Statistical Analysis for Access-Driven Cache Attacks Against AES

Liwei Zhang, A. Adam Ding, Yunsi Fei, and Zhen Hang Jiang

[1] Department of Mathematics, Northeastern University, Boston, MA 02115
[2] Department of Electrical and Computer Engineering
Northeastern University, Boston, MA 02115
zhang.liw@husky.neu.edu,a.ding@neu.edu,
yfei@ece.neu.edu,jiang.zhen@husky.neu.edu

Abstract. In recent years, side-channel timing attacks utilizing architectural behavior have been applied to cloud settings, presenting a realistic and serious cyber threat. Access-driven cache attacks allow the adversary to observe side-channel leakage (cache access pattern) of a critical cryptographic implementation to infer the secret key. However, what the attackers observe may deviate from the real cache footprint of the victim process, affecting the effectiveness of cache-based timing attacks using the observed leakage. Various countermeasures, including secure cache and architectures design, should also be evaluated accurately for their side-channel resilience. To address this need, this paper proposes a mathematical model for access-driven cache attacks, and derives explicit success rate formulas for those attacks. It is the first theoretical model that explicitly considers the misclassification errors for cache access and cache non-access by the victim cryptographic process. We implement several access-driven cache attacks and use our models to evaluate them. We demonstrate that the proposed statistical model predicts the success rate of cache-based timing attacks accurately. We also apply the model onto various cache defense architectures for evaluation.

Keywords: AES, side-channel analysis, access-driven cache attacks, statistical model

1 Introduction

Side channel analysis (SCA) attacks based on cache behavior have been invented to break various cryptographic implementations since the pioneer work of a decade ago [1, 2]. For instance, Tsunoo et al. [3] studied the cache based attack on Data Encryption Standard (DES). Bernstein applied it onto Advanced Encryption Standard (AES) [4]. A variety of cache-based side-channel attacks were developed, falling into three categories, trace-driven, time-driven, and access-driven attacks.

In trace-driven attacks [5–11], the adversary is able to derive information about individual cache events from side-channel traces, such as power consumption or electromagnetic emanation of the cryptographic execution. Time-driven

attacks exploit the execution time of an algorithm which depends on the cache access pattern directed by the secret key value [4, 12, 13, 11]. Access-driven attacks allow the adversary to gain information about cache lines being accessed during the execution [14–17]. Recent work has shown the possibility of cross-virtual machine side-channel attacks in cloud computing [18–22], making the access-driven SCA a real and serious cyber threat.

Various secure cache design and architectures have been proposed [23–25] to protect against cache-based SCA. Theoretical models play an integral role in understanding cache-based SCAs and guiding effective countermeasure design. For trace-driven and time-driven attacks, Page [26, 27] first described and simulated theoretical attacks. For time-driven attacks, Tiri et al. [28] presented an analytic model, and provided formulas for the number of measurements needed for the attack to succeed. Rebeiro and Mukhopadhyay [32] first quantified the leakage due to sequential and arbitrary stride prefetching in time-driven attacks through the probability distribution of cache access (non-access) for one cache line. Savas and Yilmaz [11] proposed a generic model of conditional entropy (used as the leakage metric) for trace-driven, time-driven and access-driven attacks. They theoretically analyzed the access-driven model on the basic cache architecture (without countermeasures) under the assumption that the adversary can detect accesses to specific cache lines by the cryptographic implementations precisely. And they simulated different noise levels and showed their overall effect on the side-channel leakage. Domnitser et al. [30] proposed a framework to estimate the probability of correctly identifying the leaky cache accesses under various cache architecture. Zhang and Lee [31] further proposed a framework utilizing mutual information to evaluate cache architectures' vulnerability based on non-interference property. While these works provided quantitative metrics for the cache architecture's side-channel vulnerability, these metrics were not explicitly linked to effectiveness of concrete cache-based SCAs. To clearly quantify the effects, two types of error should be separated: type I error, mistaking cache access for cache non-access; and type II error, mistaking cache non-access as cache access. Advanced cache architectures often utilize features such as prefetching, partitioning and randomization [29], which can increase rates of these two types of errors.

Our Contribution: In this work, we provide a theoretical statistical model for cache access-driven attacks that explicitly accounts for these two types of errors. We start from the basic cache architecture, establishing a statistic model and presenting corresponding properties. Then we extend it to include the effects of the two types of errors. Further more, we consider the cache access based attacks and cache non-access based attacks, and derive, for the first time, their explicit theoretical success rate formulas of retrieving individual key bytes in a divide-and-conquer fashion. The formulas quantify the effect of each factor relating to the countermeasures' effectiveness, and thus provide valuable insights in the directions of designing more secure cache architectures and countermeasures.

