Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers*

Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich

Institute for Applied Information Processing and Communciations (IAIK),
Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria
{elisabeth.oswald, stefan.mangard, christoph.herbst,
stefan.tillich}@iaik.tugraz.at

Abstract. In this article we describe an improved concept for second-order differential-power analysis (DPA) attacks on masked smart card implementations of block ciphers. Our concept allows to mount second-order DPA attacks in a rather simple way: a second-order DPA attack consists of a pre-processing step and a DPA step. Therefore, our way of performing second-order DPA attacks allows to easily assess the number of traces that are needed for a successful attack. We give evidence on the effectiveness of our methodology by showing practical attacks on a masked AES smart card implementation. In these attacks we target inputs and outputs of the SubBytes operation in the first encryption round.

1 Introduction

Higher-order DPA attacks were already mentioned in Kocher *et al.*'s pioneering article [KJJ99]: "Of particular importance are high-order DPA functions that combine multiple samples from within a trace." Subsequently, several researchers have tried to implement attacks based on this very brief sketch. Messerges was the first researcher to successfully report on a second-order DPA attack in [Mes00].

Since the publication of these two articles little progress has been made. Only recently, the topic was picked up again, see [ABG04], [WW04], [PSDQ05], [SPQ05] and [JPS05]. However, none of these articles have tackled the practical issues that arise when performing higher-order DPA attacks on software implementations on smart cards.

In this article we present a way to formulate second-order DPA attacks that are practical for smart card implementations. Our attacks are simple to mount, it is easy to assess their complexity and they can be applied to any implementation that uses additive masking as DPA countermeasure. Our results are compelling: we can attack a masked AES implementation on an 8-bit micro controller with

* The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507270 SCARD and through the Austrian Science Fund (FWF) under grant number P16952.

D. Pointcheval (Ed.): CT-RSA 2006, LNCS 3860, pp. 192–207, 2006.

no more than 400 traces. The exact moments of time when intermediate values are being manipulated do not need to be known.

This article is organized as follows. In Sect. 2 we review Messerges' original second-order DPA attack and survey related work. In Sect. 3, we explain our concept of second-order attacks, assess the complexity and formulate various attack scenarios that are relevant for masked implementations of block ciphers. In Sect. 4 we show the results of our new attacks on a masked AES implementation. We conclude this article in Sect. 5. There are two appendices to this paper. In App. A we briefly discuss an alternative pre-processing method. In App. B we provide a graphical description of the AES implementation that is targeted in Sect. 4.

2 Second-Order DPA Attacks

The attack described in [Mes00], targets the exclusive-or (short: XOR) operation of a byte of the key and a byte of masked data. It is assumed that in the implementation under attack, the mask is generated and subsequently exclusive-ored with the data prior to the exclusive-or operation that involves the key byte:

```
t=1: m = rand()  (generate mask-byte)
t=2:  x = p ⊕ m  (XOR mask with plaintext-byte)
t=3:  y = x ⊕ k  (XOR masked plaintext with key-byte)
```

Fig. 1. A code sample of a typical masked key addition

In the attack, the point in the power trace $s_j[t = 1]$ that corresponds to the time when the mask is generated (line 1 in Fig. 1) is subtracted from the point in the power trace $s_j[t = 3]$ that corresponds to the time when the masked data is XORed with the key byte (line 3 in Fig. 1). The joint distribution of these two power samples allows to derive the key-byte bit by bit. For every bit in the plaintext byte the adversary calculates the mean values $\overline{S_0} = \sum_j |s_j[t = 1] - s_j[t = 3]|$ (if the plaintext bit is 0) and $\overline{S_1} = \sum_j |s_j[t = 1] - s_j[t = 3]|$ (if the plaintext bit is 1). If $\overline{S_0} - \overline{S_1} > 0$ then the key bit is 1, otherwise it is zero. A proof for the soundness of the attack is given in [Mes00].

In the attack, it is mandatory to use the absolute value of the differences, because otherwise the difference of means is 0 in both cases. In addition, it is necessary that the mean value of the power traces are roughly the same, otherwise the difference of means also does not lead to conclusive results. This property can be achieved by using the distance-of-mean (short: DOM) test as described in statistic textbooks or by using the Pearson correlation coefficient.

Several questions arise when studying this methodology. An important one for practical attacks is how to identify the interesting points in the power trace? Other questions are how many traces are required for reliable statistics and whether the approach can be improved by using different statistics. In his article, Messerges tries to answer the last of these questions. He concludes that using the absolute value of the difference is a sound approach.

2.1 Related Work

Waddle *et al.* [WW04] were the next to investigate this topic in detail. They investigated how higher-order attacks can be mounted in a way to minimize the additional effort compared to standard DPA attacks. More precisely, their goal was to formulate higher-order attacks as standard DPA attacks with a pre-processing step. Our idea will follow this line of thought.

