Entropy Reduction for the Correlation-Enhanced Power Analysis Collision Attack

Andreas Wiemers, Dominik Klein

Bundesamt für Sicherheit in der Informationstechnik (BSI)
{firstname.lastname}@bsi.bund.de

Abstract. Side Channel Attacks are an important attack vector on secure AES implementations. The *Correlation-Enhanced Power Analysis Collision Attack* by Moradi et al. [13] is a powerful collision attack that exploits leakage caused by collisions in between S-Box computations of AES. The attack yields observations from which the AES key can be inferred. Due to noise, an insufficient number of collisions, or errors in the measurement setup, the attack does not find the correct AES key uniquely in practice, and it is unclear how to determine the key in such a scenario. Based on a theoretical analysis on how to quantify the remaining entropy, we derive a practical search algorithm. Both our theoretical analysis and practical experiments show that even in a setting with high noise or few available traces we can either successfully recover the full AES key or reduce its entropy significantly.

1 Introduction

Kocher's [11] groundbreaking paper on side channel attacks has led both science and industry to focus on attacking and hardening their implementations [1]. Due to its popularity and de-facto standard w.r.t. symmetric cryptographic algorithms, AES [4, 9, 10, 14, 15] is of particular interest. Despite its theoretical cryptographic strength, a secure AES implementation that does not leak information about processed data remains to be a challenge. A popular countermeasure to minimize leakage about the AES key is *masking*. Different masking schemes exist, but the general idea of masking is that whenever secret data is about to enter critical stages of operation, some reversible operation that makes the data appear to be random is applied. Any cryptanalysis of intermediate data of the processing step is thus worthless. After leaving the critical stage of operation, the operation is reversed and the processed result can be used. Appropriate masking schemes can successfully prevent several attacks.

One particular class of attacks against AES are collision attacks. In collision attacks, one exploits the fact that sometimes leakage of the device can indicate that the same intermediate value has been processed during some critical stage of operation. By using this observation, one can gather information about secret data and cryptanalyze the device. In particular attacks that detect internal collisions are of interest. This kind of attack method was originally applied to DES [12, 17], but later applied to AES [3, 16] as well.

A very powerful kind of collision attack against AES was applied in [13], and later improved in [8]. The attack works by feeding data into a device in order to create collisions. A major important observation by [13] is that since the S-Box in AES is the same for every key byte (as opposed to i.e. DES), in most implementations the S-Box is the same for every key byte as well. This implies that for two same processed values, the resulting power consumption should be the same as well. Their idea is to create collisions such that this leakage *between* S-Box computations of different key byte positions is exploitable. This makes the attack very powerful — it is shown in in [13] that a device with S-Boxes that are masked using the state-of-the-art Canright S-Box implementation [5, 6] can be broken with a reasonable amount of available traces. It is important to note here that in general the leakage of the device attacked in [13] was minimal, and in particular a typical state-of-the-art template attack [7] was close to impossible to execute. In particular the amount of trace data needed to mount a successful attack was magnitudes lower for the correlation attack than for a template attack.

The attack gives some information about the correct AES key. However, the attack might not find the correct AES key uniquely in practice. There are several reasons for this: Noise, an insufficient number of collisions, errors in the measurement setup, or simply the device itself, i.e. the design and implementation of the cryptographic co-processor for a hardware implementation, or the processor design and execution flow in a software implementation. Moreover, it is not clear a priori how to find a set of key candidates that fit to the observations of the attack. A naive approach, i.e. enumerating all possible key candidates is computationally infeasible due to the large search space.

It is also unclear how to assess the leakage of the device; in particular it leaves open the question how many measurements (traces) are required to successfully mount the attack. Obviously, if the key uniquely identified for a certain amount of traces, this gives an upper bound. However what if less measurements are available?

In this paper we provide an algorithm to recover the AES key in the above scenario. The theoretical motivation of the algorithm is the basis for our analysis on how to quantify the remaining entropy, which can be used to assess the leakage of a device. Both our theoretical analysis and practical experiments show that even in a fuzzy setting with high noise or few available traces, we can either successfully recover the full AES key or reduce its entropy significantly.

This paper is structured as follows. In Section 2, we first briefly recall the attack by Moradi et al. as formulated in [13] and then introduce our algorithm in Section 3. For the algorithm we give a thorough theoretical justification in Section 4. Then in Section 5 we analyze the success rate of the algorithm, i.e. its impact on the entropy of a vulnerable system w.r.t. its leakage, and give upper and lower bounds of the remaining entropy. Our theoretical findings are verified by providing experimental data in Section 6. Finally, we conclude our presentation in Section 7.

