Multiple-Differential Side-Channel Collision Attacks on AES

Andrey Bogdanov

Horst Görtz Institute for IT Security
Ruhr University Bochum, Germany
abogdanov@crypto.rub.de,
www.crypto.rub.de

Abstract. In this paper, two efficient multiple-differential methods to detect collisions in the presence of strong noise are proposed - binary and ternary voting. After collisions have been detected, the cryptographic key can be recovered from these collisions using such recent cryptanalytic techniques as linear [1] and algebraic [2] collision attacks. We refer to this combination of the collision detection methods and cryptanalytic techniques as *multiple-differential collision attacks* (MDCA).

When applied to AES, MDCA using binary voting without profiling requires about 2.7 to 13.2 times less traces than the Hamming-weight based CPA for the same implementation. MDCA on AES using ternary voting with profiling and linear key recovery clearly outperforms CPA by requiring only about 6 online measurements for the range of noise amplitudes where CPA requires from 163 to 6912 measurements. These attacks do not need the S-box to be known. Moreover, neither key nor plaintexts have to be known to the attacker in the profiling stage.

Keywords: side-channel attacks, collision detection, multiple-differential collision attacks, AES, DPA.

1 Introduction

Side-channel attacks have become mainstream since their first publication in [3]. Differential power analysis (DPA) [4] and correlation power analysis (CPA) [5], a generalization of DPA, are probably the most wide-spread practical attacks on numerous cryptographic embedded systems such as smart-card microcontrollers [6] and dedicated lightweight ASICs [7].

Collision attacks represent another class of side-channel attack techniques being essentially based on the cryptanalytic properties of attacked cryptographic algorithms. Collision attacks on block ciphers were proposed in [8] for DES. The idea is due to Hans Dobbertin and was also discussed in the early work [9]. Since then there has been quite a bit of research in this area: [10] improves the collision attack on DES, [11] applies the technique to AES, [12] suggests a collision attack

E. Oswald and P. Rohatgi (Eds.): CHES 2008, LNCS 5154, pp. 30–44, 2008.

on an AES-based MAC construction, [13] combines collision attacks on AES with differential cryptanalysis to overcome several masked rounds.

Recently such improvements as linear collision attacks [1] and algebraic collision attacks [2] for AES have been proposed which require a very low number of measurements for the key recovery procedure to succeed with a high probability and within a feasible time span. However, these attacks as well as those in [11] and [12] are rather theoretical being substantially based on the assumption that the implementation allows the attacker to reliably detect if two given S-box instances process the same value.

The contribution of this paper is two-fold. On the theoretical side, two collision detection techniques are proposed called *binary* and *ternary voting*. We refer to the combination of the statistical collision detection methods and cryptanalytic collision attacks as *multiple-differential collision attacks (MDCA)*. On the practical side, we apply MDCA to a hardware implementation of AES for a wide range of noise amplitudes using advanced power consumption simulation.

MDCA works in the two scenarios: where profiling is either allowed (ternary voting) or not allowed (ternary voting without profiling and binary voting). Note that the notion of profiling for our collision detection techniques is different from that for template attacks [14], [15]. While template attacks require detailed knowledge of the implementation in the profiling stage, the only information needed in the profiling stage of the collision detection methods is the time interval when the S-boxes are executed.

MDCA based on the binary voting method for the given AES implementation needs about 2.7 to 13.2 times less traces than Hamming-weight based CPA in the range of noise levels we studied. While MDCA based on ternary voting without profiling does not exhibit any advantages over CPA, the required number of online measurements for ternary voting with profiling is considerably lower than that for CPA for all noise amplitudes we investigated. For instance, if $\leq 10^6$ profiling measurements are allowed, MDCA based on ternary voting with profiling and linear key recovery requires only 6 online measurements in the noise amplitude range where the standard CPA would require from 163 to 6912 measurements. A further advantage of the proposed collision detection techniques combined with the linear collision attacks is that they work with *secret S-boxes*. Moreover, ternary voting with profiling also requires *neither keys nor inputs/outputs to be known* in the profiling stage. However, as already mentioned, the attacker has to know when the S-boxes are executed within the implementation.

The remainder of the paper is organized as follows. Section 2 discusses the attack scenarios, introduces some notation and briefly mentions the linear collision attacks. Section 3 presents the multiple-differential collision detection techniques and theoretically investigates some of their properties. Section 4 characterizes the underlying least-square based binary comparison test for an AES implementation, applies MDCA to this implementation and compares the results to CPA. We conclude in Section 5.

2 Preliminaries

2.1 Attack Flows

There are two basic attack scenarios we consider: collision attacks *without profiling* and collision attacks *with profiling*. A collision attack without profiling consists of an online stage and an offline stage, while a collision attack with profiling additionally contains a profiling stage.

In the *online stage*, some random known 16-byte plaintexts $P_i = \{p_j^i\}_{j=1}^{16}$, $p_j^i \in GF(2^8)$, are sent to the attacked device implementing AES, where they are added with the first 16-byte subkey $K = \{k_j\}_{j=1}^{16}$, $k_j \in GF(2^8)$. Then each of the 16 values $a_j^i = p_j^i \oplus k_j$, $a_j^i \in GF(2^8)$, is processed by the AES S-box. The *online traces* $T_i = \{\tau_j^i\}_{j=1}^{16}$, $\tau_j^i = (\tau_{j,1}^i, \ldots, \tau_{j,l}^i) \in \mathbb{R}^l$, corresponding to these S-box calculations are acquired by the measurement equipment (e.g. they can contain such side-channel parameters as power consumption or electromagnetic radiation).