The rest of papers is organized as follows. Section 2 presents the background of prime+probe attacks and some related probability theory. Section 3 proposes

our cache models, and provides some basic probability properties about the basic and advanced cache architectures. Section 4 analyzes attacks focused on the first and last rounds of AES, respectively, and derives the explicit success rate formulas for attacks based on cache access or cache non-access. Finally, Section 5 reports attack simulations and practical attack experiments that verify the soundness of our models.

2 Preliminaries

We first present some basic probability formulas, and then describe the prime+probe cache attacks.

2.1 Probability Theory

As cache attack is based on cache behavior, including cache conflicts where two memory blocks map onto the same cache line and therefore resulting in a cache line replacement, we first introduce basic probability theory for replacement.

On a sample space $S = \{s_1, \cdots, s_m\}$, a subject randomly selects n items with replacement. The output is represented by a vector $O_1 = \{o_{1,1}, \cdots, o_{1,n}\}$. Then the number of times that object s_i being selected in the experiment is $\boldsymbol{m}(s_i) = \sum_{j=1}^{n} \mathbb{I}(o_{1,j} = s_i)$, where $\mathbb{I}$ is the indicator function, yielding 1 when $o_{1,j} = s_i$, otherwise 0. Thus the expected number of times that s_i being selected is $\mathbb{E}[\boldsymbol{M}(s_i)] = \sum_{j=1}^{n} \mathbb{P}(o_{1,j} = s_i)$.

Furthermore, let n_E subjects each repeat this experiment, then the number of subjects who select object s_i is

$$\boldsymbol{N}(s_i) = \sum_{w=1}^{n_E} \mathbb{I}(s_i \in \bigcup_{j=1}^{n} o_{w,j}). \tag{1}$$

Its expected value, assuming independence among these q subjects, is

$$\mathbb{E}[\boldsymbol{N}(s_i)] = \sum_{w=1}^{n_E} \mathbb{P}(s_i \in \bigcup_{j=1}^{n} o_{w,j}) = \sum_{w=1}^{n_E} [1 - \prod_{j=1}^{n} \mathbb{P}(s_i \notin o_{w,j})]. \tag{2}$$

2.2 Prime+Probe Cache Attacks

In this section, we describe some last-level Prime+Probe cache timing attacks [22] that we have implemented and evaluated on a 2.5 GHz Intel Core i5 CPU, targeting an OpenSSL1.0.1f implementation of AES. This is used as a concrete example for the abstract and general model studied in Sections 3 and 4. The numerical results from the evaluation are presented in Section 5 to confirm the theoretical results. In this attack, we assume that the adversary (the spy process) and victim (the encryption process) are sitting on the same physical machine but on different cores, which share the last-level cache.

Targeted platform: Our targeted platform contains an Intel core i5, which is an Ivy Bridge architecture, with 2 physical cores, and features a 3MB 12-way set associative shared last-level cache (L3 cache). Since this CPU has two cores, the L3 cache is split into two equal slices, which are connected by a ring bus, for allowing parallel cache accesses by two cores. However, each core can access either slice, but it will take more time to access the remote slice than its local one. The size of each cache line is 64 bytes, and each slice therefore has 2048 cache sets.

Attack challenges: For the attack to be feasible in the cross-core setting, we need to address two challenges: first, we should be able to evict data for different cores from caches. In modern CPUs, most last-level caches have the inclusiveness property, which means if any data in the last-level cache is being evicted, it will also be evicted in all upper caches (L1 and L2 caches). Thus, further request of access to the evicted data will be served by the main memory, resulting in a distinguishably high latency. Second, we should be able to evict a targeted cache line in the L3 cache using virtual addresses. Although the L3 cache is physically tagged and physically addressed, we can utilize the 2MB ($=2^{21}$) large page memory, which will have 21 offset bits that will be directly translated into physical address. 21 offset bits are more than enough to address 2048 cache sets, which only requires 11 index bits, in each cache slice in the L3 cache. Thus, we can allocate two or more continuous large pages and select all addresses that mapped to the targeted cache line. By accessing those selected addresses, we can effectively evict the targeted cache line from the L3 cache.

