Beyond the Limits of DPA: Combined Side-Channel Collision Attacks

Andrey Bogdanov[1] and Ilya Kizhvatov[2]

[1] Katholieke Universiteit Leuven, ESAT/COSIC and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
andrey.bogdanov@esat.kuleuven.be
[2] Université du Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
ilya.kizhvatov@uni.lu

Abstract. The fundamental problem of extracting the highest possible amount of key-related information using the lowest possible number of measurements is central to side-channel attacks against embedded implementations of cryptographic algorithms. To address it, this work proposes a novel framework enhancing side-channel collision attacks with divide-and-conquer attacks such as differential power analysis (DPA). An information-theoretical metric is introduced for the evaluation of collision detection efficiency. Improved methods of dimension reduction for side-channel traces are developed based on a statistical model of Euclidean distance.

The theoretical and experimental results of this work confirm that DPA-combined collision attacks are superior to both DPA-only and collision-only attacks. The new methods of dimension reduction lead to further complexity improvements. All attacks are treated for the case of AES-128 and are practically validated on a wide-spread 8-bit RISC microcontroller whose architecture is similar to that of many smart cards.

Keywords: side-channel attacks, collision attacks, DPA, AES

1 Introduction

1.1 Motivation

Keyed cryptographic algorithms employ secret information to protect user data and can provide its confidentiality, integrity, authenticity, non-repudiation — services crucial for almost any security-related application. Numerous analysis methods have been proposed for cryptographic algorithms. While the traditional mathematical attacks are solely based on the inputs and outputs of an algorithm, side-channel attacks rely upon the fact that any real-world implementation of the algorithm is not ideal and leaks some physically observable parameters that are dependent on the key processed. Such parameters can include time [17], power consumption [18], electro-magnetic radiation [24] and algorithm behaviour under actively induced execution faults [4]. Since the attacker often has immediate physical access to embedded systems, they are most vulnerable to side-channel attacks. The fundamental problem of side-channel analysis is as follows:

Problem 1 (Fundamental for side-channel analysis). Extract the highest possible amount of key information given the lowest possible amount of side-channel information for a fixed implementation of a cryptographic algorithm.

Side-channel collision attacks provide a natural basis for solving this problem, possessing the unique combination of three important properties which are not simultaneously present in any other side-channel analysis technique known today: First, they are essentially based on the *algorithmic properties* of the attacked cryptographic algorithm, which allows the adversary to use more side-channel information from one algorithm execution. Second, they are *not based on any particular leakage model* which opens up the possibility of using all relevant side-channel information, not limited to a specific model. Third, they do *not require any significant apriori knowledge* of the implementation (a major limitation in many side-channel attacks), however, being able to profit from profiling. Side-channel collision attacks have also further attractive features such as that essential parts of the cryptographic algorithm can remain unknown to the attacker which makes many algorithmic masking techniques transparent to collision attacks.

In this work, we come up with two novel techniques significantly enhancing side-channel collision-based analysis and propose a general framework naturally incorporating them.

1.2 Collision attacks in the context

In this subsection, we aim to draw attention to some of the beneficial features of collision attacks mentioned above that they exhibit in the context of other approaches to side-channel analysis.

Regarding the method of *extracting key-related information*, there are two large classes of side-channel attacks: *leakage-model* oriented and *pattern-matching* oriented. With respect to the *key-recovery procedure*, side-channel attacks fall into two categories: *divide-and-conquer* attacks (which provide distinguishers for small key chunks) and *analytic* attacks (which recover the entire key e.g. by solving systems of equations). Correspondingly, when classifying according to information extraction method and key-recovery procedure, one can speak about the four types of side-channel attacks represented in Table 1. Note that, optionally, side-channel attacks can use profiling, which is not considered in this comparison.

Differential power analysis (DPA) [18] and correlation power analysis (CPA) [10], a generalization of DPA, are probably the most wide-spread practical attacks on numerous embedded systems such as smart-card microcontrollers and dedicated ASICs. They are based on guessing a chunk of the key, classifying traces according to this hypothesis and performing a statistical test in a leakage model such as Hamming weight or Hamming distance. Similarly to DPA, mutual information analysis (MIA) [13], [23], [3] is based on subkey guessing and classifying traces. However, the test performed for each key hypothesis uses an information-theoretic metric which does not necessarily imply a leakage model.

Template attacks [11], [1] belong to another class of powerful side-channel attacks and are optimal in an information theoretic sense. They do not rely on any particular leakage model but require a profiling stage and, as DPA, are mainly limited to key chunks. Stochastic methods [26], [14] can be seen as a version of template attacks allowing one to simplify template building, further increase the resolution and, thus, decrease the total number of measurements needed.

Algebraic side-channel attacks [25] use Hamming weights of intermediate variables detected by observing side-channel traces to simplify the systems of nonlinear equations on the full key. Thus, algebraic side-channel attacks imply that the implementation leaks Hamming-weight related side-channel information.

Side-channel collision attacks [7], [8], [9], [28], [27] use pattern-matching techniques (like template attacks) being however essentially based on the cryptanalytic properties of attacked cryptographic algorithms (by attacking key as a whole as in algebraic side-channel attacks) and not relying on any complex profiling stages (similarly to DPA).

Table 1. Side-channel attacks: Methods of extracting key-related information and key-recovery procedure

	Leakage model	Pattern-matching
Divide-and-conquer	DPA[18] CPA [10] MIA [13]	template [11] stochastic [26] MIA [13]
Analytic	algebraic [25]	collision [7], [27]

Side-channel attacks with analytic key-recovery tend to be more efficient in terms of measurement complexity. Side-channel attacks using pattern-matching information extraction are independent of a concrete leakage model (such as Hamming weight or Hamming distance), thus, being a way more universal. Collision attacks share both these benefits.

Recently, some side-channel techniques using more than one method of extracting key-related information have been proposed. For instance, differential cluster analysis [2] and mutual information analysis [13] are divide-and-conquer attacks that generally use pattern-matching but can benefit from the knowledge of leakage model. However, these attacks do not use the advantages of analytic key recovery.

In this work, we show that the analytic key-recovery procedures of collision attacks allow for extensions and propose a general framework for incorporating key-related information resulting from divide-and-conquer attacks such as DPA and template attacks into collision techniques.

1.3 Our contributions

In this work, we introduce the *combined collision attack* which is a novel technique for combining side-channel collision attacks with divide-and-conquer attacks such as DPA and template attacks, thus, using *both* divide-and-conquer and analytic key recovery as well as *both* leakage models and pattern-matching extraction (Sections 3 and 4). This combination of very different side-channel techniques allows us to use more key-relevant information contained in the side-channel traces, omitted by each of these techniques when applied separately. We theoretically compute the success probability and expected computational complexity of combined collision attacks.

Starting from the basic Euclidean distance, we propose new techniques of efficient dimension reduction and collision detection. We study some of their statistical properties in a formal way. We propose the usage of λ-*divergence* as a metric for the comparison of different collision detection methods and prove that it is equivalent to mutual information in this context (Section 5).

We practically demonstrate that DPA-combined collision attacks are more efficient than both conventional collision attacks and DPA (Section 6). On the theoretical side, this fact naturally implies that neither the usual pattern-matching methods of collision detection nor the correlation techniques of DPA use all information available in the traces. On the practical side, the new findings allow us to further reduce the measurement complexity of side-channel collision attacks. We conclude with a discussion and open problems in Section 7.

2 Basics of Collision Attacks

2.1 Internal collisions

An *internal collision* in a cryptographic algorithm $\mathcal{A}$ occurs with respect to a *target function* ϕ, if ϕ delivers the same output y given some two inputs x_1 and x_2: $y = \phi(x_1) = \phi(x_2)$ that are not necessarily equal.

Generally speaking, if $\mathcal{A}$ is an iterative cipher and ϕ is applied in its first iterations, it can be very difficult for the attacker to say if $\phi(x_1) = \phi(x_2)$ using black-box queries (plaintext-ciphertext pairs) only. However, side-channel leakage can help him to detect internal collisions.

Assume that some function ψ processes the output of ϕ. If ϕ returns an equal value y for two inputs, $\mathcal{A}$ performs two identical calculations $\psi(y)$ in these two cases. If ϕ returns two unequal values y_1 and y_2, the corresponding calculations $\psi(y_1)$ and $\psi(y_2)$ are distinct. The attacker can observe this behaviour in the power consumption or electromagnetic radiation of the device implementing $\mathcal{A}$ during the application of ψ — similar side-channel traces for equal outputs of ϕ and diverse side-channel traces for unequal outputs of ϕ. Because of this, ψ is called a *collision detection function*.

Once an internal collision is detected, it can be interpreted as a key-dependent equation $\phi(x_1) = \phi(x_2)$ delivering some information about the key, if ϕ and/or x_1 as well as x_2 are key-dependent.

However, if there are several applications of ϕ within the algorithm $\mathcal{A}$ and ϕ is invertible, one does not need a separate collision detection function ψ and can employ ϕ as both a target function and a collision detection function. Moreover, a collision now leads to a much simpler algebraic equation $x_1 = x_2$. The latter observation has yielded the idea behind *generalized collisions* proposed in [7] and provides a major advantage over other collision-based attacks such as [27].