In their article, Waddle *et al.* suggest to multiply the appropriate points in the power trace in order to produce a DPA peak (this approach was already mentioned by Chari *et al.* in [CJRR99]). Waddle *et al.* also tackle some of the questions that we raised at the end of the previous section. That is, they deal with the issue of finding the interesting points in the power trace. They suggest two methods to find the points of interest. Firstly, they propose the so-called zero-offset 2DPA which works if the masked value and the mask are processed at the same time. If this is the case, there is only one point of interest and the power traces can simply be squared. Secondly, they propose the so-called FFT 2DPA which essentially is a DPA on the FFT (fast fourier transform) of the power traces. For both proposals they investigate how the number of samples needs to be increased for reliable statistics. They conclude for both cases that a significant increase is to be expected due to the pre-processing step.

Peeters *et al.* [PSDQ05] have implemented an attack similar to the zero-offset 2DPA on an FPGA. They have concluded that the zero-offset 2DPA idea works but requires significantly more traces than a standard DPA.

Joye *et al.* [JPS05] have analyzed how the height of the DPA peak is related to the number of samples and the power consumption model under a certain definition of a signal-to-noise ratio.

Summarizing the related work it turns out that so far the arising questions have only been answered in part. Joye *et al.*'s article gives theoretical foundations and allows to assess the efficiency of higher-order attacks in theory in a certain model. Peeters *et al.* have confirmed that some of the ideas of Waddle *et al.* can be applied to FPGA implementations. Messerges has shown that if the points of interest can somehow be found, software implementations on smart cards can be attacked.

We aim to develop an attack strategy for software implementations on smart cards that is versatile, simple to implement, and easy to analyze.

3 Practical Second-Order DPA Attacks

In this section we outline our strategy for second-order DPA attacks. We first explain the assumptions that we make, then we explain the idea of our strategy and last we develop different attack scenarios.

In the following we assume that the instantaneous power consumption of the device under attack depends linearly on the Hamming-weight (short: HW) of the processed data.

Assumption. Let a be a value $\in \{0,1\}^n$ and $C(a)$ denote the power consumption of the value a. Then the power consumption C of the device at the time when a is processed is proportional to the Hamming-weight of the value a : $C(a) \approx HW(a)$.

In this paper, we focus on implementations where $n = 8$, *i.e.* we study 8-bit micro controllers. We use the following simple observation to explain a large class of second-order DPA attacks.

Observation. Let a and b be values $\in \{0,1\}$, let $\oplus$ denote the exclusive-or operation, and let $HW(x)$ denote the Hamming-weight of x. Then the following relation holds with probability one:

$$HW(a \oplus b) = |HW(a) - HW(b)|. \tag{1}$$

Consequently, we can correctly predict $|C(a) - C(b)|$ with $HW(a \oplus b)$ if $a, b \in \{0,1\}$.

We can use this observation to mount second-order DPA attacks: In the first step, the adversary chooses a point in a power trace, subtracts it from the rest of the trace and takes the absolute value of the result. In the second step, the adversary tests for all keys whether the Hamming-weight of the exclusive-or of the two intermediate values under attack correlates to the pre-processed power traces. Only for the correct key and for the correct point, a peak will occur in the power trace. If there is no peak for any key then the attacker chooses another point. We can formalize our approach as follows.

Second-Order DPA Attack. Let $\mathcal{T}$ be the set of power traces that were acquired during the execution of a known algorithm using a set of known texts P_i, using a set of unknown masks M_i and using an unknown key K. We define a standard DPA attack to be a first-order DPA attack that is based on the Pearson correlation coefficient. Let $F_v(P_i)$ denote an intermediate value that is computed by the algorithm with input P_i and with a part of the unknown key K. We attack two intermediate values $F_1(P_i) \oplus M_i$ and $F_2(P_i) \oplus M_i$.

1st Step: We fix an interval I of length l for all power traces $T \in \mathcal{T}$. This interval is determined by an educated guess for the time frame in which $F_1(P_i) \oplus M_i$ and $F_2(P_i) \oplus M_i$ are processed. For each trace T we do the following. We calculate a pre-processed trace that contains all values $|I_a - I_b|$ $\forall I_a, I_b \in I \subseteq T$.

2nd Step: We make a standard DPA attack on the pre-processed power traces. In this attack, we guess a part of the key K to predict the value $HW(F_1(P_i) \oplus F_2(P_i))$.

The value $|C(F_1(P_i) \oplus M_i) - C(F_2(P_i) \oplus M_i)|$ occurs in the pre-processed traces. This value is due to the two attacked intermediate results $F_1(P_i) \oplus M_i$ and $F_2(P_i) \oplus M_i$. In the DPA attack on the pre-processed traces there occur peaks at these positions if the key guess is correct.

Remark. It is important to notice that we have given a description that is more general than Messerges' original approach. Whereas his predictions are always

made for individual bits of one intermediate result, we allow to predict several bits. When using several bits in the attack, then Observation 1 does not hold in general; it only holds for some values.

Remark. The result of the pre-processing step are traces of length $\frac{l(l-1)}{2}$. This is because $|I_a - I_b| = |I_b - I_a|$.

