Improved Collision-Correlation Power Analysis on First Order Protected AES

Christophe Clavier[1], Benoit Feix[1,2], Georges Gagnerot[1,2], Mylène Roussellet[2], and Vincent Verneuil[2,3]

[1] XLIM-CNRS, Université de Limoges, France
firstname.familyname@unilim.fr
[2] INSIDE Secure, Aix-en-Provence, France
firstname-first-letterfamilyname@insidefr.com
[3] Institut de Mathématiques de Bordeaux, France

Abstract. The recent results presented by Moradi et al. on AES at CHES 2010 and Witteman et al. on square-and-multiply always RSA exponentiation at CT-RSA 2011 have shown that collision-correlation power analysis is able to recover the secret keys on embedded implementations. However, we noticed that the attack published last year by Moradi et al. is not efficient on correctly first-order protected implementations. We propose in this paper improvements on collision-correlation attacks which require less power traces than classical second-order power analysis techniques. We present here two new methods and show in practice their real efficiency on two first-order protected AES implementations. We also mention that other symmetric embedded algorithms can be targeted by our new techniques.

Keywords: Advanced Encryption Standard, Side Channel Analysis, Collision, Correlation, DPA, Masking.

1 Introduction

Side-channel analysis was introduced by Kocher et al. in 1998 [9] and marks the outbreak of this new research field in the applied cryptography area. Meanwhile, many side-channel techniques have been published. For example Brier et al. proposed the Correlation Power Analysis (CPA) [4] which has shown to be very efficient as it significantly reduces the number of curves needed for recovering a secret key, and more recently the Mutual Information Analysis from Gierlichs et al. [7] has generated a lot of interest.

Since side-channel attacks potentially concern any kind of embedded implementations of symmetric or asymmetric algorithms, it is recommended to apply various masking countermeasures (among others) in sensitive products [1,14]. Second-order or higher-order side-channel analysis can however defeat such countermeasures by combining leakages from different instants of the execution of an algorithm and canceling the effect of a mask [12,13]. Such attacks are considered very difficult to implement and generally require an important number of power curves.

B. Preneel and T. Takagi (Eds.): CHES 2011, LNCS 6917, pp. 49–62, 2011.

A specific approach for side-channel analysis is using information leakages to detect collisions between data manipulated in algorithms. Side-channel collision attacks against a block cipher were first proposed by Schramm et al. in 2003 [19]. Their attack uses differential analysis to exploit collisions in adjacent S-Boxes of the DES algorithm. In [18] an attack against the AES is proposed to detect collisions in the output of the first round `MixColumns`. Later, Bogdanov [2] improved this attack by looking for equal S-Boxes inputs in several AES executions. He then studied in [3] statistical techniques to detect collisions between power curves. Two recent papers have updated the state-of-the-art by introducing correlation based collision detection: Moradi et al. [15] proposed a collision attack to defeat an AES implementation using masked S-Boxes, while Witteman et al. [22] applied a cross-correlation analysis to an RSA implementation using message blinding.

In this paper, we present two collision-correlation attacks on software AES implementations protected against first-order power analysis using masked S-Boxes and practical results on both simulated and real power curves. Our attacks are much more efficient and generic compared to the one presented in [15]. Moreover we believe our techniques to be applicable to other embedded implementations of symmetric block ciphers.

The remainder of the paper is organized as follows: Section 2 presents the two AES first-order protected implementations targeted by our study. Then in Section 3 we present our attacks and practical results on simulated power curves and on a physical integrated circuit. In Section 4 we compare our technique with second-order power analysis. Section 5 deals with the possible countermeasures and finally we conclude this paper in Section 6.

2 Targeted Implementations

The AES Algorithm. For the sake of simplicity in this paper we focus on the AES-128 which includes 10 rounds, each one decomposed into four functions: `AddRoundKey`, `SubBytes`, `ShiftRows` and `MixColumns`. It encrypts a 128-bit message $M = (m_0, \ldots, m_{15})$ using a 128-bit secret key $K = (k_0, \ldots, k_{15})$ and produces a 128-bit ciphertext $C = (c_0, \ldots, c_{15})$. Note however that the techniques presented in this paper are easily applicable to AES-192 and AES-256.

The only non-linear function of the AES is `SubBytes` (also referred to as the S-Boxes S in the following) which is a substitution function defined by the pseudo-inversion I in $\mathrm{GF}(2^8)$ and an affine transformation. In this paper, we consider the two following solutions that have been proposed to protect this function against first-order attacks.

