Correlation-Enhanced Power Analysis Collision Attack

Amir Moradi[1], Oliver Mischke[1], and Thomas Eisenbarth[2]

[1] Horst Görtz Institute for IT Security
Ruhr University Bochum, Germany
{moradi, mischke}@crypto.rub.de
[2] Department of Mathematical Sciences
Florida Atlantic University, Boca Raton, FL 33431, USA
teisenba@fau.edu

Abstract. Side-channel based collision attacks are a mostly disregarded alternative to DPA for analyzing unprotected implementations. The advent of strong countermeasures, such as masking, has made further research in collision attacks seemingly in vain. In this work, we show that the principles of collision attacks can be adapted to efficiently break some masked hardware implementation of the AES which still have first-order leakage. The proposed attack breaks an AES implementation based on the corrected version of the masked S-box of Canright and Batina presented at ACNS 2008 which is supposed to be resistant against first-order attacks. It requires only six times the number of traces necessary for breaking a comparable unprotected implementation. At the same time, the presented attack has minimal requirements on the abilities and knowledge of an adversary. The attack requires no detailed knowledge about the design, nor does it require a training phase.

Key words: Side Channel Analysis, Collision Attack, DPA, Masking, AES, Hardware Implementation

1 Introduction

Ten years after the introduction of side-channel attacks [2, 10, 13, 16, 23], the creation of a DPA-resistant cryptographic hardware implementation remains a challenge. During the last years several countermeasures to prevent power and EM-analysis have been proposed [12, 20, 21, 29, 30]. One of the main targets of the side channel community are implementations of the AES. AES [19], having been the NIST symmetric encryption standard for about 10 years, is probably the most widely used cipher in practical applications. Despite of its high cryptographic security in a black box scenario, implementations of AES are a popular and easy target for side channel attacks such as DPA and SPA. Correspondingly, the efficient and leakage-minimized implementation of AES is a well-studied problem [8, 9, 21, 24, 25].

At the same time attacking techniques have been improved and defeated many of these countermeasures. The first practical evaluation was performed on one additive and one multiplicative masking scheme of AES [17]. It has been shown that though they are resistant to classical DPA attacks considering standard Hamming Weight (HW) and Hamming Distance (HD) models, more sophisticated attacks using more precise power models, e.g., the toggle count model [17], are capable of overcoming the masking countermeasure. However, these attacks usually require detailed information about the implementation such as the netlist of the target device. Later it was shown in [18] that XOR gates of the mask multipliers of the masked S-box play the most significant role in the susceptibility of the evaluated schemes, but to our knowledge the proposed solutions have not been practically evaluated.

Another approach for attacking implementations using a power or EM side channel are collision attacks. Here, the attacker concludes from the leakage that two identical intermediate values have been processed and uses this information to cryptanalize the encryption scheme. The practicability of these attacks has been shown against DES in [15, 27]. Successful attacks against AES have been presented in [6, 26]. Collision attacks remain less popular than DPA-like attacks because of their sometimes complicated setup, their strong dependence on noise, and the more complex key recovery phase. Although the number of traces actually used in an attack is usually lower than that of classical DPAs, the number of traces needed to generate a collision normally makes the attacks less efficient than, e.g., correlation DPAs. Finally, with the advent of randomizing-like countermeasures, collision attacks seem to be infeasible against protected implementations.

1.1 Our Contribution

In this work we present a method to identify and exploit collisions between masked S-boxes in a very efficient manner. In fact, we use correlation to combine the leakage of all possible collisions and thereby including the full set of obtained measurements in the attack. Since practical evaluation of attacks and countermeasures by means of making a state of the art ASIC chip is not a time- and cost-effective approach, we have applied our attack on a masked version of the AES, implemented on a Xilinx Virtex-II Pro chip mounted on the Side-channel Attack Standard Evaluation Board (SASEBO) [1]. Our implementation generates all masks for each plaintext byte uniformly at random and none of the mask bytes is reused in later encryptions. Our investigation shows that the applied masking scheme is capable of resisting against those first-order DPA attacks which use common and well-known power models, e.g., HD and HW. From the results of [17] it can be expected that the masking scheme can be overcome when using a more accurate power model, e.g., toggle count, or when applying template-based DPA attacks. These attacks, however, assume a powerful adversary, because detailed knowledge such as a back annotated netlist of the layout is needed, or a training phase using a controllable target implementation has to be performed. None of these requirements have to be met to perform the attack presented in this article.

Our proposed attack reduces the effect of randomness by means of averaging over known (not chosen) inputs, and detects the collisions on the S-box input/output by examining the leakage of averaged power traces. In fact, our attack reveals that in our target implementation even uniformly distributed masks cannot prevent a first-order leakage depending on the unmasked values. It should be noted that our attack does not depend on a specific leakage model. The experimental results show that our attack is able to recover the key by means of less than 20 000 traces while the secret starts leaking out by a zero value attack using at least 1 000 000 traces of the same implementation. For a second-order zero-offset DPA, even around 8 000 000 traces are needed to recover the secret key.

1.2 Organization

The remainder of this article is organized as follows: Section 2 describes the target implementation of the AES and sets it into the context of related work. In Section 3 we analyze the AES implementation with classical methods, before we detail on the proposed collision attack in Section 4. A conclusion is given in Section 5.

2 Hardware Implementation of the AES

Several optimizations for hardware implementations of the AES have been proposed. To minimize circuit area consumption of the AES, Rijmen [24] suggested the use of subfield arithmetic in $GF(2^4)$ to compute the inverse in $GF(2^8)$. The idea was taken further by Satoh et al. [25] using the "composite-field" approach / "tower-field" representation by Paar [22] to implement the inversion in $GF(2^4)$ by the use of sub-subfield arithmetic in $GF(2^2)$. Along with other innovations this resulted in a very compact AES S-box, which was further improved by Canright [8] through choosing the normal bases which yielded the smallest circuit size.

Several masking schemes have been proposed to create a masked AES S-box using either multiplicative or additive methods. Unfortunately, multiplicative ones [4, 11] are vulnerable to certain attacks, especially the so-called zero-value attack, because a zero input value does not get masked by multiplication. The solution is to use the tower-field representation for an additive masking scheme because the inversion in $GF(2^2)$ is equivalent to squaring which is linear. The first at least algorithmically provable secure additive masking scheme was proposed by Blömer et al. [5]. Later Oswald et al. [21] proposed a more efficient scheme by using different bases and reusing some mask parts. Canright et al. [9] applied this idea to his very compact S-box resulting in the most compact masked S-box to date. [33] is also another design showing the interest of the research community on this topic.