In the following section we look at the complexity of a second-order DPA attack and then, we formulate various second-order DPA attack scenarios that are relevant for masked implementations of block ciphers.

3.1 Complexity of Our Second-Order DPA Attack

The complexity of a DPA attack is typically determined by the number of traces that have to be acquired for a successful attack. Another factor that is relevant for a practical application is the length of the traces. We discuss both complexity aspects.

Number of traces. Our approach of performing second-order DPA attacks consists of two steps. The first step consists of pre-processing the acquired power traces. The second step consists of performing a standard DPA attack. Remember that we have pointed out that we perform this standard DPA attack by predicting several bits of an intermediate value. These predictions do not always coincide with what happens inside the device because Observation 1 does not hold if $a, b \in \{0,1\}^n, n > 1$. We have to compensate these errors by increasing the number of measurements. The pre-processing step potentially increases the noise in the measurements. The effect of pre-processing has been studied in [CCD00] and [Man04]. In these articles the effect of the increase of uncorrelated noise has been investigated. However, in many micro controllers the noise that occurs in subsequent clock cycles is highly correlated. Thus, the increase of the noise due to pre-processing is not necessarily severe.

We conclude that the complexity of the second-order DPA attack is mainly determined by the number of predictions that do not match the internal value that is being processed. The influence of the pre-processing step is small and depends on the device under attack. In the attacks that we have performed in practise it turned out that the pre-processing step has virtually no influence on the number of samples.

Length of traces. In our approach we work with traces of length $\frac{l(l-1)}{2}$. In comparison to a standard DPA attack on an interval of length l, the complexity is squared.

3.2 Attacking One Masked Table Look-Up

One way to protect an implementation of a block cipher against (first-order) DPA attacks is to mask the intermediate values that occur during the computation. This is typically achieved by adding (exclusive-oring) a random value to the plaintext. The description of the block cipher is modified such that it maintains the masking.

Assumptions. Assume that the table S of the original cipher is replaced by a masked table S' such that $S'(X \oplus M) = S(X) \oplus M$ for a fixed mask M. There are two possibilities to attack such a table look-up; either one attacks the first or one attacks the last encryption round. Because both attacks follow the same principle we explain the attack on the first round only.

Attack on the first round. We use the input of the table look-up $P \oplus K \oplus M$ and the output of the table look-up $S'(P \oplus K \oplus M) = S(P \oplus K) \oplus M$ in the first encryption round for our attack. We assume that we have recorded the power trace of the first round of the algorithm.

In the first step of the attack, we locate the sequence of table look-up operations. We make an educated guess for the time frame when $S(P \oplus K) \oplus M$ and $P \oplus K \oplus M$ are computed and perform the pre-processing step. In the second step, we predict $|C(S(P \oplus K) \oplus M) - C(P \oplus K \oplus M)|$ with $HW(S(P \oplus K) \oplus (P \oplus K))$ and perform a standard DPA attack. Therefore we need to know one byte of the plaintext byte and guess one byte of the key.

Number of traces needed. In previous work we have shown that the number of traces in a standard DPA is determined by the correlation ρ between the correct predictions and the traces, see [Man04]. Based on this correlation coefficient, the number of traces can be calculated as follows

$$N = 3 + 8 \left(\frac{Z_\alpha}{\ln \frac{1+\rho}{1-\rho}} \right)^2 . \tag{2}$$

In order to assess the correlation coefficient for this scenario in practice, we study the correlation in the idealized model where $C(a) = HW(a)$ and a has 8 bits. We use the AES S-box for the table S in our calculations. Then, we calculate the correlation between $|HW(S(P \oplus K) \oplus M) - HW(P \oplus K \oplus M)|$ and $HW(S(P \oplus K) \oplus (P \oplus K))$. This can be done easily with a computer. It turns out that the correlation coefficient for a second-order DPA attack on one masked 8-bit table look-up is 0.2405.

Setting $Z_{0.9999} = 3.7190$ and $\rho = 0.2405$, and evaluating (2) shows that $N = 462$ is an upper bound for the number of traces.

The immediate question that arises is whether we could do better by either using less bits in our predictions or by applying a simple but non-linear function (for instance raising to the power β) to our pre-processed data as suggested by [JPS05]. The answer can be easily obtained by calculating the correlation between $|HW(S(P \oplus K) \oplus M) - HW(P \oplus K \oplus M)|^\beta$ and $HW(S(P \oplus K) \oplus (P \oplus K))$. We have calculated this correlation for different values of β and for attacks on different numbers of bits of $S(P \oplus K) \oplus (P \oplus K)$. It turns out that attacking a full byte is the best choice[1] and that varying β does not lead to a significant improvement of the correlation coefficient (see Tab. 1).

[1] In App. A we show that using multiplication for pre-processing, such as suggested in [WW04] and [CJRR99], leads to smaller correlations.

Table 1. Exact correlation values for the scenario described in Sect. 3.2. The correlation increases when more bits are used in the prediction. The correlation increases slightly for $\beta = \{2, 3\}$ but decreases for higher values of β.

