Error-Tolerance in Trace-Driven Cache Collision Attacks

Jean-François Gallais and Ilya Kizhvatov

University of Luxembourg
Faculty of Science, Technology and Communication
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
{jean-francois.gallais, ilya.kizhvatov}@uni.lu

Abstract. We present enhancements of the trace-driven cache collision attack against embedded AES implementations presented at WISA 2010. First, we improve the attack to reduce the remaining exhaustive search complexity from 2^{32} to at most 10 AES encryptions. Second, we extend the tolerance to errors in cache event detection to the full attack and show that the attack is efficient even for the significant error probabilities. Finally, we show that previous univariate models for estimating attack complexity are not good, and present the multivariate model which is easy to simulate. Our attack is comparable to DPA in terms of complexity, while being of a different nature. We also show by further explorations on an ARM platform that cache events are distinguishable in practice.

1 Introduction

Cache-collision attacks. Cache mechanism of the microprocessors has been known to be able to leak key-dependent information and therefore to be exploitable in side-channel attacks for about 10 years [9,17]. Since then, many cache-based attacks were reported; they fall into three different types. *Time-driven attacks* like [2] exploit the dependence of the execution time of an algorithm on the cache accesses. In *access-driven attacks* presented in [13,16], an attacker learns which cache lines were accessed during the execution by pre-loading the cache with the chosen data (which requires running a spy process on the device in parallel to the target algorithm). Finally, in *trace-driven attacks* first suggested in [3] an adversary derives information from individual cache events from the side-channel trace of an execution, such as registered power consumption or electromagnetic emanations. Trace-driven attacks pose a particular threat to embedded devices since the latter are exposed to a high risk of power or electromagnetic (EM) analysis, as opposed to desktop and server implementations that are a usual target in access- and time-driven cache attacks.

Like collision [21,4] and algebraic attacks [20], trace-driven cache collision attacks recover the key from the relations yielded by the side channel leakage. In this, they are different from differential power analysis (DPA) [10] which is a divide-and-conquer attack known to be very practical. Our motivation here is to make trace-driven cache collision attacks comparable, if not superior, to DPA in terms of both online (number of measurements) and offline (recovering the key from the measurements) complexity.

Our contributions. In this work, we elaborate on trace-driven attacks applied to embedded software AES implementations. Our practical explorations show that cache events corresponding to AES S-Box lookups are well distinguishable in the EM radiation of the device (Section 4.1). However, side-channel measurements may be noisy in a real-life scenario, making it hard to distinguish cache events reliably. In this case, our attacks allow to tolerate errors (uncertainties) in cache event detection. We show (Section 4) that even for significant probabilities of detection uncertainties, our attacks still require a feasible number of measurements to succeed. Our attacks can also tolerate another realistic case when a part of the lookup table is already in the cache by the start of the execution. We also improve the

complexity of the offline attack stage compared to [8], from 2^{30} to just 10 AES encryptions (Section 3). Finally, we scrutinize the theoretical model of the attacks and show that both models of [1] and [8] are not good since they are univariate, whereas the real attack follows the multivariate model. This model is easy to simulate. We conclude with the practical results in Section 6.

2 Generalities and Previous Work

Previous work on trace-driven cache attacks was done in [3,12,1,7,22]. These works considered at most simulations of the side-channel leakage of the cache and assumed a noise-free environment. Recent work [8] addressed the issue of error tolerance in trace-driven attacks and explored the leakage of cache events in the power consumption of an ARM microcontroller. A novel trace-driven cache attack against CLEFIA [19] exploiting differential cryptanalysis methods has been carried out in practice using the power consumption measurements on the PowerPC processor of a Xilinx Virtex-II FPGA. In this section, we present the assumptions about the cache mechanism and the implementation, and recall the attack introduced in [8].

2.1 Notation

Throughout the paper, we follow the notation of [8]. We denote

- the most and the least significant nibbles of a byte b with $\widehat{b}$ and $\check{b}$ correspondingly;
- the input of the `SubByte` function in the first AES round as x_i, equal to $p_i \oplus k_i$, where p_i and k_i respectively represent a plaintext byte and key byte, $0 \leq i \leq 15$;
- addition and multiplication over GF(2^8) by $\oplus$ and $\bullet$ respectively.

We index the bytes row-wise and *not* column-wise as in the AES specification [23,6], *i.e.* in our notation p_0, p_1, p_2, p_3 is the first row of a 16-byte plaintext.

2.2 Cache Mechanism and Assumptions

Here we present our assumptions about the cache mechanism and the implementation of the AES lookup table that will be used in our attacks. In general, we follow the description made in [5].