2.1 Blinded Lookup Table

The first targeted implementation uses a *masked* substitution table as proposed by Kocher et al. [10] and Akkar et al. [1]. This masked table S' is defined by $S'(x_i \oplus u_i) = S(x_i) \oplus v_i$, with u_i (resp. v_i) the mask of the i-th input byte x_i (resp. output byte) of function `SubBytes`, $x_i, y_i, u_i, v_i \in \mathrm{GF}(2^8)$, $0 \leq i \leq 15$.

This table is usually computed before the AES execution and stored in volatile memory.

We further consider that the same masks u and v are applied on all S-Boxes during one execution (or a round at least) of the algorithm, i.e. $u_i = u$ and $v_i = v$ for $0 \leq i \leq 15$. We believe that this hypothesis is realistic for embedded security products considering that an expensive recomputation of the 256-byte substitution table S' is necessary for each new pair (u, v) and that the storage of many masked tables is not conceivable in memory constrained devices.

2.2 Blinded Inversion Calculation

An alternative solution has been proposed by Oswald et al. [16] and improved on by Canright et al. [5]. It consists in computing the inversion in $\mathrm{GF}(2^8)$ using a multiplicative mask. To do this efficiently it is proposed to decompose the computation using inversions in the subfield $\mathrm{GF}(2^4)$ (and possibly in $\mathrm{GF}(2^2)$). Such masking method is well suited for hardware implementations.

We recall some properties of the masked inversion. Let I' denote the masked pseudo-inversion such that $I'(x_i \oplus u_i) = I(x_i) \oplus u_i$. The element $x_i \oplus u_i$ in $\mathrm{GF}(2^8)$ is mapped to a couple $(x_{i,h} \oplus u_{i,h}, x_{i,l} \oplus u_{i,l})$ of $\mathrm{GF}(2^4)$ such that $x_i \oplus u_i \cong (x_{i,h} \oplus u_{i,h})X + (x_{i,l} \oplus u_{i,l})$. As detailed in [16] many calculations occur on these subfield elements to compute the masked inversion of $x_i \oplus u_i$. The exact details of these computations can be found in [16]. Note that in these formulas neither $x_{i,h}$ nor $x_{i,l}$ is directly inversed in $\mathrm{GF}(2^4)$ but the following value:

$$d_i \oplus u_{i,h} = x_{i,h}^2 \times 14 \oplus (x_{i,h} \times x_{i,l}) \oplus x_{i,l}^2 \oplus u_{i,h} \ .$$

Then the masked inversion in $\mathrm{GF}(2^4)$ of $d_i \oplus u_{i,h}$ gives $d_i^{-1} \oplus u_{i,h}$ and is used to compute $I'(x_i \oplus u_i)$.

The 16 input bytes of `SubBytes` are blinded using different masks u_i, but one can notice that input and output masks of the inversion stage are identical. Therefore another threat to take into consideration is the zero value power analysis. This technique has been introduced in [8] and [11], and recently implemented on the masked inversion in [15]. Finally, note that the technique presented in this paper also applies to the improved version of Canright et al. [5] when input and output of the inversion are masked with the same value.

2.3 Measurements and Validation of Implementations

Curve Acquisition. We have developed software implementations on a contact smart card using a 16-bit RISC CPU with low power consumption. Two different methods were used to validate our attacks.

First, we used *simulated curves*: a proprietary tool was used to simulate power curves based on the chip architecture and the code executed. This tool generates ideal power consumption curves without any noise which enables to validate in practice the resistance of an implementation to a set of side-channel attacks leaving aside the acquisition and signal processing problems.

Second, we used *real curves*: we made physical measurements on the chip itself using a MicroPross MP100 reader and a Lecroy WavePro numerical oscilloscope.

First-Order Resistance Validation. Since our aim was to present techniques able to defeat first-order protected devices, we performed the classical first-order differential and correlation analysis on the two implementations presented above, before testing our collision attacks.

To do so, we applied DPA and CPA on the `AddRoundKey`, `SubBytes` and `MixColumns` functions at the first and the last rounds of our implementations. We also performed detailed SPA for each input byte value using many average curves to detect any noticeable (biased) power traces that would reveal a potential leakage. In any case no leakage were observed. We also verified that both implementations were immune to zero value power analysis and to the attack presented by Moradi et al.

We have thus verified that to the best of our knowledge both considered AES implementations are resistant to known first-order attacks. Nevertheless we present in the next section two new collision-correlation techniques which jeopardize these implementations.