In the optional *profiling stage*, the device is triggered to perform a number of cryptographic operations with some unknown profiling inputs for some unknown keys. The *profiling traces* are acquired by the measurement equipment. The profiling stage takes place before the online stage and can be reused by several attacks on the same implementation.

The *offline stage* recovers the key. This occurs in two steps. First, collisions are detected in the online traces T_i by means of signal processing. The collision detection with profiling additionally uses the profiling traces. Second, an AES key candidate is obtained using the detected collisions and the corresponding inputs P_i. Note that one or several plaintext-ciphertext pairs produced with the attacked key may be needed to identify the correct key candidate in the offline stage.

If averaging is applied, the attacker has to be able to send several unknown equal inputs to the device and to fix some unknown key for these measurements in the profiling stage. Additionally, he has to be able to send several copies of the known random plaintexts to the implementation in the online stage.

The attack complexity is defined by three parameters. $C_{\text{profiling}}$ is the number of inputs to AES for which measurements have to be performed in the profiling stage (number of profiling measurements). Obviously, $C_{\text{profiling}} = 0$ for collision attacks without profiling. C_{online} is the number of inputs to AES for which measurements have to be performed in the online stage (number of online measurements). C_{offline} is the computational complexity of the key recovery, that is, the number of operations needed to solve the resulting systems of linear or nonlinear equations and to identify the most probable solution.

2.2 Key Recovery from S-Box Collisions

AES-128 performs 160 S-box operations in the data path for each run, which are different for different inputs, and 40 additional S-box computations in the key schedule, which remain the same for a given key. If two of these S-box instances

in one or two distinct runs process the same value, there is *a generalized internal collision*. The power of the improved collision attacks [1] on AES origins from the fact that the number of generalized collisions grows quadratically with the linear increase of the number of unique inputs considered. So, even if the key schedule is ignored, there are about 40.9 colliding S-boxes for just one input and already about 555.2 collisions for 5 inputs.

When collisions have been detected, the AES key has to be recovered. In this paper we use the linear collision attacks [1] for this purpose. A *linear collision* in AES is a generalized collision within the first AES round. Given such a linear collision, the attacker obtains a binomial linear equation over $GF(2^8)$ of the form $k_{j_1} \oplus k_{j_2} = p_{j_1}^{i_1} \oplus p_{j_2}^{i_2}$ for $j_1 \neq j_2$.

Let γ be the number of different random inputs P_i to the algorithm for which collisions have to be detected in order for the key to be recovered with probability π within C_{offline} operations. In this paper, we apply the variant of linear collision attacks with $\gamma = 6$, $\pi = 0.854$ and C_{offline} equal to $2^{37.15}$ encryptions, see [1] for details and [2] for some more advanced techniques.

3 Multiple-Differential Collision Detection

The goal of the collision detection is to decide if two S-box instances in AES have had equal inputs based on side-channel traces.

For the direct binary comparison of S-box instances, the least-square based test was used in the original collision attack on AES in [11], which is essentially a computation of the Euclidean distance between two real-valued traces. Its resolution can be increased by suppressing noise through averaging.

However, there are other collision detection methods substantially using the simple binary comparison, two of which - binary voting and ternary voting - we propose in this section. Both methods can be combined with averaging. Additionally, the ternary voting test enables performance gains through profiling.

3.1 Binary Comparison

Definition. Given two traces $\tau_{j_1}^{i_1} = (\tau_{j_1,1}^{i_1}, \ldots, \tau_{j_1,l}^{i_1}) \in \mathbb{R}^l$ and $\tau_{j_2}^{i_2} = (\tau_{j_2,1}^{i_2}, \ldots, \tau_{j_2,l}^{i_2}) \in \mathbb{R}^l$, respectively corresponding to S-box j_1 for plaintext P_{i_1} and to S-box j_2 for plaintext P_{i_2}, the binary comparison test $\mathfrak{T}^{BC}$ can be defined as:

$$\mathfrak{T}^{BC}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) = \begin{cases} 0 \text{ (no collision)}, & \text{if } \mathfrak{S}^{BC}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) > Y^{BC} \\ 1 \text{ (collision)}, & \text{if } \mathfrak{S}^{BC}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) \leq Y^{BC}, \end{cases}$$

where Y^{BC} is a decision threshold and

$$\mathfrak{S}^{BC}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) = \sum_{r=1}^{l} (\tau_{j_1,r}^{i_1} - \tau_{j_2,r}^{i_2})^2,$$

which can be seen as a correlation characteristic of two reduced templates. Let $\mathfrak{T}^{BC}$ be characterized by the following type I and II error probabilities:

$$\alpha_1 = \Pr\{\mathfrak{T}^{BC}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) = 0 | a_{j_1}^{i_1} = a_{j_2}^{i_2}\},$$
$$\alpha_2 = \Pr\{\mathfrak{T}^{BC}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) = 1 | a_{j_1}^{i_1} \neq a_{j_2}^{i_2}\}.$$

Note that α_1 and α_2 depend on the implementation and the value of Y^{BC}. Of course, there is a strong dependency on the noise as well. See Section 4 for estimations of α_2 with a given α_1 for one implementation example and a wide range of noise amplitudes.

