MDASCA: An Enhanced Algebraic Side-Channel Attack for Error Tolerance and New Leakage Model Exploitation*

Xinjie Zhao[1], Fan Zhang[2], Shize Guo[3],
Tao Wang[1], Zhijie Shi[2], Huiying Liu[1], and Keke Ji[1]

[1] Ordnance Engineering College, Shijiazhuang, Hebei, China
zhaoxinjieem@163.com
[2] University of Connecticut, Storrs, Connecticut, USA
fan.zhang@engineer.uconn.edu, zshi@engr.uconn.edu
[3] The Institute of North Electronic Equipment, Beijing, China

Abstract. Algebraic side-channel attack (ASCA) is a powerful cryptanalysis technique different from conventional side-channel attacks. This paper studies ASCA from three aspects: enhancement, analysis and application. To enhance ASCA, we propose a generic method, called Multiple Deductions-based ASCA (MDASCA), to cope the multiple deductions caused by inaccurate measurements or interferences. For the first time, we show that ASCA can exploit cache leakage models. We analyze the attacks and estimate the minimal amount of leakages required for a successful ASCA on AES under different leakage models. In addition, we apply MDASCA to attack AES on an 8-bit microcontroller under Hamming weight leakage model, on two typical microprocessors under access driven cache leakage model, and on a 32-bit ARM microprocessor under trace driven cache leakage model. Many better results are achieved compared to the previous work. The results are also consistent with the theoretical analysis. Our work shows that MDASCA poses great threats with its excellence in error tolerance and new leakage model exploitation.

Keywords: Algebraic side-channel attack, Multiple deductions, Hamming weight leakage, Cache leakage, AES.

1 Introduction

How to improve the efficiency and feasibility of side-channel attacks (SCAs) has been widely studied in recent years. The objective is to fully utilize the leakage information and reduce the number of measurements. This can be achieved from two directions. One is to find new distinguishers for key recovery [3, 8, 34]. The

* This work was supported in part by the National Natural Science Foundation of China under the grants 60772082 and 61173191, and US National Science Foundation under the grant CNS-0644188.

W. Schindler and S.A. Huss (Eds.): COSADE 2012, LNCS 7275, pp. 231–248, 2012.

other is to combine SCA with the mathematical techniques, such as differential based [17], linear based [30], collision based [31], cube based [11], algebraic based [16, 27–29] SCAs. This paper studies algebraic based SCAs.

Algebraic cryptanalysis converts the key recovery into a problem of solving a boolean equation system. The main idea was proposed by Shannon [32] and first applied to block ciphers by Courtois *et al.* in ASIACRYPT 2002 [9]. However, the complexity of algebraic cryptanalysis increases exponentially with the number of rounds. As a result, it is mainly effective to the reduced-round block ciphers. SCA can derive additional equations by analyzing the physical leakages (e.g., timing, power, EM, cache, etc) and help to solve the equation system. The combination of two techniques leads to Algebraic side-channel attacks (ASCA) [16, 24, 27–29].

In ASCA, the targeted cipher is first represented with a system of algebraic equations. The adversary chooses a leakage model and several intermediate states, according to his measurement capability and attack strategy. After that, the physical leakages are measured and used to deduce the output values of the leakage function (model) for these targeted states. Then additional equations representing these values are derived and added into the equation system. Finally, the equations system is solved with the solvers, such as Gröbner basis-based [12] or SAT-based [33] solvers, to recover the key bits.

1.1 Related Work

In the original ASCA [27, 28], the key recovery was converted to a *Boolean satisfiability (SAT) problem* and the zChaff solver was used. It has been shown that ASCA can exploit the Hamming weight (HW) leakages in all rounds and recover the key with a single trace even when both the plaintexts and the ciphertexts are unknown [27, 28]. Further research on ASCA has been focused on three aspects.

The first is to leverage error tolerant ASCA to improve its practicability. The work in [27, 28] was mainly based on an error-free assumption. Although it was mentioned that a pair of HWs can be used to build the algebraic equations even if one of them is incorrect, the error must be small. The details of the method and the error rates in practice were not discussed. In CHES 2010, an error tolerant ASCA (TASCA) [24] was proposed based on a *pseudo-Boolean optimization (PBOPT) problem*. The SCIP solver [5] was used to handle the wrong HW deductions. TASCA works on Keeloq when the error rate is less than 20% but fails on AES. Designing an error tolerant ACSA on AES is a challenge.

The second is to analyze the dependencies of ASCA. In ACNS 2010, it was shown in [29] that the success rate of ASCA depends on the representations, leakages and ciphers. However, it is difficult to predict the success rate because of numerous parameters. At least 252 consecutive HW leakages are required in ASCA on AES in [29], but the reason was not given. In COSADE 2011, the work in [16] showed that the success rate highly depends on the algebraic immunity and the distribution of leakage information. Still, it remains as an open problem to estimate the number of leakages required for a specific cipher.

The third is to exploit new leakage models in ASCA. Previous work has studied the combination of algebraic cryptanalysis with fault attacks, e.g, attacks on

DES in eSmart 2010 [10] and Trivium in COSADE 2011 [21]. The data complexity required in the attacks [10, 21] can be further reduced. As addressed in [27, 28], ASCA is a generic framework and can be applied to more leakage models. However the diversity, error, and complexity of leakage models make it difficult to adopt new models in ASCA.

1.2 Our Work

In this paper, we study ASCA in three aspects.

Enhancement. We initiate our work by addressing the error tolerance in ASCA. We observe that not only the errors but also the leakage models may cause multiple results when inferring the output value of the leakage function for a targeted state. The ability of handling such multiple values for a state is critical to improving the error tolerance and extending the applications of ACSA. In Section 2, we introduce an enhanced ASCA technique named as *Multiple Deductions-based ASCA (MDASCA)* and propose a generic method to represent multiple values.

Analysis. In Section 3, we analyze the possible application scenarios of MDASCA. For the first time, we can exploit cache leakage models, which are widely studied in recent years [1, 2, 4, 6, 7, 13–15, 20, 22, 25]. More specifically, we use MDASCA to exploit access driven [2, 22, 25] and trace driven [1, 6, 7, 13–15, 20] cache leakage models. In Section 4, we take another approach to evaluate ASCA different from [16, 29]. We estimate the minimal amount of leakages required for ASCA on AES. Many new and better results are given based on theoretical analysis, and later confirmed by experiments. For example, in HW-based ASCA on AES (standard NIST implementation in [23]), we show that: under known plaintext/ciphertext scenario, only one round of HW leakages is required instead of three rounds in [28]; under unknown plaintext/ciphertext scenario, only two rounds of HW leakages are required instead of three rounds in [28].