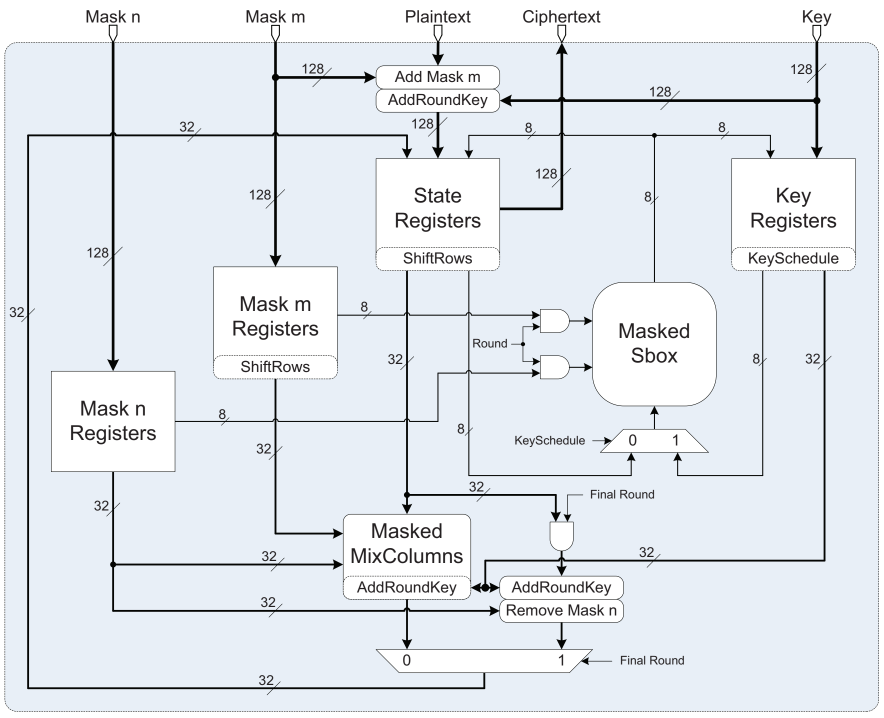


Fig. 1. Architecture of the AES design

2.1 Our Implementation

Our goal is the evaluation of a hardware implementation of the AES that is supposed to be secure against first-order side-channel attacks. To cover a wide range of possible implementations, we decided to implement two different architectures of the AES. The first one is designed to achieve low power consumption and has a low area requirement. This is achieved by choosing an 8-bit data path and features a single S-box that is sequentially used for SubBytes operations and the key scheduling. All registers are implemented as byte-wise shift-registers which can be clocked independently. The full data path of the complete AES engine excluding the key registers is masked. The mask values are generated internally by means of a PRNG, and the (uniform) distribution of the generated random values have been verified. The masks are different for each plaintext byte and differ in each execution of the encryption. The high level architecture of our AES design is depicted in Fig. 1. Unless stated otherwise, our analysis focuses on this implementation.

To verify that our attack also works in the presence of noise, we implemented a second AES engine that has a 32-bit data path and features four parallel S-boxes. Details on this engine can be found in Appendix B.

The design of the masked S-box is identical to [9] which uses two independent masks, m and n to randomize the input and the output. We retrieved the Verilog code from D. Canright's website and paid special attention that the order of the operations and other suggestions to maintain the masking scheme have been strictly kept by the synthesis tool.

Encryption starts by providing 128-bit plaintext, key, and masks m and n. The masks are independent and uniformly distributed and differ for each plaintext byte and each encryption execution. At the beginning of each round ShiftRows is performed on both the masked data state and the input mask m. The S-box is then first used by the key schedule unit to compute the first 32-bit part of the next round key without using any masks. In the following four clock cycles the masked S-box performs the SubBytes transformation on the first column. The consecutive masked MixColumns and AddRoundKey transformations are performed using a 32-bit wide datapath. During this operation the mask of the state is also changed back to mask m because during SubBytes the input mask m is replaced by the output mask n. This sequence of four times SubBytes followed by MixColumns and AddRoundkey is repeated four times to complete the round function.

2.2 Details on the Masked AES S-box

The general structure of the used masked S-box is depicted in Fig. 2 omitting the tower-field conversion. While only the $GF(2^8)/GF(2^4)$ module is shown, the $GF(2^4)/GF(2^2)$ module uses the same structure the only difference being that instead of an $GF(2^2)$ inversion module, this step is merged as squaring to the overall design. As can be seen the additional elements in the datapath are all additive (XORs). It is important to introduce a new mask before adding the masked products since the distribution of the sum of two masked products is otherwise not uniformly distributed as explained in [9]. By doing all summations in the correct order the result of every computation is either uniformly distributed or has the random product distribution independent of the used plaintext and key. Therefore, as stated in [9], the scheme is considered to theoretically achieve perfect masking on an algorithmic level by the definition of [5].

3 Analysis of the AES Implementation

The whole design has 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]. To better understand the leakage of our implementation we performed several tests of our platform. We performed tests to identify when certain leakages occur. Subsequently we analyzed the vulnerability of our implementation to first-order DPA attacks based on correlation, both in the unmasked and in the masked case.

All tests are performed on the power consumption of the Virtex-FPGA containing our implementation. Measurements are performed using a LeCroy

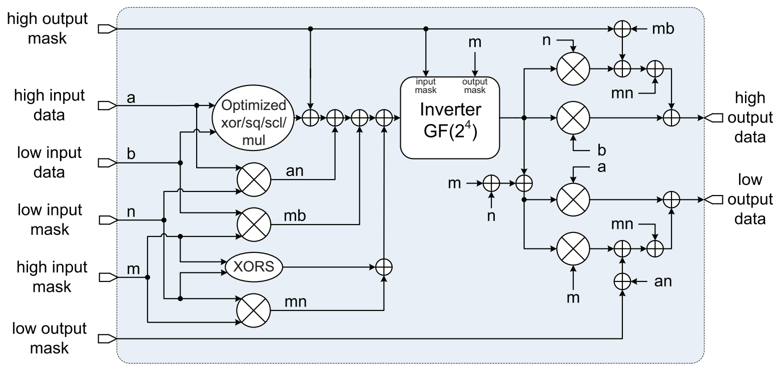


Fig. 2. Block diagram of the used masked $GF(2^8)$ inverter

WP715Zi 1.5GHz oscilloscope at a sampling rate of 5GS/s[1] and by means of a differential probe which captures the voltage drop over an 8Ω resistor in the VDD (3.3V) supply of the FPGA. In all the experiments the clock signal is provided by a 24MHz oscillator which is divided by 8 using a frequency divider, i.e., our cryptographic engine is clocked at a frequency of 3MHz.

