You cannot hide behind the mask : power analysis on a provably secure S-box implementation

Citation for published version (APA):
Pan, J., Hartog, den, J. I., & Lu, J. (2009). You cannot hide behind the mask : power analysis on a provably secure S-box implementation. In H. Y. Youm, & M. Yung (Eds.), *Information Security Applications (10th International Workshop, WISA 2009, Busan, Korea, August 25-27, 2009, Revised Selected Papers)* (pp. 178-192). (Lecture Notes in Computer Science; Vol. 5932). Springer. https://doi.org/10.1007/978-3-642-10838-9_14

DOI:
10.1007/978-3-642-10838-9_14

Document status and date:
Published: 01/01/2009

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

You Cannot Hide behind the Mask: Power Analysis on a Provably Secure S-Box Implementation[⋆]

J. Pan, J.I. den Hartog, and Jiqiang Lu

Eindhoven University of Technology,
Den Dolech 2, 5612 AZ, Eindhoven, The Netherlands

Abstract. Power analysis has shown to be successful in breaking symmetric cryptographic algorithms implemented on low resource devices. Prompted by the breaking of many protected implementations in practice, researchers saw the need of validating security of implementations with formal methods. Three generic S-box implementation methods have been proposed by Prouff el al., together with formal proofs of their security against 1st or 2nd-order side-channel analysis. These methods use a similar combination of masking and hiding countermeasures. In this paper, we show that although proven resistant to standard power analysis, these implementation methods are vulnerable to a more sophisticated form of power analysis that combines Differential Power Analysis (DPA) and pattern matching techniques. This new form of power analysis is possible under the same assumptions about power leakage as standard DPA attacks and the added complexity is limited: our experiments show that 900 traces are sufficient to break these algorithms on a device where 150 traces are typically needed for standard DPA. We conclude that the defense strategies—hiding by repeating operations for each possible value, and masking and hiding using the same random number—can create new vulnerabilities.

Keywords: Power analysis, side-channel analysis, provable security, block cipher S-box.

1 Introduction

With the expansion of electronic data-processing systems, small cryptographic devices like smart card and key tokens have tremendous possibilities for devising solutions for crucial applications, such as financial transaction and user identification. Based on cryptography, these devices could safely store secret keys and execute cryptographic algorithms. The security of cryptographic devices is therefore of vital importance and is the prime concern during design and development of such a device and the software running on it.

Power analysis has shown to be a practical threat to the security of cryptographic devices. Using the fact that the power consumption of the device is

[⋆] This work is funded by the Dutch Sentinels-project PinpasJC TIF.6687.

dependent on the data that is being processed within the device, power analysis extracts confidential information—such as the secret key used in a cryptographic algorithm—and leads to powerful and easy to conduct attacks. Ensuring the security of cryptographic devices against power analysis is an ongoing arms race at the forefront of scientific research, with new attacks being discovered, new countermeasures being introduced (see [1,2,7,14] for example).

As many implementations which are supposedly secured with effective countermeasures turn out to be broken, be it by incorrect use of the countermeasure or use of inadequate countermeasures, the need for a formal proof of the security of an implementation arises. Prouff et al. [9,10] proposed three generic S-box implementation methods that were demonstrated secure against 1st/2nd-order DPA attacks within the proof-of-security framework presented in [12]. In their methods, a combination of masking and hiding is used. On the one hand, the input and output of the S-box are masked by random values; on the other hand, the S-box look-up is executed for all possible values in a loop, during which the expected result, the masked output of S-box for the given input, is produced at a random moment in time for each execution. As will be explained in detail in Section 2, these implementations can thwart many powerful power analysis attacks.

In this paper, we introduce a new form of power analysis that can break the S-box implementation methods in [9,10], and any implementation methods that are of the same fashion. The attack makes use of the facts that the S-box look-up is executed for all possible values in an order that is predictable, and that the random value used for masking the intermediate result is also the value used for shuffling the look-ups. The new attack does not require a leakage of information that is more than required for standard DPA and the added complexity is limited. Using a device where 150 power traces are typically needed to break an unprotected implementation of AES S-box, practical experiments show that our attack on the protected implementation of AES S-box requires only 900 traces for success. Our theoretical analysis indicates that even when the noise is relatively high (e.g. SNR=0.015), our attack is still less than two orders of magnitude (i.e. 100 times) harder than a standard DPA attack. Moreover, the cost of our attack is independent from the number of random values used for the protection, as long as they are handeled in the same fashion as in the implementation methods presented in [9,10]. As a result, this new form of power analysis will reasonably be applicable in any setting where standard DPA is possible. Thus, though the implementation methods in [9,10] are formally proven to resist (1st/2nd-order) DPA attacks, they are not secure in any setting where such attacks are possible.

The remainder of the paper is organized as follows. Section 2 summarizes some popular power analysis techniques and countermeasures, followed by brief descriptions of the attacked S-box implementation methods proposed in [9,10]. Section 3 describes our attack strategy in detail. The practical and the theoretical aspects of the attack are discussed in Section 4 and Section 5, respectively. Finally, Section 6 provides some conclusions.

2 Preliminary

2.1 Power Analysis and Countermeasures

Many forms of power analysis have been discussed in literature in the past decade. Among them, the most common ones are Simple Power Analysis (SPA) [6] attacks and Differential Power Analysis (DPA) [4] attacks. With detailed knowledge of the implementation of the cryptographic algorithm under attack, SPA attacks can derive confidential information directly from a small number of power traces measured from the device. DPA attacks, on the other hand, combine a large number of power traces to extract secrets and thus remain applicable even when there is a lot of noise in the measurements. Moreover, they do not require detailed knowledge of the attacked device. Other popular forms of power analysis are template attacks [3] and collision attacks [11]. Template attacks can extract information from a single power trace by matching it against some pre-built templates. Collision attacks detect equal intermediate values that the device manipulates by comparing the power consumptions that correspond to different executions of the cryptographic algorithm under attack or to different moments in time in the same execution.

