Algebraic Side-Channel Attacks on the AES: -Why Time also Matters in DPA -

Mathieu Renauld*, François-Xavier Standaert**, Nicolas Veyrat-Charvillon*
UCL Crypto Group, Université catholique de Louvain, B-1348 Louvain-la-Neuve.
e-mails: mathieu.renauld,fstandae,nicolas.veyrat@uclouvain.be

Abstract. Algebraic side-channel attacks have been recently introduced as a powerful cryptanalysis technique against block ciphers. These attacks represent both a target algorithm and its physical information leakages as an overdefined system of equations that the adversary tries to solve. They were first applied to PRESENT because of its simple algebraic structure. In this paper, we investigate the extent to which they can be exploited against the AES Rijndael and discuss their practical specificities. We show experimentally that most of the intuitions that hold for PRESENT can also be observed for an unprotected implementation of Rijndael in an 8-bit controller. Namely, algebraic side-channel attacks can recover the AES master key with the observation of a single encrypted plaintext and they easily deal with unknown plaintexts/ciphertexts in this context. Because these attacks can take advantage of the physical information corresponding to all the cipher rounds, they imply that one cannot trade speed for code size (or gate count) without affecting the physical security of a leaking device. In other words, more intermediate computations inevitably leads to more exploitable leakages. We analyze the consequences of this observation on two different masking schemes and discuss its impact on other countermeasures. Our results exhibit that algebraic techniques lead to a new understanding of implementation weaknesses that is different than classical side-channel attacks.

1 Introduction

Template attacks [9] are usually considered as the most powerful type of side-channel attacks and can be viewed as divided in two distinct phases. First, an adversary profiles the device that he targets. That is, he builds a probabilistic model for the leakages of this device, usually referred to as templates. Then in a second phase, the adversary uses these templates to compare key-dependent predictions of the leakages with actual measurements. By repeating successive measurements and comparisons, he can eventually identify the secret data that is manipulated by the leaking implementation. This key recovery is generally performed using a divide-and-conquer strategy, recovering small pieces of the key one by one. Following, this description, an important practical question in template attacks is to determine which templates to build. For example, one can decide to profile key bits directly or to profile intermediate operations occurring

* Work supported in part by the Walloon Region research project SCEPTIC.
** Associate researcher of the Belgian Fund for Scientific Research (FNRS - F.R.S.).

in a cryptographic implementation. But once templates have been built, it also remains to determine which information to extract from the leaking device. Most side-channel attacks in the literature directly recover key bytes. In this paper, we tackle the question to know whether it is necessary to extract such a precise information or if extracting some function of the intermediate values in a cryptographic computation (that is easier to guess with the side-channels than key bytes) can lead to a successful cryptanalysis with smaller data complexity.

As a matter of fact, this question is not new and several papers already dealt with very similar issues. Most notably, collision attacks such as, *e.g.* [4, 5, 18, 24, 25] don't use the side-channels to recover key bytes. They rather detect pairs of plaintexts giving rise to similar leakages for some intermediate values in the target algorithm. Then, in an offline cryptanalysis step, they use these collisions to recover the complete block cipher key. More generally, papers such as [3, 7, 15] also combine the side-channel leakages with black box (differential, impossible differential, square) cryptanalysis techniques. Still, and as classical side-channel attacks, these attacks mainly exploit the information provided by the first block cipher rounds, where the diffusion is not yet complete. Very recently, a new type of side-channel attacks, denoted as algebraic, has been introduced in order to get rid of this limitation [23]. These attacks (inspired from [10]) first write the target block cipher as a system of quadratic (or cubic, ...) equations. But since solving such systems of equations is generally too hard, they additionally add the physical information leakages provided by any intermediate computation during the encryption process to the system. In the previously investigated example of the block cipher PRESENT implemented in an 8-bit controller, this allowed to recover the cipher key with the observation of a single encryption. Compared to classical side-channel attacks, algebraic techniques bring two interesting features. First, they can take advantage of the information leakages in all the cipher rounds. Second, they can exploit any type of information leakage. In other words, whatever information about the intermediate computations can be added to the system and the more intermediate computations (*i.e.* clock cycles, generally), the more powerful the key recoveries. Hence, they can be viewed as an extreme version of template attacks in which more intermediate leakage points are targeted but less information need to be recovered from them.

The contributions of this paper are threefold. Since algebraic attacks have first been applied to the block cipher PRESENT, due to its simple algebraic structure, it is interesting to evaluate if a more conservative cipher would lead to significantly different results. Hence, we follow the ideas of [23] and apply them to the AES Rijndael. We show that algebraic attacks are still applicable in this context, but also exhibit an increase in the attacks time complexity. Second, we analyze the security of two different masking schemes and show how algebraic attacks modify previous intuitions (*e.g.* more mask bits do not always imply more security anymore). Eventually, we discuss the influence of different design choices and countermeasures in view of the new algebraic techniques.