Application. To demonstrate the excellent error tolerance and new leakage model exploiting ability of MDASCA, in Section 5, we conduct a series of physical experiments on AES under different leakage models. Under the HW leakage model, AES implemented on an 8-bit microcontroller can be broken even when the HW deduction has 80% errors with a single power trace or 100% errors with two traces, which is better than previous results in CHES 2009 [28] and CHES 2010 [24]. Under the access driven cache leakage model, AES implemented on two typical microprocessors can be broken with only 1 and 36 cache traces, respectively, compared with 100 in IEEE S&P 2011 [2] and 300 in CT-RSA 2006 [25]. Under the trace driven cache leakage model, AES implemented on a 32-bit ARM microprocessor can be broken with only 5 cache traces instead of 30 in COSADE 2011 [15]. Moreover, all the experimental results of MDASCA are consistent with our theoretical analysis.

We describe the impacts of MDASCA in Section 7 and conclude this paper in Section 8.

2 MDASCA: Multiple Deductions-Based ASCA

2.1 Notations

To make our discussions concise and consistent, we first clarify some notations.

Deduction. In SCA, the output value of the leakage function for the targeted state obtained from side-channel leakages is called *deduction*, denoted as d. The specific meaning of the deduction highly depends on the leakage model.

Multiple Deductions. Due to the inaccurate measurements or the interferences from other components in the cryptosystem, the deduction from SCA is not always equal to the correct value. Instead, multiple values are obtained during the process, which are also referred to as *multiple deductions*.

Deduction Set. Multiple deductions are placed in a set, which is referred to as *deduction set*, denoted as D. Note that in the attack, the adversaries may also exploit the complement of the deduction set, denoted as $\bar{D}$, which includes the impossible values. The size of D ($\bar{D}$) is denoted as S_p (S_n) and is very important to the efficiency of ASCA. The elements in D ($\bar{D}$) are denoted as d_i ($\bar{d}_i$), $1 \leq i \leq S_p(S_n)$. We assume that the correct deduction d is always in D and not in $\bar{D}$ throughout this paper.

Deduction Offset. The distance between the deductions d_i and the correct one d is referred to as *deduction offset*, denoted as o_i, $o_i = d_i - d$. The value of o_i is very important when choosing the solving strategies (solvers) in ASCA.

Error Rate. In ASCA, the number of the targeted states where deductions are made is denoted as N_T. As to the possible deduction set D, the number of the targeted states where deductions are wrong is denoted as N_E. We define the *error rate* e as $e = \frac{N_E}{N_T}$.

2.2 MDASCA

Existing ASCAs [27, 28] add only a few equations to the algebraic system, assuming the deduction from leakages is single and correct. As a result, they are sensitive to errors and likely to fail in practical attacks. In this section, we propose an enhanced ASCA technique, named *Multiple Deductions-based ASCA (MDASCA)*, in which a deduction set of multiple values is created and converted into a constraint equation set. As long as the deductions are enumerable, the whole equation system can be solved by a SAT solver in a reasonable amount of time. Next, we describe the core of MDASCA, the representation of multiple deductions with algebraic equations.

Suppose a targeted state X can be represented with m one-bit variables x^j, j=1..m. $\phi(X)$ denotes the output value of the leakage function for X. If the correct deduction d can be deduced accurately, d can be calculated as in Eq. (1).

$$d = \phi(X), \qquad X = x^1 x^2 \dots x^m \tag{1}$$

Representing multiple deductions can be divided into the following two steps.

1. Building equations for the deduction set D or $\bar{D}$. $d_i \in D, 1 \leq i \leq S_p$, is a "possible deduction" on X. New variables B_i are introduced to represent X's value that generates d_i. Each B_i is represented as m one-bit variables b_i^j. New equations can be built as shown in Eq. (2).

$$B_i = b_i^1 b_i^2 \dots b_i^m, \qquad d_i = \phi(B_i), \quad 1 \leq i \leq S_p \tag{2}$$

Also, each $\bar{d}_i \in \bar{D}$ is an "impossible deduction". Similar to Eq. (2), new variables $\bar{B}_i$ and $\bar{b}_i^j$ are introduced. New equations can be built as in Eq. (3).

$$\bar{B}_i = \bar{b}_i^1 \bar{b}_i^2 ... \bar{b}_i^m, \quad \bar{d}_i = \phi(\bar{B}_i), \qquad 1 \leq i \leq S_n \tag{3}$$

Which set to use (D, $\bar{D}$, or both) is highly dependent on the leakage model and the adversaries' ability. Typically, if $S_p < S_n$, D is used because it leads to a less complicated equation system. Otherwise, $\bar{D}$ is preferred.

2. Building equations for the relationship between d and D (or $\bar{D}$). Note that if B_i is equal to X, d_i=d. $m \times S_p$ one-bit variables ${e_i}^j$ are introduced to represent whether ${b_i}^j$ is equal to x^j. ${e_i}^j$=1 if ${b_i}^j = x^j$; otherwise ${e_i}^j$=0. S_p one-bit variables c_i are introduced to represent whether d_i is correct or not. c_i=1 if d_i=d; otherwise c_i=0. c_i can be represented by Eq. (4), where $\neg$ denotes the NOT operation.

$${e_i}^j = \neg(x^j \oplus {b_i}^j), \quad c_i = \prod_{j=1}^{m} {e_i}^j \tag{4}$$

Since only one element in D is equal to d, only one c_i is 1. This can be represented as:

$$c_1 \vee c_2 \vee \dots \vee c_{S_p} = 1, \quad \neg c_i \vee \neg c_j = 1, \quad 1 \leq i < j \leq S_p \tag{5}$$

As for the impossible deductions, none of the elements in $\bar{D}$ is the correct deduction d. This can be represented by Eq. (6).

$${e_i}^j = \neg(x^j \oplus {b_i}^j), \quad c_i = \prod_{j=1}^{m} {e_i}^j = 0 \tag{6}$$

Let $n_{v,\phi}$ and $n_{e,\phi}$ denote the number of the newly introduced variables and ANF equations in representing one deduction d_i ($\bar{d}_i$). $n_{v,\phi}$ and $n_{e,\phi}$ depend on ϕ. According to Equations (2), (4), (5), $(1 + 2m + n_{v,\phi})S_p$ variables and $1 + (1 + m + n_{e,\phi})S_p + \binom{S_p}{2}$ ANF equations are introduced to represent D. According to Equations (3), (6), $(1+2m+n_{v,\phi})S_n$ variables and $(1+m+n_{e,\phi})S_n$ ANF equations are introduced to represent $\bar{D}$.

The new constraint equations mentioned above are quite simple. They can be easily fed into the SAT solver [33] to accelerate the key search. To launch MDASCA, it is important to choose ϕ and determine the deduction set $D/\bar{D}$ under different models, which is addressed in Section 3.