Combination with Averaging. To increase the resolution of the collision detection one can use averaging. That is, each plaintext is sent t times to the device. Respectively, t measurements are performed for each plaintext. Then the obtained traces for each distinct plaintext are averaged. If the noise is due to normal distribution with the zero mean value and a standard deviation σ, then the noise amplitude of the trace averaged t times will be $\sigma/\sqrt{t}$.

3.2 Binary Voting Test

In this subsection we propose a more efficient method to suppress noise which is called *binary voting*. Like in averaging, traces for multiple copies of the same plaintexts are first obtained. However, instead of averaging, the attacker tries to detect collisions using binary comparison for each pair of the traces and applies voting to filter for correct ones.

Definition. We have to reliably detect collisions for γ different plaintexts. Then each of these plaintexts is sent M^{BV} times to the device. So we have a group $\tilde{\tau}_j^i = \{\tau_j^{i,m}\}_{m=1}^{M^{BV}}$, $\tau_j^{i,m} \in \mathbb{R}^l$, of traces for each S-box instance and each plaintext. That is, the direct application of binary voting requires $C_{\text{online}} = \gamma \cdot M^{BV}$ measurements.

The binary voting test is based on the following statistic which uses a binary comparison test (for instance, the least-square based one as defined above):

$$\mathfrak{S}^{BV}(\tilde{\tau}_{j_1}^{i_1}, \tilde{\tau}_{j_2}^{i_2}) = \sum_{m=1}^{M^{BV}} \mathfrak{T}^{BC}(\tau_{j_1}^{i_1,m}, \tau_{j_2}^{i_2,m}),$$

where the multiple traces for two S-box instances are pairwisely compared to each other. The test $\mathfrak{T}^{BV}$ to decide if there has been a collision is then defined as

$$\mathfrak{T}^{BV}(\tilde{\tau}_{j_1}^{i_1}, \tilde{\tau}_{j_2}^{i_2}) = \begin{cases} 0 \text{ (no collision)}, & \text{if } \mathfrak{S}^{BV}(\tilde{\tau}_{j_1}^{i_1}, \tilde{\tau}_{j_2}^{i_2}) < Y^{BV} \\ 1 \text{ (collision)}, & \text{if } \mathfrak{S}^{BV}(\tilde{\tau}_{j_1}^{i_1}, \tilde{\tau}_{j_2}^{i_2}) \geq Y^{BV}, \end{cases}$$

where Y^{BV} is a decision threshold. The idea is that the distribution of statistic $\mathfrak{S}^{BV}$ will be different for $a_{j_1}^{i_1} = a_{j_2}^{i_2}$ and for $a_{j_1}^{i_1} \neq a_{j_2}^{i_2}$.

Properties. As the individual binary comparisons are independent, the distribution of $\mathfrak{S}^{BV}$ is due to the binomial law with M^{BV} experiments and success probability p. If $a_{j_1}^{i_1} = a_{j_2}^{i_2}$, the success probability is $p = p_e = 1-\alpha_1$. If $a_{j_1}^{i_1} \neq a_{j_2}^{i_2}$, it is $p = p_{ne} = \alpha_2$. For sufficiently large group sizes M^{BV}, the distribution of $\mathfrak{S}^{BV}$ can be approximated by a normal distribution $\mathcal{N}(M^{BV}p, M^{BV}p(1-p))$. That is, the problem of collision detection is reduced to the problem of distinguishing between two normal distributions in this case. Thus, the required value of M^{BV} can be obtained using

Proposition 1. *Let α_1 and α_2 be type I and II error probabilities, respectively, for $\mathfrak{T}^{BC}$. Then the number of S-box traces in each group needed to distinguish between $a_{j_1}^{i_1} = a_{j_2}^{i_2}$ and $a_{j_1}^{i_1} \neq a_{j_2}^{i_2}$ using binary voting test $\mathfrak{T}^{BV}$ can be estimated as*

$$M^{BV} \approx \frac{(u_{1-\beta_1}\sqrt{\alpha_1(1-\alpha_1)} + u_{1-\beta_2}\sqrt{\alpha_2(1-\alpha_2)})^2}{(1-\alpha_1-\alpha_2)^2},$$

where:

- *β_1 and β_2 are the required type I and II error probabilities for $\mathfrak{T}^{BV}$,*
- *$u_{1-\beta_1}$ and $u_{1-\beta_2}$ are quantiles of the standard normal distribution $\mathcal{N}(0,1)$.*

Combination with Averaging. The required value of M^{BV} depends on α_1 and α_2 which in turn can be seen as functions of the noise amplitude σ. For this reason we will write $M^{BV}(\sigma)$ where this dependency is important.

The binary voting technique can be combined with averaging. The traces are first averaged t times. Then the statistic $\mathfrak{S}^{BV}$ is computed. That is, one deals with $M^{BV}(\sigma/\sqrt{t})$ instead of $M^{BV}(\sigma)$.

