Improved Algebraic Side-Channel Attack on AES

Mohamed Saied Emam Mohamed [1], Stanislav Bulygin [2], Michael Zohner [2], Annelie Heuser [2] and Michael Walter [1]

1: Technische Universität Darmstadt, Integrated Circuits and Systems Lab, Hochschulstraße 10, 64289 Darmstadt, Germany
e-mail: mohamed@cdc.informatik.tu-darmstadt.de, michael.walter@swel.com
2: Center for Advanced Security Research Darmstadt (CASED), Mornewegstraße 32, 64289 Darmstadt, Germany
e-mail: {stanislav.bulygin,michael.zohner,annelie.heuser}@cased.de

Abstract. In this paper we present improvements of the algebraic side-channel analysis of the Advanced Encryption Standard (AES) proposed in [9]. In particular, we optimize the algebraic representation of AES and the algebraic representation of the obtained side-channel information in order to speed up the attack and increase the success rate. We study the performance of our improvements in both known and unknown plaintext/ciphertext attack scenarios. Our experiments indicate that in both cases the amount of required side-channel information is less than the one required in the attacks introduced in [9]. Furthermore, we introduce a method for error handling, which allows our improved algebraic side-channel attack to escape the assumption of an error-free measurement and thus become applicable in practice. We demonstrate the practical use of our improved algebraic side-channel attack by inserting predictions from a single-trace template attack.

Keywords: Algebraic Side-Channel Attack, AES, Error Tolerance, SAT solving, template attack

1 Introduction

When implementing a cryptographic algorithm, such as a block cipher, it is not only important to verify that the algorithm itself is secure against cryptanalysis, but also that an implementation of an algorithm does not leak information about the processed data. Attacks that exploit such leaked information in order to recover a cryptographic secret, are called side-channel attacks. A recently introduced type of side-channel attack, the so-called *algebraic side-channel attack (ASCA)* [9, 10], combines side-channel attacks with algebraic techniques, i.e. algebraic system solving.

Adding information from a side-channel attack, into an algebraic system allows an attacker to recover the secret key even if the number of traces is too low for a statistical attack like the template attack or the DPA. Since the

algebraic representation is adaptable and descriptive, information about any processed intermediate value can be inserted into the algebraic system in order to enhance the key recovery process. Accordingly, the ASCA is a very strong side-channel attack, when a profiling based attacker is assumed. However, the applicability of ASCA suffers from the necessity of correct information [9]. If erroneous information is inserted into the algebraic system, it is not correctly solvable. This strongly limits the practical use of ASCA, since errors in the field of side-channel analysis are common, especially if an attacker is assumed that is provided with only one trace in the attack phase. Recently, there has been a contribution, which is able to deal with erroneous information [6]. However, the approach still restricts errors to a certain degree and amount and requires an immense computation time, which limits its practical use.

In this work we propose a more practical algebraic side-channel attack, the so called *improved algebraic side-channel attack (IASCA)*. The contribution of IASCA is twofold. First, we reduce the information, which is required for solving the algebraic system by finding a more efficient algebraic representation. Secondly, we introduce a method for dealing with erroneous information, thus increasing the resistance of IASCA towards errors. Both improvements contribute to the error tolerance of IASCA by decreasing potential points of failure on the one hand, and increasing the error resistance without assuming a bounded error as in [6] on the other hand. Finally, we demonstrate the practical applicability of IASCA on the block cipher AES-128.

The paper is organized as follows. In Section 2 we give some preliminaries and related work. We describe our improved algebraic representation and give experimental results in Section 3. Section 4 outlines our approach for error tolerance in the improved side-channel attack and studies the performance of our attack in case of erroneous measurements. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Algebraic Cryptanalysis