We assume that the AES implementation uses lookup tables of 256 entries. Let b be the size of a table entry in bytes. In case of an S-Box $b = 1$, in case of T-tables used in optimized implementations [6] $b = 4$. Let l be the cache line size in bytes. In modern embedded microcontrollers common sizes are $l = 16$ and $l = 32$. Then, we have $\delta = l/b$ entries per cache line and $m = 256 \cdot b/l$ cache blocks per lookup table (note that $\delta \cdot m = 256$). The value of δ (or, equivalently, m) has effect on the attack complexity since cache events are determined by equalities/inequalities of $8 - \log_2 \delta$ higher order bits of the inputs to the lookups. We assume that the lookup table is aligned with the cache. Recent work [22] has shown that the attack is also possible when lookup tables are misaligned.

As in [8], here we present our attacks for the case $b = 1$, $l = 16$. The attacks are adaptable to other cache and lookup table configurations. We carry out theoretical analysis (Section 5) in general for different cache line sizes, *i.e.* for different values of m.

We assume that in an embedded software AES implementation S-Box lookups are performed row-wise (and therefore our row-wise notation simplifies the description of the attack algorithms). The adversary is dealing with a sequence of *observed* cache events, misses or hits, occurred in the first two rounds of the implementation. We call this sequence a *cache*

trace. A cache trace can be recovered from a side channel trace, as we show in Section 4. Since there may be uncertainties in distinguishing a miss from a hit, we also introduce an additional type of observed cache event: the uncertain event.

Like the attacks of [5], our attack do not necessarily require the cache to be clean of lookup table entries prior to each run of the implementation, but can be more efficient under the clean cache assumption.

2.3 Known Plaintext Attack

Here we recall the attack presented in [8]. An online phase and an offline phase compose the attack. In the online phase, the attacker makes the device storing an unknown AES key encrypt a number of random (independent, uniformly distributed) plaintexts. From the observed side-channel leakage, the adversary deduces the sequence of cache events (hits and misses) that correspond to the lookups to the table that stores the outputs of the `SubByte` function instantiated during an AES encryption. In [8] the lookups from the first round and the two first lookups of the second round only are considered. In Section 3 we will extend this attack to the two next lookups to reduce its offline complexity.

Analysis of the first round. Provided with a set of plaintext blocks $(p_i)^{(q)}$ and the corresponding cache traces, the adversary then proceeds to the offline analysis of them, first considering the first round. In the latter, a cache **miss** occurring at the i-th lookup can be algebraically expressed as the following inequation:

$$\forall j \in \Gamma,\ \widehat{k_i \oplus p_i} \neq \widehat{k_j \oplus p_j}$$

where Γ denotes the set of indices where previously occurred a cache miss. In a similar way a cache **hit** at the i-th lookup can be described with:

$$\exists! j \in \Gamma,\ \widehat{k_i \oplus p_i} = \widehat{k_j \oplus p_j}$$

Using the relation $\widehat{k_i \oplus k_j} = \widehat{k_i \oplus k_0} \oplus \widehat{k_0 \oplus k_j}$, the terms above can be rearranged as follows:

$$\forall j \in \Gamma,\ \widehat{k_i \oplus k_0} \neq \widehat{p_i \oplus p_j} \oplus \widehat{k_j \oplus k_0}$$

when the i-th cache event is a miss, and

$$\exists! j \in \Gamma,\ \widehat{k_i \oplus k_0} = \widehat{p_i \oplus p_j} \oplus \widehat{k_j \oplus k_0}$$

when it is a hit.

If we assume to know the values of $(\widehat{k_0 \oplus k_j})_{1 \leq j \leq i-1}$, the above equations and inequations gradually reduce the number of possible values for $\widehat{k_0 \oplus k_i}$ from 16 to 1, i ranging from 1 to 15, allowing a recursive recovery of the values for all $\widehat{k_i \oplus k_0}$. Hence this suggests to sieve the traces lookup by lookup, instead of trace by trace.

Finally, the high nibbles of the key bytes form a system of 15 linearly independent equations, reducing the entropy of the key down to $128 - 4 \times 15 = 68$ bits.

Analysis of the second round. Due to the pre-computation of the round keys, no lookup to the S-Box table occurs between the rounds. Both cache hits and misses can be analyzed in the second round, as carried out in our simulations. However in the following we focus on the analysis of the misses for the sake of clarity. This **first lookup** is indexed by:

$$y_0 = 2 \bullet s(x_0) \oplus 3 \bullet s(x_5) \oplus s(x_{10}) \oplus s(x_{15}) \oplus s(k_7) \oplus k_0 \oplus 1.$$

If a miss occurs at this lookup, we know that:

$$\forall j \in \Gamma,\ \widehat{y}_0 \neq \widehat{x}_j$$

where Γ is the set of the cache miss indices in the first round. In these inequations, the values of $\widehat{k}_0$, $\check{k}_0$, $\check{k}_5$, $\check{k}_7$, $\check{k}_{10}$ and $\check{k}_{15}$, thus 24 bits, are unknown. Hence, running through the set of the 2^{24} possible 6-uplets $(\widehat{k}_0, \check{k}_0, \check{k}_5, \check{k}_7, \check{k}_{10}, \check{k}_{15})$, one can evict a portion of wrong candidates and once done, continue with the next pair of plaintext and cache trace, until the maximum amount of information is gained. That is, up to 4 candidates remain at the end.