2 Algebraic side-channel attacks

A detailed description of algebraic side-channel attacks is given in [23]. In this paper, we aim to evaluate the impact of various parameters on their effectiveness and to apply them to a practical implementation of the AES Rijndael. Hence, this section only provides a high level description of the different attack phases.

2.1 Offline phase 1: building the system of equations

The goal of this first phase is to transform the AES into a big system of low degree boolean equations. In this system, the key bits appear as variables in such a way that solving the system is equivalent to recovering them. In this paper, we exploit the techniques presented in [2] to build our system of equations. In the case of the AES Rijndael with 128-bit plaintext and key, it results in a system of approximately 18 000 equations in 10 000 variables (27 000 monomials).

2.2 Online phase: extracting the physical information

Since directly solving the system of equations representing the AES Rijndael is generally too hard with present techniques, the idea of algebraic side-channel attacks is to feed this system with additional information. Quite naturally, physical leakages are very good candidates for this additional information. As detailed in the introduction of this paper, the issue is then to decide what to extract from the target implementation (a somewhat arbitrary decision). The more physical information extracted, the easier the solving and the more interesting the algebraic techniques compared to standard DPA. Therefore, it yields two questions:

Which intermediate operations to target? This question mainly depends on the target implementation. For example, our following experiments consider the AES Rijndael with a 128-bit master key in an 8-bit PIC microcontroller. In this context, SubBytes will generally be implemented as a 256-byte table lookup and MixColumn will exploit the description of [13] in which it is implemented with four 256-byte table lookups and 9 XOR operations (giving 13 potential leakage points). It is therefore natural to profile the target device in such a way that the leakages corresponding to all these intermediate computations (additionally considering the key additions) are exploited by the adversary.

Which information to recover from each target operation? Once the target operations have been decided by the adversary, it remains to determine what to learn about them. This again depends on the target device (and on countermeasures that could possibly be included to the implementation). For example, an unprotected implementation of the AES Rijndael in the PIC is such that for each of the previously mentioned operations, the output data has to commute on an 8-bit bus. During this clock cycle, the leakages will be highly correlated to the Hamming weight of this output data. Hence, it is again natural to perform a template attack such that these Hamming weights will be recovered.

Importantly, algebraic side-channel attacks exploit the leakages of several target operations at once. It implies that one needs to recover a correct information about all these operations at once. Indeed, introducing false information into the system of equations will generally lead to inconsistencies and prevent its proper solving. Fortunately, this can be achieved for our target device. As an illustration, the left part of Figure 1 illustrates the leakage traces of different values commuting on the PIC bus and their mean for different Hamming weights. In the right part of the same figure, we plotted the probability that the Hamming weight of large amounts of target bytes can be recovered with high confidence, possibly exploiting simple error detection (ED) and likelihood rating (LR) techniques (NEDLR means that we do not use these techniques).

– *Error detection* consists in rejecting side-channel information that gives rise to incoherent input and output values for the S-boxes.
– *Likelihood rating* means that we only use a subset of all the Hamming weights values extracted with the templates, starting with the most likely ones.

We stress the fact that all our attacks use a *data complexity* of $q = 1$. That is, we use only the encryption of one single plaintext to perform our measurements. This data complexity is different from the number of traces since we can sometimes repeat the same measurement. In our experiments, we use a repetition umber of $n_r = 1$ or 2. This is significantly different than standard DPA attacks which usually require a much higher data complexity (typically, $10 \leq q \leq 100$). We see that using these techniques, roughly 200 Hamming weights can be recovered with the observation of a single encrypted plaintext and that by repeating the measurement of the same encryption, this number can be up to 700.

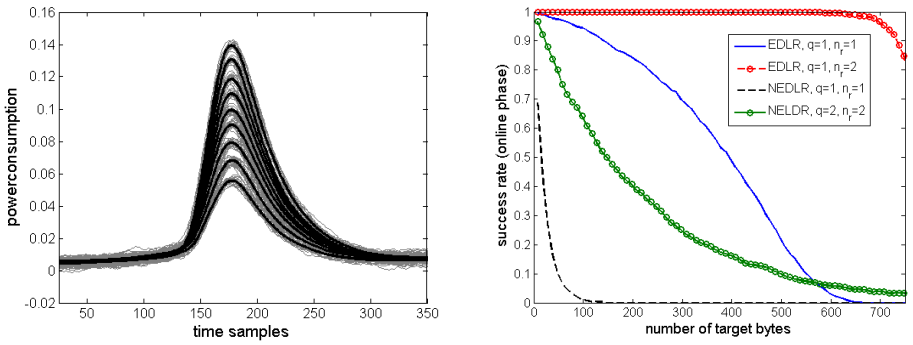

Fig. 1. Leakage traces, mean leakage traces and multiple byte success rate.