3 Analysis of Leakage Models in MDASCA

3.1 Hamming Weight Leakage Model with Errors

MDASCA can improve the error tolerance of ASCAs based on HW leakage model (HWLM). In such attacks, the adversaries try to deduce the HW of the targeted state X from measurements. In practice, due to noise, the adversaries may get wrong values of $HW(X)$ that are close to the correct one. For those implementations on some devices such as microcontrollers, the deduction offset for HW leakage is small and approximately ± 1 away from $HW(X)$, which is also addressed in [24]. Therefore, the possible HW deduction set D can be written as

$$D = \{HW(X) - 1, \quad HW(X), \quad HW(X) + 1\} \tag{7}$$

MDASCA can handle the deduction offset easily by setting ϕ=$HW()$, d=$HW(X)$. For example, if d=3, $D = \{2, 3, 4\}$.

3.2 Cache Leakage Models

Cache in the microprocessors can leak the secret information about the indexes of table(S-Box) lookups and compromise the cryptosystems. Numerous cache attacks on AES have been published. There are three leakage models in cache attacks: time driven (TILM) [4], access driven (ACLM) [2, 22, 25], and trace driven (TRLM) [1, 6, 7, 13–15, 20]. Under TILM, only the overall execution time is collected and it is difficult to deduce the internal states from few traces. Under ACLM and TRLM, adversaries can measure the cache-collisions and infer internal states with a single cache trace. Now we discuss how to use these two models in MDASCA.

Suppose a table has 2^m bytes and a cache line has 2^n bytes ($m > n$). The whole table will fill 2^{m-n} cache lines. A cipher process $\mathbb{V}$ performs k table lookups, denoted as $l_1, l_2, \ldots, l_k$. For each lookup l_i, the corresponding table index is X_i. Assume l_t is the targeted table lookup.

1.Access driven leakage model. Under ACLM [2, 22, 25], the cache lines accessed by $\mathbb{V}$ can be profiled by a malicious process $\mathbb{S}$ and used to deduce l_t. $\mathbb{S}$ first fills the cache with its own data before $\mathbb{V}$ performs the lookups, and accesses the same data after the lookups are done. $\mathbb{S}$ can tell whether a datum is in cache or not by measuring its access time. A shorter access time indicates a cache hit. A longer access time is a cache miss, implying that $\mathbb{V}$ already accessed the same cache line. If $\mathbb{S}$ knows which cache line that l_t accessed, he knows the higher $m - n$ bits of X_t. Let $\langle X \rangle$ denote the function that extracts the higher $m - n$ bits of X. Then the correct deduction for l_t is $\langle X_t \rangle$.

In practice, $\mathbb{S}$ observes many cache misses from two sources. Some are from the $k - 1$ lookups other than l_t. Some are from interfering processes that run in parallel with $\mathbb{V}$. Assume the interfering processes have accessed g different cache lines, which can be considered as g more "lookups" at $X_{k+1}, ...X_{k+g}$. All the possible values of $\langle X_t \rangle$ form a collection L. Without loss of generality, we

assume the first S_p values of L are distinct and $S_p \leq k+g$. The possible deduction set D can be written as:

$$D = \{d_1, .., d_{S_p}\}, \quad d_i = \langle X_i \rangle, \quad 0 \leq d_i < 2^{m-n} \tag{8}$$

Note that the impossible deduction set $\bar{D}$ can also be obtained, $S_n{=}2^{m-n} - S_p$. So ACLM can be easily interpreted with multiple deductions by setting $\phi = \langle \cdot \rangle$, $d = \langle X_t \rangle$ and $d_i = \langle X_i \rangle$. The values of elements in D or $\bar{D}$ are known to adversaries after deductions.

2.Trace driven leakage model. Under TRLM [1, 6, 7, 13–15, 20], $\mathbb{S}$ can keep track of the cache hit/miss sequence of all lookups to the same table of $\mathbb{V}$ via power or EM probes. Suppose there are r misses before l_t. Let $S_M(X)$ be the set of lookup indexes corresponding to the r misses, $S_M(X) = \{X_{t1}, X_{t2}, \ldots, X_{tr}\}$.

If l_t is a cache hit, the data that X_t tries to access have been loaded into the cache by previous lookups. The possible deduction set D for X_t can be written as in Eq. (9) where $S_p = r$.

$$D = \langle S_M(X) \rangle = \{\langle X_{t1} \rangle, \langle X_{t2} \rangle, \ldots, \langle X_{tr} \rangle\} \tag{9}$$

If a cache miss happens at l_t, the impossible deduction set $\bar{D}$ for X_t can be written as in Eq. (10) where $S_n = r$.

$$\bar{D} = \langle S_M(X) \rangle = \{\langle X_{t1} \rangle, \langle X_{t2} \rangle, \ldots, \langle X_{tr} \rangle\} \tag{10}$$

So TRLM can also interpreted under MDASCA by setting $\phi = \langle \cdot \rangle$, $d = \langle X_t \rangle$ and $d_i = \langle X_{ti} \rangle$. Different from ACLM, the elements in D or $\bar{D}$ are among the set of higher $m - n$ bits of table lookup indexes that cause cache misses. The exact value is unknown to adversaries even after the deductions.

4 Evaluation of MDASCA on AES

It is desirable to know how many leakages or traces are required in MDASCA. We take AES-128 as an example to evaluate it under HWLM, ACLM and TRLM. Suppose the master key is denoted as K, and the plaintext/ciphertext as P/C. We use $\xi(x)$ to denote the key search space for a variable x which is key dependent. A_i,B_i,C_i,D_i stand for the states after the AddRoundKey (AK_i), SubBytes (SB_i), ShiftRows (SR_i) and MixColumns (MC_i) in the i-th round.

4.1 HWLM Based MDASCA

Under both known P/C and unknown P/C scenarios, the attacks in [28, 29] require about 252 known HW leakages in three consecutive rounds to recover K. They assume there are 84 HW leakages in each round (16 in AK, 16 in SB, and 52 in MC) and the deduction offset is 0. However, the number of HW leakages that required can be less.

Consider a byte x, if its HW is leaked, we can calculate that $\xi(x){=}50.27{=}2^{5.65}$. It is equivalent to say that $\xi(x)$ is reduced by about $2^{8-5.65}{=}2^{2.35}$. If both $HW(x)$

and $HW(S[x])$ are leaked in SubByte, $\xi(x)$=10.69=$2^{3.42}$. These numbers are calculated by an algorithm in Appendix 1.