Data collection: In this paper, we consider the aligned T-tables, i.e., the starting memory address of a T table is mapped to the beginning of a cache line. We assume that attackers know the location of T-tables in cache. After the T-tables used by the victim are mapped into the main memory, we launch our spy process to determine which sets in the L3 cache are used by the T-tables. We use the algorithm in [22] and created one eviction data set for each cache set in the L3 cache. We sweep every set in the L3 cache while the victim is continuously encrypting the same plaintext so as to determine the range of 64 continuous sets (for each T table used by AES, each occupies 16 continuous cache sets, with one cache line in a set) that have the highest probing time. Once we find the locations of the 64 cache sets for AES, we know that the beginning 16 sets will be used by the T_0 table. By monitoring those 16 sets, we should be able to collect timing information on T_0 table.

After determining the targeted cache sets, we follow the standard Prime+Probe attack procedure. First, the spy process primes its data to occupy all these targeted 16 cache sets (including all the ways in each set); second, the spy process sends one random 16-byte plaintext to the victim process and waits for the encryption to complete; third, the spy process probes its data set by set, and record the access time for each set. For each AES encryption, the spy process records the plaintext, ciphertext, and a vector of probing times for targeted cache sets.

As in each cache set, there is at most one AES cache line, we will use cache line and cache set interchangeably in this paper.

Because the modern cache is divided into two slices, a cache line mapped to a set, say 0, can be in either cache slice depending on a hashed value of its physical address. For our data collection, we only monitor one cache slice where the first cache line of T-tables resides, and so we only probe all 16 continuous cache sets in that slice. Thus, for some of cache lines of T-tables, we would not be able to see any information leaked from our timing measurement.

Attack abstraction: Based on the probing times, the adversary classifies the cache lines into two groups: those observed as being accessed (long probing time) by the victim process, S^A, and those not being accessed (short probing time), S^{NA}. The adversary then attacks one SBox, for example, in the last round of AES with known ciphertext x. With a guessed key value k and known x, the adversary can calculate which cache line was accessed by the SBox operation. If the k value is correct, the calculated cache line must belong to the set S^A (cache line access). For an incorrect k value, the guess cache line access belongs to S^A with a probability less than one. Then over many random plaintexts, the correct k value can be identified (as the one always lead to correct cache line access guess). However, the observed set S^A may not always correspond to real accesses by the victim process due to misclassification errors. Such confusion errors can arise in modern cache architectures with features like prefetching and cache line randomization. In such cases, the adversary chooses the k value leading to most frequent agreement between the calculated cache line accesses and the observed group S^A. We analyze such attacks and provide quantitative formulas for their success rates in the next two sections.

3 Statistical Models

In this section, we establish a statistic model on cache line access probabilities. Then we derive formulas for observed cache line accesses with confusion errors. These formulas will be used in Section 4 to derive explicit success rate formulas for the prime+probe attack.

3.1 Notations and Cache Models

We denote sets by calligraphic letters (e.g., $\mathcal{X}$), denote one-dimensional random variables by capital letters (e.g., X) which take values on the sets (e.g., $\mathcal{X}$), and denote observations of the random variables by lowercase letters (e.g. x). Correspondingly, let the bold capital letters, calligraphic letters, and lowercase letters denote multi-dimensional random variables, their values and observations (e.g., $\boldsymbol{X}$, $\boldsymbol{\mathcal{X}}$, $\boldsymbol{x}$), respectively. In this paper, we consider unprotected 128-bit AES with 4 well aligned T-tables, and focus on the cache-access model for one chosen T table on divide-and-conquer fashion. Let K,P, C and $\boldsymbol{L}$ denote the random variables for the targeted key byte, plaintext, ciphertext and cache line for the

selected T table, respectively, and each takes values on sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and $\boldsymbol{\mathcal{L}}$ respectively. Here, the values of $\boldsymbol{L}$ are d_L-dimensional vectors, $\boldsymbol{l}_i = \{i \cdot d_L, i \cdot d_L + 1, ..., i \cdot d_L + d_L - 1\}$, for $i = 0, ..., n_L - 1$, with $n_L = |\boldsymbol{\mathcal{L}}|$ being the number of all possible cache lines in $\boldsymbol{\mathcal{L}}$ and d_L being the number of table elements in one cache line. In general, for block ciphers, $|\mathcal{K}| = |\mathcal{P}| = |\mathcal{L}| = d_L \cdot n_L$. For 128-bit AES and the cache structure in our target platform, $n_K = |\mathcal{K}| = 256$, $d_L = 16$ and $n_L = 16$. Let n_A be the number of times that one certain T table is accessed during one AES encryption execution. Let k_c, k_g denote the secret (correct) key and some false key. $\mathbb{P}$ and $\mathbb{E}$ are the notations for the probability and expectation.