2 Correlation-Enhanced Power Analysis Collision Attack

Let $K_1, \ldots, K_{16}$ be the correct key which is used in the first round of an AES encryption. We denote by small letters $k_1, \ldots, k_{16}$ candidates for the key. We briefly recall the Correlation-Enhanced Power Analysis Collision Attack as described in [13].

During the measurement phase we record N power consumption traces of the first round of an AES-128 encryption. These traces consists of 16 single S-Box computations. The measurement of each single S-Box computation is given as a vector of T numbers. We denote by $b_{i,w,t}$ this power consumption trace of a single S-Box computation i of a known plaintext $p_{w,i}$, $1 \leq i \leq 16$, $1 \leq w \leq N$, $1 \leq t \leq T$. As a first step we compute the average value $M_{i,\beta,t}$ over all w with $\beta = p_{w,i}$. Secondly, for any i, j and t we derive the empirical correlation coefficient $C_{i,j,\alpha,t}$ between $M_{i,\beta,t}$ and $M_{i,\beta\oplus\alpha,t}$ for any byte value α, where we treat β as a random variable uniformly distributed on all 256 byte values. At last, we set $c_{i,j}(\alpha)$ for the maximum of all $C_{i,j,\alpha,t}$, where t runs over all time points.

The idea of this approach is as follows: If the measurement $b_{i,w,t}$ is slightly dependent on the input byte $p_{w,i} \oplus K_i$ of the S-Box computation i, the average $M_{i,\beta,t}$ depends on $\beta \oplus K_i$ even more significantly. Now the input bytes $\beta \oplus K_i$ of S-Box i and $\beta \oplus K_j \oplus \alpha$ of S-Box j are the same for the choice $\alpha = K_i \oplus K_j$. Therefore, we can hope that the correlation $C_{i,j,\alpha,t}$ has — at least for some t — a significantly higher value for the correct choice $K_i \oplus K_j$ of α.

3 Recovering the AES Key

In this section we formulate our algorithm for computing candidates for the full AES key. We assume that we have given $120 \cdot 256$ values in the form

$$c_{i,j}(\alpha)$$

for $1 \leq i < j \leq 16$, where α runs over all byte values. If for each i, j the value $c_{i,j}(K_i \oplus K_j)$ is always the highest among all $c_{i,j}(\alpha)$, then it is easy to derive the full key. Here we are interested in the situation, where for each i, j, the value $c_{i,j}(K_i \oplus K_j)$ has only a tendency of being large compared to other $c_{i,j}(\alpha)$ with $\alpha \neq K_i \oplus K_j$. The idea of our approach is to consider the ad-hoc evaluation function

$$B = \sum_{i<j} c_{i,j}(k_i \oplus k_j)$$

for any key candidate $(k_1, \cdots, k_{16})$ and choose the key candidate with the highest value in B. Since this is not feasible in a straightforward manner, we instead try to compute B via partial sums. To this end, we fix an integer W, resp. integers $g_2, \cdots, g_{16}$.

Algorithm 1 Recovering the AES Key

```
Set $k_1 = 0$, $S_1 = \{k_1\}$ and $B_1 = 0$.
for $s = 1, \ldots, 15$ do
    for each key candidate $k_1, \ldots, k_s$ in $S_s$ do
        for each value of the next key bytes $k_{s+1}$ do
            compute the evaluation function
                $B_{s+1} = B_s((k_1, \ldots, k_s)) + \sum_{1 \le i \le s} c_{i,s+1}(k_i \oplus k_{s+1})$
        end for
    end for
    select subset of candidates $k_1, \ldots, k_s, k_{s+1}$ w.r.t. some criteria and store in $S_{s+1}$:
        Variant I: Select $W$ candidates $k_1, \ldots, k_s, k_{s+1}$ with largest $B_{s+1}$
        Variant II: Select all $k_1, \ldots, k_s, k_{s+1}$ with $B_{s+1} \ge g_{s+1}$
end for
return
```

Remarks and Observations

- Since B_s and B only depend on $\oplus$-sums of key bytes, we can choose one key byte as a fixed value. Here, we set $k_1 = 0$.
- The success probability of both variants of our algorithm for finding the correct key depends on the input parameters W, resp. $g_2, \cdots, g_{16}$. If we choose $g_s = B_s((K_i \oplus K_j))$, Variant II of Algorithm 1 is guaranteed to output the correct key. However, in this case S_s might become too large to store in practice.
- W can be treated as a measure of the workload (i.e. the number of computational steps) of Variant I of Algorithm 1.
- Both variants of the algorithm assume a fixed order of key byte positions. The result of the algorithm depends on that assumed order of the key bytes. One can repeat the algorithm with different orders. As s grows, the order becomes less important. We investigate the effect of the order on the success of the algorithm in Section 6.
- The choice of the order could take into account the actual distribution of the values $c_{i,j}(\alpha)$. Those i, j with significantly high values in $c_{i,j}(\alpha)$ could be considered first. In a practical setting, visual inspection of $C_{i,j,\alpha,t}$ could give a hint, cf. for example Figures 5a and 5b. In general however, we are more interested in the situation where $c_{i,j}(K_i \oplus K_j)$ is not automatically the highest value among the $c_{i,j}(\alpha)$, but is only larger on average over all i, j.