2.2 Previous work

The principle of using internal collisions in cryptanalysis is due to Hans Dobbertin and was also discussed in the early work [30]. In [28], a collision attack on DES was proposed, several adjacent DES S-boxes representing the target function ϕ. This attack was enhanced in [20] using the notion of *almost collisions* which are internal states of an algorithm that are very similar. In [27], the separate bytes of each of the four 4-byte linear MixColumn mappings in the first AES round are treated as target functions ϕ, S-boxes of the second round representing the collision detection function ψ. In [5], it is shown that similar side-channel collision attacks can be applied to AES-based MACs such as Alpha-MAC to mount selective forgery attacks that do not require any knowledge of the secret key. The results in [6] suggest that collision attacks can help overpass the random masking of some AES implementations. Overpassing random masking with collision attacks for the case of DES was done in [15] and improved in [16], but these works consider collisions in Hamming weights and therefore imply the leakage model.

As mentioned above, side-channel collision attacks on AES were improved in [7] by introducing the notion of generalized collisions that occur if two S-boxes at some arbitrary positions of some arbitrary rounds process an equal byte value within several runs. Here both the target and collision detection functions ϕ and ψ coincide being the 8-bit AES S-box. The S-box remains the same for all executions, rounds and byte positions within the round (as opposed to DES). This increases the number of function instances to compare, *i.e.* the number of potential collisions to be used afterwards for key recovery.

While [7] treats the *linear collisions* (resulting in linear equations on the key) of AES which are generalized collisions that occur in the first AES round only, the work [9] also considers *nonlinear collisions* (respectively, resulting in nonlinear equations). A set of such collisions can be considered as a system of equations over a finite field. Ways to deal with unreliable collision detection are discussed in [8], including the techniques of binary and ternary voting.

2.3 Linear collision-based key recovery for AES

To simplify representation, we chose to study collision attacks at the notable example of the U.S. encryption standard AES [12]. More precisely, we use the key-recovery [7] for AES-128 based on linear collisions for this purpose. Note that all techniques of this work can be successfully applied to other ciphers as well as to other collision-based key-recovery techniques.

We use the following notation to represent the variables of AES. $K = \{k_j\}_{j=1}^{16}$, $k_j \in \mathbb{F}_{2^8}$ is the 16-byte user-supplied key (the initial AES subkey). AES plaintexts are denoted by $P^i = \{p_j^i\}_{j=1}^{16}$, $p_j^i \in \mathbb{F}_{2^8}$, where $i = 1, 2, \dots$ is the number of an AES execution.

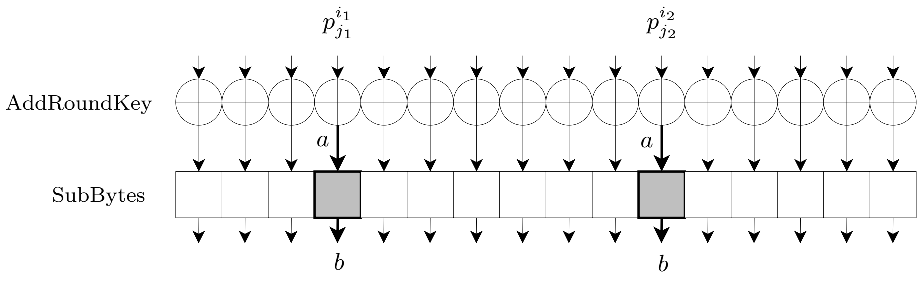


Fig. 1. A linear collision for a pair of AES executions

Given a collision within the first round of AES (linear collision)

$$S(p_{j_1}^{i_1} \oplus k_{j_1}) = S(p_{j_2}^{i_2} \oplus k_{j_2}), \tag{1}$$

one obtains a linear equation with respect to the key over $\mathbb{F}_{2^8}$ of the form

$$k_{j_1} \oplus k_{j_2} = p_{j_1}^{i_1} \oplus p_{j_2}^{i_2} = \Delta_{j_1,j_2} \text{ for } j_1 \neq j_2. \tag{2}$$

If D collisions have been detected, they can be interpreted as a system of linear equations over $\mathbb{F}_{2^8}$:

$$\begin{cases} k_{j_1} \oplus k_{j_2} & = & \Delta_{j_1,j_2} \\ & \dots & \\ k_{j_{2D-1}} \oplus k_{j_{2D}} & = & \Delta_{j_{2D-1},j_{2D}} \end{cases} \tag{3}$$

This system cannot have the full rank due to the binomial form of its equations. Moreover, for small numbers of inputs to AES the system is not connected and it can be divided into a set of h_0 smaller independent (with disjunct variables) connected subsystems with respect to the parts of the key. Each subsystem has one free variable. Let h_1 be the number of all missing variables, and $h = h_0 + h_1$. We call each of these h subsystems or missing variables a *chain*.

Without loss of generality, a chain ζ of length n can be represented as the following subsystem of the equation system (3):

$$\begin{cases} k_{j_1} \oplus k_{j_2} & = \Delta_{j_1,j_2} \\ k_{j_2} \oplus k_{j_3} & = \Delta_{j_2,j_3} \\ & \dots \\ k_{j_{n-2}} \oplus k_{j_{n-1}} & = \Delta_{j_{n-2},j_{n-1}} \\ k_{j_{n-1}} \oplus k_{j_n} & = \Delta_{j_{n-1},j_n}, \end{cases} \tag{4}$$

or alternatively as an n-tuple of byte indices $\zeta = (j_1, \dots, j_n)$ in a short form. Each chain (4) has 2^8 possible solutions, since it is sufficient to guess one key byte in the chain to unambiguously determine all other $n-1$ bytes of the chain. If the system (3) has h chains, then it has 2^{8h} solutions.

That is, 2^{8h} guesses have to be performed, which is the offline computational complexity of this basic key-recovery method. Each key hypothesis is then tested using a known plaintext-ciphertext pair with the full AES to rule out wrong candidates. Note that 2^{8h} quickly becomes feasible as the number of distinct inputs P^i grows. The work [7] demonstrates the probability that $2^{8h} \leq 2^{40}$ $(h \leq 5)$ to be about 0.85 for just 6 inputs, if all collisions can be detected.

2.4 Collision detection with Euclidean distance

For a collision attack to be successful, one has to decide if two S-boxes accept equal inputs using side-channel information obtained from the side-channel leakage (of the implementation) of the attacked cryptographic algorithm. Given two side-channel traces

$$\tau_1 = (\tau_{1,1}, \dots, \tau_{1,l}) \in \mathbb{R}^l \text{ and } \tau_2 = (\tau_{2,1}, \dots, \tau_{2,l}) \in \mathbb{R}^l,$$

respectively corresponding to a pair of S-box executions with some unknown inputs a_1 and a_2, it has to be decided whether $a_1 = a_2$ for collision detection.

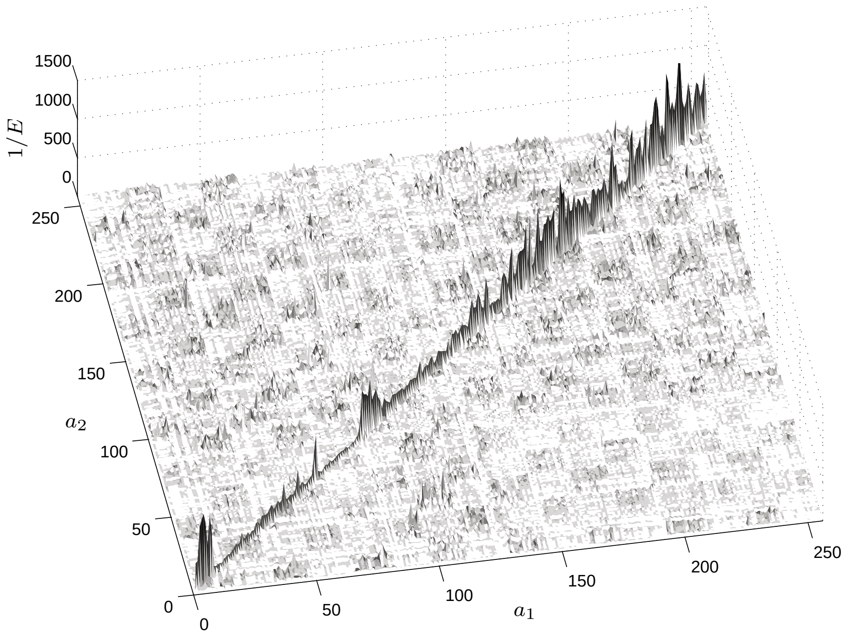


Fig. 2. Inverse Euclidean distance $1/E$ between power consumption traces for all input pairs of the AES S-box as implemented on 8-bit RISC μC ATMega16

The two traces can be considered as two vectors in the Euclidean space of dimension l. The Euclidean distance E between them is defined as

$$E(\tau_1, \tau_2) = \sum_{r=1}^{l} (\tau_{1,r} - \tau_{2,r})^2.$$

One expects that E will be higher for non-collisions and lower for collisions. Our experiments with a popular microcontroller (μC) (see Figure 2) show that this intuition is indeed justified, at least when noise is somewhat reduced by averaging traces.