Since each plaintext P_i is sent $t \cdot M^{BV}(\sigma/\sqrt{t})$ times to the device, binary voting with averaging requires $C_{\text{online}} = \gamma \cdot t \cdot M^{BV}(\sigma/\sqrt{t})$ measurements. Depending on the concrete implementation and on the range of σ, the measurement complexity can be reduced, if $\gamma \cdot t \cdot M^{BV}(\sigma/\sqrt{t}) < \gamma \cdot M^{BV}(\sigma)$ for some t. In the sequel, we will refer to binary voting with averaging simply as binary voting, since binary voting with averaging for $t = 1$ corresponds to the basic binary voting.

3.3 Ternary Voting Test

Ternary voting is another statistical technique we propose to reliably detect collisions. It is based on indirect comparisons of traces, where two given S-box traces (*target traces*, a subset of online traces) are compared through a pool of other ones (*reference traces*, profiling traces if any and possibly a subset of online traces).

While the ternary voting test is less efficient than the binary voting one in terms of the overall number of traces needed, it allows for profiling. That is, the reference traces can be acquired in the profiling stage and shared by several attacks, which can significantly amplify the performance of the online stage.

Definition. Let N^{TV} be the number of S-box instances whose (reference) traces $\{\tau_m\}_{m=1}^{N^{TV}}$, $\tau_m \in \mathbb{R}^l$, are available to the attacker for some random unknown

inputs $\{a_m\}_{m=1}^{N^{TV}}$, $a_m \in GF(2^8)$. Let $\tau_{j_1}^{i_1}$ and $\tau_{j_2}^{i_2}$ be the traces for two further S-box instances for which we have to decide if $a_{j_1}^{i_1} = a_{j_2}^{i_2}$. Then the ternary voting test can be defined as follows:

$$\mathfrak{T}^{TV}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) = \begin{cases} 0 \text{ (no collision)}, & \text{if } \mathfrak{S}^{TV}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) < Y^{TV} \\ 1 \text{ (collision)}, & \text{if } \mathfrak{S}^{TV}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) \geq Y^{TV}, \end{cases}$$

where

$$\mathfrak{S}^{TV}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}) = \textstyle\sum_{m=1}^{N^{TV}} F(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}, \tau_m)$$

with

$$F(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}, \tau_m) = \mathfrak{T}^{BC}(\tau_{j_1}^{i_1}, \tau_m) \cdot \mathfrak{T}^{BC}(\tau_{j_2}^{i_2}, \tau_m)$$

and Y^{TV} is some decision threshold. The key idea of ternary voting is similar to that of binary voting: The distributions of $\mathfrak{S}^{TV}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2})$ for $a_{j_1}^{i_1} = a_{j_2}^{i_2}$ and for $a_{j_1}^{i_1} \neq a_{j_2}^{i_2}$ will be different. Typically, $\mathfrak{S}^{TV}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2})$ will be higher for $a_{j_1}^{i_1} = a_{j_2}^{i_2}$ than for $a_{j_1}^{i_1} \neq a_{j_2}^{i_2}$. To decide if there has been a collision, the attacker needs to statistically distinguish between these two cases.

Properties. To explore the behaviour of F, it is not sufficient to know the type I and II error probabilities for the binary comparison test. Let $\mathfrak{T}^{BC}$ be characterized by the simultaneous distribution of the test results depending on the relations between $a_{j_1}^{i_1}$, $a_{j_2}^{i_2}$ and a_m:

$$\begin{aligned}
\chi_1 &= \Pr\{\mathfrak{T}^{BC}(\tau_{j_1}^{i_1}, \tau_m) = 1, \mathfrak{T}^{BC}(\tau_{j_2}^{i_2}, \tau_m) = 1 | a_{j_1}^{i_1} = a_{j_2}^{i_2} = a_m\}, \\
\chi_2 &= \Pr\{\mathfrak{T}^{BC}(\tau_{j_1}^{i_1}, \tau_m) = 1, \mathfrak{T}^{BC}(\tau_{j_2}^{i_2}, \tau_m) = 1 | a_{j_1}^{i_1} = a_{j_2}^{i_2} \neq a_m\}, \\
\chi_3 &= \Pr\{\mathfrak{T}^{BC}(\tau_{j_1}^{i_1}, \tau_m) = 1, \mathfrak{T}^{BC}(\tau_{j_2}^{i_2}, \tau_m) = 1 | a_{j_1}^{i_1} \neq a_{j_2}^{i_2}, a_{j_1}^{i_1} = a_m, a_{j_2}^{i_2} \neq a_m\}, \\
\chi_4 &= \Pr\{\mathfrak{T}^{BC}(\tau_{j_1}^{i_1}, \tau_m) = 1, \mathfrak{T}^{BC}(\tau_{j_2}^{i_2}, \tau_m) = 1 | a_{j_1}^{i_1} \neq a_{j_2}^{i_2}, a_m \neq a_{j_1}^{i_1}, a_m \neq a_{j_2}^{i_2}\}.
\end{aligned}$$

