Attacking State-of-the-Art Software Countermeasures—A Case Study for AES

Stefan Tillich and Christoph Herbst

Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A–8010 Graz, Austria
{Stefan.Tillich,Christoph.Herbst}@iaik.tugraz.at

Abstract. In order to protect software implementations of secret-key cryptographic primitives against side channel attacks, a software developer has only a limited choice of countermeasures. A combination of masking and randomization of operations in time promises good protection and can be realized without too much overhead. Recently, new advanced DPA methods have been proposed to attack software implementations with such kind of protection. In this work, we have applied these methods successfully to break a protected AES software implementation on a programmable smart card. Thus, we were able to verify the practicality of the new attacks and to estimate their effectiveness in comparison to traditional DPA attacks on unprotected implementations. In the course of our work, we have also refined and improved the original attacks, so that they can be mounted more efficiently. Our practical results indicate that the effort required for attacking the protected implementation with the examined methods is more than two orders of magnitude higher compared to an attack on an unprotected implementation.

Keywords: Advanced Encryption Standard, smart card, side channel attacks, power analysis, software countermeasures, masking, operation randomization, advanced DPA attacks.

1 Introduction

Today, an increasing amount of data is processed and distributed in electronic form. This trend is driven by advances in digital microprocessing and network technologies, which are leading us towards visions of "ubiquitous computing" and "ambient intelligence". One of the most pressing problems on the way to realizing these visions is the challenge of security. Cryptographic algorithms are an indispensable tool to establish reasonable security assurances for digital data, e.g. privacy, integrity, and authenticity. The presumption of most cryptographic methods is that the employed key is only known to authorized entities. A fundamental principle of cryptography is that cryptographic algorithms are designed in such a way that observable cryptographic data (e.g. the ciphertext) contains as little information about the key as possible.

E. Oswald and P. Rohatgi (Eds.): CHES 2008, LNCS 5154, pp. 228–243, 2008.

However, in practice keys have to be stored on physical devices like PCs or smart cards and it has been shown that the physical properties (the so-called side channels) of these devices can be exploited to extract information about the cryptographic keys they contain. Amongst such side channel attacks, power analysis developed by Kocher et al. [8] has proven to be a very potent method. The improvement of such attacks and possible countermeasures as defence against them has since been the topic of a wide array of scientific publications.

In power analysis, an attacker has to record the power consumption of a device while it performs cryptographic operations with an unknown key. A particularly powerful attack method is Differential Power Analysis (DPA) [8], which predicts intermediate values of the cryptographic algorithm and an according power consumption and matches it against the recorded power traces. In this fashion, the used key can be recovered even if the relevant information is deeply buried within noise.

Two principal countermeasures have been proposed against power analysis: Masking and hiding [9]. Masking tries to break the link between the predicted intermediate values and the values processed by the device. Hiding seeks to minimize the effect of the processed values on the power consumption. Many specific countermeasures have been proposed on different levels for hardware and software. For software implementations on a given platform, the options tend to be limited to masking schemes and to hiding through the randomization of executed operations in time.

Masking schemes split each intermediate value in a number of shares, which are then processed independently. Only by combining all the shares, the original value can be reconstructed. In its simplest form, a value a is split into two shares $a \circ m$ and m, where m is a random mask, so that $a = (a \circ m) \circ m$. A common choice for the operation $\circ$ is the logical XOR (Boolean masking). Masking is generally susceptible to higher-order DPA attacks. Such attacks combine information of the power consumption of the different shares (higher-order DPA preprocessing) so that the resulting power consumption is again dependent on the unprotected value a and thus susceptible to a "normal" 1st-order DPA attack.

As the effort for higher-order DPA attacks is expected to grow exponentially with the order, it is assumed that a masking scheme with enough shares will make practical attacks infeasible. A higher-order masking scheme for AES [10] based on this idea has been developed by Schramm et al. [13]. However, Coron et al. have demonstrated that this scheme is susceptible to 3rd-order DPA attacks irrespective of the number of used shares [3]. Another problem is posed by the large computational overhead which is required for refreshing the masks. In [13], it has been shown that a single AES encryption with resistance against 2nd-order DPA attacks requires over 40 times more clock cycles (about 200,000 clock cycles in total).