3 Description of Our Attacks

In this section, we present the general principle of collision-correlation attacks and then detail how it can be applied on the two considered AES implementations.

3.1 The Collision-Correlation Method

The principle of the attacks presented in this paper is to detect internal collisions between data processed in blinded S-Boxes on the first round of an AES execution. We demonstrate in the following that if i) we are able to detect that the same data is processed at instants t_0 and t_1, and ii) the S-Boxes are blinded such that either the same mask is applied to all message bytes or the mask is identical at the input and the output of each S-Box, then it is possible to infer information on the secret key with very few curves.

In the following, we will denote $(T^n)_{0 \leq n \leq N-1}$ a set of N power traces captured from a device processing N encryptions of the same message M. Then we consider two instructions[1] whose processing starts at times t_0 and t_1 and denote l the number of points acquired per instruction processing. As depicted in Fig. 1 we finally consider $\Theta_0 = (T^n_{t_0})_n$ and $\Theta_1 = (T^n_{t_1})_n$ the two series of power consumptions segments at instants t_0 and t_1.

Note that in practice the N power curves should start at the same instant of the encryption and be perfectly aligned. Such conditions generally require signal processing to be performed first. Note also that as the sampling rate is usually such that $l > 1$ points are acquired per instruction, we can generalize the definition of Θ_0 and Θ_1 as being series of l-sample curve segments instead of series of single power consumption samples.

[1] In our attacks we only consider the correlation between two identical instructions, but it may even be possible to detect that two different instructions manipulate identical data, e.g. by spotting a data bus using EMA.

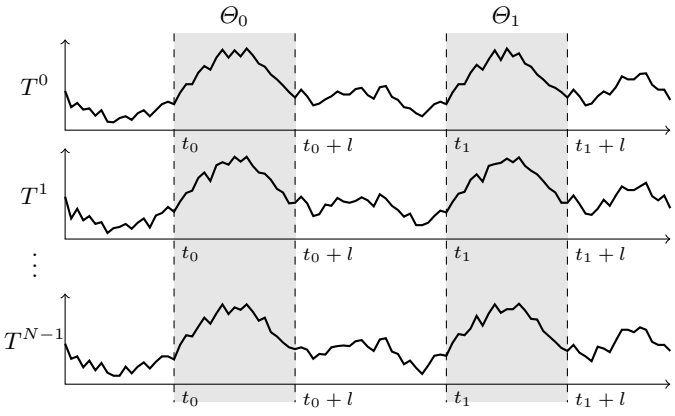

Fig. 1. General description of the collision-correlation attack

The final stage of the attack consists in applying a statistical treatment to (Θ_0, Θ_1) in order to identify if the same data was involved in $T^n_{t_0}$ and $T^n_{t_1}$ for $0 \leq n \leq N-1$. Let $\texttt{Collision}(\Theta_0, \Theta_1)$ denote a decision function returning `true` or `false` depending on whether this property is presumed to be fulfilled or not. Such a decision function would usually compare the value of a synthetic criterion with a practically determined threshold. Possible examples of such a criterion include the mean[2] squared difference, the least squared difference with binary or ternary voting [3], and the maximum Pearson correlation factor. As we used this latter criterion in our study, we recall that an estimation of the Pearson correlation factor between series of curve segments Θ_0 and Θ_1 at time offset t $(0 \leq t \leq l-1)$ expressed as

$$\begin{aligned}\hat{\rho}_{\Theta_0,\Theta_1}(t) &= \frac{\mathrm{Cov}(\Theta_0(t), \Theta_1(t))}{\sigma_{\Theta_0(t)}\sigma_{\Theta_1(t)}} \\ &= \frac{N\sum(T^n_{t_0+t}T^n_{t_1+t}) - \sum T^n_{t_0+t}\sum T^n_{t_1+t}}{\sqrt{N\sum(T^n_{t_0+t})^2 - (\sum T^n_{t_0+t})^2}\sqrt{N\sum(T^n_{t_1+t})^2 - (\sum T^n_{t_1+t})^2}}\end{aligned}$$

where summations are taken over $0 \leq n \leq N-1$, and $\Theta_i(t) = (T^n_{t_i+t})_n$ for $i \in \{0, 1\}$.

$\texttt{Collision}(\Theta_0, \Theta_1)$ thus consists in comparing $\max_{0 \leq t \leq l-1}(\hat{\rho}_{\Theta_0,\Theta_1}(t))$ to a given threshold. In our experiments a preliminary characterization of the targeted device enabled us to find proper values for l and the threshold.

