Differential Cluster Analysis*

Lejla Batina[1], Benedikt Gierlichs[1], and Kerstin Lemke-Rust[2]

[1] K.U. Leuven, ESAT/SCD-COSIC and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be
[2] University of Applied Sciences Bonn-Rhein-Sieg
Grantham-Allee 20, 53757 Sankt Augustin, Germany
kerstin.lemke-rust@h-brs.de

Abstract. We propose a new technique called Differential Cluster Analysis for side-channel key recovery attacks. This technique uses cluster analysis to detect internal collisions and it combines features from previously known collision attacks and Differential Power Analysis. It captures more general leakage features and can be applied to algorithmic collisions as well as implementation specific collisions. In addition, the concept is inherently multivariate. Various applications of the approach are possible: with and without power consumption model and single as well as multi-bit leakage can be exploited. Our findings are confirmed by practical results on two platforms: an AVR microcontroller with implemented DES algorithm and an AES hardware module. To our best knowledge, this is the first work demonstrating the feasibility of internal collision attacks on highly parallel hardware platforms. Furthermore, we present a new attack strategy for the targeted AES hardware module.

Keywords: Differential Cluster Analysis, Side-channel Cryptanalysis, Collision Attacks, Differential Power Analysis, AES Hardware.

1 Introduction

Side-channel analysis became a mature area in the past decade with many contributions to new attacks, models and countermeasures since the pioneering results of Differential Power Analysis (DPA) [14]. DPA exploits the fact that information on secret key bits can leak through a side-channel. It typically requires known input to the cryptographic algorithm. Many practical and more recently some theoretical works have been published showing the importance of applying known techniques and ideas from other research communities.

An internal collision attack (CA) is another kind of side-channel analysis. Collision attacks have been introduced by Schramm *et al.* in [22]. A collision in

* Work supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State, by FWO project G.0300.07, by the European Commission under contract number ICT-2007-216676 ECRYPT NoE phase II, by K.U. Leuven-BOF (OT/06/40), and by IWT-Tetra STRES (80138).

C. Clavier and K. Gaj (Eds.): CHES 2009, LNCS 5747, pp. 112–127, 2009.

an algorithm occurs if, for at least two different inputs, a function within the algorithm returns the same output. If this happens, the side-channel traces are assumed to be very similar in the time span when the internal collision persists. Collision attacks use the side-channel to detect collisions and afterwards offline computation with or without precomputed tables for key recovery. For both steps there are different approaches proposed in the literature. Considering the assumptions, attacks can be with either chosen or known inputs.

The work of [21] and in particular recent works on collision attacks [3,4,5,6,7] veer away from long sequences of instructions [22,15], *e.g.* collisions that persist for an entire round, and target short-scale intermediate results. Our approach follows this development and shows that internal collisions can be the source for both DPA and CA leakage.

More precisely, our work introduces Differential Cluster Analysis (DCA) as a new method to detect internal collisions and extract keys from side-channel signals. Our approach is to revisit collision attacks in the unsupervised analysis setting, which can be two-fold *e.g.* viewed as collision and DPA approach. Our strategy includes key hypothesis testing and the partitioning step similar to those of DPA. Partitioning then yields collisions for the correct key which are detected by cluster analysis. DCA typically requires known input data to the cryptographic algorithm and can be applied to arbitrary algorithmic collisions as well as to implementation specific collisions. Cluster analysis relates to some extent to standard DPA, which is obvious for the univariate case. While DCA is inherently multivariate, the technique inspires a simple extension of standard DPA to multivariate analysis. The most interesting difference is that cluster analysis is sensitive to more general leakage features and does not require a power model for multi-bit collisions.

The idea of clustering for simple side-channel attacks was already used in the work of Walter [25]. Therein, he uses clusters (called buckets) for partitioning sets of similar measurements in order to reveal exponent digits for an attack on sliding windows exponentiation. Our work is also related to Mutual Information Analysis (MIA) [12] in that both approaches can succeed without but benefit from a good power model. Also related to our work is the use of Gaussian mixture models for masked implementations [16]. In this work parameters of different Gaussian components that best fit to the observed mixed multivariate side-channel leakage are estimated without knowing the masks.

Our experiments confirm the findings on two platforms. One platform is an unprotected software implementation of the DES algorithm running on an Atmel AVR microcontroller and the other one is a hardware implementation of AES-128. Collision attacks on platforms like the latter are believed to be unfeasible due to the high parallelism of operations, *e.g.*, [5] states "the collision attacks on AES are mainly constrained to 8-bit software implementations on simple controllers".

The paper is organized as follows: Section 2 describes our new approach to collision detection by use of cluster analysis of measurement samples. Section 3 introduces the new classification of collisions into algorithmic and

implementation specific collisions and presents examples for both classes. Experimental results are provided in Section 4 and Section 5 summarizes the results of this work.

2 Differential Cluster Analysis: The General Approach

An internal collision in a cryptographic algorithm occurs if, for at least two inputs $x_i, x_{i'} \in \{0,1\}^u$ with $x_i \neq x_{i'}$ and subkey $k^\circ \in \{0,1\}^v$, values of one particular intermediate state $\Delta \in \{0,1\}^w$ collide. The intermediate state Δ is a specific property of the cryptographic algorithm. Although we provide examples from symmetric schemes the general approach is also valid for public key schemes. In the case of DES for example, the intermediate state is given by a few bits after an S-box access. Let f_k denote the key dependent function $f_k : \{0,1\}^u \mapsto \{0,1\}^w$ that computes Δ for a given input x in a cryptographic algorithm. The function f_k is said to be a *many-to-one collision function* if many inputs are mapped to the same output.