4 Theoretical Justification

In this section, we give a justification of the evaluation function B. To this end, we treat $c_{i,j}(\alpha)$ for any i, j, α as a realization of a normally distributed random variable. We assume the easiest scenario: For any i, j, α with $\alpha \neq K_i \oplus K_j$

the means and the standard deviations are equal and are denoted by a, resp. σ. Furthermore, for the correct $\alpha = K_i \oplus K_j$ the means are equal and are denoted by b and in addition, the standard deviations are equal to σ. For any key candidate $k = (k_1, \cdots, k_{16})$ we can check whether for all key candidates $\tilde{k}$

$$c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) \approx a, \text{ if } \tilde{k}_i \oplus \tilde{k}_j = k_i \oplus k_j$$
$$c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) \approx b, \text{ if } \tilde{k}_i \oplus \tilde{k}_j \neq k_i \oplus k_j$$

As a likelihood measure for any key candidate we seek a function in the single probability density functions as

$$\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) - a)^2}{2\sigma^2}), \text{ resp.}$$
$$\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) - b)^2}{2\sigma^2})$$

The cumulative probability density function of two *independent* random variables is just the product of the single probability density functions. Therefore, we are led to use as an evaluation function the product over all single probability density functions. Taking logarithms we get

$$\sum_{\tilde{k}} \left[\sum_{\substack{i<j, \\ \tilde{k}_i \oplus \tilde{k}_j = k_i \oplus k_j}} (c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) - a)^2 + \sum_{\substack{i<j, \\ \tilde{k}_i \oplus \tilde{k}_j \neq k_i \oplus k_j}} (c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) - b)^2 \right]$$

An equivalent evaluation function is therefore

$$\sum_{i<j} c_{i,j}(k_i \oplus k_j).$$

5 Success Rate of the Algorithm (Variant II)

In this section we want to give theoretical estimates of the success rate and workload for the second variant of the algorithm. The purpose of this section is to find relations between those theoretical estimates and basic properties of the distributions of $c_{i,j}(\alpha)$. To make the derivation as simple as possible, we restrict ourselves to the scenario in the last section, i.e. $c_{i,j}(\alpha)$ for any i, j, α is treated as a realization of a normally distributed random variable with mean a, resp. b, and standard deviation σ. Furthermore, for any key candidate the evaluation function

$$B_s = \sum_{i<j\leq s} c_{i,j}(k_i \oplus k_j)$$

is considered as a sum of *independent* random variables. Therefore, B_s is a normally distributed random variable. For a randomly chosen key candidate we have the expectation value

$$\mathrm{E}(c_{i,j}(k_i \oplus k_j)) = \frac{255}{256}a + \frac{1}{256}b \approx a$$

since b is assumed to be only slightly larger than a. Therefore, the mean and standard deviation of B_s are $\binom{s}{2}a$, resp. $\sqrt{\binom{s}{2}}\sigma$. For the correct key, $B_s((K_1, \ldots, K_s))$ is normally distributed with mean $\binom{s}{2}b$.

For having B_s near to its mean value, we want to avoid small values in $\binom{s}{2}$. In practice, we set in Variant II of the algorithm $g_s = -\infty$ for $s \leq 4$. For the ease of presentation we want to assume

$$B_s((K_1, \ldots, K_s)) \geq \binom{s}{2} b \text{ for } s \geq 5$$

Therefore, we set here $g_s = \binom{s}{2}b$.

5.1 An upper Bound of the Remaining Entropy

Variant II of the algorithm only finds key candidates for which $B_s \geq g_s$ for *all* s, $5 \leq s \leq 16$. In every step, the set S_s is a subset of all key candidates for which the condition $B_s \geq g_s$ is fulfilled. The size A_s of this larger set can be approximated by

$$\# S_s \leq A_s = 2^{(s-1)8}\, \mathbf{P}\left(B_s \geq \binom{s}{2} b\right)$$

and $\log_2(A_{16})$ is an upper bound for the remaining entropy.[1] $A_{16} \approx 1$ means that the correct key has been found more or less uniquely, and $\max_s A_s$ is an upper bound for the workload of variant II of Algorithm 1. The inequality of integrals