Irrespective of the security aspects, overheads of this order are not likely to be acceptable for every implementation. Therefore, it is necessary to resort to more "light-weight" countermeasures. A possible solution which requires significantly

less overhead is a combination of 1st-order masking and operation randomization as proposed in [4]. Advanced DPA attacks targeted at such a combination of masking and randomization have been proposed in [15], but so far no practical evaluation of their effectiveness has been available. The work described in this paper puts these new attacks to the practical test. Our goals were twofold: First, we sought to verify that these attacks are practicable in a state-of-the-art measurement setup. Second, we wanted to collect empirical evidence for the degree of protection offered by this combination of countermeasures. Note that we did not have the goal to develop new attacks on these countermeasures. Furthermore, we stress that—as with any practical evaluation—our results may not necessarily be optimal for the targeted device. Therefore, the increase in the number of required power traces for the protected implementation indicated in Section 5 should not be taken as a fixed "security gain factor" but only as an upper border of this factor.

The rest of this paper is organized as follows. In Section 2, we describe the protected software implementation of AES which we attacked and we give details on the countermeasures. The advanced DPA methods which have been proposed to break these countermeasures and which we have evaluated practically are presented in Section 3. Some details on the attacked smart card device (especially a characterization of its power leakage) are given in Section 4. In Section 5, we present the results of our practical work. A further discussion of some issues relating to the effectiveness of the attacks in dependence on the attacker's capabilities follows in Section 6. Finally, conclusions are drawn in Section 7.

2 Protected AES Software Implementation

The protection of our AES software implementation is based on the strategy of combining masking, shuffling and dummy operations as published in [4]. At the beginning and at the end of the AES operations there are so-called "randomization zones". Within each zone all intermediate values are protected with a single mask and the sequence of processing of the bytes of the State is randomized. In [4], the initial zone extends to the first MixColumns transformation. Jaffe showed that it is possible to attack the AES after the MixColumns operation [6]. This principally means that a protection of the first round alone would be insufficient. However, Tillich et al. showed that it is quite easy to extend this randomization zone beyond the second SubBytes operation [15], thus thwarting Jaffe's attack[1].

We are using the same masking scheme with six different random masks[2] as published in [4]. All intermediate values of the State and the key are masked. In our implementation, the randomization is achieved by shuffling of the sequence of operations on the bytes of the State. Furthermore, it would be possible

[1] The randomization zone at the end can be extended in a similar fashion to protect against Jaffe's attack.

[2] One mask for S-box inputs, one for S-box outputs, and one for each State row before MixColumns. All other occurring masks (e.g. after MixColumns) are derived from these six masks.

to process additional dummy States to increase the degree of randomization. The randomization is controlled by random values which are—similarly to the mask bytes—unknown to the attacker. We denote the degree of randomization with R, i.e. the number of points in time where a specific State byte can be processed during a specific AES encryption. In our case, the 16 bytes of the AES State are fully shuffled and no dummy States are added. This means that the randomization degree $R = 16$. Hence, a specific byte of the State will be processed with a probability of $p = \frac{1}{16}$ at one of the 16 possible points in time. One protected AES encryption requires less than 12,000 clock cycles on our targeted platform, which is described in Section 4. The cost per additional dummy State would be about 1,000 clock cycles.

The consequences of this randomization for a plain 1st-order DPA attack are illustrated in the following. In Figure 1, the result of an attack on the first S-box output of an unmasked and randomized implementation is shown. Note that only the section of the trace which corresponds to the SubBytes transformation was used. One can clearly identify four groups of peaks. When zooming into one of these groups, again four pairs of peaks can be identified. Each group of peaks corresponds to the transformation of a single State column, which again encompasses four State bytes. As a specific State byte can occur at any of these times, there are all in all 16 different positions where the output value of the unmasked S-box correlates. Furthermore if we compare the achieved correlation of this attack to the achieved correlation of an attack on the unprotected implementation (cf. also Figure 5 in Section 4) we end up with a correlation which is reduced by a factor of approximately 16. As expected, the correlation coefficient scales down linearly with the degree of randomization R [9].

3 Description of the Advanced DPA Attacks

From an implementor's view, a combination of masking and operation randomization countermeasures is a good bet for protecting software implementations of secret-key cryptographic algorithms against power analysis. Proper masking can prevent 1st-order DPA attacks while the randomization of operations in time can offer some protection against more elaborate attacks like higher-order DPA and template-based methods. At the same time, the implementation complexity and overhead can be kept within somewhat acceptable limits.

Higher-order DPA attacks and template attacks have been shown to be very effective to circumvent masking. In higher-order DPA attacks, power leakage of several intermediate values is combined in such a way that the resulting power consumption value is again dependent on the original unmasked value [7,12,14,16]. Template-based methods can be applied to enhance higher-order DPA attacks or to make first-order DPA attacks feasible [9,11]. On the other side, the technique known as "windowing" is a good way to limit the protection of randomization of operations [2]. In this method, all R possibilities of appearance of a protected value are considered and combined so that the effective protection of the countermeasure is lowered.