3.1 Analysis of the Unprotected Architecture

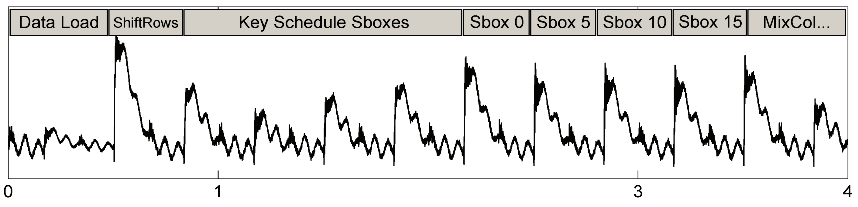
In a first step we analyze the leakage of an unprotected implementation that employs the highly compact unmasked AES S-box design of Canright [8]. A power trace of this unprotected implementation during the first 12 clock cycles is shown in Fig. 3(a). The processing order and hence the occurrence of leakages over clock cycles will pretty much stay the same for the masked implementation, as the high level architecture remains constant.

Similarly to what was observed in [17], DPA attacks using the HW of the S-box input/output are not successful. We get a good estimation about the leakage strength of the implementation platform performing a DPA attack predicting the HD of 8 bits of the state register[2]. The result of this HD-based DPA is shown by Fig. 3(b). As shown in Fig. 3(d), the leakage of approx. 3 000 traces suffices to perform a successful attack using a HD model.

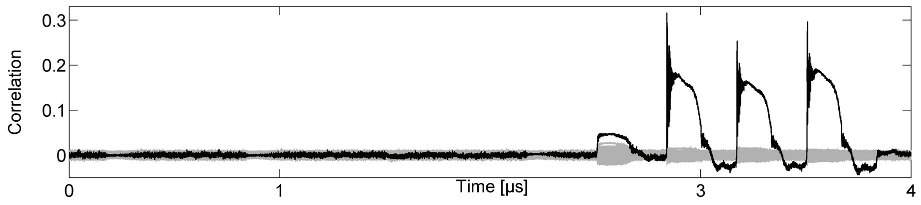
As explained in [16], most of the time implementations of the AES S-box consume less power for the zero input value than for the other cases. It holds here as well, and an attack using the zero value model is possible which is shown by Fig. 3(c). Moreover, according to Fig. 3(d) 4 000 measurements are required for succeeding with the zero value attack.

[1] This oversampling is not essential here; however, since glitches and toggles in hardware occur at very high frequencies, we decided to keep a high sampling rate, but we have confirmed the feasibility of the attacks using lower sampling rates, e.g., 1GS/s.

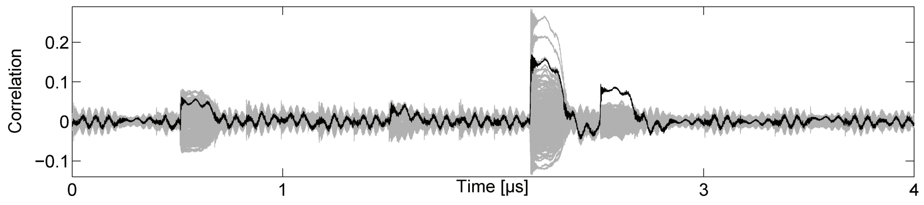
[2] Note that to predict the HD of the state register in our target architecture, two key bytes amongst 2^{16} hypotheses should be guessed at the same time.



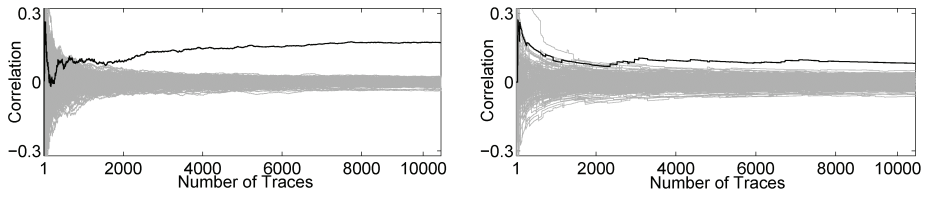
(a)



(b)



(c)



(d)

Fig. 3. (a) A measured power traces of an undefended implementation, (b) DPA attack result predicting toggles in the state register, (c) DPA attack using zero value model predicting the S-box input, and (d) the required number of traces in attacks using (left) HD model and (right) zero value model.

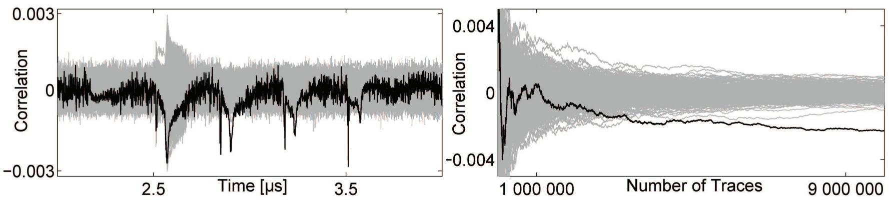


Fig. 4. Result of a zero-offset second-order DPA attack on the masked implementation using a HD model (left) by 10 000 000 traces and (right) at point 2.9μs over the number of traces.

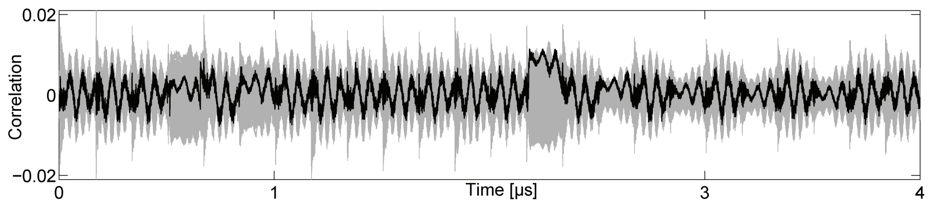


Fig. 5. Result of a zero value DPA attack on the masked implementation using 1 000 000 traces.

