idea of [14] since by applying techniques

of correlation-based DPAs to find collisions it does not require any knowledge about the characteristics of the target combinational circuit. The other one exploits a newly observed leakage which is the fact that given a fixed fault intensity the distribution of the resulting faulty ciphertext bytes is not completely random but data dependent.

It is indicated in [14] that while masking does not prevent DFA attacks, it may actually provide security against FSA-based attacks because of the randomized inputs of the combinational functions. However, by breaking all DPA-protected cores of the mentioned ASICs we have shown that randomizing countermeasures itself cannot prevent data-dependent timing of the combinational circuit, and they therefore remain vulnerable to the attacks introduced here.

Using the attack exploiting the faulty ciphertext byte distributions two DPA protected cores could be broken. Furthermore, using the attack focusing on the timing of the combinational circuits all SASEBO LSI2 and LSI3 cores could be broken, including the one applying an algorithmic fault detection scheme. In short, the results shown in this work imply the need for a special unit in the – especially side-channel protected – designs in order to detect the clock glitches to thwart such kind of attacks.

Acknowledgment

The authors would like to thank Akashi Satoh and the Research Center for Information Security (RCIS) of Japan for the prompt and kind help in obtaining SASEBOs and cryptographic LSIs. The authors of the Ruhr University of Bochum (Germany) have been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. The authors of the University of Electro-Communications (Japan) have been supported by the Strategic International Cooperative Program (Joint Research Type), Japan Science and Technology Agency.

References

1. Cryptographic Circuits with Logic Level Countermeasures against DPA. Information and Physical Security Research Group, YOKOHAMA National University `http://ipsr.ynu.ac.jp/circuit/`.
2. Side-channel Attack Standard Evaluation Board (SASEBO-R). Further information are available via `http://staff.aist.go.jp/akashi.satoh/SASEBO/en/board/sasebo-r.html`.
3. ISO/IEC 18033-3 Standard Cryptographic LSI – with Side Channel Attack Countermeasures – Specification, ver 1.0. `http://staff.aist.go.jp/akashi.satoh/SASEBO/resources/crypto_lsi/CryptoLSI2_Spec_Ver1.0_English.pdf`, 2009.
4. Standard Cryptographic LSI Specification – Countermeasures against Side Channel Attacks (65nm) – Specification, ver 0.9. `http://staff.aist.go.jp/akashi.satoh/SASEBO/resources/crypto_lsi/CryptoLSI3_Spec_Ver0.9_English.pdf`, 2010.

5. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *CRYPTO 1997*, volume 1294 of *LNCS*, pages 513–525. Springer, 1997.
6. A. Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In *CHES 2008*, volume 5154 of *LNCS*, pages 30–44. Springer, 2008.
7. S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. Satoh. An on-chip glitchy-clock generator and its application to safe-error attack. In *COSADE 2011*, pages 175–182, 2011.
8. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In *CHES 2001*, volume 2162 of *LNCS*, pages 251–261. Springer, 2001.
9. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In *CHES 2008*, volume 5154 of *LNCS*, pages 426–442. Springer, 2008.
10. J. D. Golic and C. Tymen. Multiplicative Masking and Power Analysis of AES. In *CHES 2002*, volume 2523 of *LNCS*, pages 198–212. Springer, 2002.
11. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO 1996*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
12. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
13. Y. Li, K. Ohta, and K. Sakiyama. Revisit Fault Sensitivity Analysis on WDDL-AES. In *HOST 2010*, pages 148–153. IEEE Computer Society, 2010.
14. Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta. Fault Sensitivity Analysis. In *CHES 2010*, volume 6225 of *LNCS*, pages 320–334. Springer, 2010.
15. 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 Archive, Report 2010/297. `http://eprint.iacr.org/`.
16. A. Moradi, O. Mischke, and C. Paar. Collision Timing Attack when Breaking 42 AES ASIC Cores. Cryptology ePrint Archive, Report 2011/162, 2011. `http://eprint.iacr.org/`.
17. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.
18. S. Nikova, C. Rechberger, and V. Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches. In *ICICS 2006*, volume 4307 of *LNCS*, pages 529–545. Springer, 2006.
19. S. Nikova, V. Rijmen, and M. Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. In *ICISC 2008*, volume 5461 of *LNCS*, pages 218–234. Springer, 2008.
20. G. Piret and J.-J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In *CHES 2003*, volume 2779 of *LNCS*, pages 77–88. Springer, 2003.
21. J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *E-smart 2001*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.
22. M. Saeki, D. Suzuki, K. Shimizu, and A. Satoh. A Design Methodology for a DPA-Resistant Cryptographic LSI with RSL Techniques. In *CHES 2009*, volume 5747 of *LNCS*, pages 189–204. Springer, 2009.
23. A. Satoh, T. Sugawara, N. Homma, and T. Aoki. High-Performance Concurrent Error Detection Scheme for AES Hardware. In *CHES 2008*, volume 5154 of *LNCS*, pages 100–112. Springer, 2008.