$$\int_x^\infty e^{-t^2/2} dt \leq \int_x^\infty \frac{t}{x} e^{-t^2/2} dt = \frac{1}{x} e^{-x^2/2}$$

can be used to give an upper bound for the standardized normal distribution $\mathcal{N}_{0,1}$:

$$\mathcal{N}_{0,1}(x, \infty) \leq \frac{1}{x\sqrt{2\pi}} e^{-x^2/2}$$

We set

$$\tau = \frac{b-a}{\sigma}$$

and derive

$$\begin{aligned} A_s &= 2^{(s-1)8}\, \mathbf{P}\left(\frac{B_s - \binom{s}{2}a}{\sigma\sqrt{\binom{s}{2}}} \geq \tau \sqrt{\binom{s}{2}}\right) \\ &\leq 2^{(s-1)8} \frac{1}{\tau\sqrt{2\pi\binom{s}{2}}} e^{-\frac{1}{2}\binom{s}{2}\tau^2} = \frac{1}{\tau\sqrt{2\pi\binom{s}{2}}} 2^{(s-1)8 - \frac{1}{2\ln(2)}\binom{s}{2}\tau^2} \end{aligned}$$

[1] Since one key byte cannot be determined by the algorithm, the accurate remaining entropy is more properly $\log_2(A_{16}) + 8$.

This approximation of $\log_2(A_s)$ has roughly the form of a parabola in s. The condition $A_{16} \approx 1$ corresponds to the equation

$$\tau = \frac{b-a}{\sigma} \approx \sqrt{2\ln(2)} \approx 1.2$$

Some results are provided in Table 1. This can be interpreted in that we can expect that for $\frac{b-a}{\sigma} \geq 1$, Variant II of Algorithm 1 is successful with workload $\leq 2^{36}$ and remaining entropy ≤ 29.

Table 1. Upper Bounds for the Remaining Entropy

$\frac{b-a}{\sigma}$	$\log_2(A_{16})$	$\log_2(A_4)$	$\max_{s\geq 5}\log_2(A_s)$
1.4	0	24	15
1.2	0	24	23
1.1	10	24	29
1.0	29	24	36
0.9	45	24	46

5.2 A lower Bound of the Remaining Entropy

We want to analyze variant II of Algorithm 1 step by step. We expect that the size of $\#S_{s+1}$ can be approximated by a *conditional* probability of the form

$$\#S_{s+1} \approx \#S_s 2^8 \mathbf{P}\Bigg(B_{s+1} \geq \binom{s+1}{2} b \mid B_s \geq \binom{s}{2} b, \\ B_{s-1} \geq \binom{s-1}{2} b, \ldots, B_5 \geq \binom{5}{2} b\Bigg)$$

Note that

$$X_s = \frac{B_s - \binom{s}{2} a}{\sigma}$$

is the sum of $\binom{s}{2}$ $\mathcal{N}_{0,1}$-distributed independent random variables. Therefore, X_s represents a Gaussian random walk. We write the conditional probability in the form

$$\mathbf{P}\left(B_{s+1} \geq \binom{s+1}{2} b \mid B_s \geq \binom{s}{2} b, B_{s-1} \geq \binom{s-1}{2} b, \ldots, B_5 \geq \binom{5}{2} b\right)$$
$$= \mathbf{P}\left(X_{s+1} \geq \binom{s+1}{2} \tau \mid X_s \geq \binom{s}{2} \tau, X_{s-1} \geq \binom{s-1}{2} \tau, \ldots, X_5 \geq \binom{5}{2} \tau\right)$$

The probability

$$\mathbf{P}\left(X_{s+1} \geq \binom{s+1}{2}\tau, X_s \geq \binom{s}{2}\tau, X_{s-1} \geq \binom{s-1}{2}\tau, \ldots, X_5 \geq \binom{5}{2}\tau\right)$$

can be interpreted as the probability of a Gaussian random walk with at least linear growth at special steps. We get a lower bound of the conditional probability if we omit the conditions on all $s' < s$.

$$\#S_{s+1} \geq V_s 2^8 \mathbf{P}\left(B_s + \sum_{1\leq i\leq s} c_{i,s+1}(k_i \oplus k_{s+1}) \geq \binom{s+1}{2} b \mid B_s \geq \binom{s}{2} b\right)$$

for $s \geq 5$ and $\#S_5 = A_5$. Since $\binom{5}{2} = 10$, we use for $\#S_5 = A_5$ the formula

$$\#S_5 = 2^{32}\mathcal{N}_{0,1}\left(\sqrt{10}\frac{b-a}{\sigma}, \infty\right)$$

The probability $\mathbf{P}\left(B_s + \sum_{1\leq i\leq s} c_{i,s+1}(k_i \oplus k_{s+1}) \geq \binom{s+1}{2} b \mid B_s \geq \binom{s}{2} b\right)$ only depends on s and $\tau = \frac{b-a}{\sigma}$. This probability and therefore all lower bounds of $\#S_{s+1}$ can be calculated numerically. Table 2 extends Table 1 above. We can expect that the remaining entropy and the workload of variant II of Algorithm 1 are within the limits of this table.