Taking the assumptions as in [28, 29], we evaluate MDASCA on AES under unknown P/C scenarios. With 16 bytes leaked in AK_1 and SB_1, $\xi(B_1)$=$2^{3.42\times16}$= $2^{54.72}$. In MC_1, four bytes in a column can be calculated with 13 steps [28, 29]. Simply considering the last four steps which write the output to D_1, we estimate that each column can reduce $\xi(D_1)$ for about $2^{2.35\times4}$. Taking all four columns into account, $\xi(D_1)$ should be reduced to $2^{54.72-2.35\times4\times4} \approx 2^{17.12}$ approximately. $\xi(D_1)$ can be further reduced to 1 if other nine leakages can be utilized. We believe 84 HW leakages can obtain A_1,B_1,D_1, which also applies to A_2,B_2,D_2.

We verify this with a C program and later with experiments in Section 5.2. Two rounds of leakages can obtain the second round key by XORing D_1 and A_2. Then the master key K can be recovered via solving the equations on the key schedule of AES. Note that the known P/C scenario is just a special case of above. One round of leakages should be enough to recover the first round key K since both A_1 and P are known.

4.2 ACLM Based MDASCA

ACLM based cache attacks on AES have been widely studied [2, 22, 25]. This paper takes AES implementations in OpenSSL1.0.0d as the target, where n=6 and m=11,10,8 for table with the size of 2KB, 1KB or 256 bytes. Accordingly, the leakages (*i.e.*, $\epsilon = m - n$) are the higher 5, 4 and 2 bits of table lookup index, respectively. This is equivalent to say that one leakage reduces $\xi(K)$ about 2^{ϵ}. We consider two typical ACLM models.

1.Bangerter model. Bangerter *et al.* [2] launched attacks on AES in OpenSSL 1.0.0d with one 2KB table and 100 encryptions, assuming $\mathbb{S}$ can profile the accessed cache lines per lookup. In the attack, the CFS Scheduler of Linux and Hyper thread techniques are required. We refer to their work as *Bangerter model*.

In the attack, there are 16 leakages of table lookups in each round. After the first round analysis, $\xi(K)$ can be reduced to $2^{(8-\epsilon)\times16}$. After the second round analysis, $\xi(K)$ can be approximately reduce to $2^{(8-2\epsilon)\times16}$. As to ϵ=4 or 5, two rounds of leakages from one cache trace are enough to recover K. As to ϵ=2, after the first round analysis, $\xi(K)$ can be reduced to $2^{6\times16}$. After the second round analysis, three cache traces are enough to recover $\xi(K)$ into $2^{(6-2\times3)\times16} = 1$.

2.Osvik model. Osvik *et al.* [25] and Neve *et al.* [22] conducted ACLM based cache attacks on AES in OpenSSL 0.9.8a, assuming $\mathbb{S}$ can profile the accessed cache lines per encryption. We refer to their work as *Osvik model*.

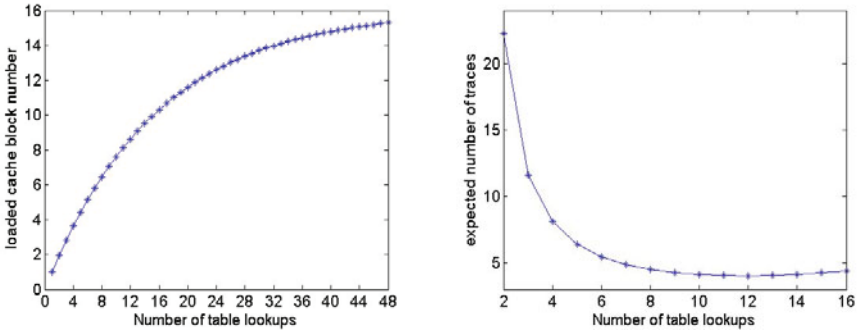
In OpenSSL0.9.8a, four 1KB tables (T_0, T_1, T_2, T_3, m=10) are used in the first nine rounds, and a table T_4 is used in the last round. The attacks in [25] can succeed with 300 samples by utilizing the 32 table lookups in the first two rounds of AES. The attack in [22] can break AES with 14 samples by utilizing the 16 T_4 table lookups in the final round of AES. AES in OpenSSL1.0.0d removes T_4 and uses T_0, T_1, T_2, T_3 throughout the encryption. As a result, it is secure against [22] and increase the difficulties of [25].

We implemented AES in OpenSSL1.0.0d with m=10, n=6 and ϵ=4. In total, there are 40 cache accesses to the same table in one encryption. Under Osvik model [25], $16 \times (\frac{15}{16})^{40} \approx 1.211$ cache lines will not be accessed by $\mathbb{S}$, which means on average there are 1.211 elements in the impossible deduction set for the table lookup indexes. Theoretically speaking, around $\frac{15}{1.211} \approx 12.386$ traces with one round of leakages can recover the high 4-bit of table lookup indexes, which reduces $\xi(K)$ to 2^{64}. Utilizing the 16 table lookups in the second round, $\xi(K)$ can be further reduced to 1. In practice, the number of traces required will increase a little bit, as shown in Section 5.3.

4.3 TRLM Based MDASCA

The first TRLM based cache attack on AES was proposed by Bertoni *et al.* [6] and later improved in [13] through real power analysis. More simulation results with 1KB table were proposed in [1, 6, 7, 20]. Recently, several real-world attacks were proposed in [14, 15] on AES with 256B table on a 32-bit ARM microprocessor. In [14], the cache events are detected with a power probe and only the first 18 lookups are analyzed. 30 traces reduce $\xi(K)$ to 2^{30}. In [15], the attacks are done with an EM probe and $\xi(K)$ is further reduced to 10 when two more lookups are considered.

Let p_i and k_i be the i-th byte of the plaintext and the master key. The i-th lookup index is $p_i \oplus k_i$. In TRLM, the XOR between two different lookup indexes ($p_i \oplus k_i$ and $p_j \oplus k_j$) can be leaked. From the 16 leakages in the first round, $\langle p_i \oplus k_i \oplus p_j \oplus k_j \rangle$ can be recovered, $1 \leq i < j \leq 16$. $\xi(K)$ can be reduced to $2^{128-15*\epsilon}$ and further reduced to 1 by analyzing leakages in the later rounds.



(a) $\#\ell$ (b) N_l

Fig. 1. Estimations in TRLM based MDASCA on AES

To calculate N_ℓ, the number of cache traces required for a successful attack, we need to know $\#\ell$, the expected number of cache lines that are updated by $\mathbb{S}$ after the first l lookups. Let N_c=2^{m-n} be the number of cache lines that one table can fill. The probability for one cache line not updated by $\mathbb{S}$ after l

table lookups is $(\frac{N_c-1}{N_c})^\ell$. And $\#\ell = N_c(1-(\frac{N_c-1}{N_c})^\ell)$. Let y_ℓ be the ℓ-th lookup index, and ρ_ℓ be the reduced percentage of $\xi(y_\ell)$ due to the ℓ-th lookup. Then $\rho_\ell = (\frac{\#\ell}{N_c})^2 + (1-\frac{\#\ell}{N_c})^2$, as also shown in [1]. Let $N_c \times (\rho_\ell)^{N_l} \leq 1$, $N_\ell \approx -log_{\rho_\ell}^{N_c}$.