24. E. Trichina. Combinational Logic Design for AES SubByte Transformation on Masked Data. Cryptology ePrint Archive, Report 2003/236, 2003. http://eprint.iacr.org/.
25. N. Veyrat-Charvillon and F.-X. Standaert. Generic Side-Channel Distinguishers: Improvements and Limitations. In *CRYPTO 2011*, volume 6841 of *LNCS*, page 348. Springer, 2011. the extended version is available on ePrint Archive, Report 2011/149. http://eprint.iacr.org/.
26. XILINX. Virtex-II Pro and Virtex-II Pro X FPGA User Guide. Technical report, version 4.2, 2007. http://www.xilinx.com/support/documentation/user_guides/ug012.pdf.

Appendix

Table 1. Specification of AES cores of the three targeted LSIs including the Δt ranges and the number of captures used to mount the attacks

IP core	Description	LSI2 130nm		LSI2 90nm		LSI3 65nm	
		Δt range [ps]	No. of Captures	Δt range [ps]	No. of Captures	Δt range [ps]	No. of Captures
AES_Comp	composite field S-box	6450 Δ : 10 5000	10 000	5320 Δ : 10 5130	10 000	3650 Δ : 10 3370	10 000
AES_TBL	table look-up S-box by case statement	5475 Δ : 25 4900	1 000 000	3960 Δ : 20 3550	1 000 000	3570 Δ : 10 3420	1 000 000
AES_PPRM1	S-box by 1-stage AND-XOR	11350 Δ : 25 7775	10 000	6135 Δ : 20 5555	10 000	5325 Δ : 25 5000	10 000
AES_PPRM3	S-box by 3-stage AND-XOR	6425 Δ : 25 5150	10 000	5230 Δ : 10 5130	10 000	3650 Δ : 10 3420	10 000
AES_Comp ENC_top	composite field S-box, only encryption	6325 Δ : 25 5100	10 000	5200 Δ : 10 5130	10 000	3700 Δ : 10 3410	10 000
AES_PKG	composite field S-box, precomp. roundkeys	6325 Δ : 25 5100	10 000	5360 Δ : 10 5130	10 000	3850 Δ : 10 3370	10 000
AES_MAO	DPA count. by Masked And Operation	8475 Δ : 25 6250	10 000	5900 Δ : 5 5850	10 000	4500 Δ : 25 4300	10 000
AES_MDPL	DPA count. by MDPL logic style	12825 Δ : 25 10850	10 000	9350 Δ : 25 8050	10 000	5800 Δ : 5 5260	10 000
AES_TI	DPA count. by Threshold Implementation	10860 Δ : 20 9800	1 000 000	5900 Δ : 5 5850	1 000 000	6340 Δ : 20 5940	1 000 000
AES_WDDL	DPA count. by WDDL logic style	6750 Δ : 10 5730	10 000	5250 Δ : 5 5150	50 000	3835 Δ : 5 3675	10 000
AES_PR	DPA count. by pseudo RSL logic style	31685 Δ : 10 31055	1 000 000	14400 Δ : 20 13840	1 000 000	6650 Δ : 25 6150	1 000 000
AES_WO	DPA count. by pseudo RSL (evaluation)	7575 Δ : 25 6475	10 000	5910 Δ : 10 5430	10 000	3900 Δ : 25 3600	10 000