Most papers on collision attacks [27], [7], [8], [9] use the Euclidean distance between two noisy traces as the basic metric for collision detection. In a recent work [22], Pearson's correlation is used to detect many collisions at once from multiple traces. We have found that collision detection can be significantly improved by dimension reduction for side-channel traces based on the properties of the Euclidean distance, which is, therefore, the metric of our choice. We detail this in Section 5.

3 Framework for Collision Attacks

In this section, we propose a general framework for the side-channel analysis of cryptographic algorithms based upon internal collisions. This framework allows the adversary to amplify collision attacks by

– Any collision detection technique and
– Any divide-and-conquer side-channel attack.

Later, we will study the core test of the framework presented in this section (Section 4) and evaluate it, combined with our novel collision detection techniques (Section 5), also in a practical setting (Section 6).

3.1 Attack flow

All collision techniques of this work are studied at the example of the U.S. encryption standard AES, which is a highly relevant target, and experimentally verified on a wide-spread 8-bit platform similar to many smart-card microcontrollers. Note that most techniques of collision attacks are also successfully applicable to other ciphers and implementations. For block ciphers with smaller S-boxes such as SERPENT or PRESENT collision attacks become even more efficient than in the case of AES as there are more collisions occurring. Moreover, collision attacks can be almost directly applied to stream ciphers using S-boxes in the output function. Thus, for the sake of simplicity, we will deal with AES-128 in this paper, though having in mind that the collision techniques are actually much more generally applicable.

Let the AES-128 implementation have a 16-byte key fixed for the entire attack and leak a key-dependent side-channel parameter (e.g. power consumption or electromagnetic radiation).

A collision attack consists of an online stage, signal processing stage, and key-recovery stage. Its procedure is outlined in Algorithm 1 and is explained here:

– In the *online stage* (steps 1-3 of Algorithm 1), N chosen 16-byte plaintexts P^i are sent to the attacked device implementing AES (**ChooseInputs**). The side-channel traces

Algorithm 1 Collision attack based on linear collisions combined with a divide-and-conquer test for AES-128

```
 1: $\mathcal{P} = (P^1, \ldots, P^N) \leftarrow$ ChooseInputs()
 2: $\mathcal{T} = (T^1, \ldots, T^N) \leftarrow$ AcquireTraces($\mathcal{P}$)
 3: [each trace $T^i$ contains 16 subtraces, one for each S-box]
 4: $\mathcal{C} \leftarrow$ DetectCollisions($\mathcal{P}$, $\mathcal{T}$)
 5: [each collision in $\mathcal{C}$ is a four-tuple $(p_{j_1}^{i_1}, p_{j_2}^{i_2}, j_1, j_2)$, see (1)]
 6: for each $k_j$ of 16 key bytes do
 7:     $\mathcal{K}_j = (\kappa_j^1, \ldots, \kappa_j^{256}) \leftarrow$ SortKeyByte($j, \mathcal{P}, \mathcal{T}$)
 8: end for
 9: [now $\mathcal{K} = (\mathcal{K}_1, \ldots, \mathcal{K}_{16})$ contains 16 sorted lists of key byte candidates of length 256 each]
10: $K' \leftarrow$ RecoverKey($\mathcal{C}$, $\mathcal{K}$)
11: return $K'$ as a key candidate
```

T^i (e.g. power consumption or electromagnetic radiation) are acquired by the measurement equipment (**AcquireTraces**) for these plaintexts. Each trace T^i contains 16 subtraces, one for each S-box:

$$T^i = \{\tau_j^i\}_{j=1}^{16}.$$

That is, T_i is a set of 16 individual traces τ_j^i for each of the 16 S-box instances in the first AES round. The trace τ_j^i is a real-valued vector of length l, $\tau_j^i \in \mathbb{R}^l$, thus, containing l measurement points.
In our attacks, we will send γ randomly drawn 16-byte plaintexts to the AES encryption, each repeated t times, which yields $N = \gamma \cdot t$.

– In the *signal processing stage* (steps 4-9 of Algorithm 1), collisions are detected in the target traces T^i (**DetectCollisions**) and the divide-and-conquer attack is applied to sort the key-byte candidates in each of the 16 byte positions (**SortKeyByte**). Before applying the signal processing, the traces corresponding to each of γ unique plaintexts are averaged t times to decrease noise. The output of the signal processing stage is the set of detected collisions $\mathcal{C}$ containing 4-tuples $(p_{j_1}^{i_1}, p_{j_2}^{i_2}, j_1, j_2)$ and 16 sorted lists $\mathcal{K}$ of 256 key byte candidates for each of the 16 byte positions. Depending on the measurement setup and implementation, one might choose to perform decimation and denoising in this stage.
– In the *key-recovery stage* (step 10 of Algorithm 1), an AES key candidate K' is computed using the list of detected collisions $\mathcal{C}$ and sorted candidates $\mathcal{K}$ (**RecoverKey** detailed in Algorithm 2). Note that **RecoverKey** can return either the right 16-byte key, a wrong 16-byte key or an empty set of keys $\emptyset$, if no key candidate has passed the final key testing. By π we denote the success probability of Algorithm 1 which is the probability that **RecoverKey** returns the right key.

3.2 Combined key recovery

This subsection outlines the new technique of combining the analytic key recovery of linear collision attacks with the divide-and-conquer key-recovery of such attacks as DPA.

The procedure of the combined key recovery is provided in Algorithm 2 and mainly relies on the test of each chain **TestChain** introduced and analyzed in Section 4. This is the major advantage of our approach compared to the conventional collision attacks where it is not possible to test the correctness of each chain separately and steps 5-10 of Algorithm 2 are missing. As opposed to that, the availability of divide-and-conquer

Algorithm 2 Key-recovery **RecoverKey** based on linear collisions and sorted key-byte candidates from a divide-and-conquer test for AES-128

```
Require: Collisions $\mathcal{C}$ and sorted key-byte candidates $\mathcal{K}$
 1: build $h$ chains $\zeta_1, \ldots, \zeta_h$ from collisions $\mathcal{C}$
 2: [a chain $\zeta_i$ of length $n_i$ is an n-tuple $(j_1, \ldots, j_{n_i})$, see (4)]
 3: for each $\zeta_i$ of $h$ chains do
 4:     $\mathcal{G}_i \leftarrow \{0, \ldots, 2^8 - 1\}$, i.e. all $2^8$ chain guesses
 5:     for each chain guess $g \in \{0, \ldots, 2^8 - 1\}$ do
 6:         if not TestChain($\zeta_i, g, \mathcal{K}, \mathcal{C}$) then
 7:             remove chain guess $g$, $\mathcal{G}_i \leftarrow \mathcal{G}_i \backslash \{g\}$
 8:         end if
 9:     end for
10:     [now $\mathcal{G}_i$ only contains survived guesses for chain $\zeta_i$]
11: end for
12: unite chain guesses to full key guesses, $\mathcal{G} \leftarrow \cup_{i=1}^{h} \mathcal{G}_i$
13: [$\mathcal{G}$ contains full key guesses survived chain filtration]
14: $K' \leftarrow$ TestKeysWithAES($\mathcal{G}$)
15: return $K'$ as a key candidate
```

information in the combined key recovery allows to test for each chain separately which can provide a significant efficiency gain. This can be reflected in the increased success probability π of the attack given some measurement complexity or in the reduced measurement complexity given some success probability π, thus, delivering a better solution to Problem 1.

3.3 Attack complexity and efficiency metric

According to the three stages of a collision attack outlined above, its complexity is defined by three parameters (see Algorithm 1):

– C_{online} is the number of inputs to AES for which measurements have to be performed in the online phase (**AcquireTraces**).
– $C_{\text{processing}}$ is the computational complexity of signal processing on side-channel traces needed to detect collisions (**DetectCollisions**) and sort key-byte candidates within the divide-and-conquer attack (**SortKeyByte**).
– C_{recovery} is the computational complexity of **RecoverKey** (Algorithm 2), that is, the number of operations needed to solve the resulting systems of linear or nonlinear equations and to identify the most probable solution.

For collision attacks in this work, we bound C_{recovery} by 2^{40} computations of AES which can be performed within several hours on a PC. Given this restriction on C_{recovery}, the online complexity $C_{\text{online}} = N = t \cdot \gamma$ becomes the major limiting factor of collision attacks, since $C_{\text{processing}}$, mainly determined by γ, will be negligible for our choices of γ.

The success of an attack is often of probabilistic nature and the success probability π of the attack has to be considered along with $C_{\mathbf{online}}$ to derive the average-case measurement complexity:

$$C = C_{\text{online}}/\pi. \tag{5}$$

This metric characterizes the expected number of measurements needed to recover the full 16-byte key of AES. In this work, our main goal is to improve C by lowering C_{online} and increasing π, given the above admissible upper bound on C_{recovery}.