3.2 Analysis of the Masked Architecture

Moving towards the masked version of the implementation, we should emphasize that neither the attacks using the HW model predicting S-box input/output nor those which use the HD model on the state register are expectedly able to reveal the secrets. Since in our architecture the state and the mask registers are shifted in the same fashion, both masked values and the masks are processed at the same time. Therefore, one can perform a zero-offset second-order DPA attack [31] by squaring the power values and by means of a HD model to predict the transitions in the state register. In practice higher-oder attacks usually require much more traces in comparison to the first-order ones, and we collected 10 000 000 traces to clearly distinguish the correct key guess amongst the others. The result of such an attack is shown by Fig. 4 indicating that around 8 000 000 traces are needed to have a successful attack. In our experiments we have examined several possible power models in first-order attacks, and interestingly the secret starts leaking by a zero value DPA attack using 1 000 000 traces. The relevant result is shown by Fig. 5. It shows that power consumption of the target implementation is not really independent of the unmasked values, and this issue motivated us to try for an alternative approach in order to decrease the number of measurements and to distinguish the secret more clearly. It should be emphasized that in our target implementation the mask values are internally generated by means of a PRNG, and the (uniform) distribution of the generated random values has been verified. Furthermore, the masks are different for each plaintext byte and differ in each execution of the encryption.

Before introducing our collision attack, we first explain some issues which we observed during practical experiments. As mentioned before, we acquired millions of traces for the aforementioned attacks. It should be noted that these traces have been recorded on randomly chosen plaintext bytes. To learn about the behavior of the implementation, we computed the average over the traces (measured from the masked implementation) based on a plaintext byte and thereby obtained 256 mean traces. By examining the variance of the mean traces we can detect in which clock cycle a function, e.g. the S-box, relevant to the selected plaintext byte is computed, if the mean traces are not ideally close to each other. If such features are detectable in the power traces, the mean traces are not independent of the unmasked values. In Fig. 6 two variance traces over the mean traces of plaintext byte 0 and byte 5 are shown[3]. The figure shows that a function over these two plaintext bytes is computed in two consecutive clock cycles, which fits to the target architecture. We have used 1 000 000 measurements to generate the variance traces shown in Fig. 6, but we have examined using less number of traces, e.g., 50 000, and the result had the same shape and the same feature.

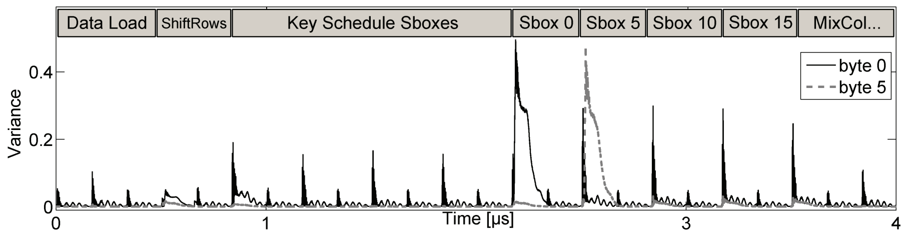


Fig. 6. Variance of mean traces for plaintext byte 0 and 5

Since the mean traces depend on the unmasked values, a couple of attacks are possible. For example, as expected by the authors of [9], a DPA using the toggle-count model should work here. Yet, for that attack the adversary needs to have access to the target netlist or layout to simulate and extract the toggle-count model. Moreover, a template-based DPA attack also might work, but the adversary needs to first create profiles for a known key. The aim of our attack is to avoid such limitations and strong assumptions.

4 Correlation-Enhanced Collision Attack

Based on the observations described in Section 3, we adapt collision attacks to be able to exploit any first-order leakage without knowing the precise hypothetical power model. The attack targets collisions in the full S-box computation. Detected collisions have the same 8-bit input and consequently the same 8-bit output value. Please keep in mind that these values are always masked, with different masks for each measurement and each S-box. The developed attack does not require any sort of training phase with a known-key device. Of course, knowledge about the position of the execution of the S-box computation are helpful, but all information needed can be extracted from the measurements of the device under attack. As described in section 3.2, one way would be to compute the mean traces and perform a variance check. Alternatively, such information can be gained by *combing* through the power traces with an offset of e.g. one clock cycle [28]. We split the attack into a measurement phase, which is comparable to previous collision attacks against unmasked implementations, and an enhanced detection phase.

[3] Note that the mean traces are computed based on each plaintext byte independently.

During the *measurement phase* we record the power consumption traces T_i of the encryption of random known plaintexts $P_i = \{p_j^i\}_{j=0}^{15}$. We know that each trace T_i contains leakage of every S-box computation of the first round $s_j^i = \mathrm{S}(p_j^i \oplus k_j)$, which we target in our collision attack. In our model, a collision occurs when two S-box computations at the byte position $j_1 = a$ and $j_2 = b$ collide, i.e., have equal output $s_a^{i_1} = s_b^{i_2}$ and due to the bijectivity of the S-box also equal input $p_a^{i_1} \oplus k_a = p_b^{i_2} \oplus k_b$. We can define the input difference $\varDelta_{a,b}$ as

$$\varDelta_{a,b} = p_a^{i_1} \oplus p_b^{i_2} = k_a \oplus k_b$$

Hence, this type of collision reveals a linear relation between two key bytes, depending only on the known difference $\varDelta_{a,b}$. By finding more first-round collisions, eventually we will have relations for all 16 key bytes k_i, reducing the key entropy to 8 bits (i.e. 256 key candidates for the full 128-bit AES key), which can easily be recovered by trial and error. This attack is labeled *linear collision attack on AES* in [6]. In theory, this attack is prevented by masking, since both input and output of the S-box are masked, destroying any relation between input difference $\varDelta$ and the plaintexts $p_a^{i_1}$ and $p_b^{i_2}$. Yet, we show that there is a remaining leakage in the masked Canright/Batina S-box that can be exploited by an adaption of the linear collision attack we describe in the following.