Unlike previous collision attacks that search for similarity between different power traces, the new key recovery attack aims at detecting significantly separated clusters as result of internal collisions. This is an unsupervised learning process. The adversary observes the side-channel response on input patterns, but has incomplete knowledge about the internal state of the system, especially she does not know any key and therefore any true labels of samples. The adversary, however, usually knows the number of different clusters, *i.e.*, the number of possible values for Δ.

DCA assumes that it is feasible to run statistics for all subkey candidates in the algorithm, *i.e.*, v is a small number. For common constructions of ciphering algorithms such as AES and DES this assumption is clearly fulfilled. In the first step of the attack, the adversary classifies measurement samples $\boldsymbol{i}_n$ ($n \in \{1, \ldots, N\}$, where N is the total number of samples) with input[1] x_n into 2^w classes according to $f_k(x_n)$ with guessed subkey hypothesis k. As result, the adversary obtains 2^w bins of classified measurements for each key guess.[2] This new attack tests whether clustering statistics, such as good cluster separation or high cluster compactness, indicates a separation of distinct clusters. Note that if $k = k^\circ$ the separation of the samples into the 2^w bins corresponds to the computation of the cryptographic device. If the side-channel leakage of different values of Δ is detectable, this in turn reveals the correct key. If the subkey guess is wrong the measurements are generally classified into bins incorrectly, *i.e.* almost all bins include samples that result from different values of Δ. As a consequence, clusters of measurements resulting from different values of Δ are expected to broaden the statistical distribution of each bin and to smooth out side-channel differences.

DCA classifies objects into classes according to special collisions that relate to known inputs and a key hypothesis. Cluster statistics are used to detect

[1] Note that this attack can be applied to both known and chosen input.

[2] Note that not all 2^w states might be possible in a given cryptographic algorithm.

the collisions. Note that this collision and clustering attack is a multivariate approach. Cluster statistics can be applied to measurements samples from a single (univariate) or from multiple (multivariate) time instants. Multivariate DCA benefits if prior knowledge on the relative distances of points in time that contain exploitable side-channel leakage is available, *e.g.* as a result of profiling. Furthermore, additional options exist for combining DCA results. One example is to combine the outcomes of DCA for w 1-bit intermediate states for the analysis of an w-bit intermediate state.

2.1 Cluster Statistics

We provide details about criterion functions for clustering and describe how to measure the clustering quality. In literature, *e.g.* [11,24], cluster statistics use a number of cluster characteristics. In Table 1 characteristics for c clusters $\mathcal{D}_i$, $i \in \{1, \ldots, c\}$ with population n_i of vectors $\boldsymbol{x}$ and total population N are summarized. Note that in case of univariate analysis all vectors have only one element.

The *sum-of-squared-error* is a widely used cluster criterion function. It computes

$$J_{SSE} = \sum_{i=1}^{c} \sum_{\boldsymbol{x} \in \mathcal{D}_i} \parallel \boldsymbol{x} - \boldsymbol{m_i} \parallel^2 .$$

This function evaluates the sum of the squared Euclidean distances between the vectors $(\boldsymbol{x} - \boldsymbol{m_i})$. Informally speaking that is the sum of the scatter over all clusters. The optimal partition minimizes J_{SSE}. An alternative is the *sum-of-squares* criterion. It evaluates the square of the Euclidean distance between the cluster centroids $\boldsymbol{m}_i$ and the total mean vector $\boldsymbol{m}$:

$$J_{SOS} = \sum_{i=1}^{c} n_i \parallel \boldsymbol{m_i} - \boldsymbol{m} \parallel^2 .$$

The optimal partition maximizes J_{SOS}. An interesting observation is that the sum of J_{SSE} and J_{SOS} is a constant, thus minimizing J_{SSE} (yielding intra cluster cohesion) is equivalent to maximizing J_{SOS} (yielding inter cluster separation) [24].

In the context of side-channel analysis computing variances can also be useful. In such cases, one can either take variances into account explicitly or normalize

Table 1. Cluster Characteristics

Mean vector for the i-th cluster:	$\boldsymbol{m_i} = \frac{1}{n_i} \sum_{\boldsymbol{x} \in \mathcal{D}_i} \boldsymbol{x}$
Total mean vector:	$\boldsymbol{m} = \frac{1}{N} \sum_{i=1}^{c} n_i \boldsymbol{m_i}$ where $\sum_{i=1}^{c} n_i = N$
Variance vector for the i-th cluster:	$\boldsymbol{v_i} = \frac{1}{n_i} \sum_{\boldsymbol{x} \in \mathcal{D}_i} (\boldsymbol{x} - \boldsymbol{m_i})^2$
Total variance vector:	$\boldsymbol{v} = \frac{1}{N} \sum_{i=1}^{c} n_i \boldsymbol{v_i}$ where $\sum_{i=1}^{c} n_i = N$
Squared Euclidean norm ($\mathbb{R}^k \rightarrow \mathbb{R}$):	$\|\|(z_1, z_2, \ldots, z_k)\|\|^2 = \sum_{j=1}^{k} z_j^2$

the measurements before evaluating cluster criteria like J_{SOS} and J_{SSE}. The variance test [23] is a criterion function that evaluates