In algebraic cryptanalysis the inputs, outputs, and a secret of a cryptographic primitive like, as in our case, AES [5] are related by a system of (non-linear) equations, usually over $GF(2)$, the field of the two elements 0 and 1. After setting the corresponding variables to the values of a known plain-/ciphertext pair, the system is attempted to be solved and the secret key is determined by the solution. However, finding a solution for such a system is non-trivial and there are several techniques available to be employed. The one we mainly focus on is SAT solving. The area of SAT solving has received a lot of research in the past decades and is one of the most efficient techniques available for algebraic cryptanalysis. In this setting, the system of equations is translated into a set of clauses constituting an equivalent satisfiability (SAT) problem instance, which is then fed into a SAT solver, e.g. CryptoMiniSat [11] (see e.g. chapter 13 of [1]). A clause is a set of literals, which are either positive, i.e. a variable, or negative, i.e. a negated variable, and related by disjunctions. The clauses are related by

conjunctions. A SAT instance taking on this form is said to be in Conjunctive Normal Form (CNF). The conversion of a solution of the SAT problem into a solution for the algebraic problem is straightforward. Another approach to solve the system of equations is the Gröbner bases technique as described in [1]. However, so far this has not been as efficient as SAT solving for attacking symmetric cryptosystems. In contrast, computing Gröbner bases provides a compact description of all the possible solutions to a system of equations. So, we apply Gröbner bases techniques over subsets of equations to improve SAT solving in this context as explained in the next section.

So far, attacking the full AES with algebraic techniques was not successful, the reason being that the resulting algebraic system (and its equivalent SAT problem) does not have enough information to be solved efficiently. On the other hand, we are able to enhance our algebraic system by additional, e.g. side-channel, information, by representing this information algebraically. In [9] Renauld *et al.* presented an implementation of ASCA employing the Hamming weight based leakage model, and analyzed the amount of side-channel information necessary to make the attack practically feasible. We build upon their work by reducing the amount of necessary information and introducing error tolerance.

When working with side-channel information one has to be careful in specifying the implementation under attack as the leakage model depends heavily on it. The focus of our study is on AES-128. AES-128 is a widely used block cipher [5], it processes 128 bit blocks with 128 bit keys. AES is a substitution permutation network having 10 rounds. The round function of AES consists of the key addition (where for each round a different key is used according to the key scheduling algorithm), substitution layer composed of 16 8-bit S-boxes, and the linear diffusion layer that is a composition of ShiftRows and MixColumns operations. See [5] for a detailed description of the algorithm. Since we are building on previous work we target AES-128 implemented as Renauld *et al.* did in [9]. We do want to point out, however, that our approach is applicable to other implementations as well. Since Renauld *et al.* elaborate sufficiently on the target operations and their potential leakages, we constraint ourselves to a brief summary (and refer to [9] for further details): we assume potential leakages at each byte of the state before and after the substitution layer and 52 additional potential leakages during the MixColumn operation, resulting in $2 \cdot 16 + 52 = 84$ leakages per round and $84 \cdot 9 + 32 = 788$ leakages over all (since the last round does not contain the MixColumn operation). Note that we do not target operations in the key scheduling algorithm.

2.2 Side-Channel Analysis

In algebraic side-channel analysis we assume the attacker is provided with a sufficient number of traces for profiling, whereas he is limited to a small number of traces (e.g., only one trace) in the attack phase [3, 7]. Note that in this scenario the attacker is not able to gain a secret key only with side-channel analysis.

However, the acquired side-channel information is sufficient to provide the algebraic system with information about Hamming weights of several internal values (e.g. Hamming weight of S-box input/output) in order to recover the secret key. In the following, we briefly sketch the idea of template attacks [3], which is one of the most common profiled side-channel attacks. In the profiling phase of a template attack the adversary is provided with power trace vectors $\{l^i_{c,v}\}^{N_1}_{i=1}$ for each class c of each attackable operation v. For example, if the adversary is interested in the Hamming weights ($c \in \{0, \ldots, 8\}$) of one S-box input and output for the first three rounds ($|v| = 6$), he requires $6 \cdot 9$ different leakage vectors each with N_1 power traces. Template attacks rely on a multivariate Gaussian noise model, accordingly the power trace vectors are considered to be drawn from a multivariate normal distribution. More precisely,

$$\mathcal{N}(l_{c,v}|\mu_{c,v}, \Sigma_{c,v}) = \frac{1}{(2\pi)^{N_t} \ |\Sigma_c|^{1/2}} e^{-\frac{1}{2}(l_{c,v}-\mu_{c,v})^T \Sigma^{-1}_{c,v}(l_{c,v}-\mu_{c,v})},$$

where N_t is the number of interesting points within the measured trace for each operation. Accordingly, the construction of these templates is based on the estimation of the expected values $\hat{\mu}_{c,v}$ as well as the covariance matrix $\hat{\Sigma}_{c,v}$.