To defend against power analysis attacks, two general countermeasures exist: Masking and hiding [5]. Masking conceals the intermediate results of a cryptographic algorithm with random values. In the most common form, an intermediate result v is split into n shares (random masks $r_1, \ldots, r_{n-1}$ and $v \oplus r_1 \oplus \ldots \oplus r_{n-1}$) which all need to be combined to find any information on v. At each point in time only a single share is used ensuring that no information is leaked about the masked sensitive intermediate result. Hiding uses techniques such as random insertion of dummy operations and shuffling of (independent) instructions to ensure that in different executions the same operation does not (or only with a small probability) happen at the same moment in time. As a result, power traces are misaligned and the power consumption in any point in time will not (or at most very weakly) be correlated to any sensitive data.

In turn, attacks exist that are efficient despite these countermeasures (see e.g. [5,13]). An n-order DPA attack examines n locations of a power trace that correspond to the secret shares of an intermediate result. Biased mask attacks force bias into the masks, for instance by using templates, enabling DPA attacks examining a sensitive intermediate result that is protected by the no longer uniformly distributed masks. In windowing, the power consumption for a complete region, the 'window', is used rather than a single point in time. This prevents misalignment as long as the targeted intermediate result is computed within the window. As the number of traces needed for a successful attack grows with the size of the window, windowing is typically combined with trace alignment techniques such as pattern matching. In pattern matching, a pattern chosen from one trace is matched to the other traces to detect the same executed operation or equal processed data, allowing realignment of a trace.

Table 1. The provably secure S-box implementation algorithms proposed in [9,10]. INPUT: $\tilde{x} = x \oplus r$ (algorithm 1) or $\tilde{x} = x \oplus r_1 \oplus r_2$ (algorithms 2, 3); OUTPUT: $S(x) \oplus s$ (algorithm 1) or $S(x) \oplus s_1 \oplus s_2$ (algorithms 2, 3).

Algorithm 1	Algorithm 2	Algorithm 3
$R_0 \leftarrow s$	$r_3 \leftarrow rand(n)$	
$R_1 \leftarrow s$	$r' \leftarrow (r_1 \oplus r_3) \oplus r_2$	$b \leftarrow rand(1)$
for $a = 0$ to $2^n - 1$	**for** $a = 0$ to $2^n - 1$	**for** $a = 0$ to $2^n - 1$
$cmp \leftarrow compare(a, r)$	$a' \leftarrow a \oplus r'$	$cmp \leftarrow compare_b(r_1 \oplus a, r_2)$
$R_{cmp} \leftarrow R_{cmp} \oplus S(\tilde{x} \oplus a)$	$T[a'] \leftarrow (S(\tilde{x} \oplus a) \oplus s_1) \oplus s_2$	$R_{cmp} \leftarrow (S(\tilde{x} \oplus a) \oplus s_1) \oplus s_2$
end	**end**	**end**
$cmp \leftarrow compare(R_0, R_1)$	**return** $T[r_3]$	**return** R_b
return $R_0 \oplus (cmp \times R_1)$		

2.2 Provably Secure S-Box Implementations

Table 1 depicts the algorithms of the three provably secure S-box implementation proposed in [9,10]. In these algorithms, an S-box look-up maps an input in $\mathbb{F}_2^n$ to an output in $\mathbb{F}_2^m$ through a table S. Sensitive intermediate result x, which is often a function of the plaintext and the key, is concealed by an input mask r (resp. $r_1 \oplus r_2$). The masked sensitive intermediate result $\tilde{x}$ is taken as the input of the algorithms and a masked S-box output $S(x) \oplus s$ (resp. $S(x) \oplus s_1 \oplus s_2$) is returned at the end. The input masks and the output masks are independent variables. The core idea of the algorithms is to compute $S(\tilde{x} \oplus a) \oplus s$ (resp. $S(\tilde{x} \oplus a) \oplus s_1 \oplus s_2$) for every $a \in \mathbb{F}_2^n$ and return only the *expected result* $S(x) \oplus s$ (resp. $S(x) \oplus s_1 \oplus s_2$) at the end.

It was formally proven in [9,10] that algorithm 1 is secure against 1st-order DPA attacks and that algorithms 2 and 3 are also secure against 2nd-order DPA attacks. The security of the implementations lies in countermeasures including masking, shuffling and insertion of dummy operations. During each execution $2^n - 1$ dummy S-box look-ups are performed, between which the look-up for the expected result is executed. The 2^n look-ups are shuffled according to the value of the input mask (i.e. r or $r_1 \oplus r_2$), randomizing the moment in time the expected result is computed.

An S-box implemented in this fashion can thwart most of the pratical power analysis reported till now [5,13]. Standard 1st/2nd-order DPA attacks do not work for these implementations because the intermediate results are masked and randomized. Biased mask attacks and windowing are no longer practical, because there are too many possible points in time where the expected result could be computed, and the attacks cannot succeed without an extreme increase of the number of power measurements (about 2^n times as many as needed for an unprotected implementation). Pattern matching can detect equal (masked) intermediate values, however, without further information about the masks this is not enough to reveal the secret key.

Table 2. A model of the algorithms shown in Table 1. INPUT: $\tilde{x} = x \oplus r$; OUTPUT: $S(x) \oplus s$.