β	1	2	3	4	5	6
1 Bit	0.0861	0.0985	0.0950	0.0869	0.0775	0.0685
2 Bits	0.1119	0.1315	0.1283	0.1189	0.1080	0.0972
3 Bits	0.1415	0.1652	0.1604	0.1482	0.1341	0.1203
4 Bits	0.1723	0.1914	0.1834	0.1674	0.1496	0.1327
5 Bits	0.1936	0.2100	0.2003	0.1822	0.1623	0.1435
6 Bits	0.2092	0.2291	0.2186	0.1987	0.1767	0.1559
7 Bits	0.2278	0.2460	0.2341	0.2125	0.1887	0.1661
8 Bits	0.2405	0.2622	0.2501	0.2273	0.2021	0.1782

3.3 Attacking Two Masked Table Look-Ups

Assumptions. Assume that the table S of the original cipher is replaced by another masked table S' such that $S'(X \oplus M) = S(X) \oplus M'$ for fixed masks M and M'. The outputs of two table look-ups are then $S(X_1) \oplus M'$ and $S(X_2) \oplus M'$. There are two possibilities to attack the table look-up outputs of such an implementation. One possibility is to target two different table look-up outputs in the first encryption round. The second option is to attack one table look-up output in the first and one table-lookup output in the last encryption round.

Attack on the first round. Assume that we attack the outputs of two table look-ups in the first encryption round. Hence we use $S(P_1 \oplus K_1) \oplus M'$ and $S(P_2 \oplus K_2) \oplus M'$ in our attack. In the first step we locate the sequence of table look-ups and make an educated guess for the time frame when $S(P_1 \oplus K_1) \oplus M'$ and $S(P_2 \oplus K_2) \oplus M'$ are computed and perform the pre-processing. In the second step we predict $|C(S(P_1 \oplus K_1) \oplus M') - C(S(P_2 \oplus K_2) \oplus M')|$ with $HW(S(P_1 \oplus K_1) \oplus S(P_2 \oplus K_2))$ and perform a standard DPA attack. We need to know two bytes of plaintext and guess two bytes of the key for this attack.

Attack on the first and the last round. Assume that we use the output of one S-box in the first encryption round and the output of one S-box in the last encryption round. Hence, we use $S(P \oplus K_1) \oplus M'$ and $S(C \oplus K_2) \oplus M'$ in our attack. In the first step, we locate the first and the last encryption round and perform the pre-processing step. In the second step, we predict $|C(S(P \oplus K_1) \oplus M') - C(S(C \oplus K_2) \oplus M')|$ with $HW(S(P \oplus K_1) \oplus S(C \oplus K_2))$ and perform a standard DPA attack. Therefore, we need to know one byte of the plaintext and one byte of the ciphertext and we need to guess one byte of the key of the first round and one byte of the key in the last round.

Number of traces needed. Because the key guess in this scenario is based on 16 bits, it gets impractical to calculate the correlation coefficient exactly. Therefore, we have decided to estimate it based on 100000 plaintexts. The estimation of the correlation between $|HW(S(P \oplus K_1) \oplus M') - HW(S(C \oplus K_2) \oplus M')|^\beta$ and $HW(S(P \oplus K_1) \oplus S(C \oplus K_2))$ leads to the values shown in Tab. 2.

Table 2. Simulated correlation values for the scenario described in Sect. 3.3. The correlation increases when more bits are used in the prediction. The correlation increases slightly for $\beta = \{2, 3\}$ but decreases for higher values of β.

β	1	2	3	4	5	6
1 Bit	0.0851	0.0894	0.0944	0.0788	0.0698	0.0587
8 Bits	0.2322	0.2563	0.2517	0.2265	0.2043	0.1755

Because we only want to illustrate that the correlation coefficients are approximately the same as in the previous attack, we only give the numbers for attacking 1 bit and for attacking 8 bits.

3.4 Attacking a Masked Key Addition

This scenario is the same as the one described by Messerges in [Mes00].

Assumptions. The plaintext P is concealed with a random mask M: $P \oplus M$. During the key addition, the masked plaintext is exclusive-ored with the key: $P \oplus M \oplus K$. The manipulation of M and the computation of $P \oplus M \oplus K$ occur somewhen during the (initial) phase of the algorithm. We assume for the attack that we have recorded the power trace of the initial phase of the algorithm.

Attack on the key addition. We use the value of the mask M and and the value $P \oplus M \oplus K$ of the key addition in our attack. In the first step, we locate the sequence of key addition operations. We make an educated guess for the time frame when M and $P \oplus M \oplus K$ are computed. In the second step, we predict $|C(M) - C(P \oplus M \oplus K)|$ with $HW(P \oplus K)$ and perform a standard DPA attack. For the prediction we need to know one bit of the plaintext and we need to guess one bit of the key.

Number of traces. In this scenario we can only attack one bit of an intermediate result at a time. We have calculated the correlation between $HW(P \oplus K)$ and $|HW(M) - HW(P \oplus M \oplus K)|^{\beta}$ for different values of β. Tab. 3 shows the results for different values of β.