The measurement phase is the same as for the normal linear collision attack. Yet, we apply a different *detection phase* to identify many collisions at once. As described above, we first have to detect where the leakage of the individual S-boxes occurs. To reduce the influence of the masks, we average the power consumption for equal input bytes p_j. We do this by browsing all of our traces T and averaging only those traces where the jth plaintext byte equals a certain value $\alpha \in GF(2^8)$. Hence, we get 2^8 average traces M_j^α for each plaintext byte position j, where M_j^α is the average of all traces T_i where $p_j^i = \alpha$.

$$M_j^\alpha = \overline{T_i \cdot \delta(p_j^i = \alpha)}$$

Unlike the classical *linear collision attack*, we do not try to detect a single collision, but directly include all possible collisions between two byte positions $j = a$ and $j = b$. We know that for one particular key pair k_a and k_b, the difference $\varDelta_{a,b} = k_a \oplus k_b$ is constant. Hence, a collision occurs whenever the plaintexts at position a and b show the same difference, i.e., $p_a = \alpha$ and $p_b = \alpha \oplus \varDelta_{a,b}$. Our approach is to guess the difference $\varDelta_{a,b}$ and verify our guess by detecting all resulting collisions $p_a = \alpha$ and $p_b = \alpha \oplus \varDelta_{a,b}$ for all $\alpha \in GF(2^8)$ at the same time. For detecting the correct $\varDelta_{a,b}$, we correlate the averaged power consumption M_a^α of the S-box lookup of $p_a = \alpha$ to the averaged power consumption $M_b^{\alpha \oplus \varDelta_{a,b}}$ of the S-box lookup of $p_b = \alpha \oplus \varDelta_{a,b}$ for all $\alpha \in GF(2^8)$. The correct difference $\varDelta_{a,b}$ of the two key bytes k_a and k_b is then given by:

$$\arg\max_{\varDelta_{a,b}} \rho\left(M_a^\alpha, M_b^{\alpha \oplus \varDelta_{a,b}}\right)$$

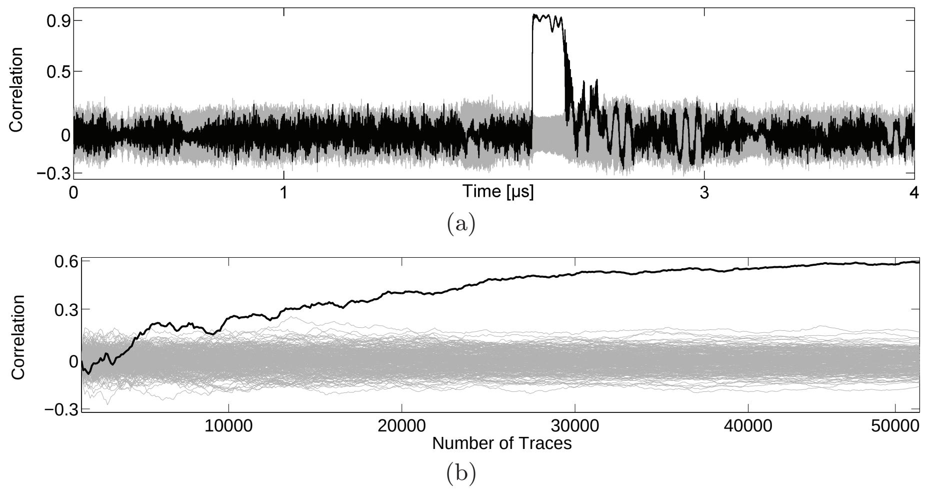


Fig. 7. (a) The result of collision attack using the mean traces of byte 0 and byte 5, (b) the result of the attack over the number of used measurements.

The correlation $\rho\left(M_a^\alpha, M_b^{\alpha\oplus\Delta_{a,b}}\right)$ is computed over all $\alpha \in GF(2^8)$ and can be computed for every point in time. It is maximum if $\Delta_{a,b}$ is correct. For wrong differences Δ, the correlation approaches zero. Hence, this attack behaves similar to a correlation attack. Unlike correlation based DPA, which correlates the power consumption to a power model that will never truly represent the real power consumption, our attack correlates the power consumption of one S-box computation to the power consumption of a different instantiation of the same S-box (processing the same value). Compared to classical collision attacks, our attack is stronger because all traces are included in calculating the correlation coefficient ρ, i.e. leakage from all traces T_i is used to recover the key relations.

If we go back to the last experimental results described in section 3.2, where millions of traces have been collected from the masked implementation while all 256 mask bits are randomly generated for each measured encryption, we had the mean traces for plaintext byte 0, M_0, and for plaintext byte 5, M_5. Also, we have shown that a function, here the S-box, exists, which processes a value depending on these two plaintext bytes at two consecutive clock cycles. Then we shift for example M_5 with the length of a clock cycle to the left to align the traces and perform the attack algorithm. The result of the attack is shown by Fig. 7(a) in which the correct hypothesis is obviously distinguishable amongst others. As mentioned before, our attack computes the correlation between the power consumption of two instances of the S-box computation; this explains why we get such a high correlation value for a correct guess of Δ. It should be noted that while initially 1 000 000 traces have been used to compute the mean traces, around 20 000 traces are enough to distinguish the correct key relation. The relevant figure for the number of required traces is shown in Fig. 7(b).

Recovering the full AES key: Repeating this attack on other sets of mean traces, e.g., between M_0 and M_1, M_0 and M_2, and so on, will reveal 15 relations between all key bytes[4]. The correct AES key can easily be distinguished from the remaining 256 candidates by simple trial and error. In cases where this might not be feasible or to recover the full key of an implementation of AES-256, we simply extend our attack to the second round. An alternative for the case when ciphertexts are unknown is to compute the output of MixColumns of the first round (of AES) for each of the 256 possible key candidates and perform only one (collision) attack using the leakages of the second round for each of the 256 128-bit key candidates separately. The attack would be possible for only one of the key candidates since the use of others will lead to wrong input bytes for the AddRoundKey and therefore to incorrect averages (to make the mean traces). In fact, a variance test approach on mean traces of the second round can reveal the correct 128-bit key candidate, and performing the attack on the second round is not even necessary to recover the full key of AES-128. No new measurements are needed for this extension.