Table 2. Bounds for the Remaining Entropy

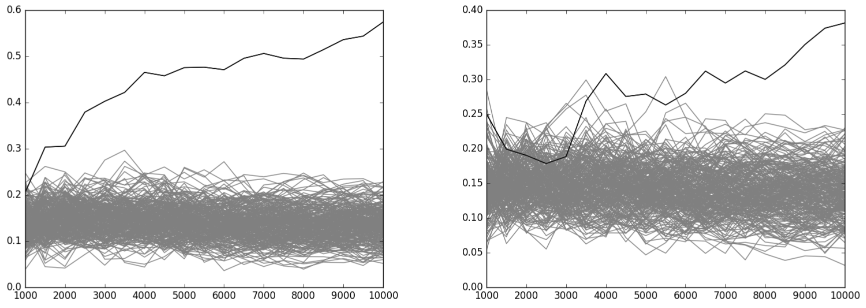
				Lower bound of	Lower bound of
$\tau = \frac{b-a}{\sigma}$	$\log_2(A_{16})$	$\log_2(A_4)$	$\max_{s\geq 5} \log_2(A_s)$	$\log_2(\#S_{16})$	$\max_{s\geq 5} \log_2(\#S_s)$
1.4	0	24	15	0	14
1.2	0	24	23	0	21
1.1	10	24	29	2	26
1.0	29	24	36	21	32
0.9	45	24	46	37	41

5.3 Probability of the Event $B_s(K_1, \ldots, K_s)) \geq \binom{s}{2} b$

We consider the event

$$B_s((K_1, \ldots, K_s)) \geq \binom{s}{2} b \text{ for } \textit{all } s = 16, 15, \ldots, 5$$

Note, that the probability of this event does not depend on b and σ, it is just a real number. On first sight, one could believe that the probability of this



(a) $c_{0,1}(\alpha)$ (b) $c_{0,9}(\alpha)$

Fig. 1. Correlation of $c_{i,j}(\alpha)$ vs # of traces. The correct value of α is shown in black.

event is 2^{-12}. But since $B_{s-1}((K_1,\ldots,K_{s-1}))$ is a subsum of $B_s((K_1,\ldots,K_s))$, the probability of $B_s(K_1,\ldots,K_s)) \geq \binom{s}{2}b$ is larger than $\frac{1}{2}$ if we already know that $B_{s-1}((K_1,\ldots,K_{s-1})) \geq \binom{s-1}{2}b$. We compute an approximation of this probability by a simulation of normally distributed random variables. We get

$$\mathbf{P}\left(B_s((K_1,\ldots,K_s)) \geq \binom{s}{2}b \text{ for } \textit{all } s = 16, 15, \ldots, 5\right) \approx 0.15.$$

To this end, the assumption $B_s(K_1,\ldots,K_s)) \geq \binom{s}{2}b$ for all s is not too restrictive.

6 Experiments

Software Implementation

Our setup consists of an AES-128 based software implementation running on an Atmel ATMEGA328P-PU. The S-Boxes are realized as lookup tables and stored in the program memory of the ATMEGA. The S-Boxes are masked using the method presented in [2]. This masking is known to have an inherent weakness, however the attack does not use any systematic way in exploiting this weakness, and as shown in [8, 13], even state-of-the-art Canright S-Boxes [5] are susceptible to correlation attacks. Hence we anticipate that our results are representative. Moreover, the masking used [2] is straightforward to implement. The ATMEGA was setup on a custom prototype board, and powered by a lab-grade power supply at 3.3 Volt. We used a LeCroy HDO6104 to record the power consumption of the ATMEGA with a resistor against ground. We recorded N traces of AES-128 encryption. The plaintext was chosen at random, and we attacked the masked_subbytes procedure of the first AES round.