We note that metric C is also applicable to pure divide-and-conquer-style attacks. In this case, it can be rewritten as $C = C_{\text{online}}/\alpha^{16}$, where α is the probability to determine a single AES key byte with C_{online} traces. Also, in a DPA-style attack, $t = 1$ is often the case, so the metric boils down to just γ/α^{16}. Again, this is the average measurement complexity which does not consider the complexity of the recovery. The latter may be non-negligible in a divide-and-conquer attack if m candidates, $m > 1$, are chosen from the sorted divide-and-conquer lists for individual key bytes (one would do this to increase α which will normally be a nondecreasing function of m). A way to capture this is suggested in work [29], which proposes a unified framework for the analysis of divide-and-conquer side-channel attacks. It presents a much more generic ways of comparing different side-channel attacks. In the following section we argue that they are only of a limited application to the attacks of the analytic class and to collision attacks in particular.

3.4 Collision attacks and the unified framework

In [29], three metrics were introduced for the comparison of side-channel attacks, namely, two actual security metrics (o-th order success rate and guessing entropy) and an information-theoretic metric based on conditional entropy. Our initial idea was to apply the metrics of the unified framework to perform a fair comparison between collision attacks on the one hand and DPA as well as template attacks on the other hand. However, while we were able to adapt the notion of the o-th order success rate and guessing entropy to collision attacks, we feel that those neither reflect our algorithmic intuition behind key recovery in this case nor practically capture the nature of the attack procedure.

This is mainly due to the fact that the unified framework is aimed at divide-and-conquer attacks such as DPA or template attacks. Collision attacks, together with algebraic side-channel attacks [25] belonging to the class of analytic attacks recover the entire key at once by solving a system of equations on the key rather than operating on small key chunks independently. This requires to define the target key class as the full key space ($\mathcal{S} = \mathcal{K}$ in terms of the unified framework), so computing the information theoretic metric *e.g.* for the 128-bit AES key becomes infeasible.

We consider this to be a limitation of the unified framework and argue that it is still an open research problem to come up with a (development of the unified) framework practically applicable also to analytic attacks.

However, as regards local side-channel leakage, we would like to stress that the unified framework is perfectly applicable. As applied to collision attacks, the quality of collision detection for a pair of traces is the most crucial local side-channel property. So we use an information-theoretic metric similar to that of the unified framework to compare methods of collision detection given two local side-channel traces.

4 Test of Chain

The test of chain **TestChain** used to filter out key candidates on chains in Algorithm 2 is the stage that determines the values of the crucial metrics C and C_{recovery}. Therefore, in this section we describe and analyze **TestChain** in detail.

4.1 Procedure

TestChain has as input:

- Chain ζ_i of length n_i consisting of key byte indices $j_1, \ldots, j_{n_i}$.

– Guess g of the chain. Without loss of generality, we assume that it is the first byte j_1 of the chain and $k_{j_1} = g$.
– The list of (linear) collisions $\mathcal{C}$ to be able to compute all the other $n_i - 1$ key bytes in the chain from k_{j_1}.
– The lists of sorted key-byte candidates $\mathcal{K}$ coming from a divide-and-conquer test (e.g. DPA). Only list $\mathcal{K}_j$ with $j \in \{j_1, \ldots, j_{n_i}\}$ are needed for the chain to be tested. Each $\mathcal{K}_j$ is a sorted list of 256 candidates for the key byte k_j.

The output of **TestChain** is *true*, if the chain ζ_i has passed the test, or *false* otherwise.

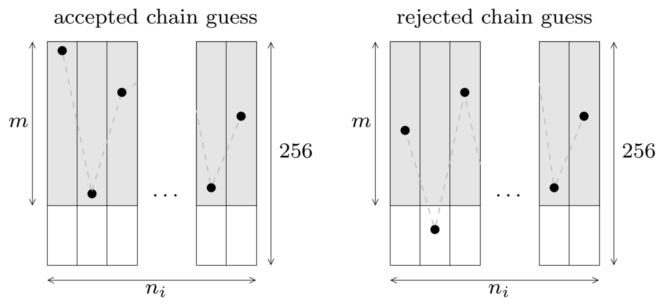


Fig. 3. Test for chain ζ_i: the guess of chain is rejected if at least one key byte falls outside the most probable m byte values as suggested by the divide-and-conquer test

The idea of the test of chain is to filter out those guesses of chains that are less probable to be compatible with the key information obtained from a divide-and-conquer test.

In each list $\mathcal{K}_j$, we will consider values among the top m positions. These are the most probable candidates for the key byte k_j as suggested by the divide-and-conquer test. We superpose the guess of the chain, computed from the byte guess g for k_{j_1}, and the n_i corresponding sorted lists $\mathcal{K}_j$ of length 256 bytes each, see Figure 3.

Now the test of chain can be described as follows:

– The guess of chain is accepted if all key bytes of the chain are among the top m candidates, each in the corresponding list $\mathcal{K}_j$.
– The guess of chain is rejected if at least one key byte of the chain falls outside the m top candidates in its corresponding list $\mathcal{K}_j$.

4.2 Success probability of combined attack

The threshold m has to be chosen in a way that the probability to filter out the correct key is small. This probability depends on the distribution of the position for the correct key byte of the divide-and-conquer test used.

Let α be the probability that the correct key byte is among the top m candidates in the sorted divide-and-conquer list. (In terms of the unified framework, α is exactly the m-order success rate for a single key byte recovery.) Under the assumption that all chain tests are independent, the probability for the full correct key to survive after passing the tests with all h chains can be computed as

$$\Pr\{\text{correct key survives after } h \text{ chains}\} = \prod_{i=1}^{h} \alpha^{n_i} = \alpha^{\sum_{i=1}^{h} n_i} = \alpha^{16},$$

since the sum of all chain lengths is $\sum_{i=1}^{h} n_i = 16$. Moreover, if all collisions have been detected correctly (i.e. collision detection yielded no false positives), this determines the success probability of the full combined collision attack

$$\pi = \alpha^{16}, \tag{6}$$

which is in fact equivalent to the success probability of the divide-and-conquer attack.

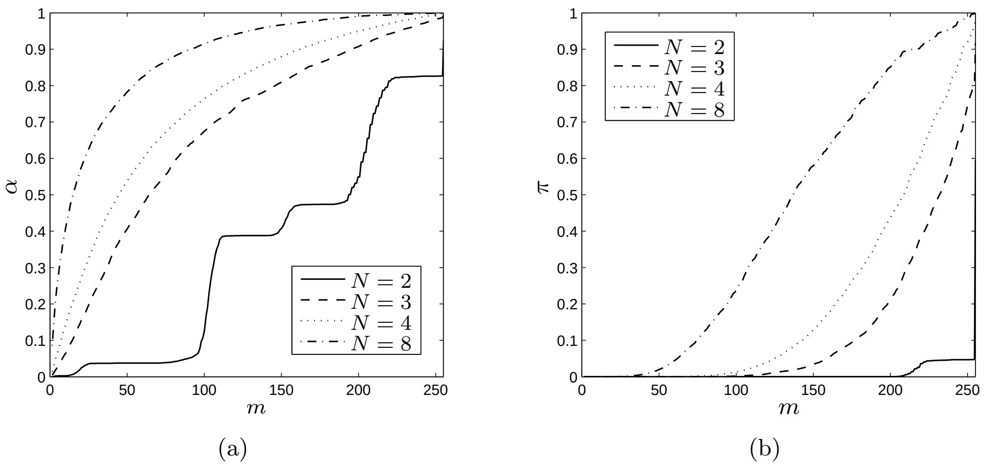


Fig. 4. Empirical dependency of α and π upon m for different numbers of traces N

As a practical example, our experiments with DPA attacks against an AES implementation on the 8-bit ATmega16 μC show that the chances for the correct key byte guess to be among the top m candidates in the sorted DPA list are quite good already for small values of m (which are preferred to have low C_{recovery}, as we will detail in the next subsection) and small values of N (which are obviously preferred to have low C_{online}). Figure 4(a) depicts the dependency of α upon m for different numbers of inputs N. The corresponding success probability for the full attack is depicted in Figure 4(b).

4.3 Complexity of key recovery for combined attack

Without the test of chain, the complexity of the collision attack is 2^{8h} AES computations, since each chain suggests 2^8 candidates for the respective subset of key bytes and there are h disjunct chains. In the combined approach, we effectively test m candidates for each chain. Moreover, we filter our improbable chain candidates separately for each chain. This results in a lower number of full 16-byte key candidates to be tested with AES at the end which determines C_{recovery}. Since the chain evaluation is much less complex than the testing of a full 16-byte candidate with the full AES, we can win in the total attack complexity significantly. Here we estimate C_{recovery} given m and α.

The expected number of wrong chain guesses to be tested in the test of chain can be computed as

$$(1-\alpha)m + \alpha(m-1) = m - \alpha\,.$$

The probability for a wrong chain guess to survive one element of chain can be derived as

$$\alpha\left(\frac{m-1}{255}\right) + (1-\alpha)\left(\frac{m}{255}\right) = \frac{m-\alpha}{255}\,,$$

since a wrong chain guess results in wrong key byte candidates suggested along the entire chain.

The expected number of correct chain guesses to be tested in the test of chain is α. The probability for a correct chain guess to survive one element of chain is α.