The cache-based SCA attack utilizes the physical measurements (timing) which relate to cache misses or cache hits, seen by the spy process, but caused by the preceding table accesses of one execution of the victim AES process. During an AES execution, there are $n_A = 40$ accesses in total of each T table for the ten rounds. For each access, the cache line index can be calculated based on the input and the key value according to the AES algorithm. After one execution, the victim process leaves some footprint in the cache for the spy process to see, but no details like when is a cache line accessed (by which byte), so the adversary just observes the group of cache lines accessed by the victim based on the probing time through the monitoring process. Due to the interference of noise, the adversary can not distinguish cache lines accesses from non-accesses. In general, the adversary shall monitor all cache lines. With all cache lines being monitored, we denote the two groups of all observed accessed and non-accessed cache lines during one encryption by $\boldsymbol{S}^{\text{ac}}$ and $\boldsymbol{S}^{\text{na}}$ respectively. We shall sometimes use the notations with subscript $\boldsymbol{S}_w^{\text{ac}}$ and $\boldsymbol{S}_w^{\text{na}}$ to indicate that the sets are for the wth encryption.

The algorithm can be assumed to satisfy two common properties approximately. One is that the n_A accesses are independent, because individual message bytes are processed independently in block ciphers. The other is that, for each access, every element in $\boldsymbol{\mathcal{L}}$ has equal probability to be accessed (over random plaintext inputs). So, when all cache lines are monitored, for the wth execution, the cache lines being taken as accessed cache lines by the adversary can be written into $\boldsymbol{O}_w$:

$$\boldsymbol{O}_w = (\boldsymbol{o}_{w,1}, \cdots, \boldsymbol{o}_{w,n_A}). \tag{3}$$

So, we have the group of all observed accessed cache lines $\boldsymbol{S}_w^{\text{ac}} = \bigcup_{j=1}^{n_A} \boldsymbol{o}_{w,j}$, while the group of all observed non-accessed cache lines $\boldsymbol{S}_w^{\text{na}} = \boldsymbol{\mathcal{L}} \backslash \bigcup_{j=1}^{n_A} \boldsymbol{o}_{w,j}$. In what follows, we use the word "observed" to emphasis that access/non-access is the observation by the adversary, such as observed cache line access, observed non-accessed cache lines, etc.

During the n_A accesses of one encryption, there exists one access which directly determines the accessed cache lines by the input (P/C) and key. Let $\boldsymbol{f}(x, k)$ denote the key-sensitive cache line predicted by the key value k calculated from the algorithm, where x is the input (known plaintext or ciphertext) of

one execution. In other words, with the input x, for some key hypothesis $k \in \mathcal{K}$, we would predict that the line $\boldsymbol{f}(x,k)$ is accessed during the encryption. In what follows, we shall sometimes use the notations with subscript $\boldsymbol{f}_{\text{last}}(\cdot,\cdot)/\boldsymbol{f}_{\text{1st}}(\cdot,\cdot)$ to indicate that it is for the last/first round of AES. To emphasis key-sensitive access, we call such cache lines $\boldsymbol{f}(\cdot,k)$ as *predicted* cache lines by the key value k. Since there are n_A times of one T table being accessed during one encryption, in addition of predicted accessed cache line, there may exist some other cache lines being accessed by the encryption. We call all cache lines which should be accessed/non-accessed during the n_A accesses calculated by algorithm *algorithm-based* accessed/non-accessed cache lines.

In this paper, we also consider the effects caused by the number of monitored cache lines. We use $\boldsymbol{S}^{\text{mo}}$ to denote the set of cache lines being monitored by the adversary, where $\boldsymbol{S}^{\text{mo}} \subset \boldsymbol{\mathcal{L}}$. Only the cache lines belonging to $\boldsymbol{S}^{\text{mo}}$, the adversary has the opportunity to observe and group into observed accessed/non-accessed cache lines, while other cache lines (outside $\boldsymbol{S}^{\text{mo}}$), the adversary can not know any information. During the wth encryption execution, with the monitored cache line set $\boldsymbol{S}^{\text{mo}}$, the common cache lines in $\boldsymbol{S}_w^{\text{ac}}$ and $\boldsymbol{S}^{\text{mo}}$ are observed as accessed cache lines by the adversary, while the common cache lines in $\boldsymbol{S}_w^{\text{na}}$ and $\boldsymbol{S}^{\text{mo}}$ are observed as non-accessed cache lines. The cache access/non-access based attack only can utilize the information of cache lines in $\boldsymbol{S}^{\text{mo}}$.