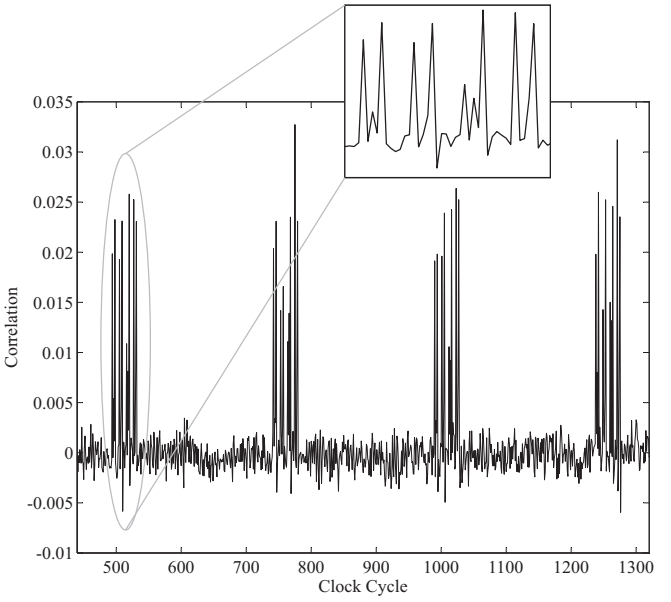

Fig. 1. Result of DPA attack on randomized AES implementation using the Hamming weight model and compressed traces

It has been shown in [15] that the attacks on masking and randomization can be combined to form effective attacks on implementations which employ a single mask and which randomize the course of operations. Three possible combinations have been presented and their effectiveness has been compared by estimations techniques and high-level simulation which neglected electronic noise.

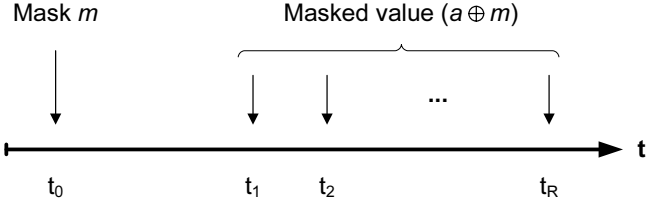

Fig. 2. Points in time relevant for an attack

Figure 2 shows the timeline for a part of the execution of a protected secret-key cipher implementation. Towards the beginning at time index t_0, the mask m is processed in some form (it is generated, stored, used in some precomputation, etc.). Subsequently, the intermediate value a masked with m appears. The occurrence of this masked intermediate value is protected by randomization, i.e.

in each execution the specific value $a \oplus m$ can appear in any of the R points at the times $t_1..t_R$ with equal probability $p = \frac{1}{R}$. The power consumption at these points in time is used in all attacks from [15]. An adversary must therefore be able to find these points in time. As will be discussed in Section 6, the constraints for $t_1..t_R$ can be relaxed, so that knowledge of the exact time indices is not necessary. The attacks are described in the following.

3.1 Biasing Masks and Windowing followed by 1st-Order DPA Attack

One precondition for effective masking is that the used masks must be uniformly distributed. If this condition is not met, 1st-order DPA attacks can become feasible. Therefore, the principal idea of this attack is to try to determine the Hamming weight of the used mask and to select a subset of the collected power traces, where the mask fulfills some property, e.g. has a high Hamming weight. This selection of power traces effectively equals a bias which is introduced into the distribution of the mask values. The selected power traces are then only protected by the randomization of the operations ($t_1..t_R$), whose effect can be minimized by windowing. A subsequent 1st-order DPA attack can be successfully applied on the selected power traces. This attack has been shown to be rather effective under most circumstances in [15].

3.2 2nd-Order DPA Attack followed by Windowing

The idea of this attack is to take the randomization of the operations into account during 2nd-order DPA preprocessing. The power consumption values for m at t_0 and $a \oplus m$ at $t_1..t_R$ are pairwise combined (2nd-order DPA preprocessing). For each pair, this preprocessing results in a joint leakage value of the two points. If the correct points in time have been chosen, one of these R joint leakage points is always dependent on the actual unmasked value. Which of these points is the correct one is determined by the randomized course of operations of the corresponding execution. When all points are windowed (i.e. summed up), the correct one is inevitably included and the resulting power consumption is to some degree dependent on the unmasked value. A 1st-order DPA attack can then be mounted to determine the correct key hypothesis. This attack has been evaluated in [15] to be less effective than the first one in most cases.