Then the probabilities

$$p_e = \Pr\{F(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}, \tau_m) = 1 | a_{j_1}^{i_1} = a_{j_2}^{i_2}\}$$

and

$$p_{ne} = \Pr\{F(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}, \tau_m) = 1 | a_{j_1}^{i_1} \neq a_{j_2}^{i_2}\}$$

can be computed using

Proposition 2. *If $a_{j_1}^{i^1}, a_{j_2}^{i_2}, a_m \in GF(2)^8$ are uniformly distributed and mutually independent, then*

$$p_e = \tfrac{1}{2^8}\chi_1 + \tfrac{2^8-1}{2^8}\chi_2$$

and

$$p_{ne} = \tfrac{2}{2^8}\chi_3 + \tfrac{2^8-2}{2^8}\chi_4.$$

Proof. If $a_{j_1}^{i_1} = a_{j_2}^{i_2}$, two cases are possible for $F(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}, \tau_m) = 1$:

- $a_{j_1}^{i_1} = a_{j_2}^{i_2} = a_m$ which happens with probability of $1/2^8$, and
- $a_{j_1}^{i_1} = a_{j_2}^{i_2} \neq a_m$ which happens with probability $\frac{2^8-1}{2^8}$.

If $a_{j_1}^{i_1} \neq a_{j_2}^{i_2}$, there are three cases leading to $F(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2}, \tau_m) = 1$:

- $a_{j_1}^{i_1} = a_m$, $a_{j_2}^{i_2} \neq a_m$ with probability $1/2^8$,
- $a_{j_2}^{i_2} = a_m$, $a_{j_1}^{i_1} \neq a_m$ with probability $1/2^8$, and
- $a_{j_1}^{i_1} \neq a_m$, $a_{j_2}^{i_2} \neq a_m$ with probability $(2^8 - 2)/2^8$.

The claims of the proposition follow. □

For the sake of simplicity, we first study the properties of $\mathfrak{T}^{TV}$ under the assumption that all applications of F to compute $\mathfrak{S}^{TV}$ are mutually independent. Under this assumption, $\mathfrak{S}^{TV}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2})$ would have a binomial distribution with N^{TV} being the number of experiments and success probability $p = p_e$, if $a_{j_1}^{i_1} = a_{j_2}^{i_2}$, or $p = p_{ne}$, if $a_{j_1}^{i_1} \neq a_{j_2}^{i_2}$. Thus, for sufficiently large values of N^{TV}, $\mathfrak{S}^{TV}(\tau_{j_1}^{i_1}, \tau_{j_2}^{i_2})$ could be approximated by normal distribution $\mathcal{N}(N^{TV}p, N^{TV}p(1-p))$. Thus, similarly to binary voting, the number N^{TV} of S-box reference instances needed to distinguish between $a_{j_1}^{i_1} = a_{j_2}^{i_2}$ and $a_{j_1}^{i_1} \neq a_{j_2}^{i_2}$ could be estimated as

$$N^{TV} \approx \frac{(u_{1-\beta_1}\sqrt{p_e(1-p_e)} + u_{1-\beta_2}\sqrt{p_{ne}(1-p_{ne})})^2}{(p_e - p_{ne})^2},$$

where β_1 and β_2 are the required type I and II error probabilities for $\mathfrak{T}^{TV}$, $u_{1-\beta_1}$ and $u_{1-\beta_2}$ are quantiles of the standard normal distribution $\mathcal{N}(0,1)$.

However, the applications of F are dependent and this result can be only used to obtain a rough estimation of N^{TV}.

Procedure, Complexity, Averaging. Now we can describe the basic procedure of ternary voting in the case that the target key is fixed in the device and the plaintexts are random and known. This is what we call *ternary voting without profiling*.

The number N^{TV} of S-box reference instances as well as the number M^{TV} of different inputs for which reference traces have to be acquired depend on the noise level σ. We will write $N^{TV}(\sigma)$ and $M^{TV}(\sigma)$, when this dependency is crucial for understanding.

First, the attacker obtains traces for $M^{TV}(\sigma)$ random plaintexts. This yields τ_m for $N^{TV}(\sigma) = 160 \cdot M^{TV}(\sigma)$ different S-box instances for AES-128, if the key schedule is not considered and all the $16 \cdot 10$ S-box traces within each AES run are acquired at a time. Then, if $M^{TV}(\sigma) \geq \gamma$, no further measurements are needed. Otherwise, the attacker acquires traces for further $\gamma - M^{TV}(\sigma)$ plaintexts. Note that some of the reference traces can be interpreted as target traces (16 S-box traces corresponding to the first round in each of some γ executions of AES).

This yields the complexity of $C_{\text{online}} = \max(\gamma, M^{TV}(\sigma))$ measurements, where

$$M^{TV}(\sigma) = \left\lceil \frac{N^{TV}(\sigma)}{160} \right\rceil .$$

Like binary voting, ternary voting can be combined with averaging to achieve better resolution. In this case each trace has to be averaged t times. Thus, the complexity of ternary voting with averaging is $C_{\text{online}} = t \cdot \max(\gamma, M^{TV}(\sigma/\sqrt{t}))$. In the sequel we refer to ternary voting both with and without averaging simply as ternary voting.

Profiling. Now we are ready to describe what we refer to as *ternary voting with profiling*. Unlike binary voting, the method of ternary voting allows for profiling. In the profiling stage, reference traces are acquired only, for which the attacker has to know *neither* the key used *nor* the plaintexts. Moreover, this also works if keys are changed between blocks of t executions. The target traces are obtained in the online phase and compared based on the pre-measured reference traces.