$$J_{VAR} = \frac{\| \boldsymbol{v} \|^2}{\frac{1}{N} \sum_{i=1}^{c} n_i \| \boldsymbol{v_i} \|^2} ,$$

i.e. the ratio between the overall variance and the weighted mean of intra cluster variances. The optimal partition maximizes J_{Var}. The student's T-test [13] evaluates the distances between cluster centroids, normalized by intra cluster variances and cluster sizes. We use it in the T-test criterion that evaluates the sum of squared distances for all pairs of clusters

$$J_{STT} = \sum_{i,j=1;i\neq j}^{c} \frac{\| \boldsymbol{m_i} - \boldsymbol{m_j} \|^2}{\sqrt{\frac{\|\boldsymbol{v_i}\|^2}{n_i} + \frac{\|\boldsymbol{v_j}\|^2}{n_j}}} .$$

Again, the optimal partition maximizes J_{STT}.

2.2 The General Approach

Here we summarize our general approach:

1. Measure N samples $\boldsymbol{i}_n$ of power traces while the targeted device computes the cryptographic algorithm with unknown fixed subkey k°. For each sample, store the associated known input x_n for $n = 1, 2, \ldots, N$.
2. For each subkey hypothesis $k \in \{0,1\}^v$ do the following[3]:
   (a) Sort the N measurements according to $\Delta_i = f_k(x_i)$ into 2^w clusters $\mathcal{D}_0, \ldots, \mathcal{D}_{2^w-1}$.
   (b) For each cluster $\mathcal{D}_i$ do the following: Compute mean value $\boldsymbol{m}_i$ and variance $\boldsymbol{v_i}$.
   (c) Compute a cluster criterion J_k (*e.g.* J_{SSE} or J_{SOS}) to quantitatively assess the quality of the cluster separation.
   (d) Store the pair of k and J_k: (k, J_k).
3. Rank the pairs (k, J_k) according to J_k.
4. Output the key candidate k with the value of J_k that leads to the best clustering quality (min. or max., depending on the criterion function).

2.3 Refinements

Several refinements of the general approach can make the attack more efficient.

(i) Known input x_n is assumed in the general approach. Noise due to non-targeted operations can be highly reduced if the input x_n can be chosen. If the input can be further adaptively chosen this allows to apply an adaptive sieving of key candidates, thereby minimizing the number of samples needed.

[3] In practice, steps might be iterated for components of $\boldsymbol{i}_n$, *e.g.*, each iteration might include samples of one or a few clock cycles of the device.

(ii) In the general approach we do not include any assumption on similarity of some clusters. Integration of a side-channel leakage model is possible by merging of clusters, *e.g.*, for a Hamming weight model the number of clusters is reduced from 2^w to $w + 1$. See Sect. 2.4 and 3 for a detailed discussion.
(iii) Depending on the algorithm and the choice of Δ related key hypotheses that lead to so called "ghost peaks" [8] exist. For such key hypotheses the formed clustering is, to a certain degree, identical to the correct clustering. If such related key hypotheses exist this may allow for a special adaption of the analysis.
(iv) Prior normalization of the N samples $\boldsymbol{i}_n$ to zero mean and unit standard deviation is often useful in practice if the spread is due to noise. Transformation to principal components is another useful pre-processing step.

2.4 Detailed Comparison with DPA

Here, we compare DCA with DPA in more detail. For DPA, different variants are known in literature [18]. The original approach for DPA [14] selects one bit of an intermediate state to test whether there is a provable imprint of this bit in the power traces. Kocher's original DPA can be derived from DCA when restricted to two clusters ($w = 1$) and one time instant. The proposed statistics, however, differ to some extent.

Essential differences between DPA and DCA occur for $w > 1$. Multi-bit DPA was first introduced by Messerges *et. al.* [19]. The main idea is that each bit in an intermediate state causes the same amount of leakage and that considering multiple bits at once in an attack may be advantageous. A drawback of this "all-or-nothing" DPA is the inefficient use of measurements: although 2^w clusters exist, only two of them are used. Correlation Power Analysis (CPA) [8] with a Hamming weight or distance model correlates the predicted side-channel leakage with the measured side-channel traces to check for provable correlation. CPA uses all available measurements and can be very efficient in practice. Potential drawbacks are the requirement for a good leakage model, which may be hard to come up with, and the fact that CPA with a Hamming model may not capture all leakage details as Pearson correlation solely tests for equidistance of cluster centroids, *i.e.* linearity.

The general DCA approach uses 2^w clusters and all available measurements. The assumption of a side-channel leakage model is only optional in multi-bit DCA, *e.g.* an adjustment to the Hamming weight or distance model works with $w+1$ clusters instead. Unlike CPA, DCA can also capture special features of the leakage, *e.g.* unequal densities, non-spherical shapes, and unequal proximities.

Example for $w = 2$ where DCA is advantageous. The following example shows that special cases of side-channel leakage exist that can neither be detected with single-bit DPA nor with Hamming weight based CPA, but still with DCA. We consider a general case with $w = 2$, *i.e.* four equally likely intermediate states "00", "01", "10", "11" that have a mean side-channel leakage of α_0, α_1, α_2, α_3, respectively.