3.3 Windowing followed by 2nd-Order DPA Attack

A third option for attacking is to reverse the order of windowing and 2nd-order DPA preprocessing. First, randomization is compensated by summing up all R points in time where the targeted masked value $a \oplus m$ can occur. Then, 2nd-order DPA preprocessing is performed with this sum and a point of the power trace which depends on the corresponding mask m. The result of this preprocessing can be attacked with a 1st-order DPA attack. In [15], this attack variant has been shown to be rather ineffective in comparison to the previous two methods.

4 Attacked Device

The device used to implement and attack the protected AES software implementation is a smart card with an ATMega163 core [1]. The ATMega163 is an 8-bit microcontroller based on AVR, which is a single-cycle instruction RISC architecture from Atmel. It is equipped with 1,024 bytes of internal RAM, 16 KB in-system self-programmable FLASH and 512 bytes of EEPROM. The core of the controller contains 32 general purpose registers which are directly connected to the arithmetic-logic unit (ALU). Three register pairs can be used to store a 16-bit address into the internal memory.

The used smart card is shipped without any software or operating system. This means the card is under full control of the designer and all parts of the software (including boot code and operating system) have to be implemented from scratch. In our scenario, there is only a minimum version of an operating system implemented which can handle the basic functions of the T=1 protocol specified in ISO 7816 [5]. The card can execute the protected AES implementation described in Section 2. For the sake of performance, the randomization parameters and mask bytes are sent to the smart card along with the plaintext. The key used for the AES encryption is stored in the EEPROM of the smart card.

In general, the device leaks the Hamming weight as well as the Hamming distance of the processed values. When attacking the non-randomized S-box output using the Hamming weight model, the maximum achievable correlation is 0.458. In a similar attack using the Hamming distance between the S-box input and the S-box output—which occur as subsequent values of a register—the maximum correlation is 0.257. The results for these attacks on uncompressed traces are displayed in Figure 3 and in Figure 4. It can be seen that for this device and the sequence of instructions used to implement the S-box lookup, the Hamming weight model leads to a higher correlation.

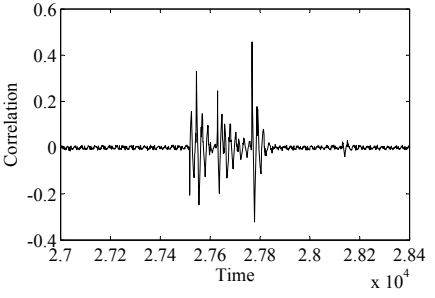

Fig. 3. Result of DPA attack on unprotected AES implementation using the Hamming weight model and uncompressed traces

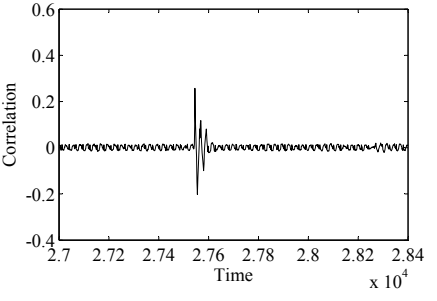

Fig. 4. Result of DPA attack on unprotected AES implementation using the Hamming distance model and uncompressed traces

For minimization of the amount of data used for an attack, it is a common technique to compress the measured traces [9]. When using the compression described in Section 5.2, the achievable correlation using the Hamming weight model reduces from 0.458 to 0.383. With the Hamming distance model the correlation reduces from 0.257 to 0.236. The results of an attack on compressed traces can be seen in Figure 5 and Figure 6. For our practical attacks, this means that most of the information is preserved in the compressed power traces.

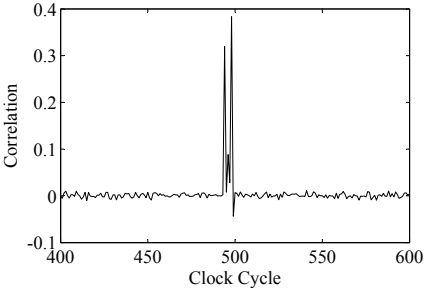

Fig. 5. Result of DPA attack on unprotected AES implementation using the Hamming weight model and compressed traces

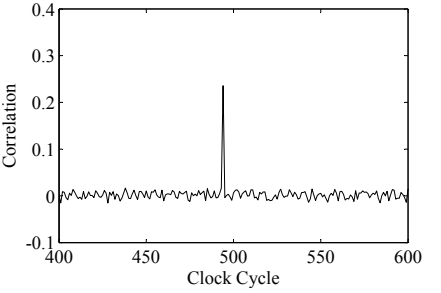

Fig. 6. Result of DPA attack on unprotected AES implementation using the Hamming distance model and compressed traces

5 Practical Results

In order to demonstrate the effectiveness and practicability of the methods described in [15], we have attacked the protected AES software implementation presented in Section 2. For the randomization of operations, we have used full shuffling of the 16 State bytes, i.e. $R = 16$. Power traces were collected with a LeCroy LC584AM digital oscilloscope and a differential probe by measurement over a $1\,\Omega$ resistor in the ground line of the smart card reader. A trigger signal has been supplied by the smart card at the beginning of encryption. We have collected a set of 500,000 power traces, which took about 134 hours in our measurement setup, i.e. a rate of approximately one trace per second. The uncompressed traces required about 50 GB of disk space. For comparison, a set of compressed traces was between 700 MB and 2 GB in size, depending on the actual compression function. An uncompressed trace contained 100,000 points, whereas a compressed trace consisted of about 1,800 points (one per clock cycle).