Thus, $C_{\text{profiling}} = t \cdot M^{TV}(\sigma/\sqrt{(t)})$ measurements have to be performed in the profiling stage, each measurement comprising all 10 rounds of AES-128. Then only $C_{\text{online}} = t \cdot \gamma$ measurements are needed in the online stage, each measurement comprising only the first round for the linear key recovery. For the latter measurements we do have to know inputs. Moreover, they all have to be performed with the key to be recovered.

3.4 Required Error Probabilities of Collision Detection

The measurement complexity of the binary and ternary voting methods depends on the success probability to be achieved. Let us take q as a desirable success probability of the whole attack and estimate the required type II error probabilities β_2 for binary and ternary voting. Recall that π is the success probability of the cryptanalytic collision attack used to recover the key after the collisions have been detected.

In the linear key recovery, there are 16γ S-box instances between which a collision can occur. That is, the voting has to be performed $w = \binom{16\gamma}{2}$ times. Then β_2 can computed as

$$\beta_2 = 1 - (q/\pi)^{1/w}.$$

For instance, if $\gamma = 6$ and $q = 0.5$, one obtains $\beta_2 \approx 1.174 \cdot 10^{-4}$. Additionally, β_1 has to be low enough to enable the detection of a sufficient number of collisions.

4 MDCA and AES: A Case Study

The purpose of this section is to estimate the real-world efficiency of different MDCA variants based on an AES implementation example and to compare

the methods to the standard Hamming-weight based CPA for the same AES implementation. In order to be able to perform this comparison for different noise levels σ, we carefully simulated the deterministic power consumption in Nanosim using dedicated power simulation libraries and added Gaussian noise of different amplitudes to it. The main results of the section are summarized in Table 1.

4.1 Implementation and Simulated Traces

The characteristics of $\mathfrak{T}^{BC}$ strongly depend on the signal-to-noise ratio of the implementation. To perform the estimations for a variety of noise levels, a serial VHDL implementation of the AES S-box has been performed (that is, only one S-box is calculated at a time). The deterministic power consumption for all 2^8 inputs was simulated using Synopsys Nanosim with the Dolphin Integration power consumption library SESAME-LP2 based on a 250nm technology by IHP [16]. The design was clocked at 10 MHz. The sampling rate was set to 10 Gsamples/s.

The S-box was implemented as combinatorial logic on the basis of an 8-bit register. Each S-box calculation $y = S(x)$ occurs in two clocks. In the first clock, the input x is read from the register and the output y is computed. In the second clock, the register is set to zero and the calculated output y is written to the register.

The simulated deterministic power traces obtained are noise-free. That is, there is neither electronic noise (power supply noise, clock generator noise, conducted emissions, radiated emissions, etc.) nor algorithmic noise (since only the relevant part of the circuit is considered) in these traces. To model noise we added random values due to univariate normal distribution[1] with the zero mean value and a standard deviation σ whose value characterizes the noise amplitude.

Note also that the simulated signal was not subject to a low-pass filter as it would have been the case for the real-world measurements of power consumption due to the presence of capacitances within the chip as well as on the circuit board where the power consumption measurements are performed. This would have cut off the high-frequency contribution to the signal reducing the advantage of high-resolution measurements. However, the effect of this circumstance is rather limited for the measurements of the electromagnetic radiation. A major limitation in this case is the bandwidth of the oscilloscope. Thus, we believe that the simulated traces with added Gaussian noise can be used for an initial analysis of the efficiency of our collision detection techniques. The main advantage of using the simulated power consumption is that one can add noise of different amplitudes to model the behaviour of attack methods for different devices and physical conditions.

To evaluate α_2 for this implementation, we chose Y^{BC} in $\mathfrak{T}^{BC}$ so that α_1 becomes sufficiently low by shifting Y^{BC} to the right. For this value of α_1, the

[1] Normal distribution is a sound noise model [17]. As a matter of fact, the noise is often distributed due to the multivariate normal distribution [17], [18]. However, only a few co-variances in the co-variance matrix of this multivariate normal distribution significantly differ from zero [18] for many implementations.

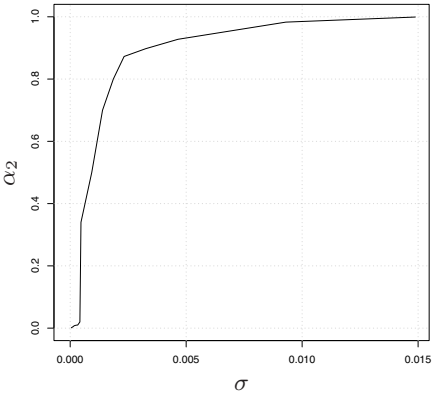

Fig. 1. Type II error probability α_2 for $\mathfrak{T}^{BC}$ as a function σ

type II error probability α_2 was estimated experimentally by executing $\mathfrak{T}^{BC}$ for random equal and unequal inputs to the S-box. We performed that for several noise amplitudes σ. The results can be found in Figure 1. Though this cannot be seen as a complete characterization of $\mathfrak{T}^{BC}$, the figure is meant to illustrate the intuition behind the multiple-differential collision detection methods.

