Improved Algebraic Side-Channel Attack on AES

Mohamed Saied Emam Mohamed*, Stanislav Bulygin*†, Michael Zohner†, Annelie Heuser†, Michael Walter*, and Johannes Buchmann*
*Technische Universität Darmstadt, Department of Computer Science
Hochschulstraße 10, 64289 Darmstadt, Germany
email: {mohamed,buchmann}@cdc.informatik.tu-darmstadt.de, michael.walter@swel.com
†Center for Advanced Security Research Darmstadt (CASED)
Mornewegstraße 32, 64293, Germany
email: {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 [1]. 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 [1]. Furthermore, we introduce a method for error handling, which allows our improved algebraic side-channel attack to escape the assumption of an error-free environment 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.**

***Index Terms*—Algebraic Side-Channel Attack, AES, Error Tolerance, IASCA**

I. 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 the algorithm does not unintentionally leak information about the processed data. Attacks that exploit such unintentionally leaking 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)* [2], [1], combines side-channel attacks with algebraic techniques, i.e. an algebraic system.

Inserting information from a side-channel attack, into an algebraic system allows an attacker to recover the secret key even if regular side-channel attacks fail. 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. 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 [1]. If false 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 [3]. 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 an easier 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 [3] on the other hand. Finally, we demonstrate the practical applicability of IASCA on the block cipher standard AES-128.

The paper is organized as follows. In Section II we give some preliminaries and related work of what is presented in this paper. We describe our improved algebraic representation and give experimental results in Section III. Section IV outlines our approach for error tolerance in the improved side-channel attack and studies the performance of our attack in an erroneous environment. Finally, Section V concludes this paper.

II. Preliminaries

A. Algebraic Cryptanalysis

In algebraic cryptanalysis the inputs and outputs of a cryptographic primitive like, as in our case, AES [4] 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 the system is non-trivial and there are several techniques available to be employed. The one we mainly focus on is SAT solving, since it appears to be the best suitable tool for the attack. The area of SAT solving has received a lot of research in

978-1-4673-2340-6/12/$31.00

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 [5]) (see e.g. chapter 13 of [6]). 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 of the set are implicitly assumed to be 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 are Gröbner bases techniques as described in [6]. However, so far this has not been as efficient as SAT solving. In contrast, we apply Gröbner bases techniques in substeps 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) are too large to be solved efficiently. On the other hand, the algebraic representation has the advantage of being able to incorporate almost any additional information, for example side-channel information, as long as it is representable in algebraic form. In [1] 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, which is a widely used block cipher [4]. 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 S-boxes, and linear diffusion layer that is a composition of ShiftRows and MixColumns operations. See [4] 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 [1]. We do want to point out, though, 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 [1] 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 $2 * 16 + 52 = 84$ leakages per round and $84 * 9 + 32 = 788$ 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.

B. 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 [7], [8]. Note that in this scenario the attacker is not able to gain the secret key only with side-channel analysis. However, the acquired side-channel information is sufficient to provide the algebraic system with information about the Hamming weight (HW) of internal values (e.g., HW of S-box input/output) in order to recover the secret key. In order to recover the HW of internal values, we utilitze an extension of the template attack [7], which we briefly sketch in the following. In the profiling phase the adversary is provided with power trace vectors $\{l_w^i\}_{i=1}^{N_1}$ for each HW w. Template attacks rely on a multivariate Gaussian noise model, so the power trace vectors are considered to be drawn from a multivariate normal distribution. More precisely,

$$\mathcal{N}(l_w|\mu_w, \Sigma_w) = \frac{1}{(2\pi)^{N_t}\ |\Sigma_w|^{1/2}} e^{-\frac{1}{2}(l_w-\mu_w)^T \Sigma_w^{-1}(l_w-\mu_w)},$$

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 $\hat{\mu}_w$ as well as the covariance matrix $\hat{\Sigma}_w$.

In the attack phase the attacker measures a trace vector $\{l_{w^*}^i\}_{i=1}^{N_2}$ with unknown w^* and uses the maximum-likelihood estimator [7], given by

$$\mathcal{L}_w = \prod_{i=1}^{N_2} P(l_{w^*}^i|w) = \prod_{i=1}^{N_2} \mathcal{N}(l_{w^*}^i|\mu_w, \Sigma_w), \tag{1}$$

for each possible w, so $P(l_{w^*}^i|w)$ denotes the probability that l_w^i was measured while w is the HW of the internal value.

The adversary then chooses the w which maximizes $\mathcal{L}_w$. Note that, in case $N_2 = 1$ he only gains the probability according to $\mathcal{N}(l_{w^*}|\mu_w, \Sigma_w)$ for each class w, which may not be decisive for only one w.