The power traces included about 1,800 clock cycles at the start of AES encryption spanning over various precomputations (parts of the masked key scheduling, mask preprocessing, and the masking of the plaintext), the initial AddRoundkey, and the first AES round. In order to keep the size of the traces small, the sampling rate has been limited to $200 \cdot 10^6$ samples/second.

All statistical analyses were carried out on a PC featuring a quad-core Intel Xeon processor at 2.33 GHz and 8 GB of RAM. Attack times were generally determined by the number and size of the analyzed traces, and not by the kind of statistical analysis. An attack using all 500,000 power traces took about 140 s for compressed traces and about 1,7 hours for uncompressed traces. For an attack with biased masks, the template-building took about 160 s when 100 traces were used for each of the nine templates. The time for attacks involving fewer traces would scale down almost linearly.

For our attacks we have used the S-box output of the first round as intermediate value a and the S-box output mask as corresponding m. The time indices $t_1..t_{16}$ for $a \oplus m$ were determined by 1st-order DPA attacks using the known masked S-box outputs as attacked intermediate values. Suitable indices t_0 were found accordingly by using the S-box output mask as attacked value. For both cases, the time indices resulting in high correlation values were used.

5.1 Results: Biasing Masks and Windowing followed by 1st-Order DPA Attack

In order to introduce a bias in the mask values, we have used templates to derive the Hamming weight of the mask m. Templates were built from the uncompressed traces for each Hamming weight of the mask. We used 100 traces per template with 16 interesting points per trace. The multivariate Gaussian distribution model has been employed. The traces used for the 1st-order DPA attack have been compressed (integration of absolute values per clock cycle). Randomization has been countered by windowing, i.e. the power consumption values at times $t_1..t_{16}$ have been summed up. The Hamming weight of an unmasked S-box output byte has been used as predicted power consumption. The attack itself is therefore similar to the one described in [11] ("Templates During Preprocessing"), except for the additional compensation of the randomization countermeasure.

In practice, it is important to find a good tradeoff between a sharp bias and a minimal number of discarded traces. For example, only choosing traces with a mask Hamming weight of 8 (i.e. $m = 0$xFF) will lead to the highest correlation but on the other hand, this would mean to discard $\frac{255}{256} = 99.6\%$ of the recorded power traces. Selecting masks with a Hamming weight greater or equal to six has been shown to be a good choice [9,11,15] and therefore we have also used it for our evaluations.

The effectiveness of the attack depends on the accuracy of the biasing process. In order to show the best outcome, we have also biased the masks following their actual values (ideal case). The results for biasing with the actual Hamming weights and with the Hamming weights predicted by template matching are show in Figures 7 and 8, respectively.

In the ideal case, the correlation for the correct key hypothesis was about -0.04 while the use of template matching yielded a correlation of about -0.025. With increasing accuracy of the template method in predicting the actual Hamming weight of the used mask, the result of the attack should get closer to that

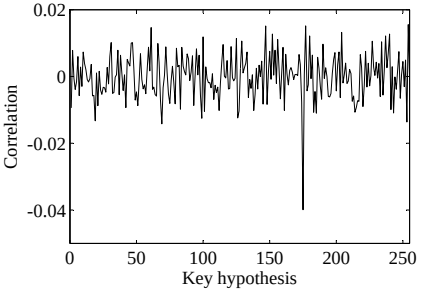

Fig. 7. Result of attack with ideal mask biasing

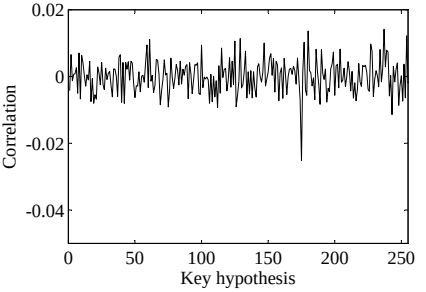

Fig. 8. Result of attack with mask biasing through templates

of the ideal case. We use the rule of thumb from [9] to estimate the required number of power traces for a successful attack. We have also taken those traces into account which were discarded during the biasing process (about 85% of all traces). For ideal biasing, about 122,000 power trace are sufficient, for our biasing with templates, about 305,000 power traces are required.