In the attack phase the attacker measures a trace vector $\{l^i_{c,v}\}^{N_2}_{i=1}$ (with unknown c) and uses the maximum–likelihood estimator [3], given by

$$\mathcal{L}_{c^*,v} = \prod_{i=1}^{N_2} P(l^i_{c,v}|C) = \prod_{i=1}^{N_2} \mathcal{N}(l^i_{c,v}|\mu_{c^*,v}, \Sigma_{c^*,v}), \tag{1}$$

for each possible c^* regarding the operation v. The adversary then chooses the c^* which maximizes $\mathcal{L}_{c^*,v}$ for each operation v. Note that, in case $N_2 = 1$ he only gains the probability according to $\mathcal{N}(l_{c,v}|\mu_{c^*,v}, \Sigma_{c^*,v})$ for each class c^* and v, which may not be decisive for only one c^*.

3 IASCA: Improved Algebraic Representation of ASCA

3.1 Approach

While Renauld *et al.* successfully demonstrated the feasibility of ASCA, its potential was not fully explored. We demonstrate in this section that it is possible to reduce solving times and the amount of necessary information, thus increasing the practicability of the attack, simply by tweaking the algebraic representation.

In order to improve the work [9] we optimize the representation of the SAT problem to reduce solving times. At first glance it is not immediately clear what characteristics a "good" representation should have. An important characteristic of a SAT instance appears to be the size of the problem, not only in the number of variables, but also the number of clauses. However, considering the way SAT solvers work - building a search tree, exploring it while learning conflict clauses and trying to cut branches efficiently ([1]) - the average length of clauses is also

a suitable heuristic measure, since short clauses may produce conflicts sooner than long clauses. It follows that introducing short clauses might improve the representation, even though increasing the size of the instance, thus yielding a trade-off to the first characteristic.

First, we targeted the representation of the S-box operation. Renauld *et al.* use a set of clauses which enumerates all possible input and output values resulting in 2048 clauses, each of length 9. We followed a different approach using Gröbner basis techniques to derive 8 high-degree equations (degree 7) of the form $y_i = f(x)$ for $i \in \{1, \cdots, 8\}$, where $x = (x_1, \ldots, x_8)$ denotes the input of an S-box, and y_i the i-th output bit of an S-box. We say that these equations provide an *explicit representation*, since they explicitly describe the dependence of the output on the input. Converting these equations to a set of clauses using PolyBoRi's CNF converter [2] resulted in 946 clauses with average length of 7.22. Table 1 lists the numbers of clauses occurrences of certain lengths in both representation.

Length	6	7	8	9
Renauld *et al.*	0	0	0	2048
Explicit representation	77	590	270	9

Table 1. Comparison of S-box clause lengths of Renauld *et al.* and ours deducted from the explicit representation

Furthermore, we tried to exploit the restrictions the S-box operation has on the Hamming weight of its input and output. Specifically, we included short clauses for the case that the input and output of a certain S-box are known.