Resistance against single-bit DPA requires that the following conditions must hold: $\alpha_0+\alpha_1 = \alpha_2+\alpha_3$ and $\alpha_0+\alpha_2 = \alpha_1+\alpha_3$ in order to not reveal any information on either the left or the right bit of the intermediate state. To achieve resistance against Hamming weight CPA the following condition must hold: $\alpha_0 = \alpha_3$.

Resistance against both single-bit DPA and Hamming weight CPA is achieved if the previous three conditions are combined, which leads to $\alpha_0 = \alpha_3$ and $\alpha_1 = \alpha_2$. The trivial solution $\alpha_0 = \alpha_1 = \alpha_2 = \alpha_3$ implies a single joint cluster and no information leakage at all. If $\alpha_0 \neq \alpha_1$ two different clusters arise: a joint cluster of intermediate states "00" and "11" as well as a joint cluster of intermediate states "01" and "10". These two separated clusters can be directly detected with DCA which, assuming that the intermediate state results from a non-injective mapping and depends on a secret subkey value, enables key recovery. We acknowledge that an adapted selection function that tells apart whether the two bits are equal or not would enable multi-bit DPA and CPA. However, the general DCA approach detects such particular features by its own nature while DPA and CPA require a guiding selection function. As mentioned earlier, coming up with such a selection function may not always be easy and can require detailed knowledge of the device's leakage behavior.

3 Differential Cluster Analysis: Applications

In this section we first distinguish between two types of collisions, algorithmic and implementation specific collisions. While an algorithm assumes a more abstract concept, an implementation is a concrete realization of it. There may exist many implementations of an algorithm. On the other hand, having an implementation, one can find just one algorithm corresponding to it.

In general, internal collisions may occur due to the algorithm or may be caused by a particular implementation. In the former case, we speak about algorithmic collisions that are results of non-injective mappings within the algorithm. Algorithmic collisions are therefore typically implementation independent. For algorithmic collisions the adversary guesses the colliding intermediate state as it is computed in the algorithm, *e.g.* a cryptographic standard. Neither a leakage model nor any other information about implementation properties are used here. The question is whether clusters for a complete or partial intermediate state can be distinguished when applying cluster criterion functions.

On the other hand, implementation specific collisions can be observed merely due to the way a certain algorithm is implemented. In other words, there can be ways to implement an algorithm that induce this type of collisions, while the collisions are not obvious in the algorithm. For implementation specific collisions the adversary adds knowledge of confirmed or assumed implementation properties to the algorithmic attack. Examples include targeting particular intermediate states of the implementation or applying a special leakage model. Clusters are built according to such a special implementation specific intermediate state or clusters are merged according to such a special leakage model. Next we present examples for both cases.

3.1 Algorithmic Collisions

Data Encryption Standard. We revisit collisions in isolated S-boxes [22]. The S-box function is $4-to-1$, *i.e.*, it maps four inputs to one output. For each S-box, the collision function $f_k = S_i$ maps a group of four values of $z \in \{0, \ldots, 2^6 - 1\}$ to one cluster of $f_k(z) \in \{0, \ldots, 2^4 - 1\}$. As $z = x \oplus k$ the mapping depends on subkey k given known x. Alternatively, the 4-bit output differential of the first round can be used as a collision function that evolves from tracking the four output bits of the active S-box to the R-register in the next round. For the correct key hypothesis, up to sixteen separated clusters are expected.

Advanced Encryption Standard. An isolated AES S-box is not a collision function because one merely obtains a permutation of cluster labels for each key guess. Targeting only r-bit ($1 \leq r < 8$) of an S-box outcome, however, is an applicable many-to-one collision function, at the cost of loosing predicted bits of the full intermediate result.

Collisions also occur in a single output byte after the MixColumns transformation as described in [21], because this is a $2^{24} - to - 1$ collision function. For the purpose of saving computation time, an attack with two measurement series using two fixed input bytes to each MixColumns transformation is advantageous, if applicable. This leads to a $2^8 - to - 1$ collision function. Let b_{00} be the first output byte of MixColumns, (x_{22}, x_{33}) the two fixed input bytes, S the AES S-box, and (k_{00}, k_{11}) the target for key recovery. Then we have $b_{00} = 02{\cdot}S(x_{00} \oplus k_{00}) \oplus 03{\cdot}S(x_{11} \oplus k_{11}) \oplus c$, where constant $c = S(x_{22} \oplus k_{22}) \oplus S(x_{33} \oplus k_{33})$. Without loss of generality we assume $c = 0$ and label clusters with the value of b_{00}, as there exists a bijective mapping from b_{00} to the correct cluster label $b_{00} \oplus c$. Similarly, the second measurement series can be used to recover the other two key bytes.

3.2 Implementation Specific Collisions

Examples for this type of collisions include hardware and software implementations at which some internal registers are used multiple times, *e.g.*, in subsequent rounds of an algorithm or during subsequent processing of functional units. Hereby, internal collisions can be caused for intermediate results that are not provided in an algorithmic description.

AES-128 hardware architecture. In Appendix A we describe an AES-128 hardware architecture that leaks such implementation dependent collisions and that we analyze in the experimental part of this paper. The architecture is very compact and suitable for smartcards and other wireless applications, which makes the attacks extremely relevant.