Note that in this collision-correlation technique we compute the correlation factor between a set of real power consumptions Θ_0 with another set of real power consumptions Θ_1, rather than with model dependent estimations. As Bogdanov already described in [3] about binary and ternary voting techniques, an interesting property of this method is that, unlike Hamming weight based CPA, our

[2] The mean being taken over the N traces as well as over the l samples.

criterion does not rely on a particular leakage model. The consequences of this are that i) the attack is more generic and requires much less knowledge of the targeted device, and ii) the secret S-Boxes may be attacked as well as known ones.

As said above, correlating two instants (curve segments) on different traces has already been applied by Moradi et al. [15] on a particular AES implementation. However they collect many traces obtained by encrypting random messages and average them according to the value of an S-Box input byte. This results in 2^8 averaged curves for each byte position, from which they try to detect collisions between two bytes. They successfully carried out this attack on their implementation of the Canright et al. [5] first-order protected implementation. However as indicated by the authors their implementation presented a remaining first-order leakage based on zero-value attack. We applied Moradi's attack to the first-order protected implementations considered in this study without success. We thus consider that this attack is not applicable to most first-order protected implementations. Indeed averaging different traces implies the use of new random mask values which should spoil the influence of the unmasked data and make the collision of intermediate values undetectable. The technique we develop in this paper improves on Moradi's attack in order to detect data collisions by comparing two instants on a same trace and repeating it on many executions without the destructive averaging process. In the following we detail two applications of our attack on two different implementations.

Remark. Collision based analyses are also known as *cross-correlation attacks* in [22] and *multiple-differential collision attacks* in [3]. We prefer the term *collision-correlation attacks* since cross-correlation may be ambiguous depending on the context, and multiple-differential collision attacks seems us too generic for our method.

3.2 Attack on the Blinded Lookup Table Implementation

First, we present an application using principle presented above on the implementation described in Section 2.1. This attack targets the execution of the first round `SubBytes` function. Each 16 masked input byte $x'_i = x_i \oplus u$ is substituted by a masked output byte $y'_i = y_i \oplus v$ where $y'_i = S'(x'_i)$. We try to detect when two `SubBytes` inputs (and outputs) are equal within the first AES round as depicted on Fig. 2.

Detecting a collision in the first AES round between bytes i_1 and i_2 yields that $x_{i_1} \oplus u = x_{i_2} \oplus u$ and considering that $x_i = m_i \oplus k_i \oplus u$ implies the following relation of the two involved key bytes:

$$k_{i_1} \oplus k_{i_2} = m_{i_1} \oplus m_{i_2} \ . \tag{1}$$

Description. Practically, we encrypted N times the same message M and collected the N traces corresponding to the first AES round. For each of the N

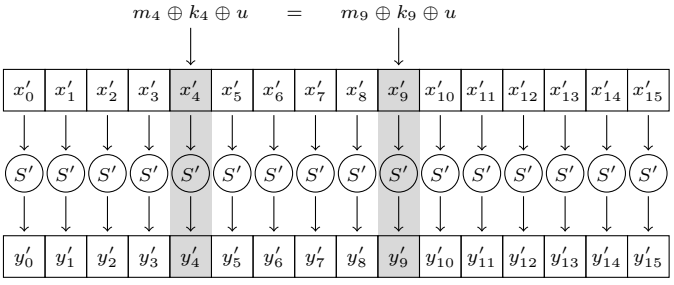

Fig. 2. Collision between the computation of two S-Boxes on bytes 4 and 9 on the blinded lookup table implementation

traces we identified the 16 instants t_i corresponding to the beginning of the computation $S'(x_i \oplus u)$. This allowed us to extract 16 segments from each trace and construct the series Θ_i used for collision-correlation as explained in Section 3.1.

Performing $\texttt{Collision}(\Theta_{i_1}, \Theta_{i_2})$ for all the 120 possible pairs (i_1, i_2) yields a set of relations $(i_1, i_2, m_{i_1} \oplus m_{i_2})$ given by Eq. (1). By repeating this process for several random messages M one can accumulate enough relations so that the secret key is recovered up to a guess on one key byte.

Based on 10 000 simulations we observed that on average 59 random messages (each one being encrypted N times) provide enough relations to retrieve the key up to an unknown byte.

Practical Results. We present hereafter our results on both simulated and real curves.