4.2 Reference Figures for CPA

We compared the efficiency of MDCA with binary and ternary voting to the Hamming-weight based CPA [5]. The Hamming-weight power consumption model is sound for the implementation in question, since the register is first set to zero and then re-written with the target byte value. CPA was applied to the same simulated traces with the same noise amplitudes as MDCA. The number of measurements needed by CPA is denoted by C_{CPA}.

For our comparison, it was assumed that traces for all 16 S-boxes in the first round are acquired within one measurement. This is very similar to MDCA based on linear key recovery considered in this paper: The traces corresponding to the 16 S-box calculations in the first round are acquired at a time in the online stage for binary voting and ternary voting with profiling.

The number of measurements needed for CPA can be potentially reduced if guessing entropy is allowed in the offline stage of CPA. To treat this point, we assumed that CPA is successful, if it returns a correct 8-bit key chunk with probability 0.5. At the same time, it was assumed for all collision attacks that the needed success probability of the complete attack is $q = 0.5$. That is, a collision attack on AES is successful, iff it returns the correct 16-byte key with probability 0.5.

Note that power consumption models are also important for collision attacks. The right choice of a power consumption model allows the attacker to perform binary comparison more efficiently. In this paper, the consideration was restricted to the Euclidean distance of two vectors. However, other binary comparison

tests can turn out to be more consistent with the power consumption of other implementations.

4.3 Online and Profiling Complexity of MDCA

In this subsection, C_{online} and $C_{\text{profiling}}$ for MDCA based on binary voting and ternary voting both with and without profiling are experimentally derived for the given implementation. The estimations are performed for the linear key recovery method with $\gamma = 6$.

Table 1. C_{online} against different values of σ for $\mathfrak{T}^{BV}$, $\mathfrak{T}^{TV}$ without profiling, $\mathfrak{T}^{TV}$ with profiling and C_{CPA}

$10^3\sigma$	0.46	0.93	2.32	3.25	4.65	6.97	9.30	11.62	13.95
C_{online}, $\mathfrak{T}^{BV}$	60	192	276	468	960	1290	1872	2976	4242
C_{online}, $\mathfrak{T}^{TV}$ w/o profiling	80	390	2605	5200	10640	23840	42320	66080	95200
C_{online}, $\mathfrak{T}^{TV}$ with profiling	6	6	6	6	6	18	30	60	120
C_{CPA}, HW based CPA	163	349	1645	4192	6912	15676	26341	39348	56025

Binary Voting. Figure 2 and Table 1 give experimental values of C_{online} for the binary voting test in a range of noise amplitudes. The values of t have been chosen that minimize the resulting number of traces needed. If σ' is the noise amplitude to be attained by averaging and σ is the given noise level, then one has to average about $t = (\sigma/\sigma')^2$ times. Thus, $C_{\text{online}} \approx \gamma \frac{\sigma^2}{\sigma'^2} M^{BV}(\sigma')$. The results demonstrate that binary voting is well-suited for our implementation providing an advantage of factor 2.7 to 13.2 for a wide range of σ.

Ternary Voting without Profiling. Figure 3 and Table 1 give concrete values of C_{online} in this case for a range of noise amplitudes. Values of t were chosen that minimize C_{online}. The performance of the ternary voting test without profiling is comparable to CPA. However, ternary voting without profiling does not exhibit any advantages over CPA in terms of measurement complexity.

Ternary Voting with Profiling. For a given σ, the attacker can reduce t which leads to a linear decrease of C_{online} and to a considerable growth of $C_{\text{profiling}}$ due to the slope of M^{TV} as a function of the noise amplitude (see Figure 3 for this dependency). We assumed that $\leq 10^6$ measurements in the profiling stage are feasible. To obtain the lowest possible online complexity within this bound on the profiling complexity, we chose t that minimizes C_{online} with $C_{\text{profiling}} \leq 10^6$ for each interesting value of σ. The resulting values of C_{online} and $C_{\text{profiling}}$ are depicted in Figure 4.3. The values of C_{online} can be also found in Table 1. Note that there is a wide spectrum of parameter choices: If there are more severe limits on $C_{\text{profiling}}$, then t and C_{online} increase. And the other way round: If the attack scenario admits for higher values of $C_{\text{profiling}}$, C_{online} can be further reduced.

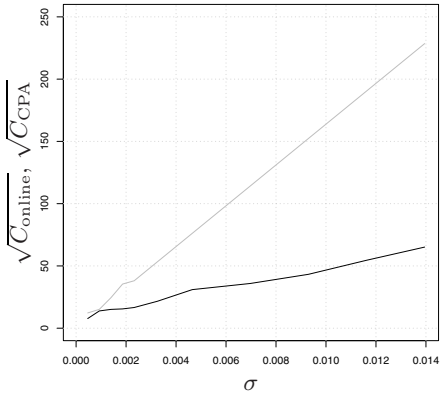

Fig. 2. Binary voting test against CPA: C_{online} (black line) and C_{CPA} (grey line) as functions of σ