III. IASCA: Improved Algebaic Representation of ASCA

A. Approach

While Renauld et al. successfully demonstrated the feasibility of ASCA, its potential was not fully exhausted. 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 on the work in [1] we optimized the representation of the SAT problem to reduce solving times. At first sight 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 in 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 ([6]) - the average

length of clauses is also a suitable, though simple, 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 [9] resulted in 946 clauses with average length of 7.22. Table I lists the numbers of occurrences of clauses 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 I
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 will denote its *weight* as the number of input/output pairs that correspond to that pair. Table II shows all Hamming weight pairs and their weight. 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:

$$\prod_{x_i \in X_{w+1}} x_i = 0 \quad \& \quad \prod_{x_i \in X_{9-w}} (x_i + 1) = 0 \qquad (2)$$

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 pairs (i.e. with weight > 0) we considered the set of equations defining the S-box operation and included the equations specifying the corresponding Hamming weights of the input and output. From this set we derived a set of short equations 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 potential 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 III shows the number of clauses of length 1, 2, and 3 included for each high weight Hamming weight pair. Finally, this yielded a set of very short clauses for each Hamming weight pair, which was 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 II
WEIGHTS OF HAMMING WEIGHT PAIRS

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

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 III
NUMBER OF CLAUSES OF LENGTH 1, 2, AND 3 INCLUDED FOR EACH HIGH WEIGHT HAMMING WEIGHT PAIR

B. Experimental Results

In this section, we present experimental results which demonstrate the performance of our improved algebraic side channel attack. We used the same assumptions as in [1]. For the experiments presented here, we assume that all Hamming weights are correct. For these experiments we used both the standard Java implementation of ASCA by Renauld [1] and

our modified implementation IASCA. We used the SAT solver CryptoMiniSat [5] 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™ Processor 8356" CPUs and 128 GB of main memory. Each CPU is running at 2.3 GHz.

Tables IV and V 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 our results obtained by using Renauld's ASCA and IASCA with the results reported in [1]. 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 IV shows that in case of consecutive Hamming weights, IASCA needs at most only the Hamming weights of two consecutive internal rounds 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 (788 HW), 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%, while ASCA needs all the 84 Hamming weights of the first round.

Attack \ Model	Res. in [1]	ASCA	IASCA
Consecutive HW	3 rounds (256 HW)	3 rounds (256 HW)	2 rounds (168 HW)
Random HW	$>$ 756 HW	551 HW	394 HW
Random R_1	-	84 HW	68 HW

TABLE IV
RESULTS FOR KNOWN PLAINTEXT/CIPHERTEXT ATTACK SCENARIO.

Table V 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 we have a 70% rate of success using IASCA. In case of a random scenario, where the known Hamming weights are distributed randomly over all potential leakages, IASCA needs only 472 Hamming weights distributed randomly over all rounds with rate of success $\geq 80\%$.

Attack \ Model	Res. in [1]	ASCA	IASCA
Consecutive HW	3 rounds (252 HW)	3 rounds (252 HW)	$<$ 3 rounds (184 HW)
Random HW	$>$ 756 HW	551 HW	472 HW

TABLE V
RESULTS FOR UNKNOWN PLAINTEXT/CIPHERTEXT ATTACK SCENARIO.

IV. ERROR TOLERANCE

A. Error Tolerance by Generalization

Most contributions on algebraic side-channel attacks, for instance [1], assume an error free set of Hamming weights. However, due to several kinds of noise (e.g., electronic noise, quantization noise, and switching noise) the emitted side-channel leakage may lead to falsified Hamming weight predictions [10]. 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 [3], [11] deal with erroneous predictions by using pseudo-Boolean optimization instead of SAT solvers. In order to verify their approach, they built a system for the Keeloq block cipher, for which they were able to solve the system in 3.8 hours assuming an error rate of 18.8%. Lately, in [11] an evaluation on AES was described, which deals with an error rate of 25% and has a solving time of 8.7 h. However, the error rate of a practical template attack is not always bound to 25%.

In the following, we extend IASCA with the capability of dealing with errors. The idea behind our approach is to generalize the clauses, inserted to the algebraic system, until they describe all Hamming weights that are predicted from the template attack with a high certainty. Let $\mathbb{W} = \{x \in \mathbb{N}_0 | x \leq 8\}$ describe all possible Hamming weights for an 8-bit value. So far, we have built the clauses by assuming that the equation $HW(x) = w$ with $w \in \mathbb{W}$ holds. This equation can also be expressed as two inequations, where one represents $HW(x) \leq w$ and the other $HW(x) \geq w$. Both inequations can be described by a set of clauses (cf. equation 2 and 3). However, in order to represent a larger range of Hamming weights, we can also change the boundary w of each inequality. For instance, if we assume the correct Hamming weight in the interval $\{w, w+1\}$ we combine the equations $HW(x) \leq w+1$ and $HW(x) \geq w$. Also, if we are more uncertain in the value of the correct Hamming weight, we can extend the interval to $\{w-1, w, w+1, w+2\}$ and describe this by combining the clauses for $HW(x) \geq w-1$ and $HW(x) \leq w+2$.