Model	Instantiations	
for $a = 0$ to 2^n-1	Algorithm 1	Algorithms 2, 3
$\leftarrow \tilde{x} \oplus a$ $\leftarrow S(\tilde{x} \oplus a)$ $\leftarrow S(\tilde{x} \oplus a) \oplus R$ **end** $\leftarrow S(x) \oplus s$	$R = \begin{cases} s & \text{if } a = r, \\ \bigoplus_{\substack{j=0 \\ j \neq r}}^{a-1} S(\tilde{x} \oplus j) \oplus s & \text{if } a \neq r. \end{cases}$	$R = s$ $r = r_1 \oplus r_2$ $s = s_1 \oplus s_2$
$S(\tilde{x} \oplus a) \oplus R = S(x) \oplus s$ if $a = r$		

3 Description of the New Power Analysis Attack

The algorithms depicted in Table 1 all use the same approach to secure an S-box implementation. Even though there are some differences in the usages of masks, registers and memory, these algorithms produce intermediate results in a similar fashion which can be summarized by an implementation model that is shown in Table 2. In this model, masks used for the same intermediate result are summed up and treated as one. Note that although it can be instantiated differently depending on the algorithm, variable R is randomized from execution to execution. The expected result is always computed during loop iteration $a = r$.

The model in Table 2 exposes three potential vulnerabilities to power analysis. First, the 2^n S-box look-ups are executed depending on the same unknown value of a few bits (e.g. $\tilde{x}$) so that guessing the unknown allows the prediction of the inputs and outputs of all the S-box during an execution. Second, the S-box look-up is performed for a large number (i.e. 2^n) of times in each execution with different inputs, opening the possibility for a 1st-order DPA attack even within a single trace. Third, the same random variable (i.e. r) is used for both masking and hiding, hence defeating one countermeasure immediately leads to the destruction of the other.

We introduce an attack that combines these vulnerabilities and can break a protected S-box implementation that is in line with the model in Table 2. This attack is illustrated in Figure 1 and consists of five steps. To simplify the notations, in the rest of this section we assume that the sensitive intermediate result x is the XOR of the plaintext and the key of the attacked device, and the variable R is always equal to the output mask s in the attacked implementation (as in Algorithms 2, 3 in Table 2). As discussed in Sections 4 and 5, the attack functions equally for implementations where a different R or a different x is used.

Step 1: Acquiring Power Traces. First, we randomly generate N plaintexts $p_1, \ldots, p_N$ and let the attacked device process these plaintexts based on its secret key K. During the execution of p_i, an input mask r_i and an output mask s_i are randomly generated on the device and the masked sensitive data $\tilde{x}_i = p_i \oplus (K \oplus r_i)$ is computed. The sensitive data x_i is thereby $p_i \oplus K$. Next, the S-box implementation is used to execute on $\tilde{x}_i$ to produce $S(x_i) \oplus s_i$.

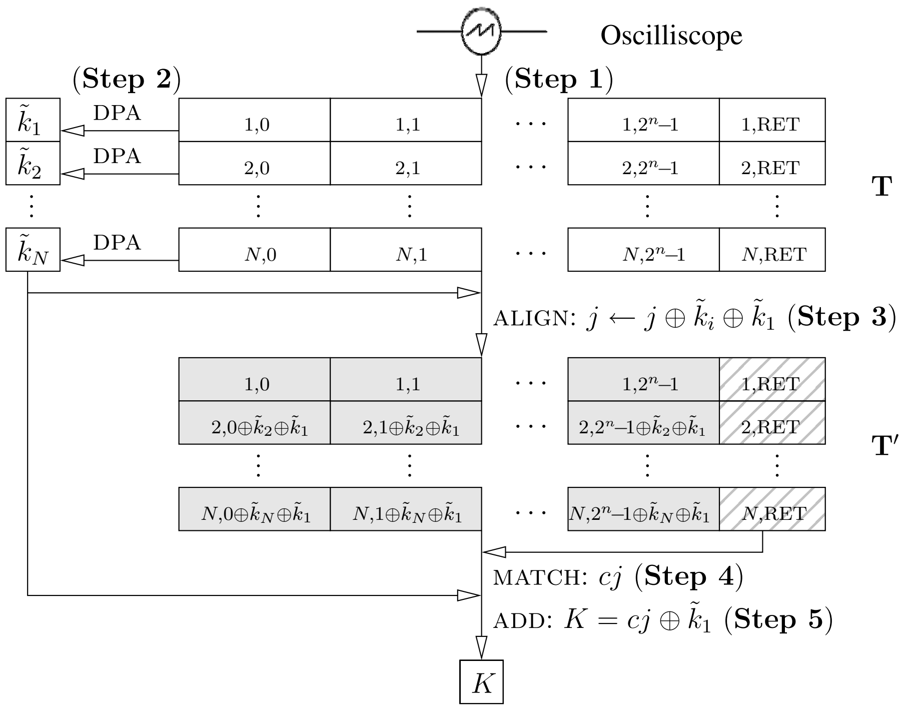
Oscilliscope
(Step 2)
(Step 1)
DPA
$\tilde{k}_1$
$\tilde{k}_2$
$\tilde{k}_N$
1,0
1,1
1,2^n–1
1,RET
2,0
2,1
2,2^n–1
2,RET
N,0
N,1
N,2^n–1
N,RET
T
ALIGN: $j \leftarrow j \oplus \tilde{k}_i \oplus \tilde{k}_1$ (Step 3)
2,0$\oplus\tilde{k}_2\oplus\tilde{k}_1$
2,1$\oplus\tilde{k}_2\oplus\tilde{k}_1$
2,2^n–1$\oplus\tilde{k}_2\oplus\tilde{k}_1$
N,0$\oplus\tilde{k}_N\oplus\tilde{k}_1$
N,1$\oplus\tilde{k}_N\oplus\tilde{k}_1$
N,2^n–1$\oplus\tilde{k}_N\oplus\tilde{k}_1$
T′
MATCH: cj (Step 4)
ADD: $K = cj \oplus \tilde{k}_1$ (Step 5)
K