As previously mentioned, the implementation of the AES-128 that we attack was profiled in such a way that we recovered the Hamming weights corresponding to AddRoundKey (16 weights), SubBytes (16 weights) and MixColumn ($4 * 13$ weights in our implementation of the AES). For the 10 cipher rounds, it corresponds to a maximum of 788 (not always necessary) correct Hamming weights.

It is essential to note that algebraic side-channel attacks in general could work with totally different choices for the target operations and leakages. As a proof of concept and because of their wide use and good connection with our target device, we extracted Hamming weights from our power consumption traces. But one can theoretically exploit any type of information leakage. A particularly interesting open problem is to apply these attacks to advanced circuit technologies with more complex leakage behaviors. It raises the question of how to best extract partial information that is both meaningful (*i.e.* allows to solve the system) and robust (*i.e.* can be easily recovered with high confidence).

2.3 Offline phase 2: solving the system

Once the system of equation including the additional side-channel information is written, it remains to attempt solving it. Different solutions have been proposed in the literature for this purpose. The original attack of Courtois and Pieprzyk proposed linearization techniques called XL or XSL [10]. Groebner basis-based techniques have then been suggested as possible alternatives, *e.g.* in [6, 12]. Yet another possibility is to use a SAT solver as in [11]. In this paper, we take advantage of this last solution. It implies that the system has to be expressed as a *satisfiability problem*. The satisfiability problem is the reference NP-complete problem and is widely studied (see [14] for a survey). It consists of determining if a boolean formula (a formula combining boolean variables, AND, OR and NOT gates) can be "satisfied", *i.e.* if it exists at least one assignment of the variables such that the whole formula is true. Most SAT solvers require a particular type of formula denoted as a Conjunctive Normal Form (CNF). A CNF is a conjunction (AND) of clauses, each clause being a disjunction (OR) of literals.

In practice, we can write a CNF from our system of equations so that the only valid assignment corresponds to the solution of the system, as detailed in [1]. But this conversion to a boolean formula can be done in a number of ways. For example, the substitution boxes of the AES can either be written as non-linear boolean equations (during the first offline phase of the attack) that are then converted into a boolean formula, or as a set of clauses that are introduced directly into the global CNF. While the first method seems more complex and introduces (much) more intermediate variables, it gave rise to better results in some of our experiments. We conjecture that the SAT solver better handles certain hard instances of the attack with some redundancy and a few additional intermediate variables. As an illustration, the first solution gives about 120 000 clauses of 4 literal per substitution box, versus 2048 clauses of 9 literals for the second solution. The final size of the CNF derived from the system of equations consequently depends on the conversion strategy. To the previous S-boxes, one has to add the linear layers (MixColumn and AddRoundKey) that produce approximately 45 000 clauses of 4 literals per round. Eventually, the additional side-channel information can be defined by approximately 70 000 clauses of up to 8 literals. This gives a formula containing between a minimum of 500 000 and up to several millions of clauses of 1 to 9 literals (the SAT solver used in our experiments [8] was not able to handle more than 5 millions of them).

3 Attacking the AES key scheduling algorithm

Before starting the investigations of algebraic side-channel attacks against the AES, a preliminary remark has to be made about attacks against its key scheduling algorithm. Because of the relatively simple structure of the key expansion in Rijndael, it is possible to write a system of equations only for this part of the algorithm and to solve it successfully with (even a part of) the Hamming weights gathered from its execution. Such attacks then closely connect to the SPA proposed by Mangard in [19]. In the following, we consequently consider the more challenging case in which round keys are pre-computed in a safe environment.

4 Attacking the AES Round functions

In this section, we assess the practicability of algebraic side-channel attacks against the AES. For this purpose, we considered several situations. First, we attacked an unprotected implementation with a known pair of plaintext/ciphertext. Then, we studied the influence of unknown plaintexts/ciphertexts. Finally, we considered two different masking schemes and analyzed their resistance against algebraic attacks. Each attack exploits the observation of a single encryption trace from which side-channel information is extracted. Additionally, the amount of information recovered by the adversary is used as a parameter in our evaluations. It is measured in "number of rounds of Hamming weights", obtained consecutively (*i.e.* the adversary recovers all the Hamming weights of a few consecutive rounds - the chosen rounds are picked starting in the middle of the block cipher) or randomly (*i.e.* the adversary recovers the same amount of Hamming weights randomly spread over the different intermediate computations).