However, this approach has an impact on the improvement we proposed in Section III. 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 (w_x, w_y) is (X, Y) and an error of e_x and e_y, i.e. $X \subseteq \mathbb{W}$

is a set with e_x elements and $Y \subseteq \mathbb{W}$ is a set with e_y elements. In this case, instead of adding the short clauses for the pair (X, Y), we add the cross product of the sets for all Hamming weight pairs in $X \times Y$, thus making sure that all Hamming weight pairs satisfy the clauses. Naturally, the number of clauses decreases with increasing errors e_x and e_y.

Note that we assume the predicted Hamming weights to be consecutive. This is a realistic assumption, since if the Hamming weight leakage model correctly describes the power consumption of our device, consecutive Hamming weights have a similar likelihood. However, if the assumed Hamming weights are not consecutive, a larger interval of Hamming weights must be chosen such that the minimum and maximum predicted Hamming weight define the limits of the interval. Also note that our approach is not restricted in the size of the interval and thus in the severity of the error.

B. Test Setup and Template Attack

In the following, we evaluate the performance of our error tolerant IASCA by inserting clauses, obtained by conducting a template attack on a microcontroller executing an AES-128 encryption. We perform the attack on an ATMega256-1 microcontroller for which the Hamming weight leakage model adequately describes the 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 an 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 III, i.e. for 788 values in total. We utilized a single-trace template attack (e.g., $N_2 = 1$), and, in order to achieve a decisive result about the precision, repeated the attack 2000 times with varying plaintext/key pairs.

C. Estimating the Error Tolerance of IASCA

When performing ASCA one would represent the predicted (in our case most likely) Hamming weight as a set of clauses, which would then be inserted into the algebraic system. However, this approach relies on a template attack that always predicts the correct Hamming weight, which is rarely possible in practice. Therefore, we suggest to utilize the generalization of clauses as discussed in Subsection IV-A in order to describe an interval of Hamming weights, thereby lessening the dependence on a correct prediction of the template attack. However, describing a larger set of Hamming weights reduces the descriptiveness of the clauses and increases the solving time of the SAT solver. Thus, we have to find a trade-off between maximizing the possibility that the correct Hamming weight is contained in the predicted set and minimizing the number of elements, contained in the set. We tackle this problem by fixing a trade-off variable, the so-called *certainty threshold*. In order to determine this certainty threshold we introduce an additional phase after the profiling phase of the template attack, the so-called *precision estimation phase*.

The precision estimation phase requires a set of measurements that were not used in the profiling phase and for which the key and plaintext are known. Ideally, this set can be obtained by leaving out some measurements in the profiling phase. In our experiments, used 1000 measurements for the precision estimation phase from which we gained $788,000$ predictions. For each prediction we obtained a likelihood vector $\mathcal{L}$ and computed the correct Hamming weight. Next, we determined the certainty threshold T. The certainty threshold is a measure of trust in our template attack and helps determining which Hamming weights of a prediction we insert into our algebraic system. Suppose we have obtained the likelihood vector $\mathcal{L}$ from our template attack. Using a pre-determined certainty threshold T we can then compute the Hamming weights we want to insert into the algebraic system using the function f:

$$f(T, \mathcal{L}) = \min_{W}\{|W| : \sum_{w \in W} \mathcal{L}_w \geq T \wedge \nexists\, W^* \subseteq \mathbb{W} \wedge \sum_{w^* \in W^*} \mathcal{L}_{w^*} > \sum_{w \in W} \mathcal{L}_w \wedge W^* \neq W\}, \quad (3)$$

for $W \subseteq \mathbb{W}$. In other words, f returns the minimal set of Hamming weights that the template attack determined as most likely and for which the accumulated likelihood exceeds the certainty threshold T.

The certainty threshold makes use of the fact that even if the template attack predicted the wrong Hamming weight as the most likely, the likelihood of the correct Hamming weight is still above a certain minimum. Thus, there always exists a certainty threshold T for which the function f outputs a set that contains the correct Hamming weight. The challenge of the precision estimation phase is to fix the value of T such that all sets of Hamming weights, returned by f, contain the correct Hamming weight while having the smallest possible number of elements. In order to determine the best choice for T we computed the size of the predicted sets for a varying T, return by f, for each of the 788,000 predictions of the prediction estimation phase. We determined whether T was chosen big enough by computing the accuracy, i.e. the number of instances where the correct Hamming weight was included in the output of f, divided by the number of predictions. The results of these tests are depicted in Table VI.