Fig. 1. Block diagram illustrating the new power analysis attack

During this execution a power trace is recorded: $\mathbf{t}_i = (t_{i,0}, \ldots, t_{i,2^n-1}, t_{i,\text{RET}})$, where *subtrace* $t_{i,j}$, $0 \leq j \leq 2^n - 1$, corresponds to loop iteration $a = j$ and subtrace $t_{i,\text{RET}}$ corresponds to the return of the expected result $S(x_i) \oplus s_i$. The power traces obtained for all the plaintexts can be written as matrix $\mathbf{T} = (\mathbf{t}_1, \ldots, \mathbf{t}_N)'$.

Step 2: Recovering Masked Keys. Note that each trace $\mathbf{t}_i$ has 2^n subtraces $t_{i,0}$ trough $t_{i,2^{n-1}}$ which all process intermediate values depending on some (unknown) masked key $\tilde{k}_i = K \oplus r_i$, (known) plaintext p_i and (known) loop iteration indices $a = 0..2^n - 1$. This allows us to recover $\tilde{k}_i$ by performing a standard 1st-order DPA attack using the subtraces. (See e.g. [5] for details of such an attack.) Note that we perform an attack for each trace, i.e. N attacks in total.

Step 3: Aligning Power Traces. The effect of the hiding countermeasure can be removed by aligning the power traces. Having found the value of the masked keys we rearrange the traces to ensure that the computation of the expected result always happens at a fixed (but unknown) location for every power trace. We use cj to denote (the index of the subtrace containing) this location. We rearrange trace $\mathbf{t}_i$ by moving subtrace $t_{i,j\oplus\tilde{k}_i\oplus\tilde{k}_1}$ to position j leaving $t_{i,\text{RET}}$ where it is: $\mathbf{t}'_i = (t_{i,0\oplus\tilde{k}_i\oplus\tilde{k}_1}, \ldots, t_{i,2^n-1\oplus\tilde{k}_i\oplus\tilde{k}_1}, t_{i,\text{RET}})$. Note that the computation of the expected result happens in loop iteration r_i. Since trace $\mathbf{t}'_1 = \mathbf{t}_1$ remains unchanged it holds that $cj = r_1$. For an arbitrary trace $\mathbf{t}'_i$, subtrace $t_{i,cj\oplus\tilde{k}_i\oplus\tilde{k}_1}$ is placed at position cj. Because $cj \oplus \tilde{k}_i \oplus \tilde{k}_1 = r_1 \oplus \tilde{k}_i \oplus \tilde{k}_1 = r_i$, in every trace $\mathbf{t}'_i$ the computation of the expected result occurs at position cj.

Step 4: Comparing Power Signals. Within matrix $\mathbf{T}' = (\mathbf{t}'_1, \ldots, \mathbf{t}'_N)$ there are two columns which work with the expected result: the column cj and the last column RET. Due to the non-linearity of the S-box and the randomness of the output masks, the expected results $S(x_i) \oplus s_i$, $i = 1..N$, are statistically independent from other intermediate values processed during the loop. Thus, column cj of $\mathbf{T}'$ will be the only column related with column RET, and because of this, we can find cj by comparing column RET to every other column of $\mathbf{T}'$—it is the column giving the highest correlation value. Note that every subtrace $t_{i,j}$ contains several power consumption signals. For the correlation value between a column of subtraces and another column of subtraces, we take the maximum value amongst the correlation coefficients between any signal in the first and any signal in the second subtrace.

Step 5: Recovering the Key. Having found the masked key $\tilde{k}_1$ and the mask $r_1 (= cj)$ we can recover now the secret key $K = \tilde{k}_1 \oplus cj$. Note that if we make an error in determining $\tilde{k}_1$ we will make the same error in realigning the traces; the column cj will be shifted by the same amount and $K = \tilde{k}_1 \oplus cj$ still gives the correct result. We provide a more detailed analysis in Section 5.

4 Practical Experiments

We validate our attack on an protected S-box implementation described in Section 3 by experiments with the AES S-box. To obtain the power traces for these experiments the power signals of an 8-bit microcontroller clocked at 3.57 Mhz (1 clock cycle per 280 ns) are sampled at rate 1 GHz (1 sample per ns). In the experiments we focus on steps 2 and 4 of the attack described in Section 3. These steps contain the statistical analysis of the traces which mostly determines the effectiveness of the attack. The steps 1, 3 and 5 include trace processing and data computations which are well known to be feasible in practice and do not need to be repeated. E.g. we create the subtraces $t_{i,j}$ (step 1) by using already measured and extracted S-box computations rather than generating them anew.