Practical Results

We recorded 10000 traces with randomly generated plaintext values. Our implementation follows closely the approach shown in [13] and we compute the correlation w.r.t. each possible value of one $C_{i,j,\alpha,t}$. Figure 5a shows the resulting correlation of $C_{0,1,\alpha,t}$ for one AES round, i.e. in our setup $t = 0, ..., 25809$. Correlation peaks at the end of the S-Box for the correct value are clearly visible. However a correlation peak is not apparent for every pair of key byte positions i, j; for example considering the correlation values $C_{9,11,\alpha,t}$ as depicted in Figure 5b, no clear peak is observable for the correct value α. Nevertheless when ranking all possible values $c_{9,11}(\alpha)$, the correct value is still at position 18. Fewer traces result in less collisions and more noise, and the rankings become more fuzzy. To give two examples, the rankings for the correct value of α for 2500 and 10000 traces are as shown in Figure 3a and 3b. If more traces are available, the correlation values $c_{i,j}(\alpha)$ for the correct value α become very distinct from those for incorrect values of α, as illustrated in Figure 1a and Figure 1b.

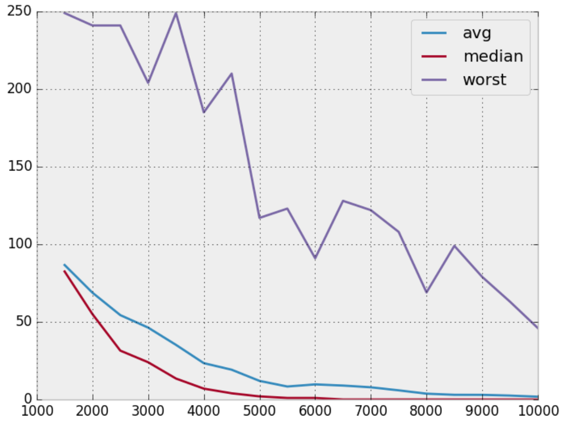


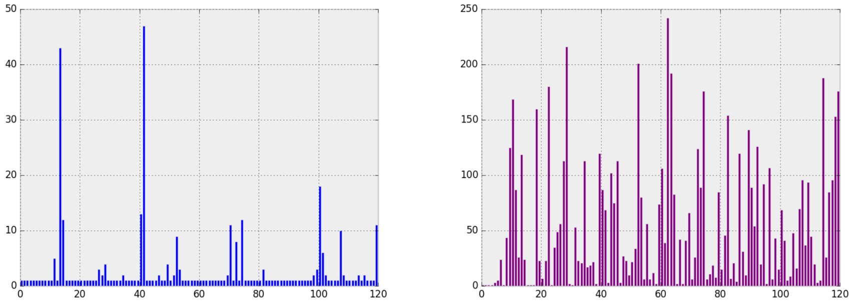
Fig. 2. Number of Traces vs. Ranking Positions of $c_{i,j}(\alpha)$ for correct α.

For $N = 10000$, the correct value of α often shows on rank 1, but there are some outliers. For $N = 2500$, there are few rankings with position 1. Nevertheless, a part of the key can be recovered and the key entropy is significantly reduced, as is shown in our experiments. Figure 2 shows average, median, and worst ranking positions for the correct value of α in relation to the number of available traces.

For Variant II of Algorithm 1 one needs to choose appropriate input values $g_2, \cdots, g_{16}$. This requires prior knowledge about the quality of the rankings $c_{i,j}(\alpha)$. If such knowledge is not available, appropriate values could also be estimated by manual analysis of the rankings $c_{i,j}(\alpha)$ and/or by visual inspection of

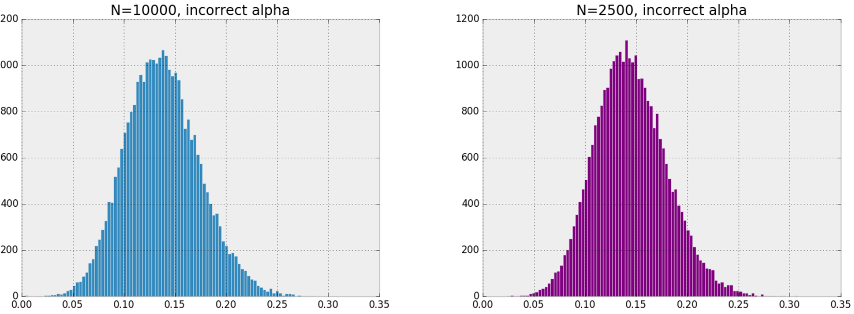
Table 3. Full Key Recovery for $W = 1500$ using Algorithm 1 (Variant I).