We first assume that no incorrect side-channel information is used (which would lead the SAT solver to fail). Hence, the experiments in Section 4.1, and 4.2 only focus on the second part of the offline phase described in Section 2.3. Then, in Section 4.4, we discuss the tradeoff between the applicability of this offline phase and the online information extraction described in Section 2.2.

Note that for difficult instances of the attack, the SAT solver is sometimes unable to solve the system in a reasonable time. We consider that an attack has failed whenever the solver has not found a solution within 3600 seconds. Eventually, the SAT solver was running on an Intel-based server with a Xeon E5420 processor cadenced at 2.5GHz running a linux 32-bit 2.6 Kernel.

4.1 Attacking an 8-bit device with known plaintext/ciphertext

The results of this first scenario are in Figure 2 (solid lines). Each dot is obtained from averaging on a set of 100 independent experiments. They illustrate that the success rate of the attacks depends heavily on whether the side-channel leakages correspond to successive operations or not. Indeed, 3 rounds of consecutive leakages (*i.e.* 252 Hamming weights) is enough to solve more than 95% of the cases,

whereas 8 rounds of randomly distributed leakages only give a 60% success rate. It is interesting to note that it is in fact the leakage of the MixColumn operation that seems to be the most critical when solving the system. Removing some of the Hamming weights obtained from this operation highly impacts the attack effectiveness. This can be justified by recalling that MixColumn consumes a large amount of clock cycles in a software implementation of the AES.

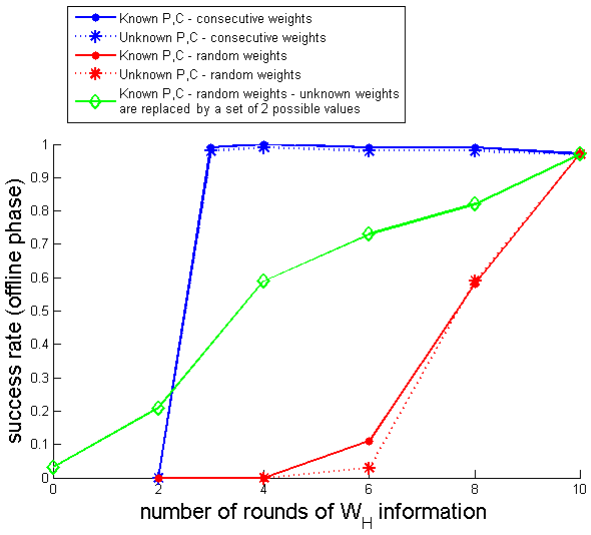

Fig. 2. Success rate of the attacks against an unprotected implementation of the AES Rijndael in function of the amount of exploited leakages. One round of side-channel information is equivalent to 84 known Hamming weights.

We finally remark that the SAT solver can also deal with less precise leakages. For example, if one Hamming weight cannot be determined exactly, it is possible to consider a pair of Hamming weights including the correct one and to add this information to the system. This context is investigated in Figure 2 (circled line).

4.2 Attacking an 8-bit device with unknown plaintext/ciphertext

The dotted lines in Figure 2 represent the results of an attack with unknown plaintext and ciphertext. It is an interesting scenario to consider since classical DPA attacks such as [17] would generally fail to recover keys in this context. Intuitively, it is also a much harder case, since the plaintext and ciphertext represent 256 unknown variables in our SAT problem. But excepted for a slight reduction of the success rate, our results show that algebraic side-channel attacks against the AES Rijndael are quite strongly immune against unknown inputs. This can be understood when considering that 3 consecutive rounds of Hamming weight are enough to recover the complete master key.

4.3 Time complexity and attack strategies

The solving times of the SAT problem for different instances of side-channel algebraic attacks (represented in Figure 3) seem to follow an exponential distribution. It means that a small number of instances require much more effort from the SAT solver, which may be due to the intrinsic difficulty of these instances, but also to some poor decisions made by the SAT solver. Therefore, the question to know if good strategies could be applied in order to better cope with this distribution directly arises. This section brings some insights with this respect.

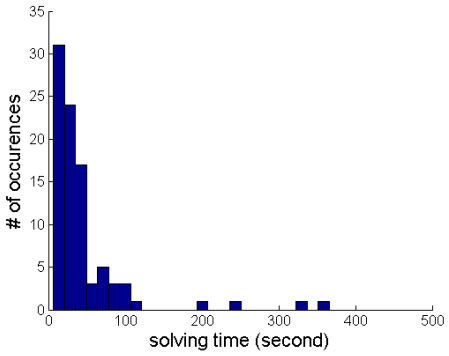

Fig. 3. Distribution of the solving times (8 consecutive rounds of Hamming weights).