Experimental Validation of Step 2. In effect, step 2 performs, for each trace, a 1st-order DPA attack on an unprotected S-box using all 2^n possible plaintexts. We take the 256 power traces measured while the microcontroller executes AES S-box look-ups using plaintexts $0, 1, \ldots, 255$ and a randomly selected key. Figure 2 shows the results of the attack for the correct key hypothesis (160 in this case). There exist multiple peaks in this graph because an S-box look-up takes more than one nanoseconds to be executed on the microcontroller. The highest peak $\rho = 0.59$ occurs at 9514 ns, denoting the point in time the output of the S-box is processed. Figure 3 plots the results of the attack for 9514 ns and all

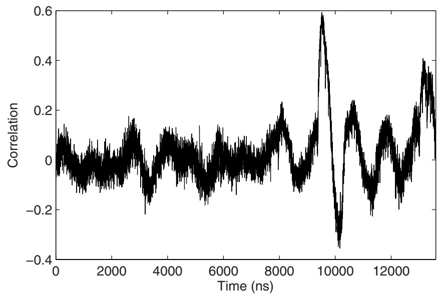

Fig. 2. The results of the DPA attack for the correct key hypothesis

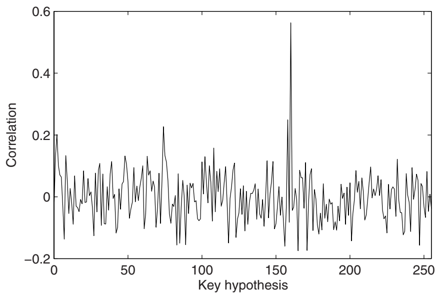

Fig. 3. The results of the DPA attack for all the key hypotheses at 9514 ns

the key hypotheses. As expected, the highest peak occur at the correct hypothesis 160 in this graph. The significance of the peaks in both figures indicates that the DPA attack has successfully revealed the key.

Note that a DPA attack may already succeed with fewer power traces. Figure 4 shows the evolution of the results for all the key hypotheses with an increasing number of traces used. The result for the correct hypothesis is plotted in black and the results for the incorrect hypotheses are plotted in light gray. The outer dark gray curves mark the confidence interval for correlation coefficient that is equal to zero (see Section 5). Within the interval is the expected region for the incorrect key hypotheses. The point where the black curve leaves this region gives an estimation for the number of traces required for a successful attack. Figure 4 shows that approximately 150 power traces are already sufficient to find the key. Using all available 256 traces the attack will almost always succeed; for nearly all traces we will find the correct masked key. Therefore, in step 3 nearly all traces can be correctly aligned.

Experimental Validation of Step 4. In step 4 of the attack the column of subtraces RET is compared to each of the other columns of subtraces. As the expected result is statistically independent from other intermediate results of the algorithm and every location of the traces corresponds to independent intermediate results, in step 4 we compare the **return** of the expected result with the computation of the same number or a random number.

For this purpose, we take the measurements of the power consumption for the transference to memory of $N = 10000$ expected results to obtain the column $(\mathbf{t}'_{i,\text{RET}})_{i=1..N}$. Then, we compare this column to the measurements during the computation of the expected results in $(\mathbf{t}'_{i,cj})_{i=1..N}$. We also compare this column to a column of computation of random results $((\mathbf{t}'_{i,j})_{i=1..N}$ for $j \neq cj)$.

Figure 5 shows the results of the comparison for different numbers of power traces used (up to $N = 10000$). The results for column cj are plotted in black and the results for the other 255 columns are plotted in light gray. Again, the outer dark gray curves indicate the expected region for the traces that are uncorrelated to cj. The point where the black curve leaves this region suggests that the number

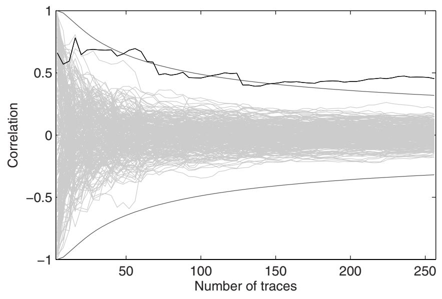

Fig. 4. The results of the DPA attack for different number of power traces

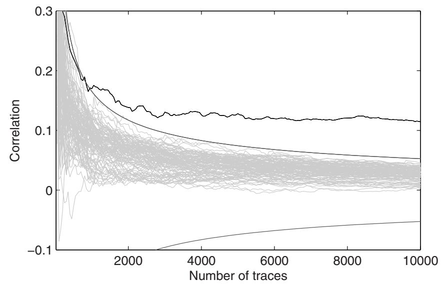

Fig. 5. The results of power signal comparison for different number of power traces

of traces required for a distinctive match between the columns cj and RET is approximately 900. A simple XOR in step 5 completes the attack.

5 Analysis of the Attack Practicality

The attack in Section 4 has succeeded with a relatively small number of power traces. However, more traces could be necessary when the measured power consumption signals have a rather low signal-to-noise ratio (SNR). In this section, we will discuss the practicality of our attack in a more general setting and provide a theoretical assessment of the effectiveness of the attack based on the SNR of the power measurements. Methods that can be used to improve the effectiveness of the attack are also suggested in this section.

Needless to say, due to the insertion of the dummy look-ups, the time required for the measurement of power traces and the hardware capability needed for the storage of the measured traces are, compared with a standard attack on an unprotected implementation, increased by a factor of approximately 2^n for our attack.

In order to mount the DPA attacks in step 2, 2^n subtraces must be extracted from each power trace that correspond to the 2^n loop iterations respectively. Such subtraces can often be spotted by visually inspecting the power trace, in that the S-box look-up is repeated for so many times in a row and that every look-up takes nearly the same amount of time (despite different processed data). Besides, the fact that an S-box look-up is usually captured by a number of points on a trace (see Section 4) can also facilitate the detection of subtraces. Note that this trace division needs to be performed only once for all traces, as in spite of different processed data the executed operations are already aligned in the raw traces for all plaintexts, hence the segmentation of one trace is also valid for other traces.

Next, N standard DPA attacks are performed (separately) using the extracted subtraces to find the masked keys. Although the total number of attacks in this step seems large, one of such attacks can usually be mounted with little effort. Please refer to e.g. [5,8] for a detailed description of a standard DPA attack.