Table 3. Exact correlation values for the scenario described in Sect. 3.4. The correlation increases only slightly for $\beta = \{2, 3\}$ but decreases for higher values of β.

β	1	2	3	4	5	6
1 Bit	0.0846	0.0912	0.0879	0.0806	0.0717	0.0626

4 Attacking a Masked AES Smart Card Implementation

In order to verify the theoretical discussions presented in Sect. 3, we have performed these attacks in practice. The target of these attacks was a masked AES

smart card implementation. This implementation is described in Sect. 4.1. For the attacks, we have executed this implementation of AES on a micro controller whose power consumption is proportional to the Hamming weight of the data it processes. The results of two second-order DPA attacks on this implementation are reported in Sect. 4.2 and 4.3.

4.1 Masking AES in Software for a Smart Card Implementation

In our masked software implementation of AES the inputs and outputs of each operation are masked additively. In the following paragraphs we briefly sketch how the masked versions of the four AES operations have been implemented. A graphical description can be found in App. B.

Masked AddRoundKey: The AddRoundKey operation does not change the mask and therefore it does not require special attention in our masked implementation. Essentially, we use the same AddRoundKey operation as in an unmasked implementation.

Masked SubBytes: We mask the SubBytes operation S with values M and M' (the masks). Therefore, we have to derive a new masked S-box S' with the property that $S'(X \oplus M) = S(X) + M'$.

Masked ShiftRows: The ShiftRows operation is done in combination with the SubBytes operation by reading and writing the state bytes in a specific order. Therefore, no separate masking effort is required for ShiftRows.

Masked MixColumns: As MixColumns is a linear operation it is sufficient to calculate the MixColumns operation with the used masks in addition to the normal calculation with the masked AES state. In order to minimize the overhead for calculation, we make sure that the state before MixColumns is always masked with the same four values. Hence, the output of MixColumns is also masked with the same four masks. The four output masks only need to be calculated once per AES encryption or decryption. The reason for using four different masks for a column is that the four bytes of the column are combined with each other during the MixColumns operation. If the same mask would be used on each byte of the column, then intermediate values of MixColumns could be processed unmasked if the masks cancel each other out.

4.2 Attacking Two S-Box Outputs in the First Encryption Round

In the scenario described in Section 3.3, we predict the power consumption by calculating the Hamming-weight of the exclusive-or of two outputs of the masked SubBytes operation: $HW(S(P_1 \oplus K_1) \oplus S(P_2 \oplus K_2))$. Using this technique, we have targeted the first two key bytes of our masked AES implementation during the first encryption round.

In order to reduce the computational effort that is needed for the attack, we have first made an educated guess for the time frame when $S(P_1 \oplus K_1) \oplus M'$ and $S(P_1 \oplus K_1) \oplus M'$ are computed. For this purpose, we have measured the power

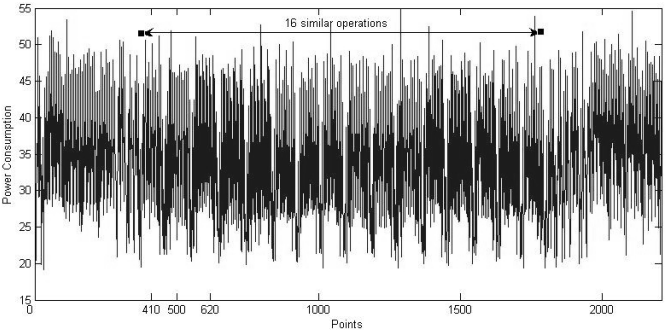

Fig. 2. A part of the first encryption round. A sequence of 16 similar operations is clearly visible in the power trace.

consumption during one execution of AES. We have inspected the power trace and we have deduced when the first round takes place. Within the first round (see Fig. 2), there is a distinct part where 16 similar operations take place. This part corresponds to the operations AddRoundKey and SubBytes. These operations are executed in combination for each byte of the AES state. Since we decided to target the outputs of the first two SubBytes operations, we selected the interval 410 to 620 of the power trace (see Fig. 2) for our attack.

After having selected this interval, we have made 3000 measurements of the power consumption of the micro controller while it was performing AES encryptions of random plaintexts. Subsequently, we have performed the pre-processing operation described in Section 3. This means that for each of the 3000 power traces we have calculated the absolute value of the difference of all pairs of points in the interval 410 to 620. We have done this computation by first subtracting the points 411 to 620 from the point 410. We refer to the absolute value of these differences as a segment. The segment that has been calculated based on point 410 consists of 210 values.

The next segment was calculated based on point 411. This segment contained the absolute value of the difference between the points 412 to 620 and 411. It consists of 209 values. Following this strategy, we have calculated corresponding segments based on all remaining points in the interval from 412 to 620. This lead to 210 segments in total, where the largest segment consisted of 210 values and the smallest one consisted of just one value.