Then the expected number of chain candidates to survive the test of chain ζ_i can be estimated as

$$\eta_i = (m - \alpha) \left(\frac{m - \alpha}{255} \right)^{n_i - 1} + \alpha \cdot \alpha^{n_i - 1} .$$

Assuming the independency of all chain tests, we obtain an estimation of the key-recovery complexity for the combined attack:

$$C_{\text{recovery}} \approx \prod_{i=1}^{h} \max(1, \eta_i) \ , \qquad (7)$$

where the maximum is taken since one has to test at least one candidate for each chain in practice.

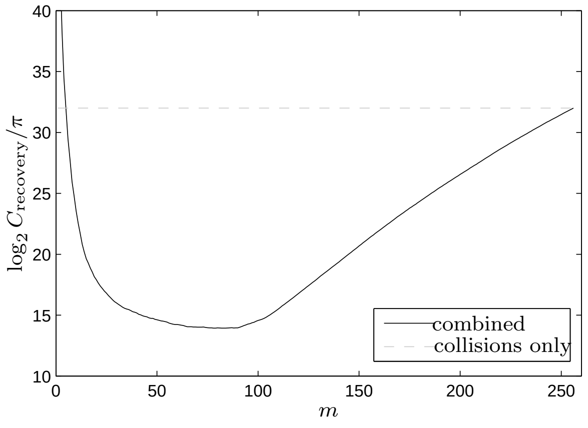


Fig. 5. Advantage of the DPA-combined key recovery over the linear collision-only key recovery with the respect to the expected complexity: Example for $h = 4$ with $n_1 = 7$, $n_2 = 6$, $n_3 = 2$, and $n_4 = 1$ (a typical case for γ=7)

Figure 5 presents a comparison between key recovery complexities for the DPA-combined attack and the linear key recovery with respect to C_{recovery}/π at the example with $h = 4$ chains of lengths $n_1 = 7$, $n_2 = 6$, $n_3 = 2$, and $n_4 = 1$ bytes, respectively, which is the expected distribution of chain lengths for $\gamma = 7$ [7]. As for the linear key recovery no DPA information is used, the complexity is 2^{32} with the success probability being exactly 1. At the same time, for the DPA-combined collision-based key recovery, the complexity C_{recovery} of (7) normalized by the success probability π of (6) will be much lower in most cases, attaining its minimum at $m = 78$ with just about 2^{14} AES computations.

Note that Figure 5 illustrates the very advantage of our combined attacks that we use to extract more information out of the side-channel traces in the key-recovery stage and, thus, to reduce the online measurement complexity C_{online}, addressing Problem 1.

5 Collision Detection

In this section, we propose improved techniques of collision detection by dimension reduction specially tailored for the Euclidean metric. First, we recall the general problem of dimension reduction in side-channel attacks. Then we develop a statistical model for the Euclidean distance, which will enable us to introduce our techniques and explain the intuition behind them. To perform the information-theoretically sound comparison of different dimension reduction techniques, we use the λ-divergence. We show that in our setting it is equivalent to mutual information. Experimental results for collision detection are provided. The practical evaluation of the dimension reduction techniques in a full (combined) collision attack will be given in Section 6.

5.1 Dimension reduction in side-channel attacks

Dimension reduction is the selection of samples from side-channel traces, usually with the purpose of improving the efficiency of an attack. In a typical setting, the clock frequencies of electronic devices are in the range of at least several MHz. Therefore, the side-channel trace acquired by the digital oscilloscope at an appropriate sampling rate of at least ten million samples per second will contain thousands, if not millions, of samples. In the presence of noise, when many traces are required, this makes the trace processing a determining part of the attack complexity and may sometimes even render the attack infeasible. On the other hand, it has long been known that only few samples in a trace would exhibit the leakage, the others being redundant. An example is the dimension reduction for a DPA attack (trace compression) [21], where we can limit ourselves to points at clock cycle maxima or to points exhibiting the highest variation across the traces corresponding to different input values. Note that while DPA is still feasible without dimension reduction, attacks employing multivariate methods (*e.g.* template-like and MIA attacks) are infeasible without an appropriate point selection.

Besides the reduced signal processing complexity, another effect of dimension reduction is the potential increase in the attack success rate. The points being removed from the side-channel traces would normally carry more noise than the informative signal, while the opposite applies to the selected points. Therefore, dimension reduction would lead to the overall increase in the signal-to-noise ratio (SNR). In most cases, this will lower the number of measurements required to reach a given success rate. Again, DPA attacks are a good example: in our experiments, selection of cycle maxima or points with the largest variation across the power traces reduces the trace count for the key-byte recovery.

We would like to stress that dimension reduction is *not* a full-scale profiling; it indeed requires some additional knowledge about the implementation, but this knowledge is much less than one would normally impose in traditional profiled attacks to build templates. As opposed to that, in its essence, dimension reduction can be seen as knowledge about time, not values being processed.

Collision attacks can benefit a lot from the dimension reduction. First, in the following, we show how to select the points from the traces to improve collision detection. We utilize Euclidean distance for trace comparison to distinguish between collisions and non-collisions. We also deal with this in the information-theoretic sense by introducing a metric for comparing our dimension reduction techniques. Second, collision detection demands for higher sampling rates (as compared to DPA or other attacks working in the Hamming weight/distance leakage model) [9] to capture subtle differences between the traces, so reduction in the number of samples in a trace is desirable to decrease $C_{\text{processing}}$, which is quadratic in the number of samples and traces.

We start with a statistical model of Euclidean distance that we first introduced in [9] and provide here for completeness to support our intuition behind the choice of dimension reduction techniques in the sequel.

5.2 A statistical model for Euclidean distance

Given two traces $\tau_1 = (\tau_{1,1}, \ldots, \tau_{1,l}) \in \mathbb{R}^l$ and $\tau_2 = (\tau_{2,1}, \ldots, \tau_{2,l}) \in \mathbb{R}^l$, we assume that each point $\tau_{i,j}$ can be statistically described as $\tau_{i,j} = s_{i,j} + r_{i,j}$, where $s_{i,j}$ is signal constant (without noise) for the given time point i as well as some fixed input to the S-box, and $r_{i,j}$ is Gaussian noise due to univariate normal distribution[3] with mean 0 and some variance σ^2 remaining the same for all time instances in our rather rough model. Let τ_1 and τ_2 correspond to some S-box inputs a_1 and a_2.

If $a_1 = a_2$, the corresponding deterministic signals are equal (that is, $s_{1,j} = s_{2,j}$ for all j's) and one has:

$$E(\tau_1, \tau_2)_{a_1=a_2} = \sum_{j=1}^{l} (\tau_{1,j} - \tau_{2,j})^2 = \sum_{j=1}^{l} \xi_j^2 = 2\sigma^2 \sum_{j=1}^{l} \eta_j^2,$$

where $\xi_j = r_{1,j} - r_{2,j}$, $\xi_j \sim \mathcal{N}\left(0, 2\sigma^2\right)$ and $\eta_j \sim \mathcal{N}(0,1)$. That is, statistic $E(\tau_1, \tau_2)_{a_1=a_2}$ follows the chi-square distribution with l degrees of freedom up to the coefficient $2\sigma^2$. As the chi-square distribution is approximated by normal distribution for high degrees of freedom, one has the following

Proposition 1. *Statistic*

$$E(\tau_1, \tau_2)_{a_1=a_2} = \sum_{j=1}^{l} (\tau_{1,j} - \tau_{2,j})^2$$

for $\tau_i = (\tau_{i,1}, \ldots, \tau_{i,l}) \in \mathbb{R}^l$ with $\tau_{i,j} \sim \mathcal{N}\left(s_{i,j}, \sigma^2\right)$ can be approximated by normal distribution $\mathcal{N}(2\sigma^2 l, 8\sigma^4 l)$ for sufficiently large l's.

Alternatively, if $a_1 \neq a_2$, one has

$$E(\tau_1, \tau_2)_{a_1\neq a_2} = \sum_{j=1}^{l} (\tau_{1,j} - \tau_{2,j})^2 = \sum_{j=1}^{l} \left(\delta_j^{(1,2)} + \xi_j\right)^2 = \; = 2\sigma^2 \sum_{j=1}^{l} \nu_j^2,$$

where

$$\delta_j^{(1,2)} = s_{1,j} - s_{2,j}, \quad \xi_j = r_{1,j} - r_{2,j}, \quad \xi_j \sim \mathcal{N}\left(0, 2\sigma^2\right) \text{ and } \nu_j \sim \mathcal{N}\left(\delta_j^{(1,2)}/\sqrt{2}\sigma, 1\right).$$