Let $x_i \in \{0,1\}^8$ ($i \in \{0, 1, \ldots, 15\}$) denote the plaintext byte. Accordingly, let $k_i \in \{0,1\}^8$ be the corresponding AES key byte. By S we denote the AES S-box. The targeted intermediate result is

$$\Delta_{ii'} = S(x_i \oplus k_i) \oplus S(x_{i'} \oplus k_{i'}) \tag{1}$$

with $i \neq i'$. This intermediate result $\Delta_{ii'}$ is, *e.g.*, given by the differential of two adjacent data cells in the studied AES hardware architecture. $\Delta_{ii'}$ depends on two 8-bit inputs to the AES S-box $(x_i \oplus k_i, x_{i'} \oplus k_{i'})$ and that is crucial for the new attacks. For the key-recovery attack we consider the pairs of known plaintext $(x_i, x_{i'})$ and fixed unknown subkey $(k_i^\circ, k_{i'}^\circ)$. Our attack is enabled because (1) is the result of a 2^8-to-1 collision function.

Using the general approach one evaluates the clustering quality for 2^{16} key hypotheses, but for this particular internal collision a special approach is feasible: for $\Delta_{ii'} = 0$ equation (1) simplifies to

$$S(x_i \oplus k_i) = S(x_{i'} \oplus k_{i'}) \Rightarrow x_i \oplus k_i = x_{i'} \oplus k_{i'} \Rightarrow k_i \oplus k_{i'} = x_i \oplus x_{i'} \,.$$

This leads to the observation that for all key guesses $(k_i,k_{i'})$ satisfying $k_i \oplus k_{i'} = k_i^\circ \oplus k_{i'}^\circ$ the elements assigned to the class $\Delta_{ii'} = 0$ are the same as in the correct assignment (caused by $(k_i^\circ, k_{i'}^\circ)$). This allows for a two-step key recovery attack:

1. Determine the correct xor-difference $k_i^\circ \oplus k_{i'}^\circ$ based on 2^8 hypotheses.
2. Determine the correct pair $(k_i^\circ, k_{i'}^\circ)$ based on 2^8 hypotheses.

We illustrate these steps in Sect. 4.2. As a result of this approach, the complexity of the key search is reduced from 2^{16} to 2^9 hypotheses.

Note that this strategy is not exclusively accredited to DCA. This implementation specific attack strategy can also be applied using standard DPA techniques.

4 Experimental Results

In this section we describe the setups for our experiments, experimental settings and provide results. The empirical evidence shows that the proposed attacks are practical and lead to a successful key recovery.

4.1 DES in Software

Our experimental platform is an unprotected software implementation of the DES running on an Atmel AVR microcontroller. The code executes in constant time and the μC is clocked at about 4MHz. For our first experiments we used a set of $N = 100$ power traces $\boldsymbol{i}_n$. The samples represent the voltage drop over a 50Ω shunt resistor inserted in the μC's ground line. Power consumption was sampled at 200 MS/s during the first two rounds of DES encryption with a constant key $k^\circ = (k_0^\circ, \ldots, k_7^\circ)$. The plaintexts $x_n = (x_{0n}, \ldots, x_{7n})$ were randomly chosen from a uniform distribution to simulate a known plaintext scenario. The targeted 4-bit intermediate result is $S_1(\tilde{x}_1 \oplus \tilde{k}_1)$ where $\tilde{x}_1$ and $\tilde{k}_1$ denote respectively the six relevant bits of plaintext and roundkey entering the first DES S-box (S_1) in the first round. Figure 1 shows results of our cluster analysis for the criteria J_{SOS}, J_{SSE}, J_{VAR} and J_{STT}. The plots show that all four criteria allow recovery of $\tilde{k}_1$ but also indicate that J_{VAR} and J_{STT} are preferable. Furthermore, Figure 1 nicely illustrates the complementary character of J_{SOS} and J_{SSE}.

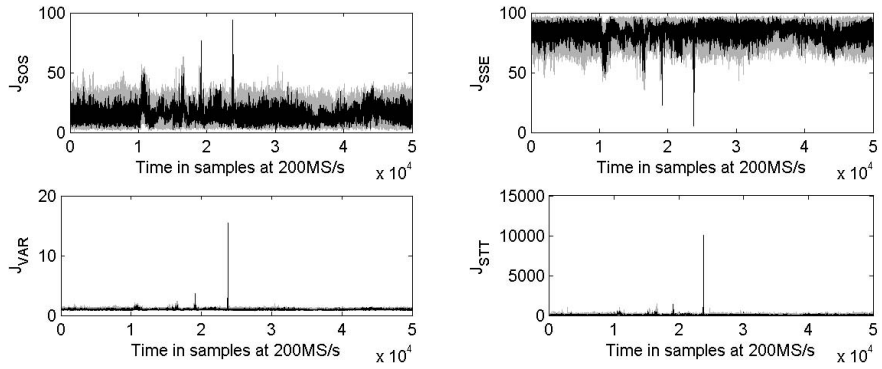

Fig. 1. Results of Cluster Analysis with J_{SOS} (top, left), J_{SSE} (top, right), J_{VAR} (bottom, left) and J_{STT} (bottom,right) when using 2^6 key hypotheses on the subkey $\tilde{k}_1$ of S_1. Results for the correct key guess are plotted in black, all other in gray.