For the attack, we have concatenated the 210 segments of each power trace. After the pre-processing step, we therefore had 3000 traces, where each trace consisted of 210 segments. Based on these traces, a standard DPA attack making hypotheses about the value of $HW(S(P_1 \oplus K_1) \oplus S(P_2 \oplus K_2))$ has been performed. Since the intermediate result $HW(S(P_1 \oplus K_1) \oplus S(P_2 \oplus K_2))$ depends on two key bytes, 65536 key guesses were necessary.

The result of this attack for the correct key guess is shown in Fig. 3. Several peaks are clearly visible in this figure. There is more than one peak in the result because the targeted micro controller manipulates the two attacked intermediate

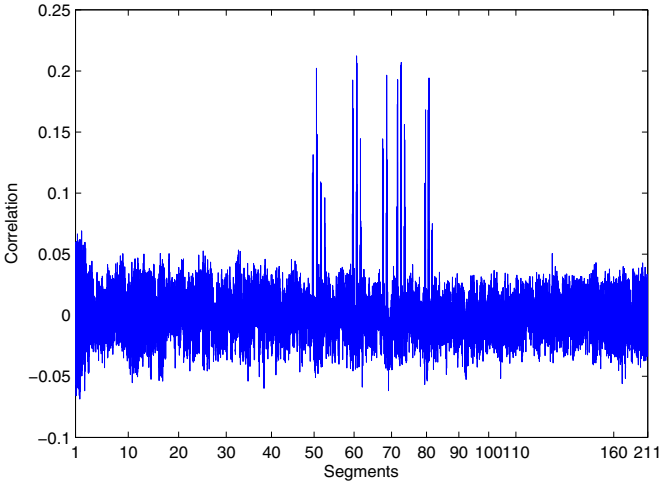

Fig. 3. Result of a second-order DPA attack on the interval 410 to 620 of the original power traces

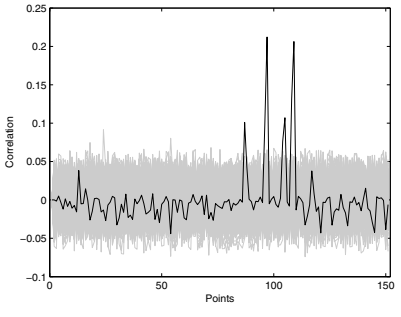

Fig. 4. The result of all 65536 key guesses in an attack on segment 61

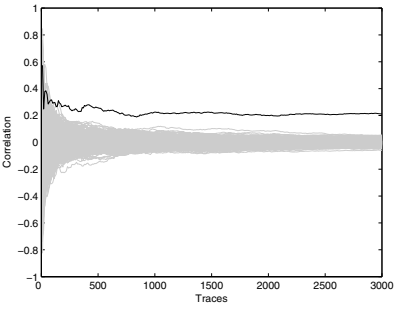

Fig. 5. Correlation coefficients for all 65536 keys depending on the number of power traces that are used in the attack

results in more than two clock cycles. The highest peak that is shown in Fig. 3 is located in segment 61 and has the value 0.21. This segment contains the result of the second-order DPA attack mounted based on the absolute values of the differences between the points 471 to 620 and 470 in the original traces. The segment consists of 150 values.

In order to show that only the correct key produces a peak in this second-order attack, we have attacked this segment based on all 65536 key hypotheses. The results of this attack are shown in Fig. 4. The results for the 65535 incorrect keys are plotted in gray. The result for the correct key is plotted in black. It can be observed that indeed only the correct key leads to significant peaks.

We have also analyzed how many samples are needed to obtain a significant peak in segment 61. Figure 5 shows how the correlation coefficients evolve depending on the number of used power traces. The correlation coefficient for the correct key guess is shown in black. The correlation coefficients for the incorrect key guesses are shown in gray. It can be observed in Fig. 5 that roughly 400 traces are needed to perform a second-order DPA attack on the output of two S-box operations. This confirms our theoretical estimate for the correlation coefficient and the number of samples given in Sect. 3.2.

For our attack we have used a standard PC with 2 GB of RAM and a standard digital oscilloscope. It took us roughly one hour to make the 3000 measurements. We compressed the power traces by integrating the absolute values of each clock cycle. The compression step required about 23 minutes. The pre-processing step for the second-order DPA attack (*i.e.*, the calculation and concatenation of the segments) took about 5 minutes. Attacking segment 61 based on 65536 key hypotheses took less than two minutes. An attack on all 210 segments can be performed within a few hours. Hence, this type of attack can be easily performed in practice. The time that is needed for the attack is mainly determined by the transfer speed of the hard disk. On a standard PC not all power traces and hypotheses can be kept in memory simultaneously.

4.3 Attacking an S-Box in the First Encryption Round

In the scenario that is described in Sect. 3.2, the power consumption of the attacked device is predicted based on the Hamming-weight of the exclusive-or of the input and output of a masked SubBytes operation: $(P_1 \oplus K_1) \oplus S(P_1 \oplus K_1)$.