For cache access/non-access based attacks, the adversary often use the number of encryptions corresponding to the predicted cache lines by each key $k \in \mathcal{K}$ as statistics to retrieve the secret key. To do cache access based attacks, with the wth encryption, for each key hypothesis $k \in \mathcal{K}$, the adversary records as follows: as follows:

$$\begin{cases} \boldsymbol{d}_w^{\text{ac}}(k) = 1, & \boldsymbol{f}(x_w,k) \in \boldsymbol{S}_w^{\text{ac}} \text{ and } \boldsymbol{f}(x_w,k) \in \boldsymbol{S}^{\text{mo}}, \\ \boldsymbol{d}_w^{\text{na}}(k) = 1, & \boldsymbol{f}(x_w,k) \in \boldsymbol{S}_w^{\text{na}} \text{ and } \boldsymbol{f}(x_w,k) \in \boldsymbol{S}^{\text{mo}}. \end{cases} \tag{4}$$

Here, "$\boldsymbol{d}_w^{\text{ac}}(k) = 1$" means the predicted cache line by the k value is observed as access by the adversary, while "$\boldsymbol{d}_w^{\text{na}}(k) = 1$" means the predicted cache line by the k value is observed as non-access. In fact, with the wth input x_w known, for $k \in \mathcal{K}$, $\boldsymbol{d}_w^{\text{ac}}(k) = \mathbb{I}(\boldsymbol{f}(x_w,k) \in \boldsymbol{S}_w^{\text{ac}}, \boldsymbol{f}(x_w,k) \in \boldsymbol{S}^{\text{mo}})$ and $\boldsymbol{d}_w^{\text{na}}(k) = \mathbb{I}(\boldsymbol{f}(x_w,k) \in \boldsymbol{S}_w^{\text{na}}, \boldsymbol{f}(x_w,k) \in \boldsymbol{S}^{\text{mo}})$, where $\mathbb{I}$ is the indicator function. Suppose there are n_E encryptions, the number of encryptions that the predicted cache lines by the k value are observed as accesses by the adversary is:

$$\boldsymbol{N}^{\text{ac}}(k) = \sum_{w=1}^{n_E} \boldsymbol{d}_w^{\text{ac}}(k), \tag{5}$$

while the number of encryptions that the predicted cache lines by the k value are observed as non-accesses by the adversary is:

$$\boldsymbol{N}^{\text{na}}(k) = \sum_{w=1}^{n_E} \boldsymbol{d}_w^{\text{na}}(k). \tag{6}$$

If the adversary monitors all cache lines and the observed cache line accesses and non-accesses are correct, the predicated cache lines by the secret key k_c are

observed, so $\boldsymbol{N}^{\mathrm{ac}}(k_c)$ is always n_E and $\boldsymbol{N}^{\mathrm{ac}}(k_c)$ is always 0. Other key hypothesis value k_g would create some mismatch for large n_E, allowing the adversary to identify k_c. However, with advanced cache architectures or due to execution noise, adversaries may encounter two types of confusion errors, i.e., a cache line access is observed as a cache line non-access or a cache line non-access is observed as an access. We call these two errors by type I and type II errors. Then $\boldsymbol{N}^{\mathrm{ac}}(k_c)$ may not be n_E, and $\boldsymbol{N}^{\mathrm{na}}(k_c)$ may not be 0. We next provide the analysis of the quantity assuming no confusion errors in Section 3.2, and accounting for confusion errors in Section 3.3.

3.2 Probability Properties Based on Cache Architecture without Confusion Errors

In this section, for noise-free channel, we study probability properties about the number of accessed cache lines for one AES encryption, which helps to set the threshold value for using the probing time to detect cache access in the next Section 3.3.