In its basic version, the SAT solver behaves deterministically, always spending the same amount of time on the same boolean formula, for the same result. However, it is possible to introduce randomness in the resolution process, thereby forcing the solver to explore the search space differently. Figure 4 shows how far the solver was able to go towards the resolution of the same SAT problem, when launched 70 times with different random seeds. A successful run has to assign 12577 variables within 3600 seconds. The bold line in the figure shows the maximum number of variables that the solver was able to assign at a given time, averaged over the 70 runs. All the other curves represent single runs. One can observe that the solving time for the same instance varies with random resolutions. As previously, it seems to follow an exponential distribution (*i*.e. the probability that a problem has not been solved after x seconds decreases exponentially in x). Also, most of the time is spent in assigning the first 3000 variables. Once this is done, finding the rest of the solution is almost instantaneous (this is observed with the vertical jump of the single runs in the figure).

These results suggest that performing a serial of short random resolutions can give better results than using a single long resolution. Unfortunately, it is not possible to predict a priori how long the resolution will take. Hence, in order to optimize the attack, we need to adapt the time limit in function of some preliminary analyzes. For example, in our experiments, 90% of the resolutions succeed in less than 100 seconds. Hence, a reasonable strategy would be to fix

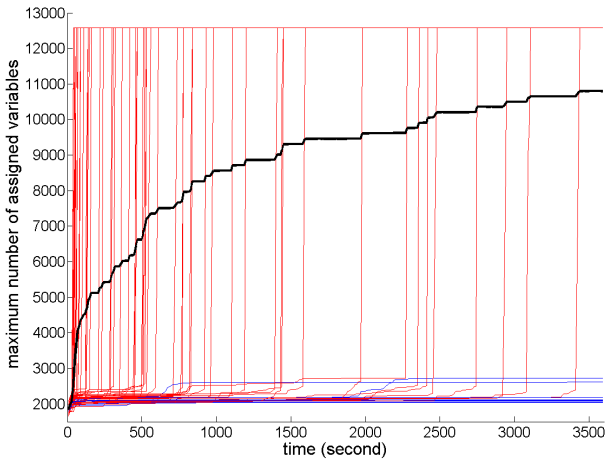

Fig. 4. Evolution of the solving for 70 random runs of the same SAT instance (8 rounds of Hamming weights are known, randomly spread over the block cipher).

the time limit at 100 seconds (rather than 3600, originally). In this setting, most of the attacks will succeed in a single run, and the harder instances will hopefully be solved by another random resolution. Of course, such strategies are mainly interesting for attacks that are successful with high probability within reasonable time constraints, which may not always be the case in practice.

4.4 Global success rate

Figure 4.4 finally presents the global success rate of our attack, combining the success rate of the online side-channel phase (given in Figure 1) and the success rate of the offline computation phase (given in Figure 2). It clearly illustrates the tradeoff between these two phases. On the one hand, the success rate of the side-channel information extraction decreases with the quantity of information the attacker tries to recover. On the other hand, the success rate of the algebraic cryptanalysis increases with the quantity of information available. This implies that for a strictly limited amount of measurements (*e.g.* $n_r = 1$ in the figure), the best global success rate does not occur for the maximum amount of Hamming weights inserted into the system. Note that this global success rate could be improved by better dealing with failures in the solving. For example, if an inconsistency is found, it is possible to reject a few Hamming weights and try again to solve the system, hoping that the incorrect leakages have been rejected.

4.5 Attacking masked implementations

The previous section suggests that attacking an unprotected implementation of the AES Rijndael in an 8-bit controller is not significantly more difficult than attacking the block cipher PRESENT in the same device in terms of data

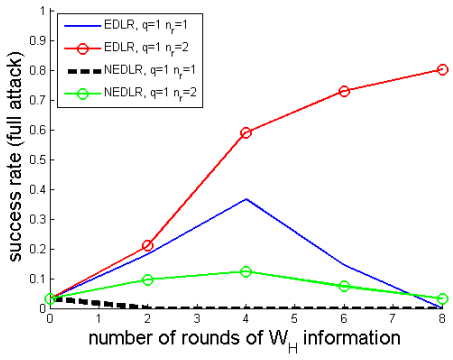

Fig. 5. Global success rate for an unprotected implementation, with known P, C.

complexity: both attacks succeed with the observation of a single encrypted plaintext. Nevertheless, attacks against the AES exhibit a significantly increased average time complexity (2.5 seconds for 31-round *vs.* 344 seconds for 10-round AES). In this section, we investigate the extent to which the increased algebraic complexity of the AES could lead to more hardly solvable systems if the target implementation is protected with a masking scheme.