Let ρ_{DPA} denote the expected correlation peak resulted from such a DPA attack for the correct hypothesis of the masked key and the correct moment in time. Let sr be the success rate of such a DPA attack and snr be the SNR of the power signal at the correct moment in time. Based on the rule of thumb introduced by Mangard et al. in [5], the relations between ρ_{DPA}, sr and snr can be roughly defined as in Eq. (1). The detailed deductions of ρ_{DPA} and sr can be found in [5] and Appendix A, respectively.

$$\rho_{\text{DPA}} = \frac{1}{\sqrt{1 + snr^{-1}}}, \qquad sr = \text{cdf}\left(\sqrt{\frac{2^n - 3}{8}\ln^2\frac{1+\rho_{\text{DPA}}}{1-\rho_{\text{DPA}}}}\right). \tag{1}$$

Having a success rate of sr in step 2 implies that there are on average $sr \cdot N$ correctly aligned power traces resulted from step 3 leaving $(1-sr) \cdot N$ traces misaligned. Thanks to the non-linearity of an S-box, the misaligned traces must contain only randomly shuffled power consumption signals.

Based on these partially aligned traces, in step 4 column RET of $\mathbf{T}'$ is matched against other columns to find cj. Let ρ_{CMP} denote the expected correlation value given by columns RET and cj. Let snr_1 and snr_2 be the SNRs of the power signals (in RET and cj respectively) that actually result in ρ_{CMP}. Eq. (2) shows some rules of thumb for the relations between ρ_{CMP}, snr_1, snr_2 and N—the number of traces needed for a successful attack, where $z_{0.9999}$ $(=3.719)$ is the quantile of the standard normal distribution $\mathcal{N}(0,1)$ for probability 0.9999. The derivations of ρ_{CMP} and N are explained in detail in Appendix A and [5] respectively. Please note that the N obtained by Eq. (2) is the number of traces needed for an entire attack.

$$\rho_{\text{CMP}} = \frac{1}{\sqrt{1 + snr_1^{-1}}} \cdot \frac{1}{\sqrt{1 + snr_2^{-1}}} \cdot sr, \qquad N = 3 + 8\,\frac{z_{0.9999}^2}{\ln^2\frac{1+\rho_{\text{CMP}}}{1-\rho_{\text{CMP}}}}. \tag{2}$$

Eqs. (1) and (2) indicate that the number of traces needed for a successful attack grows about quadratically on the success rate of the trace alignment. Increasing the correctness of trace alignment, especially in case of heavy noise, will enormously reduce the number of traces required. Hence, we provide an error-correction method that can be applied at the end of step 3 to detect misaligned traces. Since the S-box look-up is executed for all possible values in a run of the implementation, there must exists one (and only one) subtrace in each trace such that they all correspond to S-box look-ups with equal data (i.e. equal inputs and equal outputs). These subtraces can be derived based on correct masked keys obtained from step 2, while incorrect masked keys will only result in subtraces that correspond to random intermediate data thanks to the non-linearity of the S-box. For example, let us assume two different traces $\mathbf{t}_{i_0}$ and $\mathbf{t}_{i_1}$ and define relation $j_1 = j_0 \oplus \tilde{k}_{i_1} \oplus \tilde{k}_{i_0} \oplus p_{i_1} \oplus p_{i_0}$; subtraces t_{i_0,j_0} and t_{i_1,j_1} correspond to S-box look-ups with equal data if and only if $\mathbf{t}_{i_0}$ and $\mathbf{t}_{i_1}$ are correctly aligned with each other, since $p_{i_0} \oplus \tilde{k}_{i_0} \oplus j_0 = p_{i_1} \oplus \tilde{k}_{i_1} \oplus j_1$. Therefore, by demonstrating for all subtraces of all traces whether or not the relevant subtraces indeed correspond to equal processed data, we can verify if the obtained masked keys are correct and hence identify misaligned traces. To determine equal intermediate results, one can use highest-correlation or least-variance method to the corresponding power signals. If a misaligned trace is found, one can choose either to exclude it from the rest of the attack, or to make another attempt to align this trace by using the 2nd (or the 3rd, etc.) best hypothesis of the masked key resulted from step 2.

However, it is not always necessary in practice to improve the success rate of step 3; the improvement is only worthwhile when the SNR of the power measurements is relatively low. As shown later in this section, in case of the AES S-box, one percent increase of the success rate in step 3 can reduce the number of traces needed by 1686 if $snr = 1/50$ but can save only 66 traces if $snr = 1/10$.

Effectiveness of Our Attack for the AES *S*-Box. Using the methods presented in this section, we have assessed the effectiveness of our attack for such a protected implementation of the AES S-box. Figure 6 depicts the evolutions of the correlations ρ_{DPA} and ρ_{CMP}, the success rate sr and the number of traces needed N with a decreasing SNR, as well as the evolution of N with a decreasing sr. Note that in order to simplify the demonstration, we have replaced snr_1^{-1} and snr_2^{-1} in Eq. (2) with their mean snr^{-1}. Compared with using two separate SNRs, using the mean leads to a lower estimation of ρ_{CMP} and a higher estimation of N. Therefore, the results presented in Figure 6 show a worse case scenario for the attacker.

Figure 6 shows that the success rate of step 2 is extremely high even when the power measurement is relatively noisy. This is because the AES S-box is especially vulnerable to DPA attacks. The number of traces required for the entire attack, on the other hand, grows rapidly when the noise goes up, implying that the attack may encounter practical difficulties in case of very low SNRs. Fortunately, many commercial cryptographic devices used in practice can give relatively high SNRs (e.g. $\geq 1/20$). In fact, we believe that if a device requires such an elaborate protection as the attacked implementations in this paper, this device must leak a considerable amount of information once without protections; in this case our attack can be highly effective.