First note that for all possible input/output pairs of the S-box there are 47 possible corresponding Hamming weight pairs (e.g. the input/output pair $x = (1, 0, 0, 0, 0, 1, 1, 0)$ and $y = (1, 1, 1, 1, 0, 1, 1, 1)$ corresponds to the Hamming weight pair $(3, 7)$). For each Hamming weight pair we denote its *weight* as the number of input/output pairs that correspond to that pair. Table 2 shows all Hamming weight pairs and their weights. Also note that the Hamming weight of a byte x can be represented as a set of equations over GF(2) in the variables denoting the bits of x. To see this, let $HW(x) = w$, where $HW(\cdot)$ denotes the Hamming weight function. It holds that $HW(x) = w$ iff every $w+1$ sized subset of the bits of x contains at least one 0 and every $8-w+1$ sized subset contains at least one 1. These two conditions can be enforced using the following equations for all such subsets:

$$\prod_{x_i \in X_{w+1}} x_i = 0 \tag{2}$$

$$\prod_{x_i \in X_{9-w}} (x_i + 1) = 0 \tag{3}$$

where X_a is a subset of the variables representing the bits of x with $|X_a| = a$.

For each of the possibly occurring Hamming weight pair (i.e. with weight > 0) we consider the set of equations defining the S-box operation and includ the equations specifying the corresponding Hamming weights of the input and output. From this set we derive a set of short equations, again using Gröbner bases techniques, and converted them to clauses using the PolyBoRi's CNF converter. While this worked very well for low weight Hamming weight pairs, it resulted in rather long clauses for pairs with a weight larger than 7. To avoid long clauses we used a different approach for these Hamming weight pairs. Instead of computing Gröbner bases, we considered all possible clauses of length n with n being reasonably small (i.e. $n \leq 3$) and checked, which ones are satisfied by all input/output pairs that correspond to a certain (high weight) Hamming weight pair. Note, that there are exactly $\binom{16}{n} \cdot 2^n$ distinct clauses of length n, since every clause can contain n out of the 16 variables (8 input and 8 output bits) and every variable can appear as positive or negative literal. So this exhaustive search is computationally feasible for small n. Table 3 shows the number of clauses of length 1, 2, and 3 included for each high weight Hamming weight pair. Finally, this yields a set of very short clauses for each Hamming weight pair, which is added to the SAT instance in case the Hamming weights of an input/output pair of an S-box is known due to side-channel information.

in \ out	0	1	2	3	4	5	6	7	8
0	0	0	0	0	1	0	0	0	0
1	0	0	2	0	1	3	2	0	0
2	0	2	3	8	5	4	4	2	0
3	1	1	4	17	16	10	5	2	0
4	0	3	9	11	21	16	9	1	0
5	0	1	7	10	19	14	3	2	0
6	0	0	3	7	5	8	4	0	1
7	0	1	0	2	2	1	1	1	0
8	0	0	0	1	0	0	0	0	0

Table 2. Weights of Hamming weight pairs

Next, let us consider leakage of Hamming weights in the MixColumn operation. The implementation that we study employs the so-called `xtime` operation on a byte, as defined in [5], as a subroutine of the MixColumn operation. Renauld *et al.* represent this operation by 8 linear equations – one for each output bit depending on the 8 input bits. The equations have a length of 3 or 5 (including the output variables). From section 4.2.1 in [5] we obtained the following – significantly shorter – representation with input x and output y:

$$y_i = x_{i-1} + k_i \cdot x_7$$

Weight \ Length	1	2	3
(2, 3)	2	98	0
(3, 3)	0	20	788
(3, 4)	0	7	604
(3, 5)	0	36	1270
(4, 2)	0	57	0
(4, 3)	0	24	1114
(4, 4)	0	0	212

Weight \ Length	1	2	3
(4, 5)	0	12	673
(4, 6)	2	100	0
(5, 2)	1	99	0
(5, 3)	0	23	1157
(5, 4)	0	4	499
(5, 5)	0	12	838
(6, 3)	1	107	0
(6, 5)	1	100	0

Table 3. Number of clauses of length 1, 2, and 3 included for each high weight Hamming weight pair

where $k_i = 1$ for $i \in \{1, 3, 4\}$, $k_i = 0$ for all other i, and indices are computed modulo 8. In a similar fashion as described above for the S-box operation, we considered all possible pairs of Hamming weights of the input and output of the `xtime` operation, added the Hamming weight specifying equations to the representation above and computed Gröbner bases to obtain short equations. These were again translated to clauses to add to the SAT instance in case the Hamming weight of the input and the output of the operation are known. Note that for the `xtime` operation no exhaustive search for short clauses was necessary.