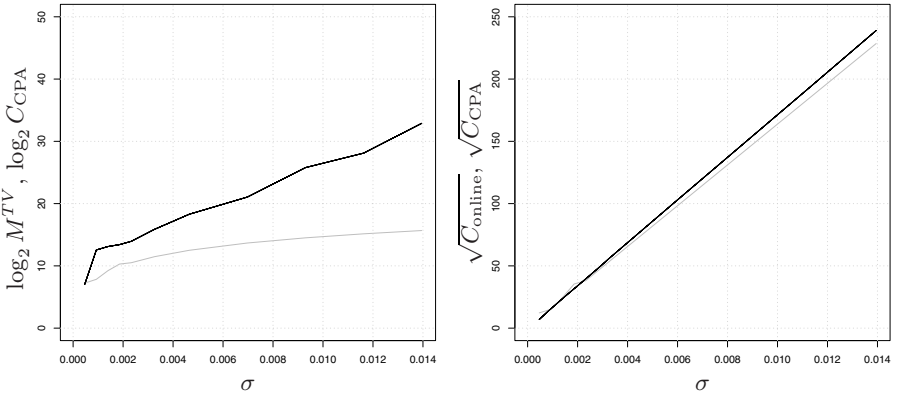

Fig. 3. Ternary voting test without profiling against CPA: $M^{TV}(\sigma)$ (on the left, black line) and C_{online} (on the right, black line) as well as C_{CPA} (both graphics, grey lines) as functions of σ

The complexity estimations for ternary voting were performed under the assumption that the attacker is able to acquire the reference traces for all S-boxes in each of the 10 AES rounds at a time. If one deals with a short-memory oscilloscope, $C_{\text{profiling}}$ increases in a linear way with respect to the decrease of the available memory volume. However, only measurements for the first round are needed for the target traces, if the linear key recovery is used.

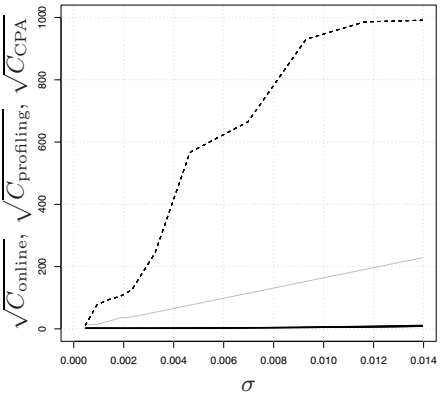

Fig. 4. Ternary voting test with profiling: C_{online} (solid black line), $C_{\text{profiling}} \leq 10^6$ (dashed black line) and C_{CPA} (solid grey line) as functions of σ

5 Conclusions and Outlooks

In this paper two statistical techniques - binary and ternary voting - allowing to safely detect collisions even in the presence of considerable noise have been proposed. An AES hardware implementation with its accurately simulated power consumption has been taken as an example to demonstrate the power of the methods. This also enables us to obtain a clear dependency of the attack efficiency from the noise amplitude in a wide range of values and to soundly compare the multiple-differential techniques with CPA for the same implementation.

The binary voting method combined with linear key recovery is well applicable to AES being 2.7 to 13.2 times more efficient than CPA in terms of measurement complexity for our implementation in the explored range of noise amplitudes. Ternary voting combined with linear key recovery and profiling needs only about 6 online measurements for the range of noise amplitudes where CPA requires from 163 to 6912 measurements for the same implementation.

Techniques similar to the ones described in this work might turn out applicable to other symmetric constructions such as stream ciphers or message authentication codes and asymmetric constructions such as digital signature schemes. There can be also some potential in using MDCA-like methods to overcome certain random masking schemes for block ciphers.

Acknowledgements. The author would like to thank Christof Paar, Emmanuel Prouff and Francesco Regazzoni for fruitful discussions as well as the anonymous referees for their constructive comments.

References

1. 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)
2. Bogdanov, A., Pyshkin, A.: Algebraic side-channel collision attacks on AES, `http://eprint.iacr.org/2007/477`
3. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
4. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
5. 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)
6. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Investigations of power analysis attacks on smartcards. In: Smartcard 1999, USENIX Association, pp. 151–161 (1999)
7. Örs, S.B., Gürkaynak, F., Oswald, E., Preneel, B.: Power-analysis attack on an ASIC AES implementation. In: ITCC 2004, pp. 546–552. IEEE Computer Society, Los Alamitos (2004)
8. Schramm, K., Wollinger, T.J., Paar, C.: A new class of collision attacks and its application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
9. Wiemers, A.: Collision Attacks for Comp128 on Smartcards. In: ECC-Brainpool Workshop on Side-Channel Attacks on Cryptographic Algorithms, Bonn, Germany (2001)
10. 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)
11. 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)
12. Biryukov, A., Bogdanov, A., Khovratovich, D., Kasper, T.: Collision attacks on Alpha-MAC and other AES-based MACs. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 166–180. Springer, Heidelberg (2007)
13. 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)
14. 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. 51–62. Springer, Heidelberg (2003)
15. Archambeau, C., Peeters, E., Standaert, F.X., Quisquater, J.J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
16. Dolphin: Description of the standard cells for the process IHP 0.25 μm. ViC specifications. SESAME-LP2. version 1.1 (2005)
17. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks and Countermeasures for Cryptographic Smart Cards: Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)
18. Lemke-Rust, K.: Models and Algorithms for Physical Cryptanalysis. PhD thesis, Ruhr University Bochum (2007)