Multivariate DPA. So far, integration of power or differential traces over small time intervals was proposed for the re-construction of a DPA signal in the presence of hardware countermeasures [10] and as a compression technique for measurement traces [18]. More generally, DPA can be extended to capture multiple well-separated leakage signals in two ways. The first approach applies standard DPA statistics and combines the (absolute) scores of multiple leakage signals afterwards. The second approach is to first combine, *e.g.* sum up, the measurement samples from multiple selected time instants before running univariate statistics. Both approaches were tested for CPA and yielded virtually identical results if the second approach takes positive and negative side-channel contributions at the different points in time into account, *e.g.* samples with positive correlation are added and samples with negative correlation are subtracted. As long as the number of instants is small, all possible combinations for combining the leakage at these instants can be tested exhaustively if the right combination is unknown.

Univariate vs. Multivariate Analysis. We study the performance of DCA for univariate and multivariate statistics. To allow a fair comparison we also provide results for univariate and multivariate CPA. Preliminary experiments indicated that the least significant bit (LSB) of $S_1(\tilde{x}_1 \oplus \tilde{k}_1)$ is a preferable target for attacks. We thus focus on attacks using two clusters, *i.e.* $w = 1$, targeting the LSB. Additionally, we identified three points in time (A,B,C) when the LSB leaks most.

For this analysis we used 5000 measurements in one hundred sets of fifty measurements. For each attack we used $N = 15, 20, 25, \ldots, 50$ measurements and repeated the attack one hundred times. We report the percentage of attack scenarios where an attack was successful, *i.e.* where the attack outputs the correct subkey value.

Table 2. Success rates in % for various univariate and multivariate attack scenarios

Test	Model	Point in time	N=15	N=20	N=25	N=30	N=35	N=40	N=45	N=50
CPA	LSB	overall	3	15	37	62	84	95	96	98
CPA	LSB	A	42	64	69	77	89	93	94	96
CPA	LSB	B	64	77	83	93	96	98	98	99
CPA	LSB	C	17	28	29	38	50	55	59	65
J_{SSE}	LSB	overall	3	15	37	62	84	95	96	98
J_{SSE}	LSB	A	42	64	70	77	89	93	94	96
J_{SSE}	LSB	B	64	78	82	93	96	98	98	99
J_{SSE}	LSB	C	18	28	31	38	50	56	59	65
CPA	LSB	AB	70	85	90	96	99	100	100	100
J_{SSE}	LSB	AB	70	83	91	97	99	100	100	100
CPA	LSB	ABC	76	90	96	99	100	100	100	100
J_{SSE}	LSB	ABC	78	94	96	99	100	100	100	100

Table 2 shows the success rates for the various scenarios. Comparing univariate CPA and DCA with the J_{SSE} criterion we observe that the success rates are very similar in all scenarios. The third block of results in the table shows success rates for DCA and CPA still targeting only the LSB but exploiting the multivariate leakage at points A,B, and C. For multivariate CPA, the measurement values were summed up, taking into account the respective signs of the correlation at each point in time, before computing correlation coefficients. Both multivariate attacks perform also very similar. It is clear that multivariate attacks are superior to univariate attacks, but they usually require knowledge of the instants A,B, and C.

4.2 AES in Hardware

Our experimental platform is a prototype chip which implements an 8051-compatible μC with AES-128 co-processor in 0.13 μm sCMOS technology without countermeasures. The architecture of the AES co-processor, which is the target of our attacks, is discussed in detail in Appendix A. The susceptibility of the chip towards templates [9,13], stochastic methods [20,13], DPA and rank correlation has already been analyzed in [2].

For our experiments we obtained a set of $N = 50\,000$ power traces $\boldsymbol{i}_n$. The samples represent the voltage drop over a 50Ω resistor inserted in the dedicated core VDD supply. Power consumption was sampled at 2 GS/s during the first round of AES-128 encryption with a constant key $k^\circ = (k_0^\circ, \ldots, k_{15}^\circ)$. The plaintexts $x_n = (x_{0n}, \ldots, x_{15n})$ were randomly chosen from a uniform distribution.

In all following examples we focus on the neighboring data cells C0,2 and C0,3 of Figure 4 that represent AES state bytes 8 and 12, respectively. However, we point out that all key bytes can be attacked in the same way since all corresponding state bytes enter the MixColumns circuit at some time.

DCA. We showed in Sect. 3.2 that clustering of the power consumption values caused by the differential $\Delta_{ii'}$ has three interesting properties. First, for the correct hypotheses (k_8, k_{12}) on both involved key bytes *all* power traces are assigned

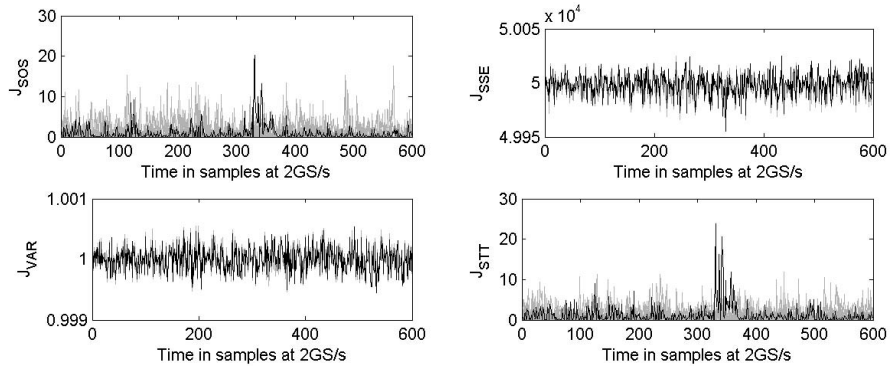

Fig. 2. Results of Cluster Analysis with J_{SOS} (top, left), J_{SSE} (top, right), J_{VAR} (bottom, left) and J_{STT} (bottom,right) when using 2^8 hypotheses on the pair (k_8, k_{12}). Results for the correct xor-difference are plotted in black, all other in gray.