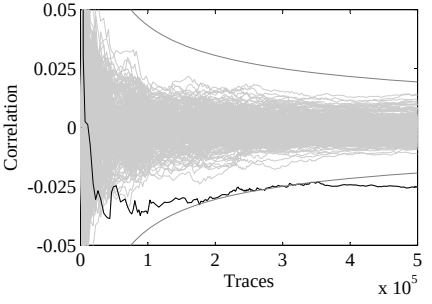

Fig. 9. Evolution of correlation in dependence on number of traces for mask biasing through templates

Figure 9 shows the evolution of the correlation with increasing number of power traces for the attack depicted in Figure 8. Note that the trace count on the x-axis also includes the discarded traces. The correct key hypothesis is plotted in black, the incorrect hypotheses are displayed in light gray. The outer dark gray lines indicate the confidence interval for $\rho = 0$. Roughly speaking, this is the expected region for incorrect key hypotheses. The point where the correct key hypothesis leaves this region gives another estimation for the number of traces required for a successful attack. In this case, the estimate lies in the vicinity of 300,000 traces, which is in line with the result from the rule of thumb from [9].

5.2 Results: 2nd-Order DPA Attack followed by Windowing

This attack can be seen as multiple 2nd-order DPA attacks in parallel, with their results combined by windowing. Nevertheless, a successful attack is not quite as simple to achieve as in a conventional 2nd-order DPA attack. Normally, most 2nd-order DPA attacks can be conducted in a more or less "brute-force" manner. More precisely, it is not necessary to determine the exact points in time which carry the most information about the targeted intermediate values and which are therefore suited most for 2nd-order DPA preprocessing. In fact, it is sufficient to predict the general regions of the power traces which are expected to contain the required points. By examining all possible combinations of the points of both regions in a 1st-order DPA attack, the correct key hypothesis can be identified without giving much thought to the actual points of the power trace which carry the required information.

When there is a need to compensate for the randomization countermeasure as well, it quickly becomes evident that this "brute-force" approach is no longer feasible. Even if all of the R parallel 2nd-order DPA attacks could be done in this manner, the subsequent windowing of the results requires that only those points are summed up which might contain information about the unmasked intermediate values. Our experiments have shown that the attack result is extremely sensitive even to slight variations in time of the two input points to the 2nd-order DPA preprocessing function. Therefore, choosing the best points from the power trace becomes a crucial precondition for windowing. Unfortunately, the best points only become known after a successful attack.

An effective way out of this dilemma can be made with a suitable compression function. If there is only a single point per clock cycle in the power trace, the 2nd-order DPA preprocessing and windowing can be done at the exact points in time where the maximal information is contained. However, care must be taken that not too much information is lost during compression. Our experiments have shown that none of the standard compression functions (maximum extraction, integration) [9] deliver satisfying results. After careful analysis of the 2nd-order DPA leakage profile of the attacked device, we have developed a new compression function, which retains most of the required information and hence delivers good results. Our new compression function extracts a small range of points around the maximum of each clock cycle and forms the average value of those points. At our sampling rate of $200 \cdot 10^6$ samples/second, a range of two points around the maximum (i.e. 5 points in total per clock cycle) was sufficient to achieve satisfying results.

As 2nd-order DPA preprocessing function we have employed the absolute of the difference of the two input points, as it has the best correspondence to single bits of unmasked values and still a good correspondence to the Hamming weight of larger values [9]. We have used the bit model (LSB of the S-box output) as power model for our attack. The results are shown in Figure 10.

The correlation peak for the correct key hypothesis has a height of approximately 0.024, requiring about 50,000 power traces for a successful attack [9].

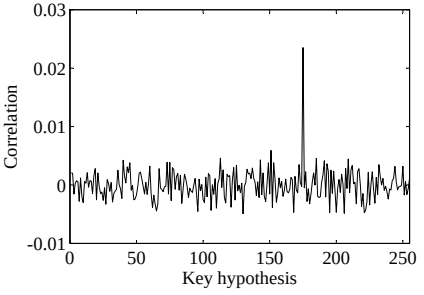

Fig. 10. Result of 2nd-order DPA attack followed by windowing

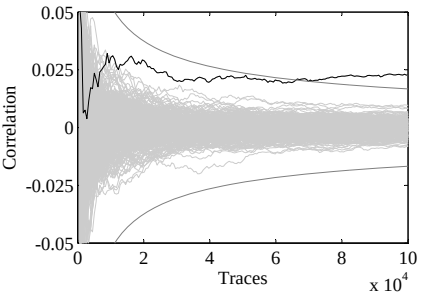

Fig. 11. Evolution of correlation in dependence on number of traces

The evolution of the correlation with the number of power traces used in the attack is shown in Figure 11. In this case, the correlation curve for the correct key hypothesis leaves the confidence interval for $\rho = 0$ at about 65,000 traces. Note that this number is a bit higher than the estimate from the rule of thumb from [9]. In our experience, the evolution of the correlation is quite dependent on the measurements, so the rule of thumb should normally be preferred for a more general prediction.