In [14, 15], N_c=16. Fig. 1(a) shows how $\#\ell$ changes with ℓ. It's clear to see that even after 48 table lookups, $\#\ell < 16$ and $\rho_\ell < 1$, which means there are still some cache misses that could be used for deductions. Fig. 1(b) shows how N_ℓ changes with ℓ, where the minimal of N_ℓ is 4, and the maximal is 22.24. If N_ℓ is 5 or 6, $\xi(K)$ can be reduced to $2^{76.10}$ and $2^{74.13}$, respectively.

Using the leakages in the second round, if N_ℓ is 5 or 6, ξ can be further reduced to $2^{76.10-48.80}$=$2^{27.30}$, and $2^{74.13-54.78}$=$2^{19.35}$ approximately. After the third round, $\xi(K)$ can be reduced to 1 with a high probability. So approximately 5 or 6 cache traces are enough to launch a successful TRLM based attacks on AES. As it is really hard to analyze the third round leakages manually, we will just verify it through MDASCA experiments, as shown in Section 5.4.

5 Application of MDASCA on AES

5.1 Experiment Setup

We conduct attacks under HWLM, ACLM and TRLM. Table 1 presents the experiment setups.

Table 1. Experiment setup of MDASCA on AES

Leakage Model	Targeted platform	AES Implementation	Note
HWLM	8-bit microcontroller ATMEGA324P	AES with compact table	
ACLM	Intel Pentium 4 processor[1], Fedora 8 Linux	AES in OpenSSL 1.0.0d	Bangerter model [2]
ACLM	Athlon 64 3000+ processor[2], Windows XP SP2	AES in OpenSSL 1.0.0d	Osvik model [25]
TRLM	32-bit ARM microprocessor NXP LPC2124	AES with compact table	

We adopt the technique in [19] to build the equation set for AES. Each S-Box can be represented by 254 ANF equations with 262 variables. The full AES-128 including both encryption and key scheduling can be described by 58288 ANF equations with 61104 variables. MDASCA does not require the full round of AES, as analyzed in Section 4. We choose the CryptoMiniSat 2.9.0 solver [33] to solve the equations. The solver is running on an AMD Athlon 64 Dual core 3600+ processor clocked at 2.0GHz. We consider that a trial of MDASCA fails when no solution is found within 3600 seconds.

5.2 Case Study 1: HWLM Based MDASCA on AES

As shown in Fig. 2(a), the instant power of AES-128 running on ATMEGA324P is highly correlated to the HW of the data. Similar to [24], we deduce the HW for 84 intermediate states in each round. According to Section 4.1, our attacks only require a few rounds of HW leakages. We calculate the Pearson correlation factor [8] when deducing the HWs from power traces.

[1] L1 Cache setting: 16KB cache size, 8 way associative, 64B cache line size.
[2] L1 Cache setting: 64KB cache size, 2 way associative, 64B cache line size.

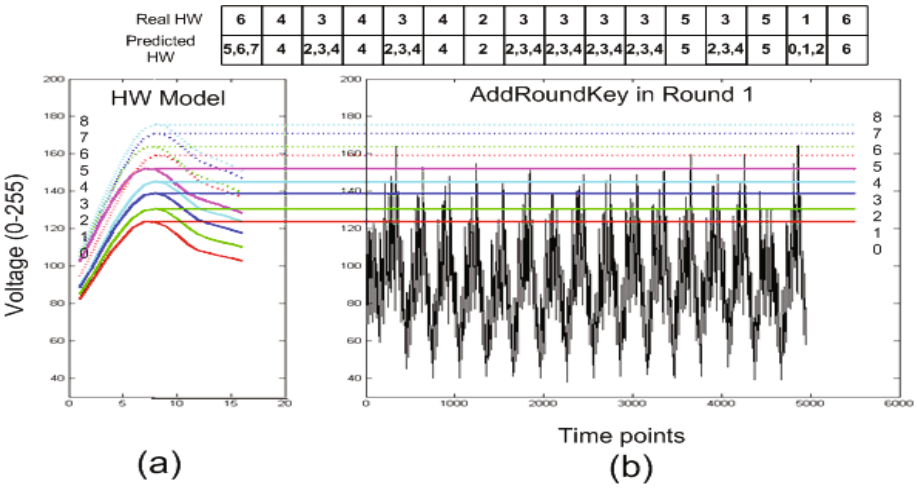


Fig. 2. HW deductions in MDASCA on AES

Fig. 2(b) shows the AddRoundKey of the first round in a single power trace where the offset is one. In this example, from Section 2.1, the error rate e is $\frac{9}{16} = 56.25\%$. The deduction set for the 8−th byte is D={2, 3, 4}, $S_p = 3$. To represent each HW deduction in D, 99 new variables (not including the 8 variables in Eq. (2)) and 103 equations are required(see Appendix 2). So, $n_{v,\phi} = 99, n_{e,\phi} = 103$. According to Section 2.2, we use 348 new variables and 340 ANF equations to represent D.

As in [28], we considered several attack scenarios: known or unknown P/C, consecutive or random distributions of correct HW deduction. Since the PBOPT solver used in [24] fails on AES even when there is no error, we only compare our results with [28]. We repeat the experiment for each scenario 100 times and compute the average time. Our results indicate that, although the SAT solver has some smart heuristics, most of the trials (more than 99%) succeed in reasonable time with small variation.

Table 2 lists how many rounds are required for different scenarios. With one power trace, when leakages are consecutive, and if P/C is known, only one round is required instead of 3 in [28]; if P/C is unknown, 2 rounds are required instead of 3 in [28]. The results are consistent with the analysis in Section 4.1.

Table 2. Comparisons of HWLM based MDASCA on AES with previous work

Scenarios	error type	leakage type	[28]	MDASCA
known P/C	error free	consecutive	3 rounds	1 round(10 seconds)
known P/C	error free	random	8 rounds	5 rounds (120 seconds)
unknown P/C	error free	consecutive	3 rounds	2 rounds (10 seconds)
unknown P/C	error free	random	8 rounds	6 rounds (100 seconds)
known P/C	80% error rate	consecutive	-	3 rounds (600 seconds)
known P/C	100% error rate	consecutive	-	2 rounds (120 seconds, 2 power traces)
known P/C	100% error rate	consecutive	-	1 rounds (120 seconds, 3 power traces)

Under HWLM, the average HW deduction error rate of a single power traces is about 75%, which is also indicated in [27, 28]. MDASCA can succeed even with 80% error rate by analyzing 3 consecutive rounds in a single trace within 10 minutes. Even when the error rate is 100% (the number of all the HW deductions is 3), AES can still be broken by analyzing two consecutive rounds of two power traces within 2 minutes, or one round of three traces with 2 minutes. From the above, we can see that MDASCA has excellent error tolerance and can significantly increase the robustness and practicability of ASCA.