3.2 Experimental Results

In this subsection, we present experimental results which demonstrate the performance of our improved algebraic side channel attack. We used the same assumptions as in [9]. For the experiments presented here, we assume that all Hamming weights are correct. For these experiments we used both the Java implementation of ASCA by Renauld [8] and our modified implementation of IASCA. We used the SAT solver CryptoMiniSat [11] to solve CNF systems produced by ASCA and IASCA. For each attack scenario each result is obtained as an average over 100 runs. Furthermore, for all experiments presented here we set 100 seconds as a time limit. We run all the experiments on a Sun X4440 server, with four "Quad-Core AMD Opteron^{TM} Processor 8356" CPUs and 128 GB of main memory. Each CPU is running at 2.3 GHz.

Tables 4 and 5 report our experiments for several attack scenarios where a plaintext/ciphertext is known and where plaintext/ciphertext is unknown, respectively. In these experiments we compared the results obtained by using Renauld's ASCA and our IASCA with the results reported in [9]. Our comparison is based on the required Hamming weights to recover the secret key with a rate of success higher than 90% in less than 100 seconds.

Table 4 shows that in case of consecutive Hamming weights, IASCA needs at most Hamming weights of two consecutive internal rounds ($R_2 - R_8$) in order to find the secret key. In this scenario it can be infered that the SAT solver uses the

system constructed by IASCA to recover a value of the round key between the selected two consecutive rounds. Afterwards, it uses the key schedule relations and the known plaintext/ciphertext to retrieve the correct value of the secret key. In the scenario of randomly distributed known Hamming weights over all rounds, IASCA requires 394 HW. Moreover, IASCA can recover the secret key using only 68 random Hamming weights from the first round with rate of success greater than 95%.

Attack \ Model	Renauld *et al.*	ASCA	IASCA
Consecutive HW	3 rounds (256 HW)	3 rounds (256 HW)	2 rounds (168 HW)
Random HW	> 756	551 (70%)	394 (50%)

Table 4. Results of known plaintext/ciphertext attack Scenarios.

Table 5 shows that for IASCA only 184 consecutive Hamming weights are sufficient to recover the secret key in an unknown plain-/ciphertext scenario. More precisely, as mentioned above, we need all the Hamming weights of two consecutive rounds R_i and R_{i+1} (168 HW) to recover the value of the round key k_i and the Hamming weights of the input state of round R_{i+2} (16 HW) to determine the correct value of k_i in case we have multiple values. However, given only 2 consecutive rounds yield a 70% rate of success using IASCA. In case of a random positions for Hamming weights, where the known Hamming weights are distributed randomly over all potential states, IASCA needs only 472 Hamming weights distributed randomly over all rounds (60%) with rate of success $\geq 80\%$.

Attack \ Model	Renauld et al.	ASCA	IASCA
Consecutive HW	3 rounds (256 HW)	3 rounds (256 HW)	< 3 rounds (184 HW)
Random HW	> 756 HW	551 HW (70%)	472 HW (60%)

Table 5. Results of unknown plaintext/ciphertext attack Scenarios.

4 Error Tolerance

4.1 Error Tolerance Classes

Most contributions on algebraic side-channel attacks, for instance [9], assume an error free set of Hamming weights. However, due to several kinds of noise (e.g., electronic noise, quantization noise, switching noise) the emitted side-channel leakage may lead to erroneous Hamming weight predictions [4]. If such an erroneous Hamming weight prediction is inserted into the algebraic system, the SAT solver will not be able to determine a correct solution.