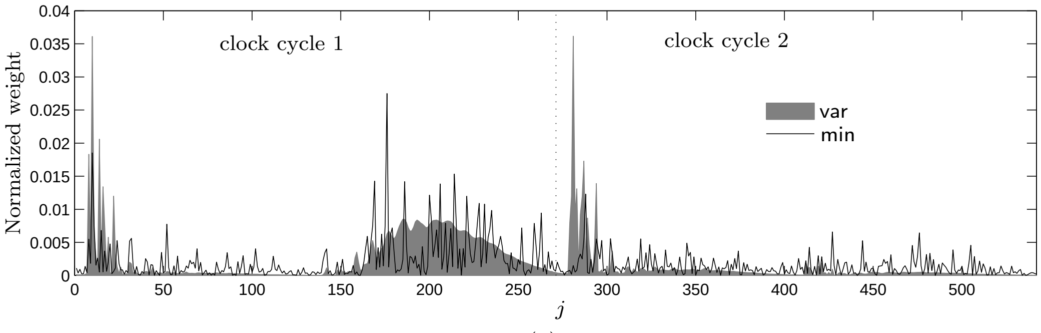
That is, statistic $E(\tau_1, \tau_2)_{a_1\neq a_2}$ follows the *noncentral* chi-square distribution with l degrees of freedom and $\lambda = \sum_{j=1}^{l} \left(\delta_j^{(1,2)}/\sqrt{2}\sigma\right)^2$ up to the coefficient $2\sigma^2$. Again, we have an approximation using

Proposition 2. *Statistic*

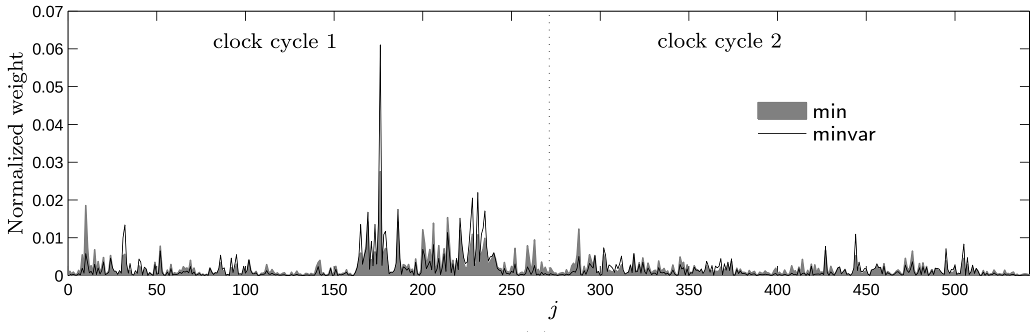
$$E(\tau_1, \tau_2)_{a_1\neq a_2} = \sum_{j=1}^{l} (\tau_{1,j} - \tau_{2,j})^2$$

[3] The real measured power consumption is often due to the generic multivariate normal distribution. However, almost all entries of the corresponding covariance matrix are close to zero. Thus, the model with independent multivariate normal distribution seems to be quite realistic.

for $\tau_i = (\tau_{i,1}, \ldots, \tau_{i,l}) \in \mathbb{R}^l$ *with* $\tau_{i,j} \sim \mathcal{N}\left(s_{i,j}, \sigma^2\right)$ *can be approximated by normal distribution* $\mathcal{N}\left(2\sigma^2(l+\lambda), 8\sigma^4(l+2\lambda)\right)$ *with* $\lambda = \sum_{j=1}^{l}\left(\delta_j^{(1,2)}/\sqrt{2}\sigma\right)^2$ *for sufficiently large* l*'s.*



(a)



(b)

Fig. 6. Informative points for DPA vs. signal difference (a), and the effect of weighting signal difference with variance (b)

5.3 Dimension reduction with signal difference

In the comparison of the two traces with the Euclidean distance, we try to distinguish between the collisions and non-collisions, *i.e.* between the distributions $E(\tau_1, \tau_2)_{a_1 \neq a_2}$ and $E(\tau_1, \tau_2)_{a_1 = a_2}$. As described above these statistics approximately follow normal distribution for large numbers of trace points. To efficiently distinguish between these two statistics it is crucial to decrease their variances while keeping the difference of their means high. For this purpose, to better distinguish between the collisions and non-collisions, we proposed to discard [9] points of traces with small minimal contribution to the difference of means[4].

To illustrate this method of dimension reduction, we assume for the moment that $\delta_j^{(1,2)} = 0$ for $j > l/2$ and $\delta_j^{(1,2)} \neq 0$ for $j \leq l/2$ with l even, that is, the second half of the trace does not contain any data dependent information. Then we can discard the second

[4] A more general way is weighting the points by their contribution to the difference of means and using weighted Euclidean distance as a metric, however our experiments have shown that point selection, being an extreme case of point weighting, is much more efficient.

halves of the both traces τ_1 and τ_2 in the comparison with the Euclidean distance and compute two related statistics on the rest of the points:

$$E'(\tau_1, \tau_2)_{a_1=a_2} = \sum_{j=1}^{l/2} (\tau_{1,j} - \tau_{2,j})^2,$$

$$E'(\tau_1, \tau_2)_{a_1 \neq a_2} = \sum_{j=1}^{l/2} (\tau_{1,j} - \tau_{2,j})^2.$$

This will adjust the means and variances of the approximating normal distributions: $\mathcal{N}\left(\sigma^2 l, 4\sigma^4 l\right)$ and $\mathcal{N}\left(2\sigma^2(l/2+\lambda), 8\sigma^4(l/2+2\lambda)\right)$, respectively. Note that the difference of means remains unaffected and equal to $2\sigma^2\lambda$. At the same time both variances are reduced, one of them by factor 2, which allows one to distinguish between these two distributions more efficiently and, thus, to detect collisions more reliably.

More generally speaking, for AES we have to reliably distinguish between inputs in each (a_{i_1}, a_{i_2}) of the $\binom{256}{2}$ pairs of byte values, $a_{i_1}, a_{i_2} \in \mathbb{F}_{2^8}$. Thus, the most informative points j of the traces are those with maximal minimums of $\delta_j^{(i_1,i_2)}$ over all pairs of different inputs, that is, points j with maximal values of

$$\min_{a_{i_1} \neq a_{i_2}} \delta_j^{(i_1,i_2)}. \tag{8}$$

We will denote this point selection criterion as min for brevity.

We estimated the values of min for all time instances j of the two (most leaky) cycles of the S-box look-up in our reference AES implementation and compared this to the signal variance in the same time points, $\mathrm{var}(s_{i,j})$, $a_i \in \mathbb{F}_{2^8}$, which is known to be an adequate indicator of the points leaking information in DPA and to which we will refer to as var, see Figure 6(a).

5.4 Dimension reduction with weighted signal difference

Figure 6(a) reveals that the points selected based on min are different from those chosen using var. Therefore, when calculating the contribution of the point j to the Euclidean distance one can consider not only (8) but also the variance in this point across the traces for all 256 possible input values. Thus, a more efficient criterion referred to as minvar can be defined choosing points that maximize

$$\min_{a_{i_1} \neq a_{i_2}} \delta_j^{(i_1,i_2)} / \mathrm{var}(s_{i,j}).$$

The intuition behind this additional weighting of points by the inverse of variance value is that the points with higher variation across different input values contribute less information but more noise to the difference of means.

In Figure 6(b), we compare minvar to just min for the same two clock cycles of our reference implementation. One can see the differences: weighting by the variance increases the contribution of some points while decreasing the contribution of the others (minvar) compared to the pure signal difference (min). This difference will be well captured by the information-theoretic metric that we will define below and use to compare the techniques in a sound way.

5.5 λ-divergence as information-theoretic metric

As mentioned above, the goal of collision detection is to efficiently distinguish between collisions and non-collisions, that is, between the distribution of the Euclidean distance for a pair of equal and non-equal inputs. Here we propose an information-theoretic measure of difference between these distributions.

Let $P(X)$ be the probability distribution over all possible secret values to be recovered using side-channel information. In the case of collision attacks, X is a set of two elements: $X = \{\text{collision, non-collision}\}$. Further let $P(L)$ be the probability distribution of side-channel leakage as measured by a collision detection method. For instance, L can be the set of all possible values of the Euclidean distance between side-channel traces for pairs of inputs to the S-box. Correspondingly, let $P(L|x)$ be the probability distribution of leakage, taken separately for collisions and non-collisions, depending on $x \in X$. We will denote $P(L_c) = P(L|x = \text{collision})$ and $P(L_n) = P(L|x = \text{non-collision})$. (Note that in Section 5.2 we have described these distributions in the independent multivariate Gaussian model.)

Our metric uses the notion of Kullback-Leibler divergence between two distributions A and B, $D_{KL}(A||B)$ [19]. For discrete distributions, it is defined as

$$D_{KL}(A||B) = \sum_i A(i) \log_2 \frac{A(i)}{B(i)}.$$

Note that D_{KL} is not commutative and $D_{KL}(A||B) \neq D_{KL}(B||A)$.

To compare collision detection methods, we use the λ-divergence between leakage distributions for collisions and non-collisions:

$$D_\lambda(L_c||L_n) = \lambda D_{KL}(L_c||L_n) + (1-\lambda) D_{KL}(L_n||L_c), \tag{9}$$

where λ is the a priori probability of a collision, in other words, $\lambda = P(x = \text{collision})$. $\lambda = 1/256$ for the 8-bit S-box of AES.

Now we show that the λ-divergence metric as introduced above and the mutual information metric [29] when applied to collision detection instead of template attacks are equivalent.

Proposition 3. $I(X, L) = D_\lambda(L_c||L_n)$.