Masking is a very popular countermeasure against side-channel attacks. It aims at rendering the power consumption of the device independent of the intermediate values used by the cryptographic algorithm. This is achieved by combining theses values with random "masks" internal to the cryptographic device (hence hidden to the adversary), that vary from execution to execution. The masks are generated at the beginning of the encryption, combined with the plaintext and key, and propagated through the cryptosystem so that every value transiting on the bus is a combination of the original intermediate value and a mask. In terms of security, masking provably prevents certain types of side-channel attacks. But it does not remove the existence of physically exploitable leakages (since the secret data is still manipulated by the device and the correct encryption result has to be produced). In terms of cost overheads, masking affects the performances of an implementation, mainly because the masks have to be propagated through the encryption process in a tractable manner. For this reason, several masking schemes have been proposed in the open literature, trading effectiveness for more random masks (hopefully implying more security).

In the following, we focus on two proposals that we believe illustrative of the state-of-the art countermeasures. Namely, we first focus on the efficient masking scheme proposed by Oswald and Schramm in [20] that uses one different mask for every plaintext byte. Then we consider the AES software implementation resistant to power analysis attacks proposed by Herbst *et al.* in [16].

1. Masking in $\mathbf{GF(2^4)^2}$. As noticed in several publications, the most difficult part of the AES to mask is its non-linear layer. This is because for a masked input $x+m$, one needs to generate a masked SubBytes such that: MSubBytes($x+m$)=SubBytes(x)$+m'$ with m' an output mask that can be propagated through the cipher rounds. The practical issue for designers is to deal with different masks efficiently. If the same mask m is used for all the AES bytes in all the rounds, only one MSubBytes table needs to be computed, which can be done quite easily prior to the encryption. But this may lead to security issues because different intermediate variables are masked with the same value. On the opposite, using a different mask for all the AES bytes implies to recompute MSubBytes anytime an S-box is computed, which is generally too expensive for practical applications. In order to get rid of this limitation, [20] proposes to take advantage of the algebraic structure of the AES S-box to deal with multiple masks efficiently. The main idea is to represent the non-linear part of the AES S-box as a multiplicative inverse in the composite field $\mathrm{GF}(2^4)^2$. Doing this, it is possible to compute a masked inverse "on-the-fly" in 14 table lookup operations and 15 XOR operations (this requires to store a total of 1024 bytes of tables in ROM).

Assuming that we can obtain the Hamming weights corresponding to all these intermediate computations, mounting an attack is straightforward. In fact, we can even attack each S-box (anywhere in the cipher) separately and find a solution in less than one second. This simplicity is due to the large quantity of side-channel information leaked (more than 20 Hamming weights for one substitution) and the extreme simplicity of the operations between two leakage points (one XOR or one table look-up operation). As an illustration, Figure 6 presents the results of such an attack mounted with partial information. It turns out that the amount of cycle makes this masked implementation even easier to target than the original (unprotected) implementation, due to the large redundancy of its leakages. We mention that we considered the compact representation described in [20] and only targeted 8-bit intermediate values. But the efficient representation in the same paper would make the situation even worse, allowing 4-bit values to be computed and observed through their Hamming weights.

2. Masking with S-box pre-computation. As previously mentioned, another approach for masking the AES, is to limit the number of 8-bit masks applied to the cipher state so that a reasonable number of masked tables can be pre-computed. The solution proposed by [16] is to use 2 masks m and m' for all the S-boxes inputs and outputs and 4 additional masks (m_1, m_2, m_3, m_4) for the MixColumn operation, resulting in only 48 random bits to generate prior to the encryption. In practice, this scheme turned out to be quite resistant to an algebraic side-channel attack. About 20% of the instances were solved in less than 24 hours of computation, and the fastest solving took about 5 hours. This is surprising, since the masking only adds 48 new variables to the problem while providing 32 more leaked values per round. From an intuitive point of view, this again comes back to the fact that such a masking scheme adds randomness to the computation without too much affecting the number of clock cycles required

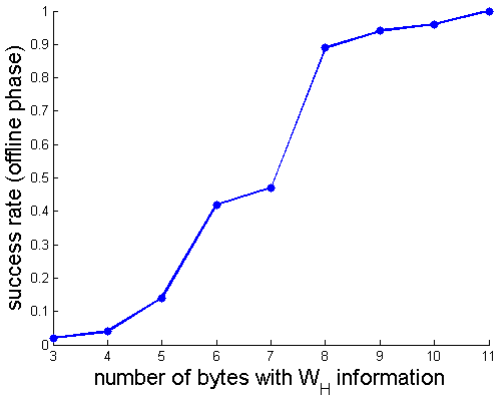

Fig. 6. Attacking one masked AES S-box with partial information leakage.

to encrypt one plaintext. It also relates to the previous observation that MixColumn is an important source of side-channel information in our attacks and becomes more complex because of the masking. For example, whereas the unprotected AES leaks $W_H(a_0 \oplus a_1)$ during the computation of MixColumn, the masked version leaks $W_H(a_0 \oplus m_1 \oplus a_1 \oplus m_2)$. That is, the masking doubles the number of bytes in each Hamming weight leakage. The quantity of information that one can extract from them is thus significantly less than that of the unmasked implementation. Such a masking scheme in therefore better suited to resist against an algebraic side-channel attack than [20].