Table 3 shows the effectiveness of our attack for some SNRs that are higher than 1/20. The number of traces need for our attack for the protected AES S-box implementation and the number of traces needed for a standard DPA attack for an unprotected AES S-box implementation are listed shoulder to shoulder in Table 3. It is indicated by our results that while a standard DPA attack requires about 150 traces to break an unprotected implementation, our attack needs about 980 traces to break the protected implementation. This estimation accords almost perfectly with the practical results presented in Section 4, which shows that our theoretical results given in Table 3 can be very close to the reality.

Important Notes. Our attack is possible for the proven secure implementations *not* because of incorrect proofs of Prouff et al.'s or inadequate security metrics of Standaert et al.'s. When directly analyzing the raw power traces that are measured during the executions of Prouff et al.'s algorithms, the security metric of Standaert et al.'s can be perfectly satisfied; therefore the algorithms were believed secure against side-channel analysis in general. However, in fact

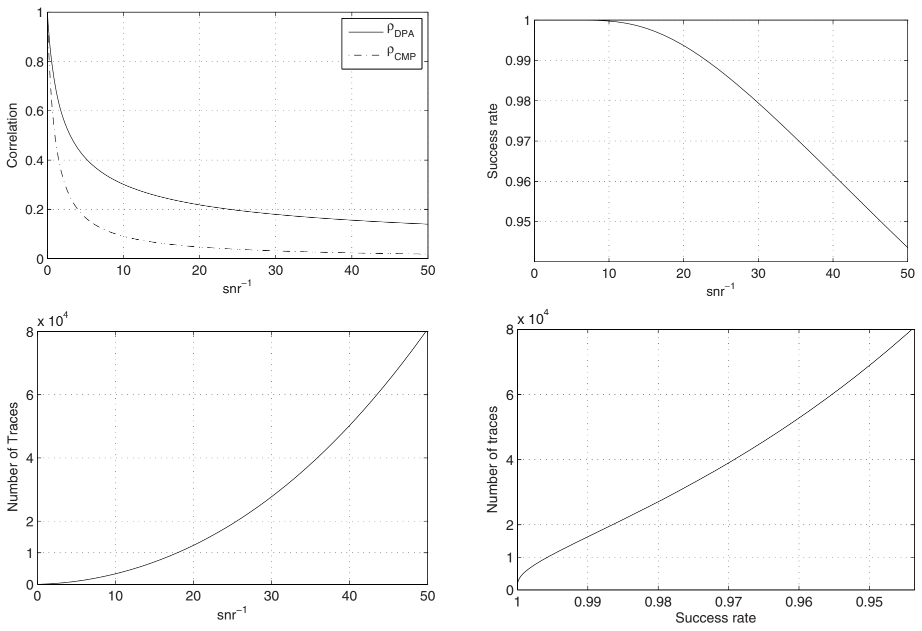

Fig. 6. The estimated results of our attack on the provably secure implementations with the AES S-box. Left-top: ρ_{DPA} and ρ_{CMP} with a decreasing SNR; right-top: sr with a decreasing SNR; left-bottom: N with a decreasing SNR; right-bottom: N with a decreasing sr.

Table 3. The number of traces needed for some high SNRs: N is for our attack on a protected AES S-box; N(DPA) is for a standard 1st-order DPA attack on an unprotected AES S-box

snr^{-1}	1	2	3	4	5	6	7	8	9	10	12	14	16	18	20
N	95	233	427	676	980	1340	1755	2225	2751	3333	4666	6227	8022	10057	12339
N(DPA)	39	67	95	122	150	178	206	233	261	289	344	399	455	510	565

the proofs have only shown the resistances against *straightforward* side-channel analysis. With a little preprocessing, such as the steps 2 and 3 in our attack, power signals that contain information about the same sensitive data can be relocated to the same position for all traces; the security of the implementation is thereby no longer provable by the metric for the aligned power traces.

We would also like to point out that our attack is *not* a high-order attack. In a high-order DPA attack, all the unknown secret shares have to be tested resulting in a total testing space that is exponential to the testing space for one share. Whereas in our attack, the required testing space is only linear to the testing space for one share because only the masked key needs to be tested. Moreover, in a noise-free scenario the resulted correlation peak of our attack is greater than that of a high-order DPA attack, e.g. it is $\rho = 0.24$ for a 2nd-order DPA attack (see [5]) and is $\rho = 1$ for our attack (see Figure 6). This means that less power traces would be necessary for our attack at least in case of high SNRs.

Finally, the attack introduced in this paper breaks a generic S-box implementation method. An implementation is vulnerable to our attack as long as it is in line with the extracted model shown in Table 2. The cost needed for breaking such implementations are almost equal in spite of the number of masks used.

6 Conclusion

In this paper, we have introduced a power analysis attack that can break a type of provably secure S-box implementations. The attack combines the standard DPA attacks and pattern matching techniques and can reveal the secret that is deliberately hidden behind the countermeasures. We have shown by both practical experiments and theoretical analysis that our attack is effective and efficient.

As general conclusions, we find that this work leads to several general observations for countermeasures against side-channel analysis. Particular care has to be taken when masking and hiding are applied based on the same random number, and when an operation is repetitively executed depending on the same unknown value. Finally, well known but worth repeating, one must be prudent when interpreting a formal proof; though it is a very useful and powerful tool, a formal proof should be used as intended and must not be seen as a final guarantee of security.