Note that MDASCA can also exploit larger number of HW deductions, e.g, 4. If the HW leakages are not enough for the solver to find the single and correct solution, a full AES encrypt procedure of an additional P/C can be added into the original equation set to verify the correctness of all the possible solutions. The time complexity might be a bit higher without increase the data complexity.

5.3 Case Study 2: ACLM Based MDASCA on AES

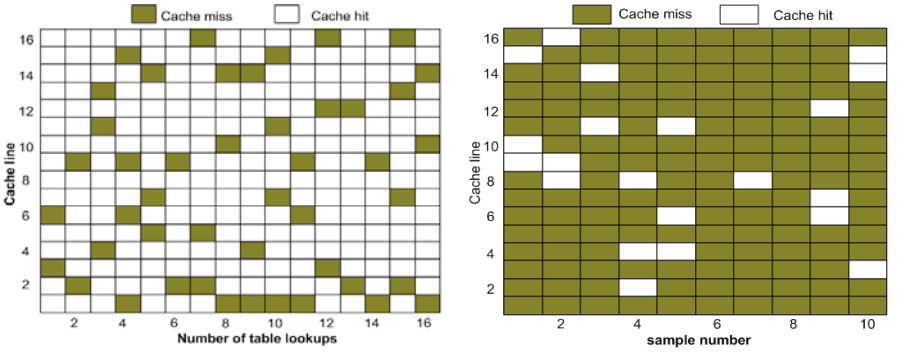
We conduct ACLM based MDASCA on AES under both Bangerter [2] and Osvik [25] model. The comparisons of MDASCA with previous work are listed in Table 3. The results are consistent with the analysis in Section 4.2.

Under Bangerter model, we apply MDASCAs to three AES implementations in OpenSSL1.0.0d with 2KB, 1KB or 256B table. Fig. 3(a) shows the cache events of 16 lookups in the first round with four 1KB tables. According to our experience, there are 1-4 cache misses during each lookup due to the noises from other system processes. Take the third column of Fig. 3(a) as an example. We have 3 possible deductions on $\langle X_3 \rangle$, and D=$\{4,11,13\}$, $S_p = 3$. To represent each deduction in D, no new variables are introduced and 4 assignment equations are required, $n_{v,\phi} = 0, n_{e,\phi} = 4$. In total, 31 ANF equations with 27 additional variables are introduced.

Table 3. Comparisons of ACLM based MDASCA with previous work

Attacks	AES implementation	Leakage model	Scenarios	samples	time
[2]	2KB table	Bangerter	known P (unknown P/C)	100	3 minutes
MDASCA	2KB table	Bangerter	known P	1	6 seconds
MDASCA	2KB table	Bangerter	unknown P/C	2	60 seconds
MDASCA	1KB table	Bangerter	known P	1	15 seconds
MDASCA	1KB table	Bangerter	unknown P/C	2	120 seconds
MDASCA	256B table	Bangerter	known P	3	60 seconds
[25]	1KB table	Osvik	known P	300	65 milliseconds
MDASCA	1KB table	Osvik	known P	36	1 hour

As shown in Table 3, only up to three traces are required in MDASCA, in contrast to 100 samples in [2]. In particular, when AES is implemented with 256B table, the attacks in [2] failed. This is because the leakages (the high 2-bit of the lookup index) is small and the number of rounds that can be analyzed is limited. MDASCA can utilize leakages of all rounds and only three cache traces are required, which is the first successful ACLM based cache attack on AES with a compact table.



(a) Cache events sampled after each table lookup

(b) Cache events sampled after one encryption

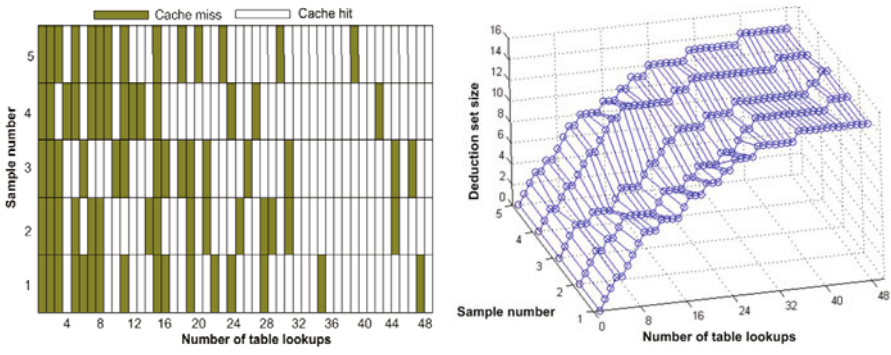
Fig. 3. Profiled ACLM based leakages of $\mathbb{V}$ by $\mathbb{S}$

Under Osvik model, we apply MDASCAs to AES implementation in OpenSSL 1.0.0d with four 1KB tables (such implementation can well defend attack in [22]). Fig. 3(b) shows the events of 16 cache lines in 10 encryptions related for T_0 in real system. About 13-16 cache lines (colored in cyan) have misses, which means that there are about 0-3 impossible deductions for the high 4-bit of every table lookup index $\langle X_i \rangle$. Take the first column as an example, there are 3 impossible deductions for $\langle X_2 \rangle$, $\bar{D}$={9,10,15}, $S_n = 3$. Therefore, 27 ANF equations with 27 variables are introduced.

From Table 3, 36 traces can recover the full key comparing to the 300 traces in [25] (in fact, to attack the same AES implementation in OpenSSL 0.9.8.a as in [25], only 30 traces are required by MDASCA). In the attack, we first try to directly use the equations generated by ACLM based leakages. Experimental results show that the whole system cannot be solved even within one day. Then, to accelerate the equation solving process, we iterate the values of several key bits into the equation system besides the equations generated from leakages. For example, four key bits need 16 enumerations. The results show that if the 4 key bits of the input value are correct, it can be solved in 40 minutes. Otherwise, the solver will output "unsatisfiable" within 10-20 seconds. We repeat the tests for about 100 times. On average one hour is enough to recover the AES-128 key.