The **second lookup**, indexed by:

$$y_1 = 2 \bullet s(x_1) \oplus 3 \bullet s(x_6) \oplus s(x_{11}) \oplus s(x_{12}) \oplus s(k_7) \oplus k_0 \oplus k_1 \oplus 1$$

is processed in a similar manner. However, the first lookup also needs to be considered. Indeed, if it is a miss, the value of $\widehat{y}_0$ is computed to be compared to $\widehat{y}_1$, along with the $\widehat{x_j}$'s of the first round that correspond to a miss. If the second lookup is a miss, we have:

$$\forall j \in \Gamma,\ \widehat{y}_1 \neq \widehat{\delta}_j$$

where Γ is the set of cache miss indices in the first round and first lookup of the second, and $(\delta_j)_{1 \leq j \leq |\Gamma|}$ are the corresponding index values. Running through the 2^{16} possible 4-uplets $(\check{k}_1, \check{k}_6, \check{k}_{11}, \check{k}_{12})$, one can evict the wrong candidates until only one remains.

3 Reducing the Offline Complexity

In [8], an exhaustive search is performed to recover the 28 to 30 remaining unknown bits: $\check{k}_2$, $\check{k}_3$, $\check{k}_4$, $\check{k}_8$, $\check{k}_9$, $\check{k}_{13}$, $\check{k}_{14}$, and the possibly up to 4 candidates for $\check{k}_7$. Here we show that analyzing the third and the fourth lookups of the second round in the same way as the first two lookups reduces the remaining exhaustive search to at most 10 full key candidates and therefore significantly decreases the overall offline complexity of the attack.

The **third lookup** is indexed by:

$$y_2 = 2 \bullet s(x_2) \oplus 3 \bullet s(x_7) \oplus s(x_8) \oplus s(x_{13}) \oplus s(k_7) \oplus k_0 \oplus k_1 \oplus k_2 \oplus 1$$

where the nibbles $(\check{k}_2, \check{k}_8, \check{k}_{13})$, thus 12 bits of information, are unknown. As for the two previous lookups, an adversary can run through the 2^{12} possibilities for $(\check{k}_2, \check{k}_8, \check{k}_{13})$ and retain only the one that satisfies all the equations and inequations derived from the previous lookups and plaintexts. These different statements possibly involve $\widehat{y}_0$ and $\widehat{y}_1$, that have to be computed for each trace.

The **fourth lookup** is indexed by:

$$y_3 = 2 \bullet s(x_3) \oplus 3 \bullet s(x_4) \oplus s(x_9) \oplus s(x_{14}) \oplus s(k_7) \oplus k_0 \oplus k_1 \oplus k_2 \oplus k_3 \oplus 1$$

where the nibbles $(\check{k}_3, \check{k}_4, \check{k}_9, \check{k}_{14})$, thus 16 bits of information, are unknown. Likewise, an adversary can identify the correct $(\check{k}_3, \check{k}_4, \check{k}_9, \check{k}_{14})$ that satisfy the constraints set by the previous lookups and plaintexts, among the 2^{16} possible candidates. Here, $\widehat{y}_0$, $\widehat{y}_1$ and $\widehat{y}_2$ may be involved in the equations and inequations, and thus have to be computed.

The advantage for the offline complexity is due to the fact that evaluating the above equations is much less complex compared to the full AES executions performed during exhaustive search. We evaluated the complexity of this extension of the analysis in the absence of detection errors in the cache trace generation (due to uncertain events or preloaded cache, as detailed in Section 4). Following the empirical figures obtained in Section 6, the cost of the analysis of the third and fourth lookups takes about 10^5 less calls to the `SubByte` and `xtime` functions than would an exhaustive search over 2^{30} candidates take on average.

We have simulated this extended attack and present the results in Section 6.

4 The Full Error-Tolerant Attack

In this section, we will describe the full attack version resistant to uncertainties in cache event detection. We begin with showing that cache events can be distinguished in the EM leakage of a microcontroller (μC) and describing a general approach to dealing with detection uncertainties.

4.1 Cache Events in Side Channel Leakage

In [8], it was shown that cache operations can be distinguished in the power consumption trace of an ARM μC. Our experiments here show that cache events can be well distinguished in the EM leakage as well with a very simple measurement setup. We have experimented with Olimex LPC-H2124 development board [15] (Figure 1(b)) carrying NXP LPC2124 [14], an ARM7 μC featuring a Memory Accelerator Module (MAM). The MAM is a single-line cache that increases the efficiency of accesses to the onboard flash memory. In Figure 1(a) we present the EM traces acquired while the μC with MAM enabled was performing a series of lookups in the AES S-Box table that was stored in the flash memory. The acquisition was performed with Langer RF-B 0.3-3 H-field probe, Langer PA 203 20 dB preamplifier and LeCroy WaveMaster 104MXi oscilloscope. The CPU clock frequency of the μC was 59 MHz, the sampling rate of the oscilloscope was set to 5 GS/s. The probe was fixed by a lab stand with the probe tip touching the surface of the μC package; the precise position of the probe was determined experimentally.