to clusters $\Delta_{ii'}$ correctly which maximizes (minimizes) the cluster statistic J_k over all clusters. Second, for a wrong hypothesis (k_8, k_{12}) where the xor-difference $k_8 \oplus k_{12}$ is correct (*i.e.* equal to $k_8^\circ \oplus k_{12}^\circ$) power traces are assigned to cluster $\Delta_{ii'} = 0$ correctly and uniformly at random to all other clusters $\Delta_{ii'} \neq 0$. Third, for a wrong hypothesis (k_8, k_{12}) power traces are assigned to all clusters $\Delta_{ii'}$ uniformly at random. The second property enables our two step attack.

The results reported in the remainder are based on either $N_1 = 50\,000$ or $N_2 = 10\,000$ measurements. We restrict to univariate analysis and computed J_{SOS}, J_{SSE}, J_{VAR} and J_{STT}.

Step 1: Detecting (k_8, k_{12}) with the correct xor-difference. In order to detect a hypothesis with the correct xor-difference one has to decide whether the cluster $\Delta_{ii'} = 0$ is different from all other clusters. We thus merge the clusters $\Delta_{ii'} \neq 0$ to analyze them as one single cluster. The statistic J_k is then evaluated for the two cluster case, $\Delta_{ii'} = 0$ and the union of the remaining clusters.

We sort N_1 measurements into two clusters $\Delta_{ii'} = 0$ and $\Delta_{ii'} \neq 0$ and evaluate the cluster criterion functions for the two cluster case. Interestingly, for this approach it does not matter whether one applies the Hamming distance model or not, since in both cases the same measurements are assigned to $\Delta_{ii'} = 0$ and $\Delta_{ii'} \neq 0$. Figure 2 shows the resulting cluster criteria. One can observe that the criteria J_{SOS} and J_{SST} yield signals at the points in time when the collision occurs.

Step 2: Detecting the correct key pair. We search all pairs (k_8, k_{12}) with the now known xor-difference $k_8^\circ \oplus k_{12}^\circ$. Detecting the correct pair is easier than detecting a pair with the correct xor-difference, because differences of many clusters yield stronger side-channel signals. We can therefore work with fewer measurements which speeds up the analysis (note that the measurements from step 1 are re-used). For each hypothesis we sort N_2 measurements into 256

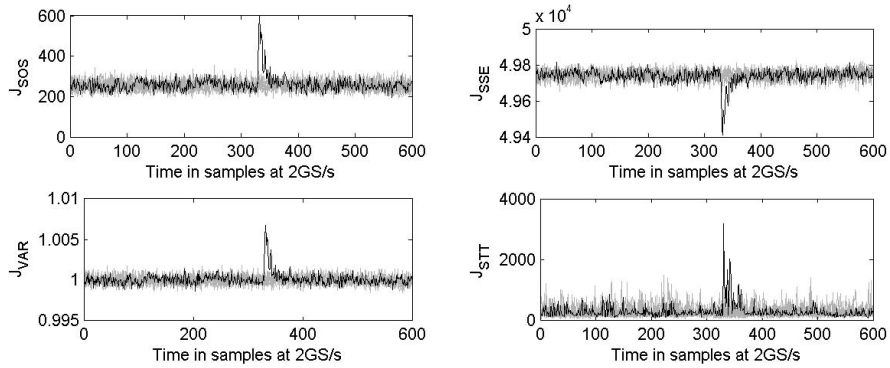

Fig. 3. Results of Cluster Analysis with J_{SOS} (top, left), J_{SSE} (top, right), J_{VAR} (bottom, left) and J_{STT} (bottom,right) when using 2^8 hypotheses on the pair (k_8, k_{12}). Results for the correct key guess are plotted in black, all other in gray.

clusters and evaluate the cluster criteria. Figure 3 shows the cluster criteria for this setting (without any power model).

We observe that all criterion functions are able to identify the correct guess. We also evaluated the cluster criteria under the assumption of the Hamming distance model, where we sorted the measurements into nine clusters according to the Hamming weight of $\Delta_{ii'}$. The results are similar and we do not present them in detail. The results demonstrate the feasibility of our approach on both platforms and show that it works with or without a power model. Among the cluster criteria that we considered J_{STT} gives in general the best results but is particularly error-prone when very few measurements are used.

Complexity. For the two-step attack, approximately 50 000 measurements were needed to reveal the correct two key bytes and $2 \cdot 2^8$ key hypotheses were tested. The general DCA approach tests 2^{16} key hypotheses and the threshold for successful key recovery is approximately at 5000 measurements. As result, our special attack strategy for the AES hardware implementation allows to reduce the number of hypotheses for which cluster criterion functions have to be computed by a factor of 2^7 at the cost of a tenfold increase of measurements.

5 Conclusion

We propose a new technique for side-channel key recovery based on cluster analysis and detection of internal collisions. The technique is broader in applications than previously known DPA attacks. It has obvious advantages when more than two clusters are used. In particular DCA does not require but can benefit from a good leakage model and it is inherently multivariate. DCA inspires a simple extension of standard DPA to multivariate analysis that is also included in this contribution. While previous works focus on internal collisions that are mainly

results of algorithmic properties, we additionally consider implementation specific collisions. Our approach is confirmed in practice on two platforms: an AVR microcontroller with implemented DES algorithm and an AES hardware module. This is the first work demonstrating the feasibility of internal collision attacks on highly parallel hardware platforms. Furthermore we present a new attack strategy for the targeted AES hardware module.