Until now only one contribution [6] deals with erroneous predictions by using pseudo-Boolean optimization instead of SAT solvers. In order to verify their approach, the authors built a system for the Keeloq block cipher, for which they were able to solve the system in 3.8 hours supporting an error rate of 18.8%. Also, an evaluation on AES was planned, but until now only the system's size was given. The main drawbacks of the approach are the required time, which grows super-linearly with the error rate, and the assumption that the distance of an erroneous Hamming weight to the correct Hamming weight is only ± 1. The assumption on the error restricts the practical use, since in profiling based side-channel attacks errors of ± 2 and higher are possible and, in case of a device with a higher noise level, rather common (cf. Table 6).

In the following, we extend the IASCA with the capability of dealing with errors. The idea behind our approach is to extend the algebraic representation of AES until it describes a range of Hamming weights, which includes the correct Hamming weight with high certainty. So far, we have built the clauses by assuming the equation $HW(x) = a$ holds. This equation can also be expressed as two inequalities, where one represents $HW(x) \leq a$ and the other $HW(x) \geq a$. Both inequations can be described by a set of clauses (cf. equations (2) and (3)). We can represent the equation by combining these clauses and thus obtain more information. However, we can also change the boundary a of each inequality in order to describe a wider range of Hamming weights. Thus, we are not restricted in the severity of the error and can assume arbitrary errors, which is a more practical assumption with regard to side-channel analysis than the one used in [6].

For our experiments we define notation to describe the severity of an error, i.e. the range of Hamming weights we assume. We define five error classes e_z, for $z \in [0, 4]$, where z denotes the number of additional consecutive Hamming weights. For instance, e_0 assumes only one Hamming weight h is possible, e_1 considers two consecutive Hamming weights $[h, h+1]$, e_2 considers three consecutive Hamming weights $[h, h+2]$, and so on.

In general it holds that the more Hamming weights we assume to be possible, the less information is added to the system. However, by choosing a larger error, we can be more certain that the inserted clauses describe the correct Hamming weight. Thus, a trade-off has to be found, which on the one hand minimizes the likelihood of erroneous information and on the other hand maximizes the information gain from an insertion into the algebraic system. To determine the

likelihood of erroneous information we use the certainty vector of the template attack (c.f. Subsect. 2.2) in our experiments. We decrease the probability of adding false information to the system by adding Hamming weights to the range until the certainty of covering the correct Hamming weight rises above a specified threshold.

Note that we require assumed Hamming weights to be consecutive in order to find a maximal informative set of polynomial equations that describe the Hamming weights. If the assumed Hamming weights were not consecutive, a bigger error class would have to be chosen such that all Hamming weights are included. However, if the Hamming weight leakage model correctly describes the power consumption of a device, the most likely HW will form an interval when grouped by their probability.

However, this approach has an impact on the improvement we proposed in Section 3. Since the Hamming weights of the input and the output of the S-boxes might be incorrect, the additional short clauses have to be adjusted accordingly. Say, the Hamming weight predictions of the input/output pair (x, y) is (w_x, w_y) and both are assumed to be in the error class e_{z_1} and e_{z_2} where $0 \leq z_1, z_2 \leq 4$, i.e. the correct Hamming weight of x (w_x) is in $[h_1, h_1 + z_1]$ and the one of y (w_y) is in $[h_2, \cdots, h_2 + z_2]$. In this case, instead of adding the short clauses for the pair (w_x, w_y), we added the intersection of the sets for all Hamming weight pairs in $\{h_1, \cdots, h_1 + z_1\} \times \{h_2, \cdots, h_2 + z_2\}$, thus making sure that all potential Hamming weight pairs satisfy the clauses. Naturally, the number of clauses decreases with increasing error.

4.2 Experiments and Results

In the following, we evaluate the performance of our error tolerant IASCA by conducting a template attack on a microcontroller and using the recovered Hamming weights in order to recover the key of an AES-128 encryption. We use an ATMega 2561 microcontroller with an 8-bit register as a target device,since the Hamming weight leakage model adequately describes its power consumption. For our measurement setup we connected the microcontroller to an external power supply and an external frequency generator and measured its power consumption with a PicoScope 6000. We implemented AES, whose start we marked by a rising trigger on an external pin, and performed a template attack using a template base of 5000 measurements. The templates were created for the S-box input and output of each round as well as the intermediate values of MixColumns, as described in Section 3. We utilized a single-trace template attack (i.e. $N_2 = 1$), and, in order to achieve a decisive result about the precision, repeated the attack 2000 times with varying plaintext/key pairs. We gained the likelihood $\mathcal{L}_{c^*,v}$ (cf. Eq. 1) according to each possible Hamming weight class c^* and each operation v (e.g. v = S-box input of the first round for byte 1).