Proof. The left-hand side of the above equality is $I(X, L) = H(L) - H(L|X)$ by definition. One can transform the right-hand side using the definitions of the Kullback-Leibler divergence and λ-divergence and obtain the left-hand side:

$$D_\lambda(L_c||L_n) = \lambda D_{KL}(L_c||L_n) + (1-\lambda) D_{KL}(L_n||L_c) =$$

$$\lambda \textstyle\sum_i P(l_c = i) \log_2 \frac{P(l_c=i)}{P(l_n=i)} + (1-\lambda) \sum_i P(l_n = i) \log_2 \frac{P(l_n=i)}{P(l_c=i)} =$$

$$[-\lambda \textstyle\sum_i P(L_c) \log_2 P(L) - (1-\lambda) \sum_i P(L_n) \log_2 P(L)] -$$

$$[-\lambda \textstyle\sum_i P(L_c) \log_2 P(L_c) - (1-\lambda) \sum_i P(L_n) \log_2 P(L_n)] =$$

$$= H(L) - H(L|X).$$

So the information-theoretic metric of the unified framework [29] applies well to the collision detection procedure.

The metric is interpreted in the sense that its lower values (*i.e.* lower mutual information) mean better distinguishing between the distributions and therefore better collision detection. In the following, we use the λ-divergence to compare our collision detection techniques to each other and to the existing techniques.

5.6 Comparison of collision detection techniques with λ-divergence

For the comparison of collision detection techniques using the λ-divergence, one has to know the distributions $P(L_c)$ and $P(L_n)$. The only way to obtain these distributions is to estimate them empirically. The same problem of estimating the leakage distribution arises in MIA attacks and several distribution estimation methods have been reported to be used [23]. We have opted for the histogram method, that is, we obtain the histograms for $P(L_c)$ and $P(L_n)$ from the samplings of Euclidean distance using the side-channel traces from our reference implementation for equal and non-equal inputs, respectively. Figure 7 shows an example of these distributions from our experiments.

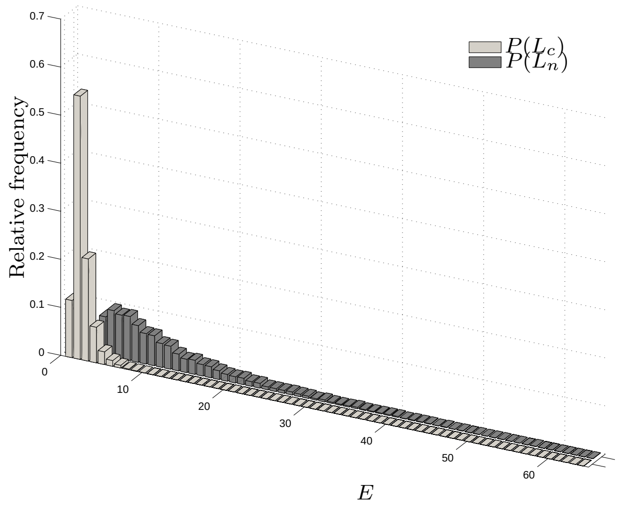


Fig. 7. Empirical distributions for $P(L_c)$ and $P(L_n)$

Then we compute $D_\lambda(L_c||L_n)$ following (9) in a straightforward way, setting $\lambda = 1/256$. Finally, for the convenience of representation, we compute $H(L) - D_\lambda(L_c||L_n)$ to have 0 for identical distributions and larger values for more distinct distributions (note that for computing $H(L)$ we need $P(L)$, which we recover from $P(L_c)$ and $P(L_n)$ knowing $P(X) = \{1/256, 255/256\}$).

We have evaluated our two new dimension reduction techniques min and minvar, the technique var commonly used in DPA, and detection without dimension reduction as a reference point. Finally, we tried averaging t traces to capture the effect of trace averaging. Figure 8 presents the experimental results for the considered techniques.

The figure shows information D_λ, measured in bits, brought by a single comparison, against the number t of trace averagings. Dimension reduction by variance, var, which works well in DPA attacks, worsens collision detection. However, both new methods, min and minvar, lead to clearly more efficient collision detection, minvar being best. As expected, the information gain grows with the averaging, since the latter increases the SNR.

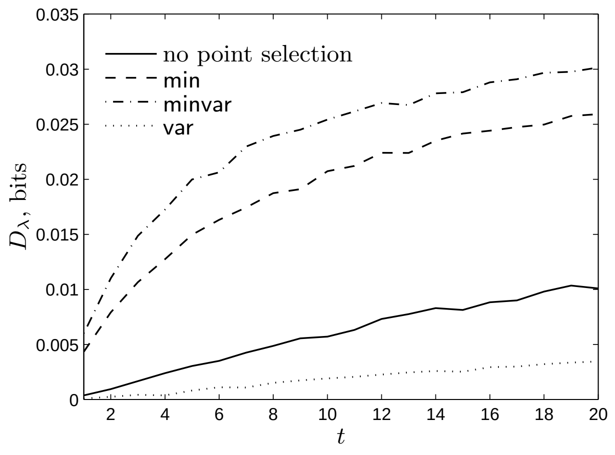


Fig. 8. Collision detection: effect of dimension reduction

6 Practical Evaluation

Here we show that the techniques we have introduced in this work perform well in practice. We implement (linear) collision attacks (Section 2) following our combination framework (Section 3) with the DPA-driven test of chain (Section 4) and employing the new collision detection methods (Section 5) against the target implementation. We experimentally estimate the efficiency of these DPA-combined collision attacks.

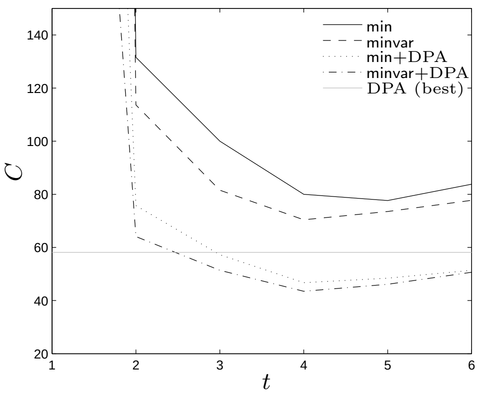
Our target implementation is AES-128 on ATmega16, a popular 8-bit μC. We measured the power consumption of the μC while it was encrypting the given plaintexts. The power consumption trace for an S-Box lookup execution comprised 4600 samples. From these, we reduced the dimension to 900 samples using our techniques min or minvar. This number was chosen empirically by observing when the attack efficiency reached saturation. The component of DPA was implementing CPA in the Hamming weight model [10].

Launching a series of 500 attacks for a given number of inputs γ and averagings t, we experimentally estimated the efficiency parameters defined in Section 3.3 of all attacks in question. Namely, besides the online complexity $C_{\text{online}} = t \cdot \gamma$, we obtained the success probability π by counting the number of successful attacks: we considered an attack successful if it was possible to recover the correct full AES key with $C_{\text{recovery}} \leq 2^{40}$. Having C_{online} and π, we computed our efficiency metric $C = C_{\text{online}}/\pi$ that reflects the expected number of measurements to mount a successful attack.

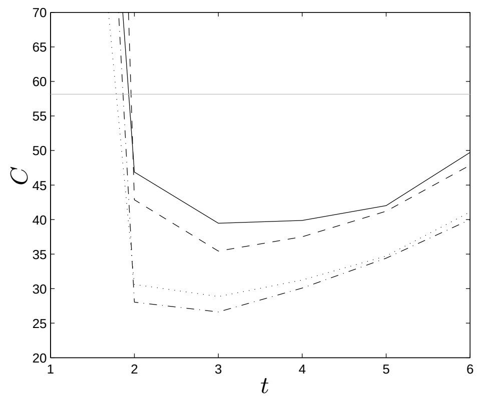
Instead of using a fixed or adaptive (like in [9]) threshold for the value of Euclidean distance E in collision detection, we follow another approach that allows to improve the attack efficiency. We consider a list of *collision candidates* consisting of all $(16\cdot\gamma)^2/2-8\cdot\gamma$ S-Box instance pairs sorted by E ascending. Taking c top candidates out of this list, starting from $c = 1$, we determine the number of chains h and their lengths n_i, $i = 1, \ldots, h$. In case of the collision-only attack, $C_{\text{recovery}} = 2^{8h}$ and we can have h at most 5 to stay within the admissible bound of 2^{40}. If h is higher, we take one more collision (increment c). Once we have enough collisions to attain $h = 5$, we perform the key recovery and check if the correct full key is among the recovered candidates. A similar approach can be used in case of the DPA-combined collision attack.

We experimentally characterized 4 specific attacks employing our techniques: collisions using min, collisions using minvar, collisions using min combined with DPA, and collisions using minvar combined with DPA. Comparison of these 4 attacks in terms of the metric

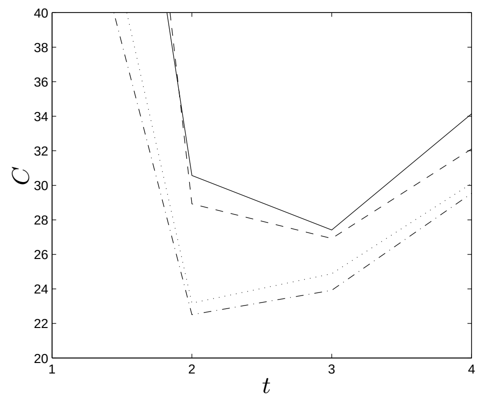
C for varying averaging t and different numbers γ of inputs is presented in Figure 9. As a reference, we also plot the best achieved case for the DPA-only attack (with C_{recovery} also bounded by 2^{40}, i.e. when the 2^{40} most probable AES key candidates as suggested by DPA are tested).