	1500 ... 3500	4000	4500	5000	5500	6000	6500
$1 \rightarrow 16$	–	✓	✓	✓	✓	✓	✓
$16 \rightarrow 1$	–	–	–	–	–	–	–
random	–	1/5	3/5	3/5	3/5	3/5	3/5
	7000	7500	8000	8500	9000	9500	10000
$1 \rightarrow 16$	–	✓	✓	✓	✓	✓	✓
$16 \rightarrow 1$	–	–	–	–	–	–	–
random	3/5	4/5	3/5	4/5	5/5	5/5	5/5



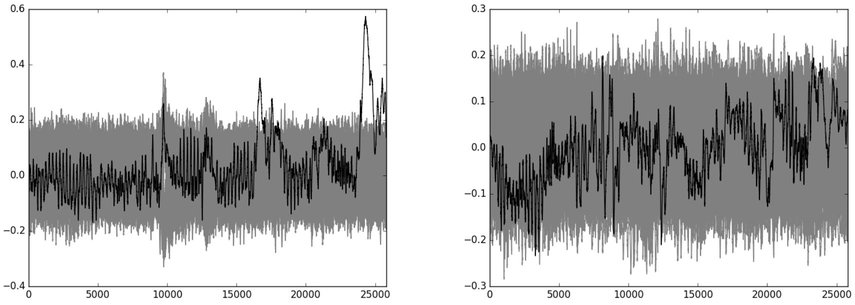
(a) $N = 10000$ (b) $N = 2500$

Fig. 3. Ranking positions for correct α.



(a) $N = 10000$ (b) $N = 2500$

Fig. 4. Distribution of $c_{i,j}(\alpha)$ for incorrect values of α.



(a) $C_{0,1,\alpha,t}$

(b) $C_{9,11,\alpha,t}$

Fig. 5. Correlation $C_{i,j,\alpha,t}$ for each timepoint t within one trace. The correct value of α is denoted in black.

$C_{i,j,\alpha,t}$. For example the comparison of Figure 5a and Figure 5b indicates that for $c_{0,1}(\alpha)$ the ranking for the correct value of α is very likely at one of the top positions, whereas for $c_{9,11}(\alpha)$ this is likely not the case. On the other hand, Variant I of Algorithm 1 requires no prior knowledge at all. This is why we have here chosen to implement Variant I of the algorithm.

As mentioned in Section 3, the success of the algorithm depends on the order in which the next key byte position is chosen, the parameter W of candidates that are kept in each iteration, and of course the number of available traces N. As mentioned in Section 2, in particular the order of choosing the next key byte position is important, as the next example illustrates:

Example 1. For simplicity, suppose we have only a key consisting of three bytes, and suppose one key byte can take only value 0 or 1. Let $c_{i,j}(\alpha)$ be as follows:

$$c_{0,1}(0) = 0.4 \qquad c_{0,1}(1) = 0.1 \qquad c_{1,2}(0) = 0.4$$
$$c_{1,2}(1) = 0.1 \qquad c_{0,2}(0) = 0.1 \qquad c_{0,2}(1) = 0.8$$

Suppose that we set $W = 2$. Assume the order $0 < 1 < 2$. We start with $S_1 = \{0, 1\}$. Since $c_{0,1}(0) > c_{0,1}(1)$ and $0 \oplus 0 = 0$ as well as $1 \oplus 1 = 0$, we yield the set $S_2 = \{00, 11\}$. Algorithm 1 terminates with $S_3 = \{001, 110\}$. On the other hand, it is not difficult to verify that for the key byte order $2 < 1 < 0$, Algorithm 1 terminates with the set $S_3 = \{100, 011\}$. Note also that Algorithm 1 is nondeterministic in general: If we set $W = 1$ in the second step during the run with order $0 < 1 < 2$, we have $B((00)) = B((11)) = 0.4$, and it is open which partial key to keep.

We executed Algorithm 1 for $N = 1500$ up to $N = 10000$ in steps of 500 traces, and for both $W = 1500$ and $W = 10000$. As for the dependency on the order, we considered the natural order of starting at key byte position 1 and moving upward to position 16 (denoted by 1 → 16 in the following), the reverse order of starting at position 16 and moving down to 1 (denoted by 16 → 1), and on five randomly chosen orders for each value of N. Table 3 and Table 4[2] show results for the full recovery of the key for $N = 1500...10000$ and the various orders. As one can see, the full key is in the computed set with good probability if at least 4000 traces are available. The probability can be increased, if a larger parameter $W = 10000$ is chosen. For less than 4000 traces, Table 5 shows the maximum number of correctly recovered key bytes in the computed set. For example, when choosing the order 1 → 16, then in the case of 2500 traces, the computed set contains a key where 12 key bytes are correctly identified. In other words, the entropy is significantly reduced, and it is not difficult to devise an algorithm that exploits the fact that one can assume that a certain amount of key byte position are correctly identified.

Table 4. Full Key Recovery for $W = 10000$ using Algorithm 1 (Variant I).