The accuracy of the template attack, computed for each S-box input, S-box output, and MixColumns intermediate value of all 2000 attack traces is depicted in Table 6. The template attack predicted the correct Hamming weight in 28% of all corresponding values. Moreover, one can see that the error is not restricted to

only ± 1 as considered in [6]. However, when we obtain the results of the template

error class	e_0	e_1	e_2	e_3	e_4
Occurence	28%	44%	24%	4%	0%

Table 6. Occurences of the error classes in the prediction of our template attack.

attack we do not know the dedicated error class of the prediction. Therefore, we have to estimate the error class of the prediction in order to generalize the clauses such that the correct Hamming weight is described. Since we have a likelihood $\mathcal{L}_{c^*,v}$ for all Hamming weight predictions $c^* \in \{0, \ldots, 8\}$, we can introduce a certainty threshold T. More precisely, we determine $C \subseteq \{0, 1, ..., 8\} =: HW$ such that

$$\min_C \{|C| : \sum_{c^* \in C} \mathcal{L}_{c^*,v} \geq T \\ \wedge \forall c^* \in C, c^{*'} \in (HW \backslash C) : \mathcal{L}_{c^*,v} \geq \mathcal{L}_{c^{*'},v}\} \ . \tag{4}$$

Note that the error class for a set of assumed Hamming weights C is $e_{(|C|-1)}$. Roughly speaking, we increase the error class, i.e. the size of the interval, until the certainty that the correct Hamming weight is contained in the interval exceeds T.

We computed the occuring error classes for $T = \{80, 85, 90, 95, 98, 99\}$ for each S-box input, S-box output, and MixColumns intermediate value of all 2000 attack traces for the single-trace template attack. The results are depicted in the side-channel attack part of Table 7. We computed the accuracy for the thresholds by verifying whether $c \in C$, i.e., whether the correct Hamming weight is contained in the error class, determined by T.

As expected, the higher the threshold, the less certain the predictions, i.e., the higher the assumed error classes. However, we only obtain an accuracy of 100% with $T \geq 95$. Thus, if we use a threshold $T \leq 90$ it is possible that $c \notin C$, i.e. that the information we include in our algebraic system might be false. Finally, we inserted the predictions of the template attack into the error tolerant IASCA. Note that the threshold acts as a trade-off between the certainty of the template attack on the one hand and the complexity of the algebraic attack on the other. In order to provide the algebraic system only with correct information, we chose a threshold at least 95. The IASCA solved the system for $T = 95$ within $84s$ and required the Hamming weight information for the first three rounds as well as the plain-/ciphertext pair. For $T = 98$ we were only able to solve the algebraic system in half of the cases while requiring $100s$ in average given the Hamming weight information of all rounds and the plain-/ciphertext pair. When we used the error classes for $T = 99$, we were not able to solve the system, even though we added all information.

Side-channel attack results							Algebraic results	
T	e_0	e_1	e_2	e_3	e_4	acc	Rounds	Time (s)
80	35%	47%	18%	0%	0%	82%	R_1	2s
85	23%	64%	13%	1%	0%	94%	R_1	3s
90	14%	45%	36%	5%	0%	99%	R_1, R_2	3s
95	9%	29%	44%	18%	0%	100%	R_1-R_3	84s
98	4%	18%	31%	43%	4%	100%	all	100s (50%)
99	0%	10%	23%	47%	20%	100%	all	-

Table 7. Percentage of error classes given certainty threshold T of template attack

Moreover, we are also interested in the performance of IASCA for the given error classes for $T = \{80, 85, 90\}$. Thus, we eliminated the errors in the predictions, such that the distribution of the error classes was maintained, and inserted them into the algebraic system. As depicted in Table 7, the IASCA was able to recover the correct key while only requiring the information of the first round for the error classes of $T = \{80, 85\}$. For $T = 90$ we could solve the system in 3 seconds using only the Hamming weight information of round 1 and round 2.