Resemblance to Template DPA: A better understanding of how and why the attack works can be gained by a comparison to a template-based DPA as described in [3, 16]. Since we do not have a training phase, the creation of templates is different. We create templates M_j^α for each input (or output) value $p_j = \alpha$ (like in some template attacks) and also for each input (or output) byte position j. The separate templates for different byte positions j are necessary, as we cannot match our templates M_j^α to specific states (or input-output combinations) of the S-boxes, because we lack knowledge about the key. In the next step, we compare pairs of these templates for two positions $j_1 = a$ and $j_2 = b$ by correlating them to each other. A template-based DPA attack, as described in [16], instead uses the template as power model which is correlated to each individual power trace. Due to the noise in each trace, the resulting correlation values are much lower when compared to our case. Our attack correlates two sets of templates, which have a much lower noise due to the averaging process. The relative distance between the correlation of the right and the wrong key hypotheses is quite similar in both attacks. Compared to a template-based DPA, our attack assumes a much weaker adversary that neither needs access to a known key implementation nor requires a training phase.

Attack in known-ciphertext scenario: Knowing only the ciphertext bytes the same attack is possible using the power traces covering the last round of the encryption because of the absence of MixColumns at the last round. Similarly to the attack on the start of the encryption, 256 128-bit candidates will remain as the last round key. Then, for each of them the input of MixColumns of the 9th round can be computed which is also the output of the S-box of the same

[4] In fact, 120 key byte relations can be computed and possibly all be evaluated by voting techniques [6, 14, 32].

round. Therefore, the variance check approach as mentioned above will reveal the correct key. We practically performed the aforementioned attack, and were able to extract the secret using the same number of traces as in the known-plaintext attack.

Attack on parallel architectures/Influence of noise: All the practical results shown are for an 8-bit architecture, and each S-box is computed in a separate clock cycle. In order to investigate the feasibility of the proposed attack in the presence of increased (switching) noise, we have examined the same attack on a 32-bit architecture where four S-boxes are in parallel running at each clock cycle. Since the mean traces are used in our proposed attack, increasing the number of measurement helps reducing the effect of Gaussian noise. An important point here is that our proposed attack compares the dependency of power traces of an instance of S-box (on its inputs) in different clock cycles. When more than one instance of S-box exist, although all of them have been implemented using the same netlist, their placement and routing, especially in FPGAs, will not be necessarily the same. As a result, the power consumption characteristic of different S-box instances will not be exactly the same. In order to perform our proposed attack in such cases the best way would be to compare the power consumption of the same instance of the S-box in different clock cycles. According to the experimental results shown in Appendix B the secret can be revealed in this way using around 250 000 measurements which shows the strength of the attack. However, if the architecture does not share any S-box instances for computations in a round (of AES), the effectiveness of our attack will depend on the similarity of power consumption characteristics of different S-box instances. Moreover, it should be noted that the proposed attack is not specific to the applied masking scheme which still has a first-order leakage. It should be efficient for any case where the mean traces are slightly different. For example, the attack works on an unprotected implementation as well, i.e., the adversary does not even need to know whether a countermeasure has been applied in the target device. We have practically evaluated this issue as well; as a result around 3 000 traces are required for the attack on an unprotected implementation using an 8-bit architecture.

Shuffling as a Countermeasure: AES does not require the S-box lookups (and also other parts of the algorithm) to be executed in a predefined order [12, 16]. Randomizing this order is a popular countermeasure for side channel analysis, because it increases the noise and reduces the leakage of one particular S-box at one instance in time. Shuffling requires an increased flexibility of the implementation and a random source to determine when to execute which S-box. In the case of AES, typically either only four S-boxes are shuffled to not conflict with the subsequent mix-columns operation, or all 16 S-boxes are processed and shuffled before the outputs are further processed. The counter-attack for shuffling is Combing [28], where the leakage of all time instances, where a particular S-box is processed, is combined by some function. Combing allows to retain the full signal compared to the unprotected case, by increasing the noise to that of a parallel processing of the S-box. The unpredicted S-box executions will enter the averages M as noise, showing a similar influence as on classic DPA. For the case where just 4 S-boxes are shuffled, we expect a similar behavior as in the 32-bit implementation, if the attacker applies combing.

Leakage due to an implementation error? One may ask about the source of leakage which we found here since we have presented the practical result on a whole AES implementation, and if some flaws in the design architecture have caused the observed strong leakage. We should emphasize that as expressed before in Section 2, we have made sure to keep all necessary requirements suggested in the original design of the masked S-box [9], like the correct order of the product additions and the masked summation of these. Moreover, we have implemented only one masked S-box on the same platform and have examined its leakage when the S-box input (including masked input and masks) solely change. The relevant results are shown in Appendix A, and confirm that the S-box computation is the source of information leakage which caused the observed vulnerability. Although we have not performed a time-consuming simulation to make the toggle-count model, we believe that the source of such a first-order leakage is toggles and glitches of the combinational circuits.

Applicability to other Algorithms: On other algorithms exhibiting a similar structure of a key addition operation followed by some kind of S-box operation (e.g. typical SPN structures), the attack can be applied in a similar fashion. Suppose that l represents the bit length of the plaintext and key portions which are processed independently from other portions at the start of the target algorithm, 8 in the case of AES because of the bit length of the S-box input, and let us represent $\odot$ as the first function which is processed over a plaintext and the relevant key portion, XOR in the case of AES because of addroundkey. As explained before, a collision in our model occurs when two $\odot$ operations at the plaintext position $j_1 = a$ and $j_2 = b$ collide, i.e., $p_a^{i_1} \odot k_a = p_b^{i_2} \odot k_b$. This can be written as:

$$p_a^{i_1^{-1}} \odot p_a^{i_1} \odot k_a = p_a^{i_1^{-1}} \odot p_b^{i_2} \odot k_b.$$

Finally, the input difference $\Delta_{a,b}$ can be expressed by:

$$\Delta_{a,b} = p_a^{i_1^{-1}} \odot p_b^{i_2} = k_a \odot {k_b}^{-1},$$

where x^{-1} is the inverse of x in the domain of $\odot$ operation.