On simulated curves. The threshold of `Collision` was fixed to having at least one point among the l points correlation curve equal to 1. Under this condition our attack was successful for $N = 16$. Since a mean of 59 different messages are required, then $16 \times 59 = 944$ traces are sufficient on the average for the attack to succeed on simulated curves.

Figures 3 and 4 show the correlation curves obtained for two different messages. Both figures present the 120 outputs of $\hat{\rho}_{\Theta_{i_1}, \Theta_{i_2}}(t)$, $i_1 < i_2$ for each message. The black curve on Fig. 3 corresponds to a collision found for the first message, whereas the second message yields no collision.

On real curves. The attack was successful using $N = 25$ so that less than 1 500 traces allow to recover the key. Notice how few traces are needed to detect a collision by correlation. This confirms that the collision-correlation technique is much more efficient than classical model-based CPA which would not obtain high correlation levels with only 25 traces. Figure 5 shows an example of a correlation peak when an equality between two S-Box outputs occurs, while Fig. 6 shows the correlation curve when all S-Box outputs are different.

Note that in the case of real curves the threshold is slightly different. To identify a clear relation between two S-Box outputs the correlation curve must

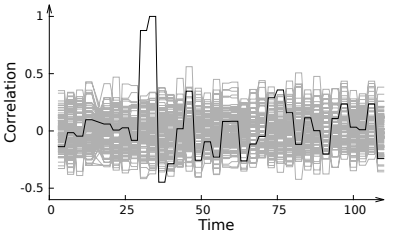

Fig. 3. Correlation curves obtained for a message giving one collision (black curve)

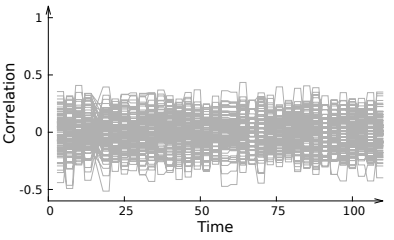

Fig. 4. Correlation curves obtained for a message giving no collision

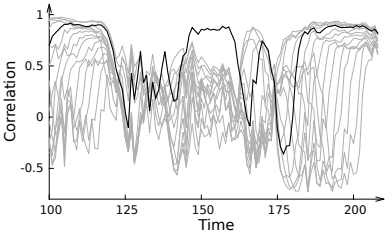

Fig. 5. Correlation peak on real curves when a collision occurs (black curve)

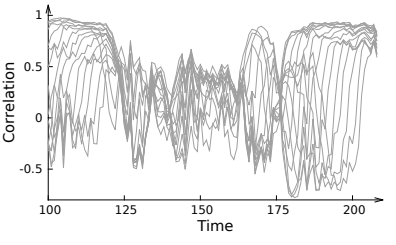

Fig. 6. No correlation peak occurs on real curves when intermediate data differ

be greater than 0.8 in the interval $[130, 160]$. So only these $l = 30$ points must be considered when computing $\texttt{Collision}(\Theta_0, \Theta_1)$.

Attack Improvement. The method for obtaining information about the key as described above basically exploits collision events where a pair (i_1, i_2) of indices gives a high correlation between Θ_{i_1} and Θ_{i_2} revealing the value of $k_{i_1} \oplus k_{i_2}$. While very informative, such collision events occur much less frequently than non-collision ones, that is when Θ_{i_1} and Θ_{i_2} show no significant correlation between each other. Non-collision events individually bring quite few information – namely that $k_{i_1} \oplus k_{i_2}$ is different from $m_{i_1} \oplus m_{i_2}$ – but they are so numerous that it appears worth trying to exploit them also.

As was already noted in [6,2], the problem of solving a set of equations involving sub-parts of the key can be formulated in terms of a labelled undirected graph. Each vertex i represents a key byte index and the knowledge of the XOR between two key bytes is represented by an edge (i_1, i_2) labelled with $k_{i_1} \oplus k_{i_2}$. At the beginning the graph does not include any edges. Each time a collision occurs between two unrelated key bytes a new edge is put on the graph and results in the merge of two connected components into a single larger one. All key byte values belonging to the same connected component can be derived from each other, and the goal of the attacker is to end up with a fully connected graph.

For a given message, only 0, 1, or 2 from the 120 pairs (i_1, i_2) lead to collisions in most cases. All other pairs reveal some impossible value for each $k_{i_1} \oplus k_{i_2}$.

Gathering all the information provided by these non-collisions, for each (i_1, i_2) we maintain a blacklist of impossible values for the XOR of the two key bytes[3].