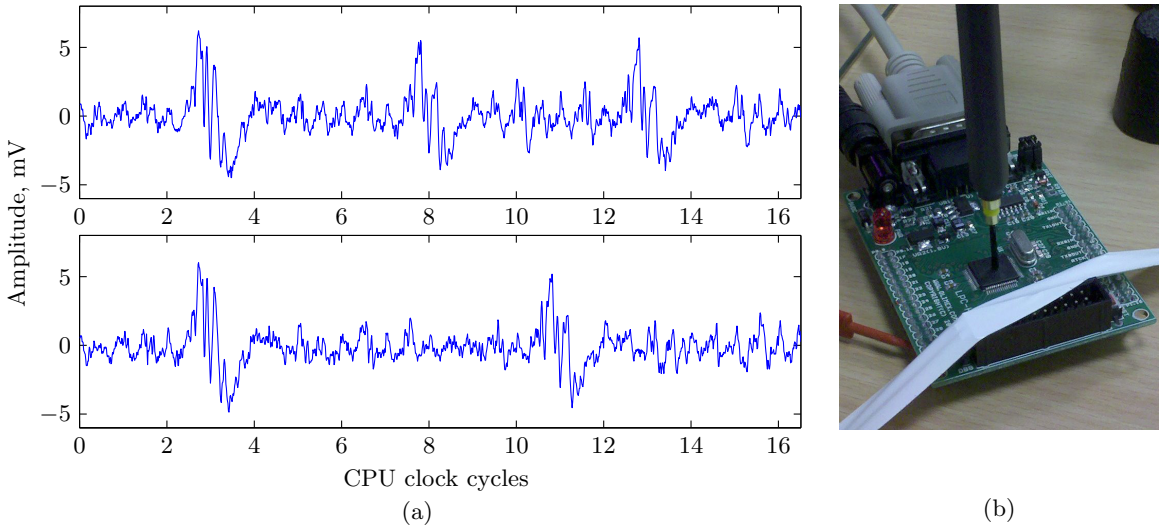

Fig. 1. (a) EM traces of an ARM7 μC with distinguishable cache event sequences: miss-miss-miss (top) versus miss-hit-miss (bottom); (b) the μC with the passive EM probe

The top trace shows a sequence of 3 cache misses, whereas the bottom trace shows a miss-hit-miss sequence. Cache misses can be seen as distinguished peaks. Note also the timing differences: Figure 1(a) suggests that the cache hit takes 2 CPU clock cycles less than the cache miss. We would like to stress that the traces were acquired *without any averaging in an unshielded setup* depicted in Figure 1(b).

In a real-life scenario, the noise contained in the measurement can disturb the extraction of the cache sequence from the side-channel trace. The use of statistics however overcomes

this obstacle. The noise is generally assumed to follow a Gaussian distribution, hence one can estimate, measuring some statistic like the respective height of the peak in the cycles where occur a cache hit or a miss. From this distribution, one can define two thresholds t_H and t_M, $t_H < t_M$. In this setting, one distinguishes between three types of events:

1. If the statistic is smaller than t_H, the event is considered as a hit.
2. If the statistic is larger than t_M, the event is considered as a miss.
3. If the statistic falls between the thresholds, we consider the event to be "uncertain".

The thresholds are chosen such that it is highly unlikely for a miss to be misinterpreted as a hit and vice-versa.

4.2 Adaptation of the Attack to Errors

Along with the cache hit H and the cache miss M, the **uncertain** U cache event integrates in the sieve. Although [8] describes the adaptation for the first round of an AES encryption only, we extend it to the second round and provide results based on simulated attacks in Section 6.

Adaptation of the first round. We consider the lookup i of the first round, eventually with uncertain events occurred before it.

1. if CT_i is a cache **hit**, then the eventual uncertain previous events of the trace like are taken for misses, such that when shrinking the set $\kappa_{0,i}$ to the set of values $\widehat{k_0 \oplus k_j}$ for $CT_j = M$, the correct value of $\widehat{k_0 \oplus k_i}$ is still included in $\kappa_{0,i}$. Hence the latter is reduced to the set $\{\widehat{k_0 \oplus k_j} | j \in \Gamma \cup \Upsilon\}$, where Γ and Υ denote the sets of indices where previously occurred respectively a cache miss and an uncertain event.
2. if CT_i is a cache **miss**, then the uncertain previous events of the trace like are taken for hits, such that when evicting from $\kappa_{0,i}$ the values $\widehat{k_0 \oplus k_j}$ for $CT_j = M$, the correct value of $\widehat{k_0 \oplus k_i}$ is not evicted. Hence, $\kappa_{0,i}$ is defined as $\kappa_{0,i} \setminus \{\widehat{k_0 \oplus k_j} | j \in \Gamma\}$.
3. if CT_i is an **uncertain** event, no action is performed on $\kappa_{0,i}$.