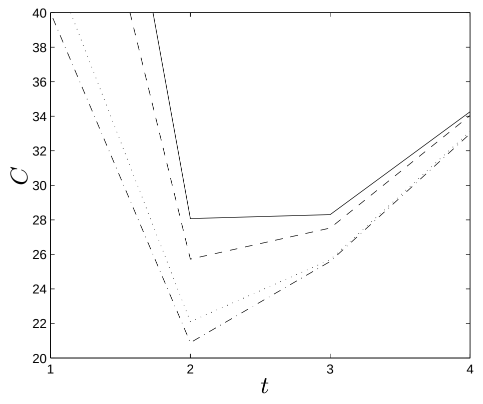
(a) $\gamma = 5$



(b) $\gamma = 6$



(c) $\gamma = 7$



(d) $\gamma = 8$

Fig. 9. Attack efficiency $C = C_{\text{online}}/\pi$ in practice, against averaging t, for different numbers γ of inputs

One can see that the combination of collision attacks with DPA clearly outperforms both collision-only and DPA-only attacks. The dimension reduction technique minvar outperforms min, thus, conforming to the information-theoretic comparison in Section 5.6.

7 Conclusions and Open Problems

In this article, we presented combined divide-and-conquer and collision attacks using side-channel leakage against implementations of cryptographic algorithms. We developed a framework for the combination of these attacks and theoretically analyzed its properties. We have also proposed dimension reduction techniques to improve side-channel collision detection using Euclidean distance, and an information-theoretic metric for comparison of collision detection techniques. We have carried out full combined DPA and collision attacks with dimension reduction techniques in practice against a real AES-128 implementation, showing that our combination is more efficient than both stand-alone DPA and collision attacks. Our experimental results also suggest that combined attacks exploit more information in the side-channel scenario than their stand-alone components. Below we present a relation of our attacks to the existing ones and outline some open problems.

Unlike collision attacks, template attacks require, in addition to a proper dimension reduction, detailed knowledge of the implementation for profiling. This appears to be a much weaker attack model than the one we use. However, the evaluation of template-combined collision attacks using our framework is still an open problem. This combination can possibly reduce the cost of the profiling stage and demonstrate that the template-only attacks, though being optimal in an information-theoretic sense [11], do not use all key-related information available to the attacker. This is due to the fact that their optimality is considered with respect to small key chunks only, not the entire key, as it is the case in collision and other analytic attacks that are algorithm-aware. Another line of future research—initiated in [8]—is using profiling to improve collision detection.

As collision and template attacks, MIA [13] does not necessarily require a leakage model. However, as demonstrated in [23], MIA tends to be significantly less efficient than DPA in terms of the required number of traces for unmasked implementations, even in the presence of strong noise. It is another open problem to evaluate MIA-combined collision attacks using our framework. As in the case of template attacks, we expect this combination to result in a reduced complexity.

While DPA, MIA and template techniques have been naturally incorporated by the unified framework [29] for comparing side-channel attacks, such analytical techniques as collision attacks, considered here, and algebraic attacks [25], cannot be reasonably captured by the unified framework directly. We consider it an important open problem to come up with a development of the unified framework both practically and generically applicable to analytic attacks. However, in this article, we successfully applied metrics similar to those of [29] to study some local properties of collision attacks.

From an information-theoretic perspective, each comparison of two traces with the purpose of collision detection, should yield in our attacks up to 0.03 bit key-related information (see Figure 8). However, not all of it is used for key recovery afterwards, where only collisions result in equations. At the same time, detected non-collisions also carry useful information ignored by the current techniques. Their usage seems to be technically problematic, since each non-collision would add an equation of a high degree to the system of equations to be solved. We leave this as another open problem.

References

1. Archambeau, C., Peeters, E., Standaert, F.X., Quisquater, J.J.: Template Attacks in Principal Subspaces. In: CHES'06. LNCS, vol. 4249, pp. 1–14. Springer-Verlag (2006)
2. Batina, L., Gierlichs, B., Lemke-Rust, K.: Differential cluster analysis. In: CHES'09. LNCS, vol. 5747, pp. 112–127. Springer-Verlag (2009)
3. Batina, L., Gierlichs, B., Prouff, E., Rivain, M., Standaert, F.X., Veyrat-Charvillon, N.: Mutual information analysis: a comprehensive study. To appear in JoC, available at `http://www.matthieurivain.com/wp-content/uploads/2010/06/joc10.pdf` (2010)
4. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: CRYPTO'97. LNCS, vol. 1294, pp. 513–525. Springer-Verlag (1997)
5. Biryukov, A., Bogdanov, A., Khovratovich, D., Kasper, T.: Collision Attacks on Alpha-MAC and Other AES-based MACs. In: CHES'07. LNCS, vol. 4727, pp. 166–180. Springer-Verlag (2007)
6. Biryukov, A., Khovratovich, D.: Two new techniques of side-channel cryptanalysis. In: CHES'07. LNCS, vol. 4727, pp. 195–208. Springer (2007)
7. Bogdanov, A.: Improved side-channel collision attacks on AES. In: SAC'07. LNCS, vol. 4876, pp. 84–95. Springer-Verlag (2007)
8. Bogdanov, A.: Multiple-differential side-channel collision attacks on AES. In: CHES'08. LNCS, vol. 5154, pp. 30–44. Springer-Verlag (2008)

9. Bogdanov, A., Kizhvatov, I., Pyshkin, A.: Algebraic methods in side-channel collision attacks and practical collision detection. In: INDOCRYPT'08. LNCS, vol. 5365, pp. 251–265. Springer-Verlag (2008)
10. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: CHES'04. LNCS, vol. 3156, pp. 16–29. Springer-Verlag (2004)
11. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: CHES'02. LNCS, vol. 2523, pp. 51–62. Springer-Verlag (2003)
12. FIPS: Advanced Encryption Standard. Publication 197. National Bureau of Standards, U.S. Department of Commerce (2001)
13. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: CHES'08. LNCS, vol. 5154, pp. 426–442. Springer-Verlag (2008)
14. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic Methods. In: CHES'06. LNCS, vol. 4249, pp. 15–29. Springer-Verlag (2006)
15. Handschuh, H., Preneel, B.: Blind differential cryptanalysis for enhanced power attacks. In: SAC'06. LNCS, vol. 4356, pp. 163–173. Springer-Verlag (2006)
16. Kim, J., Lee, Y., Lee, S.: DES with any reduced masked rounds is not secure against side-channel attacks. Computers & Mathematics with Applications 60(2), 347–354 (2010), `http://www.sciencedirect.com/science/article/B6TYJ-4Y8G1NV-3/2/ab0200615db0fd6c3a26e527602ad5d5`
17. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: CRYPTO'96. LNCS, vol. 1109, pp. 104–113. Springer-Verlag (1996)
18. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO'99. LNCS, vol. 1666, pp. 388–397. Springer-Verlag (1999)
19. Kullback-Leibler divergence. `http://en.wikipedia.org/wiki/Kullback_Leibler_divergence`
20. Ledig, H., Muller, F., Valette, F.: Enhancing Collision Attacks. In: CHES'04. LNCS, vol. 3156, pp. 176–190. Springer-Verlag (2004)
21. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer-Verlag (2007)
22. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: CHES'10. LNCS, vol. 6225, pp. 125–139. Springer-Verlag (2010)
23. Prouff, E., Rivain, M.: Theoretical and practical aspects of mutual information based side channel analysis. To appear in IJACT, available at `http://www.matthieurivain.com/wp-content/uploads/2010/06/ijact10.pdf` (2010)
24. Quisquater, J.J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: E-smart. LNCS, vol. 2140, pp. 200–210. Springer-Verlag (2001)
25. Renauld, M., Standaert, F.X., Veyrat-Charvillon, N.: Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In: CHES'09. LNCS, vol. 5747, pp. 97–111. Springer-Verlag (2009)
26. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: CHES'05. pp. 30–46. LNCS, Springer-Verlag (2005)
27. Schramm, K., Leander, G., Felke, P., Paar, C.: A collision-attack on AES: Combining side channel- and differential-attack. In: CHES'04. LNCS, vol. 3156, pp. 163–175. Springer-Verlag (2004)
28. Schramm, K., Wollinger, T.J., Paar, C.: A New Class of Collision Attacks and Its Application to DES. In: FSE'03. LNCS, vol. 2887, pp. 206–222. Springer-Verlag (2003)
29. Standaert, F.X., Malkin, T., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: EUROCRYPT'09. LNCS, vol. 5479, pp. 443–461. Springer-Verlag (2009)
30. Wiemers, A.: Collision Attacks for Comp128 on Smartcards (December 2001), eCC-Brainpool Workshop on Side-Channel Attacks on Cryptographic Algorithms, Bonn, Germany