Note that even though we were not able to solve the algebraic system for all thresholds, the results still seem promising. In case of $T = 80$, we correctly describe only 35% of all Hamming weights and were still able to solve the algebraic system in only two seconds using less information than [9] and [6]. For $T = 90$, the IASCA successfully solved the algebraic system using only the first two rounds within three seconds, even in the presence of e_3, which has not been solvable up to now. For $T = 95$, IASCA needs 84 seconds, but we assume an error rate of 91% with a presence of 18% of e_3. The limits of our approach are demonstrated by $T \geq 98$, where IASCA is not always able to solve the system in a reasonable time. However, for $T \geq 98$ we assume hardly any Hamming weights to be correct and also deal with $\geq 40\%$ of e_3 and even e_4.

5 Conclusions and Future Work

In this contribution we presented a practical algebraic side-channel attack, IASCA. The enhancements have been derived in two ways:

First, we reduced the information required for solving the algebraic system. Compared to [9] we could solve the algebraic system by only requiring 2 consecutive rounds or random 394 Hamming weights for known plain- and ciphertext. Furthermore, if the plain- and ciphertext are unknown we achieved a reduction of two rounds of required Hamming weights.

Secondly, we considered the application of IASCA under the assumption of erroneous side-channel information. We therefore conducted a single-trace template attack and analyzed the error distribution in detail, in order to obtain a more practical view on the error rate. We then conducted IASCA on the erroneous predictions of the template attack. IASCA was able to solve the algebraic

system in a few seconds, even though we provided the system with 91% erroneous predictions. Also, we were able to cope with predictions that differ from the correct Hamming weight by an arbitrary distance. Thus, we outperformed [6] in the number of errors, the restrictions on errors, and required information.

In future work, we will try to decrease the size of clauses and increase the error handling in order to further enhance the error tolerance of IASCA. Also, we will evaluate a new approach, which allows us to solve hard instances by guessing the Hamming weight at certain intermediate values. Preliminary experiments indicate that we might be able to solve the algebraic system for threshold $T = 98$ in 100% of the cases by guessing the Hamming weights of only seven intermediate values. Thus, we would be less dependent on accurate side-channel information, which would further extend the applicability of IASCA.

Acknowledgments

This work is supported by the BMBF project RESIST. We would like to thank Mathieu Renauld for his useful comments on this paper and for his valuable suggestions.

References

1. Bard, G.: Algebraic Cryptanalysis. Springer (2009)
2. Brickenstein, M.: Polybori's cnf converter. https://bitbucket.org/brickenstein/polybori/overview
3. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: CHES. pp. 13–28 (2002)
4. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
5. NIST: Advanced Encryption Standard (AES) (FIPS PUB 197). National Institute of Standards and Technology (Nov 2001)
6. Oren, Y., Kirschbaum, M., Popp, T., Wool, A.: Algebraic side-channel analysis in the presence of errors. In: Mangard, S., Standaert, F.X. (eds.) CHES. Lecture Notes in Computer Science, vol. 6225, pp. 428–442. Springer (2010)
7. Rao, J.R., Sunar, B. (eds.): Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3659. Springer (2005)
8. Renauld, M.: Simulating algebraic side channel attacks ascatocnf converter. http://www.ecrypt.eu.org/tools/ascatocnf
9. Renauld, M., Standaert, F.X., Charvillon, N.V.: Algebraic side-channel attacks on the AES: Why time also matters in DPA. In: CHES. pp. 97–111 (2009)
10. Renauld, M., Standaert, F.X.: Algebraic side-channel attacks. In: Inscrypt. pp. 393–410 (2009)
11. Soos, M.: Cryptominisat 2.5.0. In: SAT Race competitive event booklet (July 2010)