5.3 Results: Windowing followed by 2nd-Order DPA Attack

This attack had already a very low effectiveness in the simulated evaluation of [15]. For completeness, we have conducted attacks with this method on all 500,000 power traces. As suspected, this number of power traces was not sufficient to lead to a correct prediction of the key.

5.4 Dependence of Attack Efficiency on Randomization Degree

The AES implementation allows to change the degree of randomization R in order to trade performance against security. Table 1 shows how the effectiveness of the two attacks from Section 3.1 and 3.2 changes with increasing R. Conceptually, the correlation coefficient should scale down with a factor of $\sqrt{R}$ [15]. It can be seen from Table 1 that both attacks approximately follow this behavior, whereby the second one (2nd-order DPA attack followed by windowing), tends to perform worse at a higher R.

Table 1. Maximal absolute correlation coefficient in dependence on randomization degree R

R	**1**	**2**	**4**	**8**	**16**
Biasing masks	0.104	0.072	0.052	0.035	0.025
2nd-order and windowing	0.125	0.102	0.073	0.042	0.024

6 Discussion of the Practicality of the Attacks

As shown in Section 5, two of the three examined attacks succeeded with a reasonable amount of samples. Mask biasing turned out to lead to a potentially higher correlation, which is in line with the estimation results from [15]. However, this attack requires to discard a large number of power traces, which increases the total number of required power traces considerably. Furthermore, the 2nd-order DPA attack followed by windowing puts less demands on the attacker's knowledge and control over the device. In order to introduce a bias in the mask, templates for the mask values have to be built. This requires the availability of a device for profiling which is sufficiently similar to the attacked one. Moreover, the profiling device must offer the possibility to extract some information about the actually occurring mask values in order to allow template building. Depending on the attack scenario, these preconditions might not be always given.

Both methods require a windowing for the time indices $t_1..t_R$, i.e. all points in time where the attacked masked intermediate value can occur due to the randomization countermeasure (cf. Figure 2). However, for a practical attack it is not necessary to identify the exact points in time, but it is sufficient to know the distance between those points. In our attack, we have first compressed the power traces (see Section 5.2) so that there was only a single power consumption value per clock cycle. When the distance (in clock cycles) between the R points in time is known, all possible combinations of points with this distance can be used in the attack. The selection of possible combinations of R points can be seen as pulling a comb with R teeth over the power trace. The distances between the comb's teeth correspond to the clock cycle distances between the possible occurrences of the masked value $a \oplus m$. This is illustrated in Figure 12.

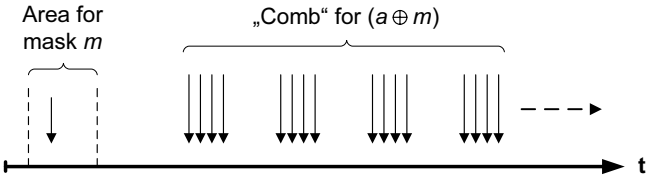

Fig. 12. Extracting viable points for 2nd-order DPA preprocessing

For each position of the comb and each point in time where the mask value m is suspected to appear ("area for mask m" in Figure 12), a new correlation value can be calculated. Thereby, 2nd-order DPA preprocessing is applied with the current mask point and each comb tooth. The resulting 16 preprocessed points are then summed up (windowed) to produce a single predicted power consumption value for the attack.

Normally, an attacker has some general idea of the order of operations which are performed by the attacked device so that it is possible to specify some areas in the power traces where certain values are likely to appear. This limits the number

of possible combinations of comb positions and mask positions and makes the attack faster. But even if all possible combinations are used in our case, the total number ranges around $3 \cdot 10^6$, which is quite feasible for an attack[3].

Depending on the scenario, an attacker can obtain information about the distance of the R randomized points (i.e. the distance between the teeth of the comb) through different means. If a device is available for profiling, then the relevant points in time can be determined through a DPA attack with known intermediate values. When the relevant sections of the implementation's source code are available, the distances can be derived with a cycle-accurate simulator. Even if those options are not available, some general knowledge about the protected implementation can be enough to establish a set of "candidate combs" with different distances between the teeth. If this set is not too large, the correct comb can be determined by trying out all combs of the set in an attack.