Acknowledgements

We sincerely thank the anonymous reviewers for their valuable and insightful comments that helped us to improve the article. We also thank Sören Rinne for providing the DES source code for the AVR microcontroller.

References

1. Efficient Implementation of the Rijndael SBox, `http://www.comms.scitech.susx.ac.uk/fft/crypto/rijndael-sbox.pdf`
2. Batina, L., Gierlichs, B., Lemke-Rust, K.: Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 341–354. Springer, Heidelberg (2008)
3. Biryukov, A., Bogdanov, A., Khovratovich, D., Kasper, T.: Collision Attacks on AES-Based MAC: Alpha-MAC. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 166–180. Springer, Heidelberg (2007)
4. Biryukov, A., Khovratovich, D.: Two new techniques of side-channel cryptanalysis. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 195–208. Springer, Heidelberg (2007)
5. Bogdanov, A.: Improved Side-Channel Collision Attacks on AES. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 84–95. Springer, Heidelberg (2007)
6. Bogdanov, A.: Multiple-Differential Side-Channel Collision Attacks on AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 30–44. Springer, Heidelberg (2008)
7. Bogdanov, A., Kizhvatov, I., Pyshkin, A.: Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 251–265. Springer, Heidelberg (2008)
8. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
9. 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)

10. Clavier, C., Coron, J.-S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
11. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & Sons, Chichester (2001)
12. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
13. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic Methods.. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 15–29. Springer, Heidelberg (2006)
14. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
15. Ledig, H., Muller, F., Valette, F.: Enhancing Collision Attacks. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 176–190. Springer, Heidelberg (2004)
16. Lemke-Rust, K., Paar, C.: Gaussian Mixture Models for Higher-Order Side Channel Analysis.. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 14–27. Springer, Heidelberg (2007)
17. Mangard, S., Aigner, M., Dominikus, S.: A Highly Regular and Scalable AES Hardware Architecture. IEEE Trans. Computers 52(4), 483–491 (2003)
18. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)
19. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Investigations of Power Analysis Attacks on Smartcards. In: Proceedings of USENIX Workshop on Smartcard Technology, pp. 151–162 (1999)
20. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis.. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
21. Schramm, K., Leander, G., Felke, P., Paar, C.: A Collision-Attack on AES Combining Side Channel- and Differential-Attack. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)
22. Schramm, K., Wollinger, T., Paar, C.: A New Class of Collision Attacks and Its Application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
23. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition vs. comparison side-channel distinguishers. In: ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2008)
24. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Reading (2006)
25. Walter, C.: Sliding Windows Succumbs to Big Mac Attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 286–299. Springer, Heidelberg (2001)
26. Wolkerstorfer, J., Oswald, E., Lamberger, M.: An ASIC Implementation of the AES SBoxes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 67–78. Springer, Heidelberg (2002)

A The AES Hardware Architecture

A similar implementation is described in [17]. The module includes the following parts: data unit, key unit, and interface. The most important part is the data unit

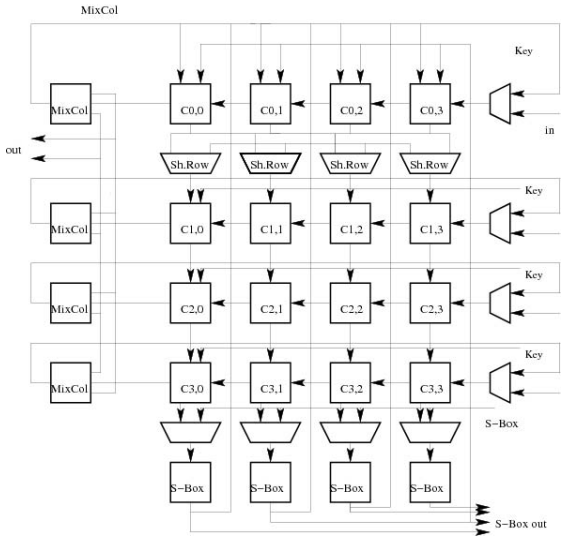

Fig. 4. The architecture of the AES module

(see Figure 4), which includes the AES operation. It is composed of 16 data cells ($C_{i,j}$, where $i, j \in \{0, 1, 2, 3\}$) and four S-Boxes. A data cell consists of flip-flops (able to store 1 byte of data) and some combinational logic (xors gates) in order to perform AddRoundKey operations. It has the ability to shift data vertically and horizontally, which is the feature exploited in our attacks. Load data is done by shifting the input data column by column into the registers of the data cells. The initial AddRoundKey transformation is performed in the fourth clock cycle together with the load of the last column. To calculate one round, the bytes are rotated vertically to perform the S-box and the ShiftRows transformation row by row. In the first clock cycle, the S-Box transformation starts only for the fourth row. Because of pipelining the result is stored after two clock cycles in the first row. S-boxes and the ShiftRows transformations can be applied to all 16 bytes of the state within five clock cycles due to pipelining. The S-Boxes in the AES module are implemented by using composite field arithmetic as proposed by Wolkerstorfer *et al.* [26] following the original idea of Rijmen [1].