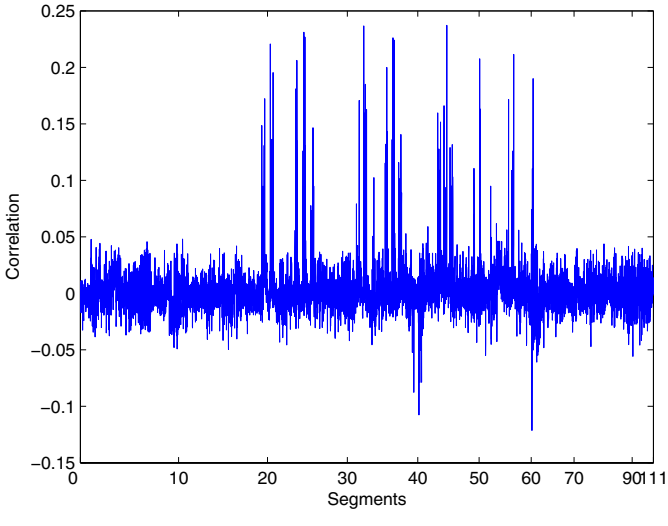

Fig. 6. Result of a second-order DPA attack on an interval of 111 points of the original power traces

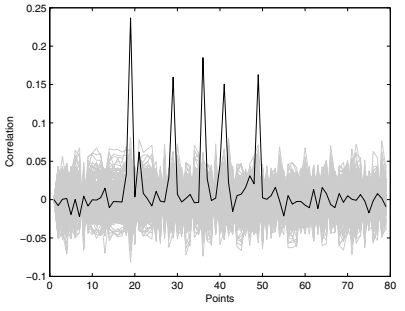

Fig. 7. The result of all 256 key guesses in an attack on segment 33

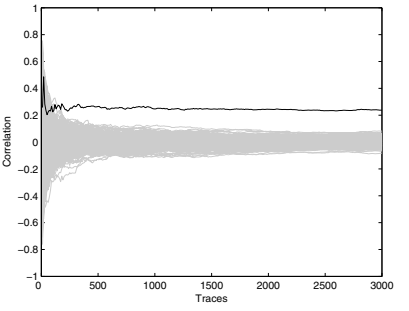

Fig. 8. Correlation coefficients for all 256 key guesses depending on the number of power traces that are used in the attack

We have targeted the first encryption round in our analysis and we have again acquired 3000 power traces of the attacked device.

Based on visually inspecting the power traces, we have made an educated guess for the time frame when the input and output of the attacked S-box is processed. We have considered an interval of 111 points for the attack. Just like in the previous section, we have performed a pre-processing step to calculate the absolute value of the difference between all possible pairs of points in this interval.

Based on each power trace, 110 segments have been calculated and concatenated. The resulting traces have been used as input for a standard DPA attack predicting the Hamming-weight of $(P_1 \oplus K_1) \oplus S(P_1 \oplus K_1)$. The result for the correct key guess is shown in Fig. 6. As expected, there are again several peaks visible in this trace. The highest peak occurs in segment 33 and has the value 0.24.

We have performed an attack on this segment based on all 256 key guesses. The result of this attack is shown in Fig. 7. The result for the correct key guess is shown in black. The result for the other key guesses is shown in gray. Figure 8 shows how the correlation coefficients depend on the number of used power traces. It can be observed that just like in the previous attack approximately 400 traces suffice to identify the correct key.

5 Conclusion

In this article, we have presented a way to formulate second-order DPA attacks that are practical for smart card implementations. Our attacks are simple to mount, it is easy to assess their complexity and they can be applied to any implementation that uses additive masking as DPA countermeasure. Our results are compelling: we can attack a masked AES implementation with no more than 400 traces without needing to know the exact moments when intermediate values are being manipulated.

Our work clearly shows that second-order DPA attacks are a practical threat for masked software implementations. Consequently, masking by itself is insuf-

ficient to protect masked smart card implementation of block ciphers against power-analysis attacks.

References

[ABG04] Mehdi-Laurent Akkar, Régis Bevan, and Louis Goubin. Two Power Analysis Attacks against One-Mask Methods. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 2004.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. On Second-Order Differential Power Analysis. In *Cryptographic Hardware and Embedded Systems - Proceedings of CHES 2005*. Springer, 2005.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Man04] Stefan Mangard. Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[Mes00] Thomas S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

[PSDQ05] Eric Peeters, Francois-Xavier Standaert, Nicolas Donckers, and Jean-Jacques Quisquater. Improved Higher Order Side-Channel Attacks with FPGA experiments. In *Cryptographic Hardware and Embedded Systems - Proceedings of CHES 2005*. Springer, 2005.

[SPQ05] Francois-Xavier Standaert, Eric Peeters, and Jean-Jacques Quisquater. On the Masking Countermeasure and Higher-Order Power Analysis Attacks. In *ITCC 2005*, 2005.

[WW04] Jason Waddle and David Wagner. Towards Efficient Second-Order Power Analysis. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.

A Using Multiplication as Pre-processing Method

Waddle *et al.* [WW04] and Chari *et al.* [CJRR99] have suggested to use multiplication as a pre-processing method. In order to illustrate that this is inferior to using the absolute difference, we study this technique for the scenario described in Sect. 3.2.