Since for a particular key pair k_a and k_b the difference $\Delta_{a,b} = k_a \odot {k_b}^{-1}$ is constant, a collision occurs when the plaintexts at position a and b show the same difference, i.e., $p_a = \alpha$ and $p_b = \alpha \odot \Delta_{a,b}$ for all $\alpha \in GF(2^l)$. At the last step, correlating 2^l average traces M_a^α and average traces $M_b^{\alpha \odot \Delta_{a,b}}$ for all $\alpha \in GF(2^l)$ can reveal the correct difference $\Delta_{a,b} = k_a \odot {k_b}^{-1}$ by:

$$\arg\max_{\Delta_{a,b}} \rho \left(M_a^\alpha, M_b^{\alpha \odot \Delta_{a,b}} \right)$$

One important property of the attack algorithm is the number of values which contribute to the computation of the correlation coefficient. In the case of AES, $l = 8$ and α can take 256 different values such that the correlation is computed over 256 points, which is not too high, but still yields a suitable estimation of the correlation. The estimation gets less reliable for target algorithms with smaller S-boxes, e.g., PRESENT [7] with $\alpha \in GF(2^4)$. To solve this problem one can define a window and perform the attack not only on a single point in time, but also using other adjacent power points, i.e., to compute the correlation in a 2-dimensional domain which equals to make vectors from matrices and get the correlation over two vectors. Alternatively, two S-boxes can be attacked in parallel by viewing them as a single one and predicting the difference Δ on two keys at the same time, if the S-boxes are processed at the same time.

5 Conclusion

In this work we have presented a collision attack that efficiently breaks a masked implementation with a remaining first-order leakage. We have further shown that combining all possible collisions via the correlation coefficient generates a highly efficient attack. The number of traces needed to overcome an implementation of the masking countermeasure of [9] only increases by a small factor of six when compared to a DPA on an unprotected implementation. Unlike other advanced attacks, the described attack is as general as a classical DPA attack, because it makes minimal assumptions about the adversary. In fact, the attack makes almost no assumptions about the leakage and does not require any detailed knowledge about the implementation (such as general architecture, layout, and netlist). Furthermore, the attack works out-of-the-box without requiring a training phase. The attack succeeds on any protected implementation as long as the means of power consumption traces for certain inputs are different, supposing a sufficient number of measurements estimating mean values. To the best of our knowledge the presented attack is the first successful collision-based attack on a masked implementation which is supposed to be resistant against first-order attacks.

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. D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-Channel(s). In *CHES 2002*, volume 2523 of *LNCS*, pages 29–45. Springer, 2003.

3. D. Agrawal, J. R. Rao, and P. Rohatgi. Multi-channel Attacks. In *CHES 2003*, volume 2779 of *LNCS*, pages 2–16, 2003.
4. M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In *CHES 2001*, volume 2162 of *LNCS*, pages 309–318. Springer, 2001.
5. J. Blömer, J. Guajardo, and V. Krummel. Provably Secure Masking of AES. In *Selected Areas in Cryptography - SAC 2004*, volume 3357 of *LNCS*, pages 69–83. Springer, 2004.
6. A. Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In *CHES 2008*, volume 5154 of *LNCS*, pages 30 – 44. Springer, 2008.
7. A. Bogdanov, G. Leander, L. Knudsen, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT - An Ultra-Lightweight Block Cipher. In *CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.
8. D. Canright. A Very Compact S-Box for AES. In *CHES 2005*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005. The HDL specification is available at author's official webpage `http://faculty.nps.edu/drcanrig/pub/index.html`.
9. D. Canright and L. Batina. A Very Compact "Perfectly Masked" S-Box for AES. In *Applied Cryptography and Network Security - 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`.
10. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In *CHES 2001*, volume 2162 of *LNCS*, pages 251 – 261. Springer, 2001.
11. J. D. Golić and C. Tymen. Multiplicative Masking and Power Analysis of AES. In *CHES 2002*, volume 2523 of *LNCS*, pages 198–212. Springer, 2003.
12. C. Herbst, E. Oswald, and S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In *Applied Cryptography and Network Security – ACNS 2006*, volume 3989 of *LNCS*, pages 239–252. Springer, 2006.
13. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
14. T.-H. Le, J. Clédière, C. Canovas, B. Robisson, C. Servière, and J.-L. Lacoume. A Proposition for Correlation Power Analysis Enhancement. In *CHES 2006*, volume 4249 of *LNCS*, pages 174–186. Springer, 2006.
15. H. Ledig, F. Muller, and F. Valette. Enhancing Collision Attacks. In *CHES 2004*, volume 3156 of *LNCS*, pages 176–190. Springer, 2004.
16. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
17. 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.
18. S. Mangard and K. Schramm. Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations. In *CHES 2006*, volume 4249 of *LNCS*, pages 76–90. Springer, 2006.
19. National Institute of Standards and Technology (NIST). Announcing the Advanced Encryption Standard (AES). *http://www.nist.gov/*, November 2001.
20. S. Nikova, V. Rijmen, and M. Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. In *Information Security and Cryptology - ICISC 2008*, volume 5461 of *LNCS*, pages 218–234. Springer, 2008.
21. E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In *Fast Software Encryption - FSE 2005*, volume 3557 of *LNCS*, pages 413–423. Springer, 2005.

22. C. Paar. *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*. PhD thesis, Institure for Experimental Mathematics, University of Essen, Germany, 1994.
23. J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Smart Card Programming and Security - E-smart 2001*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.
24. V. Rijmen. Efficient Implementation of the Rijndael S-box, 2000.
25. A. Satoh, S. Morioka, K. Takano, and S. Munetoh. A Compact Rijndael Hardware Architecture with S-Box Optimization. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 239–254. Springer, 2001.
26. K. Schramm, G. Leander, P. Felke, and C. Paar. A Collision-Attack on AES: Combining Side Channel- and Differential-Attack. In *CHES 2004*, volume 3156 of *LNCS*, pages 163–175. Springer, 2004.
27. K. Schramm, T. Wollinger, and C. Paar. A New Class of Collision Attacks and Its Application to DES. In *Fast Software Encryption - FSE 2003*, volume 2887 of *LNCS*, pages 206 – 222. Springer, 2003.
28. S. Tillich and C. Herbst. Attacking State-of-the-Art Software Countermeasures - A Case Study for AES. In *CHES 2008*, volume 5154 of *LNCS*, pages 228–243. Springer, 2008.
29. 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.
30. E. Trichina, T. Korkishko, and K.-H. Lee. Small Size, Low Power, Side Channel-Immune AES Coprocessor: Design and Synthesis Results. In *Advanced Encryption Standard - AES Conference 2004*, volume 3373 of *LNCS*, pages 113–127. Springer, 2004.
31. J. Waddle and D. Wagner. Towards Efficient Second-Order Power Analysis. In *CHES 2004*, volume 3156 of *LNCS*, pages 1–15. Springer, 2004.
32. P. Yu and P. Schaumont. Secure FPGA Circuits using Controlled Placement and Routing. In *Hardware/Software Codesign and System Synthesis - CODES+ISSS 2007*, pages 45–50. ACM, 2007.
33. B. Zakeri, M. Salmasizadeh, A. Moradi, M. Tabandeh, and M. T. M. Shalmani. Compact and Secure Design of Masked AES S-Box. In *Information and Communications Security - ICICS 2007*, volume 4861 of *LNCS*, pages 216–229. Springer, 2008.