Adaptation of the second round. The second round adapts itself to uncertain events in the same spirit. Considering lookup i in the second round, $i = 0, 1, 2, 3$:

1. if CT_i is a cache **hit**, for every possible vector of the unknown nibbles involved in y_i, $\widehat{y}_i$ must belong to $\{k_0 \oplus k_{0,j} \oplus p_j | j \in \Gamma_1 \cup \Upsilon_1\} \cup \{\widehat{y}_j | j \in \Gamma_2 \cup \Upsilon_2\}$, where Γ_i and Υ_i denote the sets of indices where previously occurred respectively a cache miss and an uncertain event in the round i, $i = 1, 2$.
2. if CT_i is a cache **miss**, for every possible vector of the unknown nibbles involved in y_i, $\widehat{y}_i$ must not belong to $\{k_0 \oplus k_{0,j} \oplus p_j | j \in \Gamma_1\} \cup \{\widehat{y}_j | j \in \Gamma_2\}$.
3. if CT_i is an **uncertain** event, no action is performed on the set of possible vectors of the unknown nibbles involved in y_i.

4.3 Partially Preloaded Cache

If the cache is partially filled with lines of the lookup table, the attack still works provided that the analysis of the cache hits is omitted. Indeed, if some lines of the lookup table are present in the cache prior to the AES encryption considered, the cache hits that occur may refer to lines previously loaded, for which an attacker has no valuable information.

5 Theoretical Model

In this section we deal with theoretically estimating the number of traces required for the attack. First, we describe the precise model to obtain the number of traces required for the error-tolerant analysis of each of the cache events. Then we show that this model, being univariate, does not estimate the number of traces well due to the statistical dependency between the cache events. The distribution in the multivariate model is too complex to be expressed theoretically so we simulate it carrying out the attacks, presenting the final results in Section 6.

5.1 Univariate Model for the Error-Tolerant Attack

In [1], a theoretical model is presented to estimate the number of traces required for the analysis of the 15 cache events in the first round. This model however does not take into consideration the dependency between the cache events. Here we present an improved model, which also takes detection errors into account. The model will provide an expected number of traces $\mathbf{E}N_i$ required for the analysis of the i-th lookup, $1 \leq i \leq 15$, for a given error probability p, for the general case of m cache lines per S-Box lookup table.

We start with obtaining the expectation $\mathbf{E}R_i$ for the fraction of candidates R_i remaining after analyzing lookup i of a single cache trace. This expectation is expressed as

$$\mathbf{E}R_i = \sum_{s=1}^{i} \Pr(T_i = s) \cdot R_i^{(s)}, \quad i \geq 1 \tag{1}$$

where T_k is the number of lookup table lines in cache (*i.e.* $|\Gamma|$) after k lookups, and $R_i^{(s)}$ is the fraction of the key candidates remaining after analysis of the i-th lookup of a single cache trace when the number of lines previously loaded into cache is s. Note that (1) works for the second round lookups as well.

The distribution $\Pr(T_k = s)$ is a classical allocation problem [11]. We have

$$\Pr(T_k = s) = \binom{m}{m-s} \left(\frac{s}{m}\right)^k \Pr_k^0(s)\,, \tag{2}$$

where

$$\Pr_k^0(s) = \sum_{l=0}^{s} \binom{s}{l} (-1)^l \left(1 - \frac{l}{s}\right)^k .$$

Note that this distribution describes the process within the device and not the attacker's observations and therefore does not depend on the error probability p.

The fraction $R_i^{(s)}$ is expressed as the sum of the products of the conditional probabilities of the three possible cache event observations (miss, hit, uncertain) and corresponding remaining fractions $R_{i,M}^{(s)}$, $R_{i,H}^{(s)}$, $R_{i,U}^{(s)}$):

$$\begin{aligned} R_i^{(s)} &= \Pr(CT_i = M | T_i = s) \cdot R_{i,M}^{(s)} \\ &+ \Pr(CT_i = H | T_i = s) \cdot R_{i,H}^{(s)} \\ &+ \Pr(CT_i = U) \cdot R_{i,U}^{(s)}\,. \end{aligned} \tag{3}$$

Now, the three probabilities are

$$\begin{aligned} \Pr(CT_i = M | T_i = s) &= \tfrac{(1-p)(m-s)}{m}\,, \\ \Pr(CT_i = H | T_i = s) &= \tfrac{(1-p)s}{m}\,, \\ \Pr(CT_i = U) &= p\,. \end{aligned}$$

Finally, recalling the error-tolerant attack description in Section 4.2, the fractions for the case of miss and hit are

$$R_{i,M}^{(s)} = \frac{m-(1-p)s}{m} ,$$
$$R_{i,H}^{(s)} = \frac{s(1-p)+ip}{m} ,$$
$$R_{i,U}^{(s)} = 1 .$$

Now, from equations (1), (2) and (3) we can obtain $\mathbf{E}R_i$ for $1 \leq i \leq 15$. The values for the case $m = 16$ and $p = 0$ are shown in Table 1.