With multiplication as pre-processing method, we use $HW(S(P \oplus K) \oplus (P \oplus K))$ to predict $(HW(S(P \oplus K) \oplus M) * HW(P \oplus K \oplus M))^{\beta}$. It is easy to exactly calculate the correlation between $HW(S(P \oplus K) \oplus (P \oplus K))$ and $(HW(S(P \oplus K) \oplus M) * HW(P \oplus K \oplus M))^{\beta}$ with a computer. We performed this calculation for different values of β and for attacks on different numbers of bits of $S(P \oplus K) \oplus (P \oplus K)$. The results are given in Tab. 4. The correlation coefficients are clearly much lower than the ones in Tab. 1 which means that this pre-processing technique is less effective.

Table 4. Exact correlation values for the scenario described in Sect. 3.2 when multiplication is used as pre-processing method. The correlation increases when more bits are used in the prediction. The correlation increases slightly for $\beta = \{2, 3, 4\}$ but decreases for higher values of β.

β	1	2	3	4	5	6
1 Bit	-0.0327	-0.0531	-0.0627	-0.0644	-0.0610	-0.0551
2 Bits	-0.0437	-0.0706	-0.0830	-0.0846	-0.0797	-0.0717
3 Bits	-0.0548	-0.0888	-0.1045	-0.1069	-0.1010	-0.0911
4 Bits	-0.0636	-0.1032	-0.1223	-0.1261	-0.1202	-0.1096
5 Bits	-0.0698	-0.1134	-0.1346	-0.1391	-0.1330	-0.1215
6 Bits	-0.0761	-0.1236	-0.1468	-0.1517	-0.1449	-0.1323
7 Bits	-0.0817	-0.1328	-0.1579	-0.1634	-0.1563	-0.1429
8 Bits	-0.0871	-0.1415	-0.1681	-0.1737	-0.1660	-0.1515

B Graphical Description of the Masked AES Implementation

Figure 9 shows, in correspondence with the notation used in Sect. 4, how our masked AES implementation works.

We use six masks in total. The masks M and M' are the input and output masks for SubBytes, respectively. At the start of an AES encryption, the masked S-box is calculated for M and M'. The masks M1, M2, M3, and M4 are the input masks for MixColumns. The corresponding output masks M1', M2', M3', and

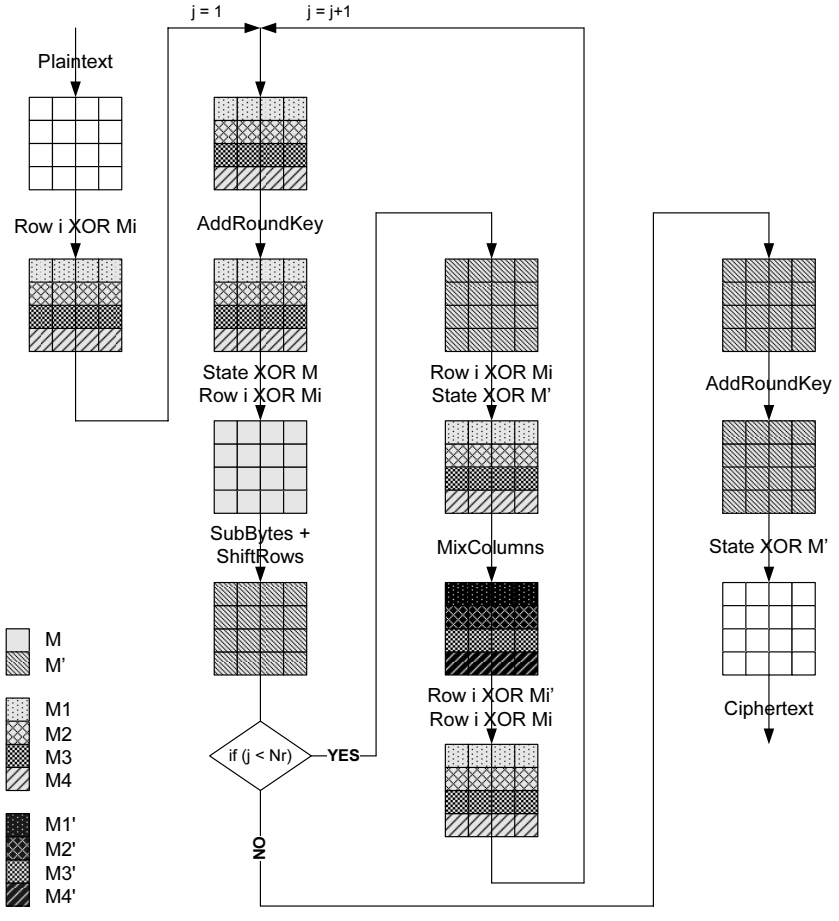

Fig. 9. Graphical description of the masked AES implementation

M4' are determined by applying MixColumns to the input masks. Fig. 9 shows how all AES operations are masked and how the masks are changed between the operations.

We have verified that our masked AES implementation is secure against first-order DPA attacks by attacking all intermediate values.