Given the information provided by previous messages to the current graph and blacklists, we adaptively choose the next message in order to maximize its usefulness which we define as the number of pairs (i_1, i_2) where one can expect new information (either positive or negative) to be obtained. As a first idea we could define the penalty of a candidate message as the number of pairs (i_1, i_2) for which $m_{i_1} \oplus m_{i_2}$ is already blacklisted. Obviously the chosen message should minimize the penalty. Actually this is slightly more complex and the definition of the penalty of a message should be refined. Indeed we must also consider cases where the message is useful for (i_1, i_2) and (i_1, i'_2) – that neither $m_{i_1} \oplus m_{i_2}$ nor $m_{i_1} \oplus m_{i'_2}$ are blacklisted – but the value of $k_{i_2} \oplus k_{i'_2}$ is known to be precisely equal to $m_{i_2} \oplus m_{i'_2}$. In such a case the two usefulness opportunities brought by the message on pairs (i_1, i_2) and (i_1, i'_2) would bring the same information so that they should count for a single one and the penalty of that message must be increased by one.

In order to find a message with minimal penalty we devised a heuristic which works in two steps. In the first step we consider some random messages (say a few hundred) and select the one with the lowest penalty. This first step ends with a somewhat good candidate. Then in a second step we repeatedly attempt to decrease further the penalty by trying small modifications on this candidate until no more improvements occur by small modifications.

We simulated our method for adaptively choosing the messages. In these simulations we assumed that the attacker is always able to correctly distinguish between collision and non-collision events. Based on 1 000 simulations with random keys, we show that the key is fully recovered (up to the knowledge of one of its bytes) with as few as 27.5 messages instead of 59 messages with the basic method. As distinguishing between a collision and a non-collision necessitates only 25 traces per message, a mere 700 executions would suffice to recover the key by analysing real curves.

3.3 Attack on the Blinded Inversion Implementation

The previous attack cannot be applied to the blinded inversion implementation described in Section 2.2 since the different S-Box input and output bytes are masked with different values u_i. However there may exist a possible leakage leading to what we may call a *Zero & One value attack*.

One can notice that values 0 and 1 produce a collision between the input and the output of the masked pseudo-inversion stage I' as depicted on Fig. 7. This is due to the following properties of the pseudo-inversion:

$$\begin{aligned} I(0) = 0 \quad &\Rightarrow \quad I'(0 \oplus u_i) = 0 \oplus u_i \\ I(1) = 1 \quad &\Rightarrow \quad I'(1 \oplus u_i) = 1 \oplus u_i \end{aligned}$$

[3] Some of these blacklists must also be updated when two connected components are merged.

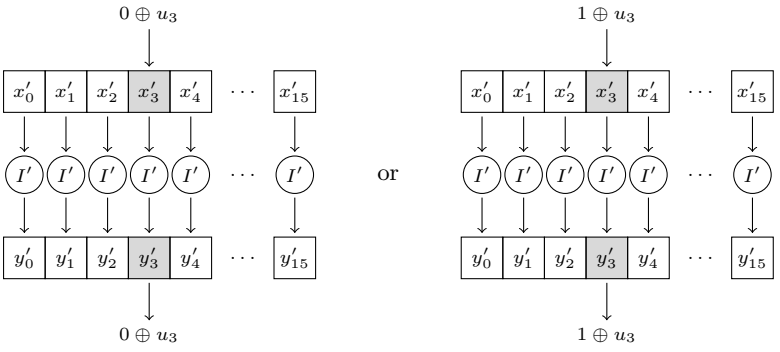

Fig. 7. Collision between the input and the output on byte 3 of the blinded inversion I' (values 0 and 1 lead to a collision)

The two cases leading to a collision are indistinguishable from one another. Detecting a collision between the input and the output of a blinded inversion gives either $x'_i = 0 \oplus u_i$ or $x'_i = 1 \oplus u_i$ which reveals a key byte except one bit:

$$k_i = m_i \quad \text{or} \quad k_i = m_i \oplus 1 \, .$$

Description. Assume we want to recover the 7 most significant bits of k_0. For every even byte value g we encrypt N times a single message M with $m_0 = g$ and collect the corresponding power consumption traces $T^{n,g}$, $0 \leq n \leq N-1$. Note that in this attack we only need to guess the 7 most significant bits because the least significant one is indistinguishable. Let's denote t_0 and t_1 the instants when $x_0 \oplus u_0$ is loaded before the pseudo-inversion I, and when the result is stored respectively. For each of the N traces we extract the two segments $T^{n,g}_{[t_0,t_0+l-1]}$ and $T^{n,g}_{[t_1,t_1+l-1]}$ and construct the series $\Theta^g_0 = (T^{n,g}_{[t_0,t_0+l-1]})_n$ and $\Theta^g_1 = (T^{n,g}_{[t_1,t_1+l-1]})_n$. For this step of our attack it is helpful to have some experience on the targeted implementation identify exactly where these two segments are located.