5 Countermeasures

The previous section underlines that algebraic side-channel attacks shed a different light on the security of cryptographic implementations. In general, resisting against such attacks should follow different guidelines that we now detail:

1. Use block ciphers with high algebraic complexity.
2. Limit the number of clock cycles in the implementations (*i.e.* have each elementary operation/instruction that has sufficient algebraic complexity).
3. Increase the algebraic complexity of the leakages (*i.e.* use large data buses, add noise and countermeasures to the implementations).

With respect to masking in an 8-bit device, the impact of these observations has been carefully analyzed in the previous section. In addition to the fact that [16] seem to better resist algebraic cryptanalysis than [20], our results obviously confirm that one cannot mask the first/last rounds of a block cipher only. We now discuss how the previous principles apply to practical countermeasures. First, using table-based implementations seems to be the most natural choice to improve resistance against algebraic side-channel attacks. The AES Rijndael has nice features with this respect (*e.g.* the ability to compute four S-boxes and MixColumn

in four 256×4-byte table lookups and three XOR operations). Although this solution is not directly applicable to small devices, preliminary experiments suggest that removing all the Hamming weights of the MixColumn operation but the 4 output ones significantly increases the resistance against algebraic side-channel attacks (we could only attack up to 70% of the SAT instances corresponding to an 4-round AES in this context). Hence, implementing table-based implementations in 32-bit (or larger) devices where the leakages would depend on larger amounts of intermediate bits (32 or more) is a challenging scenario for further investigation. Then, increasing the algebraic complexity of the leakages implies to reduce the amount of information provided by any elementary operation. Using large buses is helpful with this respect. But other countermeasures could be helpful: noise addition, dual-rail circuits, ... Eventually, we mention that time randomization would also increase the difficulty of performing the attacks, by preventing the direct profiling of the intermediate operations.

6 Conclusion and open problems

This paper shows that algebraic side-channel attacks can be applied to the AES Rijndael implemented in an 8-bit controller. The observation of a single leakage trace can be sufficient to perform a complete key recovery in this context and the attack directly applies to certain masking schemes. Our experiments also suggest interesting scenarios where less information is available (table-based implementations, typically) and for which solving the system of equations describing the target cipher and its leakages becomes challenging. Open problems consequently include the investigations of advanced strategies to attack protected devices (*e.g.* exploiting the leakages in a more flexible manner, dealing with errors, ...). In particular, circuit technologies where the leakage models are significantly different than Hamming weights would be interesting to evaluate.

To properly understand the impact of this result, it is important to relate it with theoretical and practical concerns. From a purely practical point of view, one could argue that algebraic side-channel attacks do not improve previous attacks against leaking ciphers. Indeed, these attacks require to recover the leakages corresponding to a significant amount of operations. In other words, they can be seen as (very) high-order attacks. Hence, a first-order DPA exploiting the leakages corresponding to as many leakage points (but for different plaintexts) would generally succeed as well. But from a theoretical point of view, the data complexity that is required to reach high success rates is significantly decreased. In other words, what eventually matters from a security point of view is the number of observed plaintexts required to recover a key. The attacks in this paper allow reaching a success rate of one with the observation of a single encrypted plaintext (which was never the case for previous side-channel attacks we are aware of). Even if multiple measurements have to be made (we require a maximum of one or two measurements in the present paper), they still correspond to a unique plaintext. By exploiting the leakages in all the cipher rounds, algebraic side-channel attacks also get rid of the computational limitations of classical

DPA in which enumerating key candidates is necessary. Eventually, they have an impact on the assumptions of constructions such as [21, **?**]. Hence, we believe these results have both a theoretical and practical relevance.