	1500 ... 3500	4000	4500	5000	5500	6000	6500
1 → 16	–	✓	✓	✓	✓	✓	✓
16 → 1	–	–	–	–	✓	✓	✓
random	–	1/5	3/5	3/5	3/5	3/5	3/5
	7000	7500	8000	8500	9000	9500	10000
1 → 16	–	✓	✓	✓	✓	✓	✓
16 → 1	✓	✓	–	✓	✓	✓	✓
random	3/5	4/5	3/5	4/5	5/5	5/5	3/5

Table 5. Partial Key Recovery for $W = 1500$ using Algorithm 1 (Variant I).

	1500	2000	2500	3000	3500
1 → 16	3	5	12	13	14
16 → 1	2	2	4	1	5
random #1	1	5	3	7	9
random #2	1	5	2	5	14
random #3	2	3	5	12	5
random #4	2	3	5	7	8
random #5	2	2	1	8	12

[2] Note that the miss for $N = 8000$ and 16 → 1 in Table 4 is precisely due to the nondeterministic behavior of Algorithm 1, as illustrated in Example 1.

In order to further interpret these results, we investigated the distribution $c_{i,j}$ in α for all (i,j). For each (i,j) we computed the expected value as well the standard deviation, which were very similar. Figure 4a and Figure 4b show histograms of all $255 \cdot 120$ $c_{i,j}$ with incorrect value α for $N = 10000$ and $N = 2500$, respectively. This is apparently very close to a normal distribution. Expected value and standard deviation are 0.138 ± 0.037 for $N = 10000$ and 0.144 ± 0.036 for $N = 2500$. As derived above, we expect a theoretical bound

$$\frac{b-a}{\sigma} \approx \sqrt{2\ln(2)} \approx 1.2$$

This is the smallest value b, for which we can expect that the evaluation function B succeeds. For our experimental data we yield

$$\frac{b-a}{\sigma} = \frac{0.32 - 0.138}{0.037} \approx 5.0 \text{ for } N = 10000$$

and

$$\frac{b-a}{\sigma} = \frac{0.19 - 0.1445}{0.0355} \approx 1.3 \text{ for } N = 2500$$

Apparently, the parameters for $N = 2500$ are very close to the theoretical threshold. This fits with our theoretical observations in previous sections and the experimental data.

7 Conclusion and Future Work

We have shown how to reduce the remaining key entropy of the attack introduced by [13] by providing a practical, easy-to-implement algorithm. Our theoretical analysis shows that this algorithm exploits the leakage in a natural way. Moreover, we provide a way to assess the leakage of a device w.r.t. the attack, which could be used e.g. in a Common Criteria security evaluation. Our practical evaluation supports the theoretical analysis. In particular we show that using our algorithm, a full recovery of the AES key is possible with only few available traces. That is, the key can be recovered in a setting where no visual clues w.r.t. the correct ranking are available and the attack as described by [13] would not have been applicable. The practical analysis of our algorithm for an AES implemented in hardware on an FPGA, as done originally in [13] is subject to future work.

Acknowledgements We would like to thank Sven Freud for creating the circuit board for power analysis, and Tobias Senger for his help with implementing the masking scheme.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The em side—channel(s). In: Proc. 4th CHES. pp. 29–45 (2003)

2. Akkar, M.L., Giraud, C.: An implementation of des and aes, secure against some attacks. In: Proc. 3rd CHES. pp. 309–318 (2001)
3. Bogdanov, A.: Multiple-differential side-channel collision attacks on aes. In: Proc. 10th CHES. pp. 30–44 (2008)
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Proc. 6th CHES (2004)
5. Canright, D.: A very compact s-box for aes. In: Proc. 7th CHES. pp. 441–455 (2005)
6. Canright, D., Batina, L.: A very compact "perfectly masked" s-box for aes. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) Proc. 6th ACNS. pp. 446–459 (2008)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Proc. 4th CHES. pp. 13–28 (2003)
8. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Improved collision-correlation power analysis on first order protected aes. In: Proc. 13th CHES. pp. 49–62 (2011)
9. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Proc. 3rd CHES. pp. 251–261 (2001)
10. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Proc. 19th CRYPTO. pp. 388–397 (1999)
11. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, pp. 104–113 (1996)
12. Ledig, H., Muller, F., Valette, F.: Enhancing collision attacks. In: Proc. 6th CHES. pp. 176–190 (2004)
13. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Proc. 12th CHES. pp. 125–139 (2010)
14. National Institute of Standards and Technology: FIPS PUB 197. Advanced Encryption Standard. Tech. rep. (2001)
15. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: Proc. E-smart. pp. 200–210 (2001)
16. Schramm, K., Leander, G., Felke, P., Paar, C.: A collision-attack on aes. In: Proc. 6th CHES. pp. 163–175 (2004)
17. Schramm, K., Wollinger, T., Paar, C.: A new class of collision attacks and its application to des. In: Proc. 10th FSE. pp. 206–222 (2003)