Applying the decision function $\texttt{Collision}(\Theta^g_0, \Theta^g_1)$ for all the 128 possible values g will reveal two possibilities for k_0. Repeating this step for all key bytes allows the key space to be reduced to 2^{16} values only. Note that a trick which allows to considerably reduce the number of traces is to encrypt the messages $M^g = (g, g, \ldots, g)$ with all bytes equal.

Results on Simulated Curves. As for previous attack on simulated curves, a relation is established when at least one point among the l points correlation curve is equal to 1. The attack is successful using $N = 16$ curves for each key guess. Figure 8 shows the 128 correlation curves for all possible guesses on k_0. The black curve corresponds to the correct guess for k_0.

The attack on this second implementation has thus been validated on simulated curves. We did not acquire real curves for this implementation. Based on

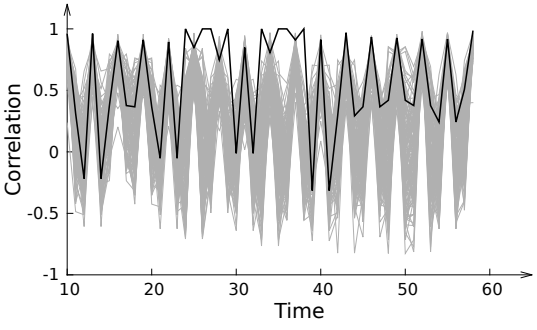

Fig. 8. Collision-correlation curves in the pseudo-inversion of the first byte in GF(2^8)

what has been observed on the previous attack (successful results obtained using simulations have led to successful results on the chip in practice), we believe that the attack would be successful on the real chip too, using a value for N of the same order to what was necessary for the first attack.

4 Comparison with Second Order Analysis

In this section, we present a brief comparison between the collision-correlation method and some known second-order attacks. Our analysis was inspired from the recent framework introduced by Standaert et al. in [20] and refined later in [21]. This comparison gives an overview on the efficiency of these different second-order techniques, and highlights how much the collision-correlation analysis improves on second-order attacks.

Our analysis targets the first implementation only. We compared the collision-correlation analysis with the second-order analysis involving the absolute difference combining function f_1, the squared absolute difference combining function f_2 and the normalized product combining function f_3, when using as distinguisher the Pearson linear correlation factor $\hat{\rho}$. Note that we did not used Mutual Information Analysis, whose results remain less efficient than the classical CPA in practice.

For sake of simplicity, we consider that the power consumption at instant t is the Hamming weight of the intermediate data involved in the computation plus a centered Gaussian noise ω_σ with standard deviation σ. Therefore $\mathrm{HW}_n(z)$ corresponds to the handling of the value z for the n-th encryption. We now define θ_0 and θ_1 as:

$$\theta_0 = (\mathrm{HW}_n(S(m_i \oplus k_i \oplus u) \oplus v) + \omega_\sigma)_{0 \le n \le N-1}$$
$$\theta_1 = (\mathrm{HW}_n(S(m_j \oplus k_j \oplus u) \oplus v) + \omega_\sigma)_{0 \le n \le N-1}$$

Let g_i (resp. g_j) denote a guess on k_i (resp. k_j). We compute the estimated values $w_{g_i,g_j} = \mathrm{HW}(S(m_i \oplus g_i) \oplus S(m_j \oplus g_j))$. Considering the N messages we obtain

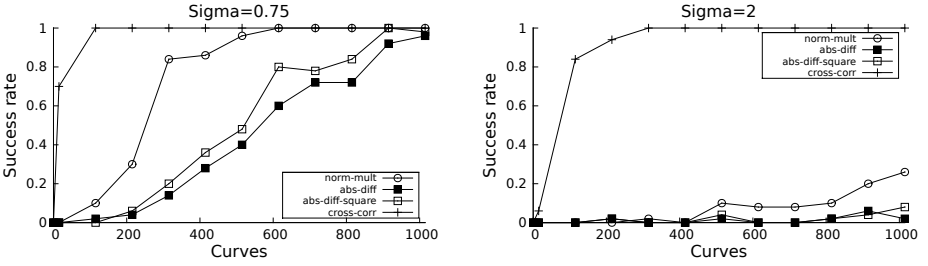

Fig. 9. Success rates of different simulated second-order attacks

the series $W_{g_i,g_j} = (w^n_{g_i,g_j})_{0 \leq n \leq N-1}$. Using the combining function f_j, the right key bytes are obtained for the highest correlation value $\hat{\rho}(f_j(\theta_0, \theta_1), W_{g_i,g_j})$.