5.4 Case Study 3: TRLM Based MDASCA on AES

In TRLM based MDASCA, we implement AES on an 32-bit ARM microprocessor NXP LPC2124 with a direct mapped cache, and profile the cache events via EM probes, as in [15]. The cache line size is 16 bytes. The table size is 256 bytes and can fill up with 16 cache lines (ϵ=4, N_c=16). According to the analysis in Section 4.3, the cache events of the first three rounds are utilized in the attack. In an EM trace, the cache miss has a distinct peak. Thus cache hit/miss events can be easily distinguished.



(a) Cache events in five encryptions (b) Deduction set size

Fig. 4. Profiled Cache trace in TRLM Based MDASCA

Fig. 4(a) shows the cache events of the first 48 lookups (first 3 rounds) in 5 cache traces. The table lookups in the first round are more likely to cause misses. The probabilities of cache hit increase in following rounds. However, even after 48 lookups, there is still high probability that full 16 cache lines on the 256B table have not been updated yet, consistent with the analysis in Section 4.3. Let the table lookup index be y_i $(1 \leq i \leq 48)$. Fig. 4(b) shows the number of deductions for 48 table lookup indexes of the 5 cache traces. This number is increased with the table lookup number, and the range is 0-15 for these 5 traces.

Take the 8-th and 9-th lookups (also l_8 and l_9) of the first sample in Fig. 4(a) as examples. As to l_8, a cache miss is observed. Then the impossible deduction set of $\langle y_8 \rangle$ is $\bar{D} = \{\langle y_1 \rangle, \langle y_2 \rangle, \langle y_3 \rangle, \langle y_5 \rangle, \langle y_6 \rangle, \langle y_7 \rangle\}$, $S_n = 6$. Note that all the variables of $\bar{D}$ have already been represented in the AES algebraic equation system, and $n_{v,\phi} = 0, n_{e,\phi} = 0$. We only needs to compute the new introduced variables and equations by Eq. (6). According to Section 2.2, 30 ANF equations with 30 variables can be generated. As to l_9, a cache hit happens. From Section 2.2, the possible deduction set of $\langle y_9 \rangle$ (higher four bits of y_9) is $D = \{\langle y_1 \rangle, \langle y_2 \rangle, \langle y_3 \rangle, \langle y_5 \rangle, \langle y_6 \rangle, \langle y_7 \rangle, \langle y_8 \rangle\}$, $S_p = 7$. As all the variables of D have been represented in the AES system, according to Section 2.2, 57 ANF equations with 35 variables can be added to the equation system.

For some table lookups, it is hard to tell whether they are cache miss or hit because the peak is not high enough. In our MDASCA, we treat uncertain cache events as cache hits. In some other scenarios, partially preloaded cache is also considered and more cache hits are observed. Our MDASCA only utilizes cache misses and still works in this case.

As in [15], we conduct several TRLM based MDASCAs on AES considering three scenarios: with both cache hit and miss events, with cache miss events only, with cache miss events and four preloaded cache lines. Each attack in the three cases is repeated for 100 times. To accelerate the equation solving procedure, we also input the candidates of 4 key bits into the equation system and launch 16 ASCA instances corresponding to 16 possible candidates. The comparisons of our results with previous work are listed in Table 4.

Table 4. Comparisons of TRLM based MDASCA on AES with previous work

Attacks	Utilized collisions	Collision type	Preloaded cache lines	Sample size	Key space	time
[13]	16 lookups	H/M	0	14.5	2^{68}	-
[14]	18 lookups	H/M	0	30	2^{30}	-
[15]	20 lookups	H/M	0	30	10	-
MDASCA	48 lookups	H/M	0	5 (6)	1	1 hour(5 minutes)
[15]	20 lookups	M	0	61	-	-
MDASCA	48 lookups	M	0	10	1	1 hour
[15]	20 lookups	M	4	119	-	-
MDASCA	48 lookups	M	4	24	1	1 hour

From Table 4, TRLM based MDASCA can exploit cache behaviors of three AES rounds (48 table lookups) and achieve better results than previous work [14, 15]. At least five cache traces are able to recover 128-bit AES key within an hour. The complexity for both online (number of measurements) and offline (recovering the key from the measurements) phases has been reduced. Moreover, the results are also consistent with the theoretical analysis in Section 4.3.

6 Impact of MDASCA

The impacts of MDASCA in this paper can be understood as follows.

The first impact is on error tolerance. Providing error tolerance can increase the robustness and practicability of ASCA. This can be addressed with two approaches. One is to embed the error toletance into the solver, as in TASCA [28]. The SCIP solver [5] used in [28] requires the small errors continuously distributed around the correct value (e.g., under HWLM), and might not work under ACLM and TRLM, where the error offset is discrete, unpredictable and large. The diversities among different leakage models play as the major barrier. The other approach is what MDASCA does. The errors are preprocessed and represented with new equations. The overhead of the approach includes new variables and equations. However, the cryptanalysts can now focus on the leakage utilization and reduce the affects from the solver. Moreover, our results of MDASCA show that the complexity of solving equations is not prohibitively high and most of the instances can be solved within reasonable times with a small variance..

The second impact is on application of attacks. Previous ASCAs [24, 27, 28] work well on some small devices (e.g., microcontroller), where the power is highly correlated to the HW and easy to measure. How can we adopt ACSA under different scenarios, such as ACLM and TRLM, where those advantages never exist? For the first time MDASCA extends the applications of ASCA to more complicated models. Considering the widely used microprocessors in common PC and embedded devices, it is difficult to launch HWLM based ASCA on them. Cache attacks are more practical. Previous attacks [1, 2, 4, 6, 7, 13–15, 20, 22, 25] on AES can only use the leakages in the first two rounds due to the complexity of representing the cache leakages of the targeted states. MDASCA can exploit the cache leakages in more rounds, even in all the rounds. Thus the complexity of the attack and the required measurements are dramatically reduced.

7 Conclusion and Future Work

Due to the existence of noises and the intrinsic feature of leakage models, correct deductions in ASCA are often hidden in multiple candidates. This paper proposes an enhanced ASCA attack called Multiple Deductions-based ASCA (MDASCA) to exploit these candidates. A generic method is described to represent multiple deductions with algebraic equations. Several leakage models suitable for MDASCA are analyzed and the details of the leakage exploitation are also provided. For the first time, we evaluate the minimal amount of leakages for MDASCA on AES under these models. To verify the practicality and theoretical analysis, we have successfully launched real MDASCA attacks under different models, and achieved better results.

From this paper, MDASCA attests again that combining algebraic techniques with SCA is a promising way to fully utilize the leakages. Future works of MDASCA consequently include the solver improvement (try different solvers for better performances and solving capabilities), application extension (to different ciphers, leakage models and implementations) and security evaluation (as a benchmark to evaluate the physical security of ciphers).

Acknowledgments. The authors would like to thank Francois-Xavier Standaert, Yu Yu, Ruilin Li, Siwei Sun, Zheng Gong and the anonymous referees for helpful discussions and comments.