References

1. G. Bard, N. Courtois, C. Jefferson, *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers*, Cryptology ePrint Archive, Report 2007/024.
2. A. Biryukov, C. De Cannière, *Block Ciphers and Systems of Quadratic Equations*, in the proceedings of FSE 2003, Lecture Notes in Computer Science, vol 2887, pp 274-289, Lund, Sweden, February 2003.
3. A. Biryukov, D. Khovratovich, *Two New Techniques of Side-Channel Cryptanalysis*, in the proceedings of CHES 2007, Lecture Notes in Computer Science, vol 4727, pp 195-208, Vienna, Austria, September 2007.
4. A. Bogdanov, *Improved Side-Channel Collision Attacks on AES*, in the proceedings of SAC 2007, LNCS, vol 4876, pp 84-95, Ottawa, Canada, August 2007.
5. A. Bogdanov, I. Kizhvatov, A. Pyshkin, *Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection*, in the proceedings of Indocrypt 2008, LNCS, vol 5365, pp 251-265, Kharagpur, India, December 2008.
6. J. Buchmann, A. Pyshkin, R.-P. Weinmann, *Block Ciphers Sensitive to Gröbner Basis Attacks*, in the proceedings of CT-RSA 2006, Lecture Notes in Computer Science, vol 3860, pp 313-331, San Jose, CA, USA, February 2006.
7. V. Carlier, H. Chabanne, E. Dottax, H. Pelletier, *Generalizing Square Attack using Side-Channels of an AES Implementation on an FPGA*, in the proceedings of FPL 2005, pp 433-437, Tampere, Finland, August 2005.
8. http://www.princeton.edu/∽chaff/.
9. S. Chari, J. Rao, P. Rohatgi, *Template Attacks*, in the proceedings of CHES 2002, LNCS, vol 2523, pp 13-28, CA, USA, August 2002.
10. N. Courtois, J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, in the proceedings of ASIACRYPT 2002, LNCS, vol 2501, pp 267-287, Queenstown, New Zealand, December 2002.
11. N. Courtois, G. Bard, *Algebraic Cryptanalysis of the Data Encryption Standard*, in the proceedings of 11th IMA International Conference, Lecture Notes in Computer Science, vol 4887, pp 274-289, Cirencester, UK, December 2007.
12. J.-C. Faugère, *Groebner Bases. Applications in Cryptology*, FSE 2007 Invited Talk, available at: http://fse2007.uni.lu/slides/faugere.pdf.
13. FIPS 197, *"Advanced Encryption Standard,"* Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, November 26, 2001.
14. J. Gu, P.W. Purdom, J. Franco, B. Wah, *Algorithms for the Satisfiability problem: a survey*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science 35:19-151, American Mathematical Society, 1997.
15. H. Handschuh, B. Preneel, *Blind Differential Cryptanalysis for Enhanced Power Attacks*, in the proceedings of SAC 2006, Lecture Notes in Computer Science, vol 4356, pp 163-173, Montreal, Canada, August 2006.
16. C. Herbst, E. Oswald, S. Mangard, *An AES Smart Card Implementation Resistant to Power Analysis Attacks*, in the proceedings of ACNS 2006, Lecture Notes in Computer Science, vol 3989, pp 239-252, Singapore, June 2006.

17. P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*, in the proceedings of Crypto 1999, LNCS, vol 1666, pp 398-412, Santa-Barbara, CA, USA, August 1999.
18. H. Ledig, F. Muller, F. Valette, *Enhancing Collision Attacks*, in the proceedings of CHES 2004, LNCS, vol 3156, pp 176-190, Cambridge, MA, USA, August 2004.
19. S. Mangard, *A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion*, in the proceedings of ICISC 2002, Lecture Notes in Computer Science, vol 2587, pp 343-358, Seoul, Korea, November 2002.
20. E. Oswald, K. Schramm, *An Efficient Masking Scheme for AES Software Implementations*, in the proceedings of WISA 2005, Lecture Notes in Computer Science, vol 3786, pp 292-305, Jeju Island, Korea, August 2005.
21. C. Petit, F.-X. Standaert, O. Pereira, T.G. Malkin, M. Yung, *A Block Cipher based PRNG Secure Against Side-Channel Key Recovery*, in the proceedings of ASIACCS 2008, pp 56-65, Tokyo, Japan, March 2008.
22. K. Pietrzak, *A Leakage-Resilient Mode of Operation*, in the proceedings of Eurocrypt 2009, LNCS, vol 5479, pp 462-482, Cologne, Germany, April 2009.
23. M. Renauld, F.-X. Standaert, *Algebraic Side-Channel Attacks*, Cryptology ePrint Archive, report 2009/179, http://eprint.iacr.org/2009/279
24. K. Schramm, T.J. Wollinger, C. Paar, *A New Class of Collision Attacks and Its Application to DES*, in the proceedings of FSE 2003, Lecture Notes in Computer Science, vol 2887, pp 206-222, Lund, Sweden, February 2003.
25. K. Schramm, G. Leander, P. Felke, C. Paar, *A Collision-Attack on AES: Combining Side Channel and Differential Attack*, in the proceedings of CHES 2004, LNCS, vol 3156, pp 163-175, Cambridge, MA, USA, August 2004.