Appendix A: Evaluation of the Masked S-box Leakage

In order to examine the leakage of the used masked S-box we have designed and implemented a test module on our experimental platform to solely measure the power consumption which belongs to the masked S-box. A diagram of the used design is shown in Fig. 8. We have added some register stages at the input and output of the S-box to isolate its activity from other parts. Two sets of power traces, 100 000 each, for two different keys, k_0 and k_1, have been measured, and the same attack has been performed on the mean traces. According to the result of the attack shown in Fig. 9(b), the masked S-box itself causes the vulnerability, and the correct difference between two key bytes, $k_1 \oplus k_2$, is obviously detectable.

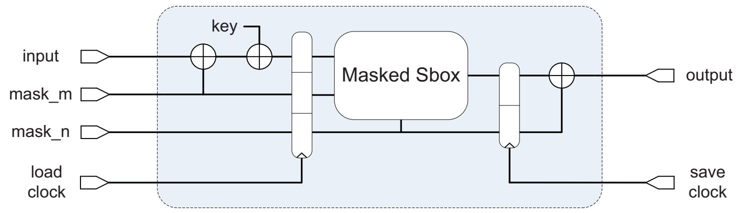


Fig. 8. Diagram of the test module

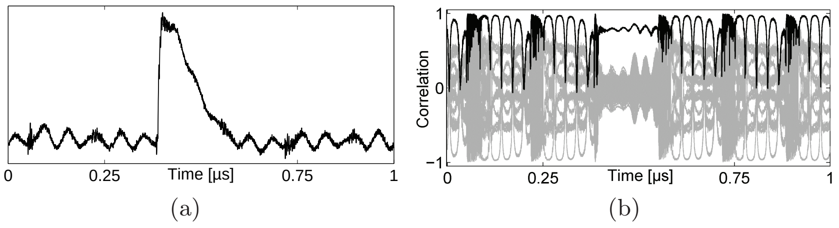
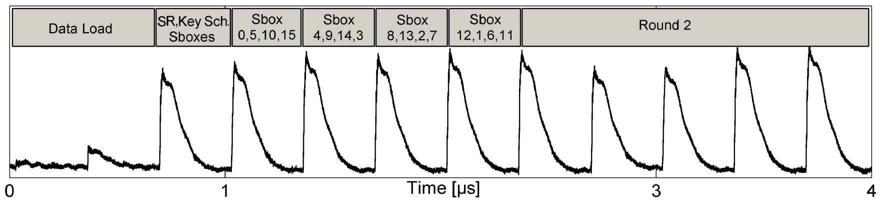


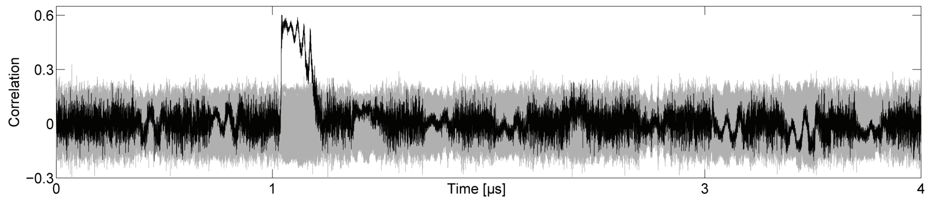
Fig. 9. (a) a power trace of the module under test, (b) the result of collision attack using the mean traces of the test module

Appendix B: Attack on a 32-bit Architecture

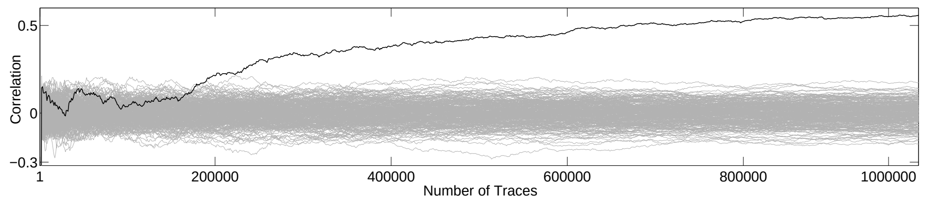
The architecture of our 32-bit implementation is similar to the 8-bit one shown in Section 2. After loading the data inputs and performing the initial AddRoundKey, the ShiftRows operation and the first round key computation are performed at the same time. Afterwards the SubBytes transformation using all four S-boxes followed by a MixColumns and AddRoundKey transformation are performed at each clock cycle. One round of the algorithm therefore needs 5 clock cycles. In fact, because of ShiftRows the S-box operations on plaintext bytes 0, 5, 10, and 15 are performed first, followed by the ones on plaintext bytes 4, 9, 14, and 3 and so on. The procedure of the attack is exactly the same as explained for the 8-bit architecture: a variance test approach identifies the clock cycles where the plaintext bytes are processed, and finally an attack using the mean traces recovers the relation between two key bytes. We have measured 1 000 000 traces. The result of the attack on byte 0 and 4, which are processed by the same S-box instance, is shown by Fig. 10(b). Fig. 10(c) also shows the number of required traces. As expected more measurement are needed in comparison to the 8-bit architecture.



(a)



(b)



(c)

Fig. 10. (a) a power trace of the 32-bit architecture, (b) the result of collision attack using the mean traces of byte 0 and byte 4, and (c) the result of the attack over the number of used measurements