As expected, a higher certainty threshold increases the size of the sets but yields a higher accuracy (acc). In order to find the best suited certainty threshold T for the attack, we have to use the lowest threshold that has an accuracy of 100%, in this case a threshold of $T = 95\%$. Using a threshold of $T \leq 90\%$ would raise the chance that we add false information to our algebraic system while a threshold of $T \geq 98\%$ would decrease the descriptiveness of the clauses and thus unnecessarily increase the solving time of the SAT solver.

After the precision estimation phase we proceeded with the attack phase of the template attack. For each of the 2000 power traces we obtained 788 predictions for which we each

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

TABLE VI
PREDICTED HAMMING WEIGHT SET SIZES FOR A GIVEN THRESHOLD T.

determined the corresponding set of Hamming weights using $T = 95\%$ and f. The resulting clauses were inserted into the algebraic system together with the plain- and ciphertext. The results are depicted in the second part of Table VI. IASCA solved the system for $T = 95\%$ within $84s$ and required the Hamming weight information for the first three rounds. 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. When we used the error classes for $T = 99\%$, we were not able to solve the system, even though we added all information.

Moreover, we were also interested in the performance of IASCA for the predicted sets for $T \in \{80\%, 85\%, 90\%\}$. Thus, we eliminated the errors in the predictions of the precision estimation phase while maintaining the size of the predicted sets and inserted them into the algebraic system. IASCA was able to recover the correct key while only requiring the predictions of the first round for a certainty threshold of $T = \{80\%, 85\%\}$. For $T = 90\%$ we could solve the system in 3 seconds using only the Hamming weight predictions 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 consider only 35% correctly predicted Hamming weights and were still able to solve the algebraic system in only two seconds using less information than [1], [3], [11]. For $T = 90\%$, IASCA successfully solved the algebraic system using only the first two rounds within three seconds, even when an a set of 4 possible Hamming weights was considered, which required a high solving time in [3], [11]. For $T = 95\%$, IASCA needs 84 seconds, but we assume only 9% correctly predicted Hamming weights and a presence of 18% of sets with 4 Hamming weights. 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 predictions to be correct and also deal with predictions that cover at least 4 Hamming weights in 47% of the cases.

V. CONCLUSION

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

First, we reduced the information required for solving the algebraic system by improving the algebraic representation of AES as well as Hamming weight information.

Secondly, we considered the application of IASCA under the assumption of erroneous information. We 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. This allowed us to generate the clauses for the algebraic system specifically for each of the predictions, therefore increasing the gain from the combination of side-channel attacks and algebraic systems. 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 [3], [11] in the number of errors, the solving time, 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.

ACKNOWLEDGMENTS

The work presented in this contribution was supported by the German Federal Ministry of Education and Research (BMBF) in the project *RESIST* through grant number 01IS10027A. The second author is supported by the German Science Foundation (DFG) grant BU 630/22-1. We would like to thank Mathieu Renauld for his useful comments on this paper and for his valuable suggestions.

REFERENCES

[1] M. Renauld, F. X. Standaert, and N. V. Charvillon, "Algebraic side-channel attacks on the aes: Why time also matters in dpa," in *CHES*, 2009, pp. 97–111.
[2] M. Renauld and F.-X. Standaert, "Algebraic side-channel attacks," in *Inscrypt*, 2009, pp. 393–410.
[3] Y. Oren, M. Kirschbaum, T. Popp, and A. Wool, "Algebraic side-channel analysis in the presence of errors," in *CHES*, ser. Lecture Notes in Computer Science, S. Mangard and F.-X. Standaert, Eds., vol. 6225. Springer, 2010, pp. 428–442.
[4] NIST, *Advanced Encryption Standard (AES) (FIPS PUB 197)*, National Institute of Standards and Technology, Nov. 2001.
[5] M. Soos, "Cryptominisat 2.5.0," in *SAT Race competitive event booklet*, July 2010.
[6] G. Bard, *Algebraic Cryptanalysis*. Springer, 2009.
[7] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *CHES*, 2002, pp. 13–28.
[8] J. R. Rao and B. Sunar, Eds., *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3659. Springer, 2005.
[9] M. Brickenstein and A. Dreyer, "Polybori: A framework for gröbner-basis computations with boolean polynomials," *Journal of Symbolic Computation*, vol. 44, no. 9, pp. 1326 – 1345, 2009, effective Methods in Algebraic Geometry.
[10] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
[11] Y. Oren and A. Wool, "Tolerant algebraic side-channel analysis of AES," Cryptology ePrint Archive, Report 2012/092, 2012, http://eprint.iacr.org/.