Table 1. Expected ratios of the remaining candidates for the first round lookups, $m = 16$, $p = 0$.

i	1	2	3	4	5	6	7	8
$\mathbf{E}R_i$	0.882813	0.787598	0.711151	0.650698	0.603836	0.568490	0.542866	0.525418
i	9	10	11	12	13	14	15	
$\mathbf{E}R_i$	0.514812	0.509903	0.509705	0.513373	0.520184	0.529519	0.540853	

Note that these values are larger than the ones obtained in [1] (denoted there by $R_{expected}^{k}$, k being i in our terms) since in our model we have correctly considered the dependency of the analysis of an i-th lookup on the events in the previous i lookups (recall that the enumeration of lookups starts from 0).

Knowing $\mathbf{E}R_i$, we can estimate the expected number of traces $\mathbf{E}N_i$ required for the analysis of an i-th lookup. We recall from Section 4.2 that in the analysis of each lookup in the first round, we want to reduce the number of candidates for the corresponding XOR difference of the key nibbles from m to 1. The traces are statistically independent if the inputs are independent (which is the assumption of our known plaintext attack), each trace leaves us with a fraction $\mathbf{E}R_i$ of the remaining candidates, so we have

$$m \cdot (\mathbf{E}R_i)^{N_i} \leq 1 ,$$
$$\mathbf{E}N_i \approx -\log_{\mathbf{E}R_i} m .$$

We estimate $\mathbf{E}N_i$ for all the lookups of the first round being analyzed assuming $m = 16$, for the case there are no errors in detection, *i.e.* $p = 0$, and for the cases $p = 0.25$ and $p = 0.5$. These theoretical estimates are shown in Figure 2 against the empirical results we obtained in attack simulations. One can see that our model captures the behaviour of the attack and the effect of the errors.

The model for the second round is similar and was described in [8] for the case without detection errors. It can be adjusted in the same way to include error probability p. The model is also easily adjustable for the misses-only analysis in the case of partially preloaded cache.

5.2 The Multivariate Model

The full attack measurement complexity is determined by the maximum number of traces required for the analysis of each lookup. If the cache events were statistically independent, the expectation for the maximum number of traces $\mathbf{E}(\max_i N_i)$ would be equal to the maximal expected number of traces among each of the lookups $\max_i(\mathbf{E}N_i)$, so one could use the model described above.

However, the cache events are dependent, therefore

$$\mathbf{E}(\max_i N_i) \neq \max_i(\mathbf{E}N_i) ,$$

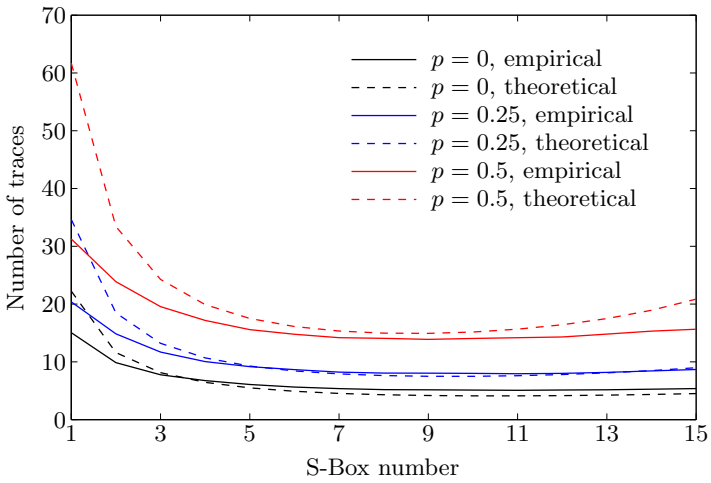

Fig. 2. Expected number of traces for the lookups of the first round. Theoretical and empirical figures for different error probabilities p are shown.

and so the univariate model is not applicable for the estimation of the full attack complexity. This is well seen when one considers the results of attack simulations: for example, in case $p = 0$ for the first round in the univariate model we have $\max_i(\mathbf{E}N_i) = \mathbf{E}N_1 = 15$, whereas $\mathbf{E}(\max_i N_i) = 19$. The same holds for the second round: in [8] the numbers of traces for the analysis of the first and second lookups are 27 and 18, whereas our experiments here show that the expectation for the maximum in the second round is 30.