References

1. Brier, É., 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)
2. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
3. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
4. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
5. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. In: Advances in Information Security. Springer, Heidelberg (2007)
6. Mayer-Sommer, R.: Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 78–92. Springer, Heidelberg (2000)
7. Oswald, E., Mangard, S., Herbst, C., Tillich, S.: Practical second-order DPA attacks for masked smart card implementations of block ciphers. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 192–207. Springer, Heidelberg (2006)
8. Pan, J., den Hartog, J., de Vink, E.: An operation-based metric on cpa resistance. In: Jajodia, S., Samarati, P., Cimato, S. (eds.) SEC, International Federation for Information Processing, pp. 429–443. Springer, Boston (2008)
9. Prouff, E., Rivain, M.: A generic method for secure sbox implementation. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 227–244. Springer, Heidelberg (2008)

10. Rivain, M., Dottax, E., Prouff, E.: Block ciphers implementations provably secure against second order side channel analysis. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 127–143. Springer, Heidelberg (2008)
11. Schramm, K., Wollinger, T.J., 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)
12. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. Cryptology ePrint Archive, Report 2006/139 (2006), `http://eprint.iacr.org/2006/139`
13. Tillich, S., Herbst, C.: Attacking state-of-the-art software countermeasures-a case study for AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 228–243. Springer, Heidelberg (2008)
14. Tillich, S., Herbst, C., Mangard, S.: Protecting AES software implementations on 32-bit processors against power analysis. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 141–157. Springer, Heidelberg (2007)

A The Derivation of Eqs. (1) and (2)

Mangard et al. [5] introduced a rule of thumb to assess the number of power traces needed for a DPA attack. According to them, the number of traces N that are necessary to mount a DPA attack with a confidence α can be calculated by Eq. (3), where $z_\alpha = \mathrm{cdf}^{-1}(\alpha)$ is the quantile of $\mathcal{N}(0,1)$ for probability α. They also suggested that the number of traces needed for a successful attack can be calculated by Eq. (3) with $\alpha = 0.9999$. Inversely, the success rate α of a DPA attack using N power traces can be derived as in Eq. (4). Letting $N = 2^n$ we obtain the success rate of the DPA attack in step 2 of our attack as the sr in Eq (1).

$$N = 3 + 8\,\frac{z_\alpha^2}{\ln^2 \frac{1+\rho_{\text{DPA}}}{1-\rho_{\text{DPA}}}}\,, \quad (3) \qquad \alpha = \mathrm{cdf}\left(\sqrt{\frac{N-3}{8}\ln^2\frac{1+\rho_{\text{DPA}}}{1-\rho_{\text{DPA}}}}\right). \quad (4)$$

To deduct ρ_{CMP} in (2) we first consider the following general cases. Assume that M_1 and M_2 are two measurements for the same random data variable, of which the Hamming-weight is H. Let P_1 and P_2 denote the noise in M_1 and M_2, respectively. Therefore, the SNR of M_1 is $snr_1 = \sigma^2(H)/\sigma^2(P_1)$ and the SNR of M_2 is $snr_2 = \sigma^2(H)/\sigma^2(P_2)$. The correlation coefficient $\rho\,(M_1, M_2)$ can be calculated by Eq. (5), where the simplification is made based on the fact that P_1 and P_2 are statistically independent from H.

$$\begin{aligned}\rho\,(M_1, M_2) &= \rho\,(H + P_1,\, H + P_2) \qquad (5)\\ &= \frac{E\left((H+P_1)\cdot(H+P_2)\right) - E(H+P_1)\cdot E(H+P_2)}{\sigma(H+P_1)\cdot\sigma(H+P_2)}\\ &= \frac{E(H^2) - E^2(H)}{\sigma(H+P_1)\cdot\sigma(H+P_2)} = \frac{\sigma^2(H)}{\sqrt{\sigma^2(H)+\sigma^2(P_1)}\cdot\sqrt{\sigma^2(H)+\sigma^2(P_2)}}\\ &= \frac{1}{\sqrt{1+snr_1^{-1}}}\cdot\frac{1}{\sqrt{1+snr_2^{-1}}}\,.\end{aligned}$$

Now, let us consider the case where error exists in one of the measurement. Let I denote a random variable that has the same distribution as M_1 but is statistically independent from M_1. Let X_1 be an erroneous measurement of the processed data such that $Pr(X_1 = M_1) = sr$ and $Pr(X_1 = I) = 1 - sr$. The correlation coefficient $\rho\,(X_1,\, M_2)$ then corresponds to the expected correlation coefficient ρ_{CMP} in step 4 of our attack for the correct loop iteration index (see Eq. (2)), which can therefore by developed as in Eq. (6) based on Eq.(5).

$$\begin{aligned}
\rho_{\text{CMP}} = \rho\,(X_1,\, M_2) &= \frac{E(X_1 \cdot M_2) - E(X_1) \cdot E(M_2)}{\sigma(M_2) \cdot \sqrt{E(X_1^2) - E^2(X_1)}} \qquad (6)\\
&= \frac{\begin{array}{l} E(M_1 \cdot M_2) \cdot sr + E(I \cdot M_2) \cdot (1 - sr) \\ \qquad\qquad - (E(M_1) \cdot E(M_2) \cdot sr + E(I) \cdot E(M_2) \cdot (1 - sr)) \end{array}}{\sigma(M_2) \cdot \sqrt{E(M_1^2) \cdot sr + E(I^2) \cdot (1 - sr) - (E(M_1) \cdot sr + E(I) \cdot (1 - sr))^2}}\\
&= \frac{(E(M_1 \cdot M_2) - E(M_1) \cdot E(M_2)) \cdot sr}{\sigma(M_2) \cdot \sigma(M_1)} = \rho\,(M_1,\, M_2) \cdot sr\\
&= \frac{1}{\sqrt{1 + snr_1^{-1}}} \cdot \frac{1}{\sqrt{1 + snr_2^{-1}}} \cdot sr\,.
\end{aligned}$$