References

1. Acıiçmez, O., Koç, Ç.: Trace Driven Cache Attack on AES. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 112–121. Springer, Heidelberg (2006)
2. Bangerter, E., Gullasch, D., Krenn, S.: Cache Games - Bringing Access-Based Cache Attacks on AES to Practice. In: IEEE S&P 2011, pp. 490–505 (2011)
3. Batina, L., Gierlichs, B., Prouff, E., Rivain, M., Standaert, F.X., Veyrat-Charvillon, N.: Mutual Information Analysis: A Comprehensive Study. Journal of Cryptology 24, 269–291 (2011)
4. Bernstein, D.J.: Cache-timing attacks on AES (2004), `http://cr.yp.to/papers.html#cachetiming`
5. Berthold, T., Heinz, S., Pfetsch, M.E., Winkler, M.: SCIP C solving constraint integer programs. In: SAT 2009 (2009)
6. Bertoni, G., Zaccaria, V., Breveglieri, L., Monchiero, M., Palermo, G.: AES Power Attack Based on Induced Cache Miss and Countermeasure. In: ITCC 2005, pp. 586–591. IEEE Computer Society (2005)
7. Bonneau, J.: Robust Final-Round Cache-Trace Attacks Against AES. Cryptology ePrint Archive (2006), `http://eprint.iacr.org/2006/374.pdf`
8. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
9. Courtois, N., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)

10. Courtois, N., Ware, D., Jackson, K.: Fault-Algebraic Attacks on Inner Rounds of DES. In: eSmart 2010, pp. 22–24 (September 2010)
11. Dinur, I., Shamir, A.: Side Channel Cube Attacks on Block Ciphers. Cryptology ePrint Archive (2009), http://eprint.iacr.org/2009/127
12. Faugère, J.-C.: Gröbner Bases. Applications in Cryptology. In: FSE 2007 Invited Talk (2007), http://fse2007.uni.lu/slides/faugere.pdf
13. Fournier, J., Tunstall, M.: Cache Based Power Analysis Attacks on AES. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 17–28. Springer, Heidelberg (2006)
14. Gallais, J., Kizhvatov, I., Tunstall, M.: Improved Trace-Driven Cache-Collision Attacks against Embedded AES Implementations. In: Chung, Y., Yung, M. (eds.) WISA 2010. LNCS, vol. 6513, pp. 243–257. Springer, Heidelberg (2011)
15. Gallais, J., Kizhvatov, I.: Error-Tolerance in Trace-Driven Cache Collision Attacks. In: COSADE 2011, pp. 222–232 (2011)
16. Goyet, C., Faugre, J., Renault, G.: Analysis of the Algebraic Side Channel Attack. In: COSADE 2011, pp. 141–146 (2011)
17. Handschuh, H., Preneel, B.: Blind Differential Cryptanalysis for Enhanced Power Attacks. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 163–173. Springer, Heidelberg (2007)
18. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
19. Knudsen, L.R., Miolane, C.V.: Counting equations in algebraic attacks on block ciphers. International Journal of Information Security 9(2), 127–135 (2010)
20. Lauradoux, C.: Collision Attacks on Processors with Cache and Countermeasures. In: WEWoRC 2005. LNI, vol. 74, pp. 76–85 (2005)
21. Improved Differential Fault Analysis of Trivium. In: COSADE 2011, pp. 147–158 (2011)
22. Neve, M., Seifert, J.: Advances on Access-Driven Cache Attacks on AES. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 147–162. Springer, Heidelberg (2007)
23. FIPS 197, Advanced Encryption Standard, Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, November 26 (2001)
24. 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 2010. LNCS, vol. 6225, pp. 428–442. Springer, Heidelberg (2010)
25. Osvik, D.A., Shamir, A., Tromer, E.: Cache Attacks and Countermeasures: The Case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)
26. Percival, C.: Cache missing for fun and profit (2005), http://www.daemonology.net/hyperthreading-considered-harmful/
27. Renauld, M., Standaert, F.-X.: Algebraic Side-Channel Attacks. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) Inscrypt 2009. LNCS, vol. 6151, pp. 393–410. Springer, Heidelberg (2010)
28. Renauld, M., Standaert, F., Veyrat-Charvillon, N.: Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 97–111. Springer, Heidelberg (2009)
29. Renauld, M., Standaert, F.-X.: Representation-, Leakage- and Cipher- Dependencies in Algebraic Side-Channel Attacks. In: Industrial Track of ACNS 2010 (2010)
30. Roche, T.: Multi-Linear cryptanalysis in Power Analysis Attacks. MLPA CoRR abs/0906.0237 (2009)

31. Schramm, K., Wollinger, T.J., Paar, C.: A New Class of Collision Attacks and Its Application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
32. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal 28 (1949); see in particular page 704
33. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT Solvers to Cryptographic Problems. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 244–257. Springer, Heidelberg (2009)
34. Whitnall, C., Oswald, E., Mather, L.: An Exploration of the Kolmogorov-Smirnov Test as Competitor to Mutual Information Analysis. Cryptology ePrint Archive (2011), http://eprint.iacr.org/2011/380.pdf

Appendix 1: Algorithm to Calculate $\xi(x)$ under HWLM

Algorithm 1 computes $\xi(x)$ from two input parameters. The first is n, the number of bits in x. The second is m. If m is 1, the algorithm outputs $\xi(x)$ for the cases that $HW(x)$ is known. Otherwise, it outputs $\xi(x)$ for the cases that both $HW(x)$ and $HW(S(x))$ are known, where $S(x)$ is the S-Box result of x.

Algorithm 1. Compute the search space of x

```
1: Inputs: n, m
2: Output: ξ(x) the expected search space of x
3: int i, j, sum=0;
4: for i=0 to 2^n do
5:     for j=0 to 2^n do
6:         if(HW(i) == HW(j))
7:             if(m == 1 or HW(S(i)) == HW(S(j)))
8:                 sum++;
9:             end if
10:        end if
11:    end for
12: end for
13: return ξ(x) = ( float)sum/2^n;
```

Appendix 2: Hamming Weight Representation of a Byte

Suppose X is a byte,containing 8 bits $(x_7 \dots x_0)$. $HW(X)$ can be represented with a 4-bit value $Y{=}(y_3 \dots y_0)$. x_0 and y_0 denote the LSBs. Bits in Y can be calculated as:

$$\begin{aligned} y_3 &= \prod_{i=0}^{7} x_i, \qquad y_2 = \sum x_i x_j x_m x_n (0 \le i < j < m < n \le 7), \\ y_1 &= \sum x_i x_j (0 \le i < j \le 7), \qquad y_0 = \sum_{i=0}^{7} x_i \end{aligned} \tag{11}$$