Then as in [21] we execute many times the attack with the different combining functions and calculate the success rate of each one. Figure 9 shows two comparison graphs, one for $\sigma = 0.75$ and the other for $\sigma = 2$. Both graphs plot the success rates on 50 runs with respect to the number of curves used.

We emphasise that in this comparison the second-order attacks are shown in a very favorable light. Indeed the correlation model used here is exactly the one applied to simulate the curves. In practice an attacker would not have such good properties.

5 Countermeasures

The attacks presented in this paper defeat first-order protected implementations. Therefore, an obvious countermeasure would be to apply second-order masking. To the best of our knowledge, the best solution should be the countermeasure presented by Rivain et al. [17]. It allows the implementation of proven d-order DPA resistant AES for any $d \geq 1$.

Another countermeasure against our first attack may simply consist in executing the `SubBytes` function in a random order. Even if this method is not theoretically perfect, it may be sufficient to practically resist to second-order attacks. Considering the second implementation, we think that its main weakness is the use of a same mask before and after each byte pseudo-inversion. If the result is masked with a different value then the collision-correlation attack is no longer feasible.

It is also necessary to consider that depending on the quality of the hardware countermeasures provided by the device, these attacks can become much more complicated in practice.

6 Conclusion

We have presented a new collision-correlation analysis method on first-order secured AES implementations. We highlighted the fact that this kind of attack is

more powerful and practicable than previous second-order power analyses, and increases the risk of these implementations being broken in practice. This confirms the necessity for developers to take into account how collisions of masked data may be unsafe in cryptographic implementations. A possible countermeasure could be the use of second (or higher) order resistant schemes.

Though we presented practical results on software implementations, we believe that this technique may also be a threat for hardware coprocessors. Therefore the collision-correlation threat should be taken into consideration by developers and designers during their embedded cryptographic design.

Acknowledgments. The authors would like to thank Sean Commercial for his valuable comments and advice on this manuscript. We would also like to thank the anonymous reviewers of this paper for their fruitful comments and advice.

References

1. Akkar, M.-L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)
2. Bogdanov, A.: Improved side-channel collision attacks on AES. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 84–95. Springer, Heidelberg (2007)
3. Bogdanov, A.: Multiple-differential side-channel collision attacks on AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 30–44. Springer, Heidelberg (2008)
4. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
5. Canright, D., Batina, L.: A Very Compact "Perfectly Masked" S-Box for AES. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 446–459. Springer, Heidelberg (2008)
6. Clavier, C.: An improved SCARE cryptanalysis against a secret A3/A8 GSM algorithm. In: McDaniel, P., Gupta, S.K. (eds.) ICISS 2007. LNCS, vol. 4812, pp. 143–155. Springer, Heidelberg (2007)
7. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
8. Di Golic, J., Tymen, C.: Multiplicative masking and power analysis of AES. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 198–212. Springer, Heidelberg (2003)
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
10. Kocher, P.C., Jaffe, J.M., June, B.C.: DES and Other Cryptographic Processes with Leak Minimization for Smartcards and other CryptoSystems, Journal = US Patent 6,278,783 (1998)
11. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)

12. Mangard, S., Pramstaller, N., Oswald, E.: Successfully Attacking Masked AES Hardware Implementations. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 157–171. Springer, Heidelberg (2005)
13. Messerges, T.S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
14. Messerges, T.S.: Securing the AES Finalists Against Power Analysis Attacks. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001)
15. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
16. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A Side-Channel Analysis Resistant Description of the AES S-Box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
17. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
18. Schramm, K., Leander, G., Felke, P., Paar, C.: A Collision-Attack on AES: Combining Side Channel- and Differential-Attack. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)
19. Schramm, K., Wollinger, T., Paar, C.: A New Class of Collision Attacks and Its Application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
20. Standaert, F.-X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
21. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The World Is Not Enough: Another Look on Second-Order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010)
22. Witteman, M.F., van Woudenberg, J.G.J., Menarini, F.: Defeating RSA multiply-always and message blinding countermeasures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 77–88. Springer, Heidelberg (2011)