So, to estimate the number of traces required for the full attack, one has to consider the distribution of $\max_i N_i$ for the case of statistically dependent random variables N_i, which requires the multivariate distribution for $(N_1, N_2, \ldots, N_{19})$. This multivariate distribution is hard to express analytically, therefore it is easier to simulate it carrying out the attack and sampling the values of $\max_i N_i$. This is what we do to obtain the results presented in Section 6.

To better illustrate the behaviour of the attack, in Figure 3(a) we present the empirical bivariate distribution for the case of the first round lookups 1 and 2. In Figure 3(b) we show the distribution for $\max(N_1, N_2)$ against the independent distributions for N_1 and N_2. The mean of the former is 18.04. One can see that it is greater than the maximum of the means for N_1 and N_2 and is actually quite close to the number of traces required for the analysis of all the 15 lookups in the first round.

Nevertheless, the univariate model provides an estimate for the lower bound for the number of traces required for the full attack, so it can be still applied when one requires this bound.

6 Results

We have conducted simulated attacks with different values for the probability p of uncertainty for a cache event and the number of preloaded lines in the cache for 10^4 random keys in every case. The results are depicted in Figure 4.

One can see that our algorithm tolerates errors well: in case error probability is 0.8, one would need about 160 measurements to preform the full key recovery in case it is known that the cache is clean prior to each run of the algorithm. If the cache is not clean from the lookup table lines, then the numbers of traces (required for the misses-only analysis) are of course higher, but the attack is still feasible.

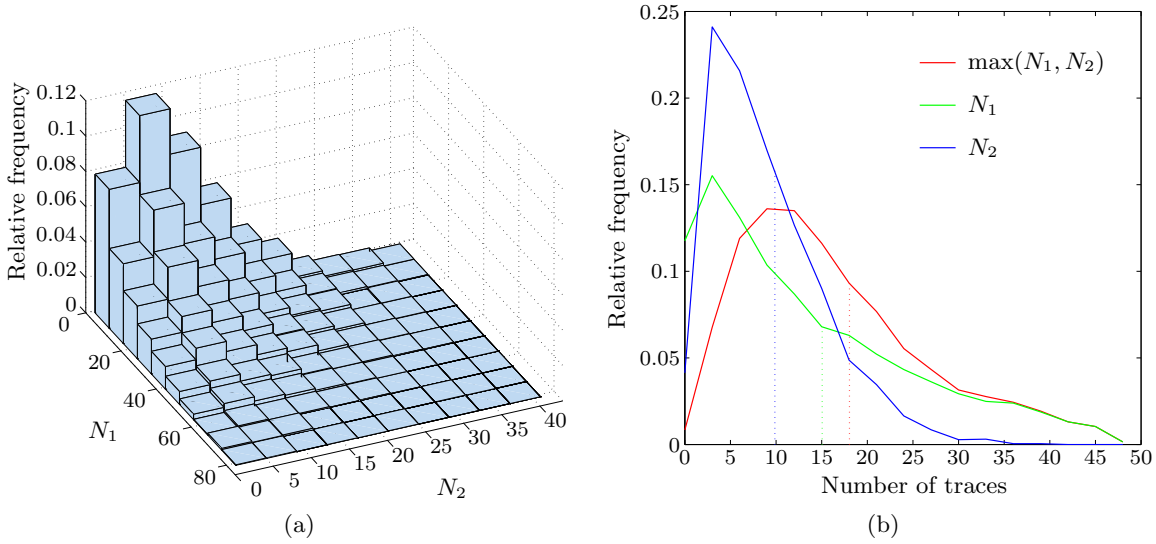

Fig. 3. (a): empirical bivariate distribution for (N_1, N_2); (b): empirical univariate distributions for N_1, N_2, and for $\max(N_1, N_2)$, dashed lines showing the corresponding means.

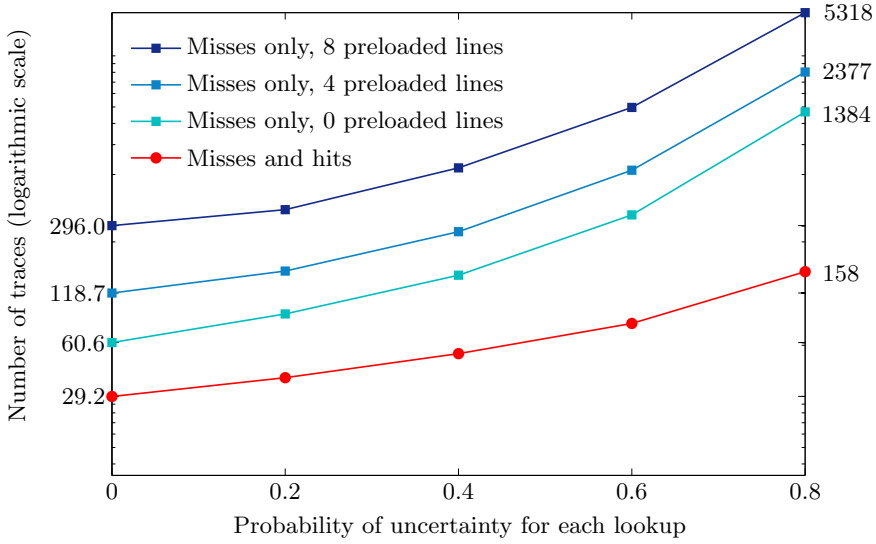

Fig. 4. Average number of traces required for the full key recovery.