For our protected AES implementation it would be sufficient to know that there is a randomization of the columns of the State and a randomization of the bytes within each column. The distance between the processing of the columns and between the four S-box lookups of one column are constant. Thus there are only two configuration parameters for the comb, resulting in a manageable set of candidate combs.

Hence, the method of 2nd-order DPA attack followed by windowing can be regarded as a fairly generic attack which requires only little more knowledge about the implementation than a plain 1st-order DPA attack.

7 Conclusions

In this paper we have practically demonstrated the effectiveness of advanced DPA attacks on an AES smart card implementation with state-of-the-art software countermeasures. We have evaluated the three principal attack methods described in [15], which have so far only been subject to theoretical estimation and high-level simulation. Two of these methods work well in defeating the masking and randomization of operations countermeasures of the AES software implementation. One of the methods leads to a potentially higher correlation, but requires the attacker to be able to profile the attacked device in detail. The second attack is not quite as effective but is more general. It can be mounted without profiling and requires only little knowledge about the implementation. Nevertheless, it must not be overlooked that the effort for the attacks is considerably larger than for an unprotected implementation. While 100 power traces are normally enough to break the unprotected implementation, the advanced DPA attacks on the protected implementation require a minimal number of about 50,000 traces for success. Hence, DPA becomes more than two orders of magnitude (i.e. 100 times) harder under use of the described attacks. Although there is no guarantee that there are no better attacks on a specific implementation,

[3] There are about 1,800 points per compressed trace, which is hence the upper limit for comb positions and mask positions. The maximal number of combinations is therefore $1,800 \cdot 1,800 = 3,240,000$.

our work has delivered empirical evidence that a combination of masking and operation randomization can offer significant protection against advanced DPA attacks.

Acknowledgements. The research described in this paper has been supported by the Austrian Science Fund (FWF) under grant number P18321-N15 ("Investigation of Side-Channel Attacks"), by the European Commission under grant number FP6-IST-033563 (Project SMEPP) and, in part, by the European Commission through the IST Programme under contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. Atmel Corporation. 8-bit Microcontroller with 16K Bytes In-System Programmable Flash. Available online at `http://www.atmel.com/dyn/resources/prod_documents/doc1142.pdf`, February 2003.
2. C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Çetin Kaya Koç and C. 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.
3. J.-S. Coron, E. Prouff, and M. Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 28–44. Springer, September 2007.
4. C. Herbst, E. Oswald, and S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2006*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 2006.
5. International Organisation for Standardization (ISO). ISO/IEC 7816-3: Information technology - Identification cards - Integrated circuit(s) cards with contacts - Part 3: Electronic signals and transmission protocols. Available online at `http://www.iso.org`, September 1997.
6. J. Jaffe. Introduction to Differential Power Analysis, June 2006. Presented at ECRYPT Summerschool on Cryptographic Hardware, Side Channel and Fault Analysis.
7. M. Joye, P. Paillier, and B. Schoenmakers. On Second-Order Differential Power Analysis. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

8. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. 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.
9. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007. ISBN 978-0-387-30857-9.
10. National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001. Available online at `http://www.itl.nist.gov/fipspubs/`.
11. E. Oswald and S. Mangard. Template Attacks on Masking—Resistance is Futile. In M. Abe, editor, *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007, Proceedings*, volume 4377 of *Lecture Notes in Computer Science*, pages 243–256. Springer, February 2007. ISBN 978-3-540-69327-7.
12. E. Oswald, S. Mangard, C. Herbst, and S. Tillich. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.
13. K. Schramm and C. Paar. Higher Order Masking of the AES. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.
14. F.-X. Standaert, E. Peeters, and J.-J. Quisquater. On the Masking Countermeasure and Higher-Order Power Analysis Attacks. In *International Conference on Information Technology: Coding and Computing (ITCC 2005), April 4-6, 2005, Las Vegas, Nevada, USA, Proceedings*, volume 1, pages 562–567. IEEE Computer Society, April 2005. ISBN 0-7695-2315-3.
15. S. Tillich, C. Herbst, and S. Mangard. Protecting AES Software Implementations on 32-bit Processors against Power Analysis. In J. Katz and M. Yung, editors, *Proceedings of the 5th International Conference on Applied Cryptography and Network Security (ACNS 2007)*, volume 4521 of *Lecture Notes in Computer Science*, pages 141–157. Springer, June 2007.
16. J. Waddle and D. Wagner. Towards Efficient Second-Order Power Analysis. In M. Joye and J.-J. 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.