The attack measurement complexity for low error probabilities (which are quite realistic looking at our practical explorations in Section 4.1) is comparable to that of the DPA. The offline complexity is negligible: recovering the full key from a set of cache traces takes less than a minute on a standard PC.

The countermeasures against trace-driven cache-collision attacks have been discussed in the previous works on the subject [3,12,7,5] and are similar to the countermeasures against cache attacks in general [18].

References

1. Onur Acıiçmez and Çetin Kaya Koç. Trace-driven cache attacks on AES (short paper). In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS'06*, volume 4307 of *LNCS*, pages 112–121. Springer, Heidelberg, 2006.
2. Daniel J. Bernstein. Cache-timing attacks on AES. `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, 2004.
3. Guido Bertoni, Vittorio Zaccaria, Luca Breveglieri, Matteo Monchiero, and Gianluca Palermo. AES power attack based on induced cache miss and countermeasure. In *ITCC'05*, volume 1, pages 586–591. IEEE, 2005.
4. Andrey Bogdanov. Improved side-channel collision attacks on AES. In Carlisle Adams, Ali Miri, and Michael Wiener, editors, *SAC'07*, volume 4876 of *LNCS*, pages 84–95. Springer, 2007.
5. Joseph Bonneau. Robust final-round cache-trace attacks against AES. Cryptology ePrint Archive, Report 2006/374, 2006. `http://eprint.iacr.org/2006/374`.
6. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer, 2002.
7. Jacques Fournier and Michael Tunstall. Cache based power analysis attacks on AES. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP'06*, volume 4058 of *LNCS*, pages 17–28. Springer, 2006.
8. Jean-François Gallais, Ilya Kizhvatov, and Michael Tunstall. Improved trace-driven cache-collision attacks against embedded AES implementations. Cryptology ePrint Archive, Report 2010/408, 2010. `http://eprint.iacr.org/2010/408`.
9. John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side channel cryptanalysis of product ciphers. *Journal of Computer Security*, 8:141–158, 2000.
10. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
11. Valentin F. Kolchin, Boris A. Sevastyanov, and Vladimir P. Chistyakov. *Random Allocations*. V. H. Winston & Sons, Washington, D.C., 1978.
12. Cédric Lauradoux. Collision attacks on processors with cache and countermeasures. In Christopher Wolf, Stefan Lucks, and Po-Wah Yau, editors, *WEWoRC'05*, volume P-74 of *Lecture Notes in Informatics*, pages 76–85. Gesellschaft für Informatik, Bonn, 2005.
13. Michael Neve and Jean-Pierre Seifert. Advances on access-driven cache attacks on AES. In *SAC'06*, volume 4356 of *LNCS*, pages 147–162. Springer, 2007.
14. NXP B.V. LPC2114/2124 single-chip 16/32-bit microcontrollers. `http://www.nxp.com/documents/data_sheet/LPC2114_2124.pdf`, 2007.
15. Olimex. LPC-H2124 header board for LPC2124 ARM7TDMI-S microcontroller. `http://www.olimex.com/dev/lpc-h2124.html`.
16. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In David Pointcheval, editor, *CT-RSA'06*, volume 3860 of *LNCS*, pages 1–20. Springer, 2006.
17. Daniel Page. Theoretical use of cache memory as a cryptanalytic side-channel. Technical report CSTR-02-003, University of Bristol, 2002.
18. Daniel Page. Defending against cache-based side-channel attacks. *Information Security Technical Report*, 8(P1):30–44, 2003.
19. Chester Rebeiro and Debdeep Mukhopadhyay. Cryptanalysis of CLEFIA using differential methods with cache trace patterns. In *CT-RSA'11*, LNCS. Springer, 2011.
20. Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES: Why time also matters in DPA. In *CHES'09*, volume 5747 of *LNCS*, pages 97–111. Springer, 2009.
21. Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A collision-attack on AES: Combining side channel- and differential-attack. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES'04*, volume 3156 of *LNCS*, pages 163–175. Springer, 2004.
22. Xin-Jie Zhao and Tao Wang. Improved cache trace attack on AES and CLEFIA by considering cache miss and S-box misalignment. Cryptology ePrint Archive, Report 2010/056, 2010. `http://eprint.iacr.org/2010/056`.
23. FIPS PUB 197: Specification for the Advanced Encryption Standard, 2001. `http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.