Without confusion errors, for one encryption, the set of observed accessed cache lines is exactly same as the set of algorithm-based accessed cache lines. If a certain cache line $\boldsymbol{l}_i \in \boldsymbol{\mathcal{L}}$, $i = 1, \cdots, n_L$, is not accessed during one encryption, we have $\boldsymbol{l}_i \notin \boldsymbol{S}^{\mathrm{ac}}$. So, the corresponding probability of $\boldsymbol{l}_i$ being not accessed by one encryption is $\mathbb{P}(\boldsymbol{l}_i \notin \boldsymbol{S}_w^{\mathrm{ac}}) = (1 - 1/n_L)^{n_A}$. With n_E executions, the two number of times that $\boldsymbol{l}_i$ is accessed or non-accessed are $\sum_{w=1}^{n_E} \mathbb{I}(\boldsymbol{l}_i \in \boldsymbol{S}_w^{\mathrm{ac}})$ and $\sum_{w=1}^{n_E} \mathbb{I}(\boldsymbol{l}_i \notin \boldsymbol{S}_w^{\mathrm{ac}})$, denoted by $\boldsymbol{M}^{\mathrm{ac}}(\boldsymbol{l}_i)$ and $\boldsymbol{M}^{\mathrm{na}}(\boldsymbol{l}_i)$, respectively. In Property 1, we give the expected values of these two numbers.

Property 1 *Among n_E executions, the expected number of times that one certain cache line $\boldsymbol{l}_i \in \boldsymbol{\mathcal{L}}$, $i = 1, \cdots, n_L$, is observed as access/non-access is:*

$$\mathbb{E}[\boldsymbol{M}^{ac}(\boldsymbol{l}_i)] = n_E \cdot [1 - (1 - \frac{1}{n_L})^{n_A}], \qquad \mathbb{E}[\boldsymbol{M}^{na}(\boldsymbol{l}_i)] = n_E \cdot (1 - \frac{1}{n_L})^{n_A}, \quad (7)$$

where n_L and n_A are the total number of cache lines in the cache and the total number of accesses during one execution, respectively.

For one execution, the number of cache lines in the set $\boldsymbol{\mathcal{L}}$ being not accessed $\mid \boldsymbol{S}^{\mathrm{na}} \mid$ ranges from $\max(0, n_L - n_A)$ to $n_L - \min(1, n_A)$. In the following property, we give the probability that there are exactly r non-accessed cache lines lines, (see proofs in Appendix 8.1).

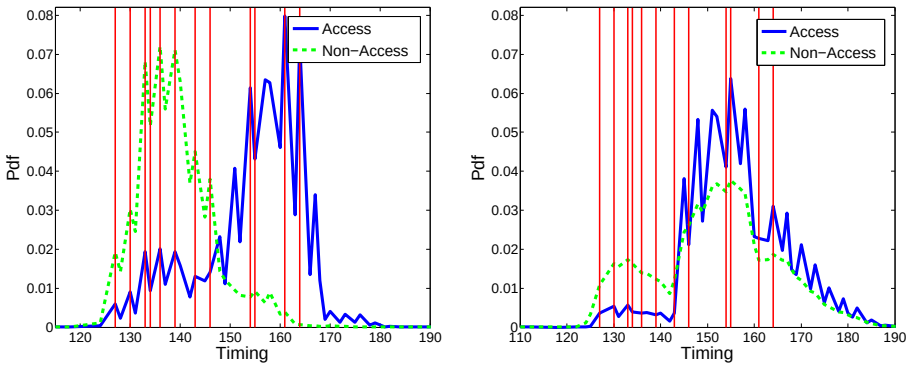
Property 2 *For one execution, the probability that there are exactly r cache lines that are not accessed is*

$$\mathbb{P}(\mid \boldsymbol{S}^{na} \mid = r) = \frac{\binom{n_L}{r} \sum_{j=0}^{r} (-1)^j \binom{r}{j} (n_L - j)^{n_A}}{(n_L)^{n_A}}. \quad (8)$$

3.3 Mixture Distribution of Probe Timing

Some recent advanced cache architectures can cause confusion errors, so that cache lines accesses observed by the adversary may differ from the real cache line accesses by the victim process [30–32]. Even without any advanced cache architectures, confusion errors can occur due to interference effects from other concurrently running processes or operations. Therefore an observed cache line access (identified by the probing time) does not always mean that this cache line is accessed by the algorithm. It may be accessed by other interfering or operating system processes. The same is true for an observed cache non-access (due to prefetching etc). In this section, we explicitly consider the effect of these two types of errors occurring during the executions.

For one monitored cache line, the adversary uses the probing time to classify this line into observed cache access (long timing) or non-access (short timing). There will be confusion errors unless the cache accesses and non-accesses based on algorithm correspond to two apart groups of probing time range. Fig. 1 plots the probability density functions (pdf) of probing timing (number of cycles) for algorithm-based cache line accesses (solid blue line) and cache line non-accesses (dash green line), respectively, for the first cache line and last cache lines on one real AES data set. This figure shows that the timing can not completely separate

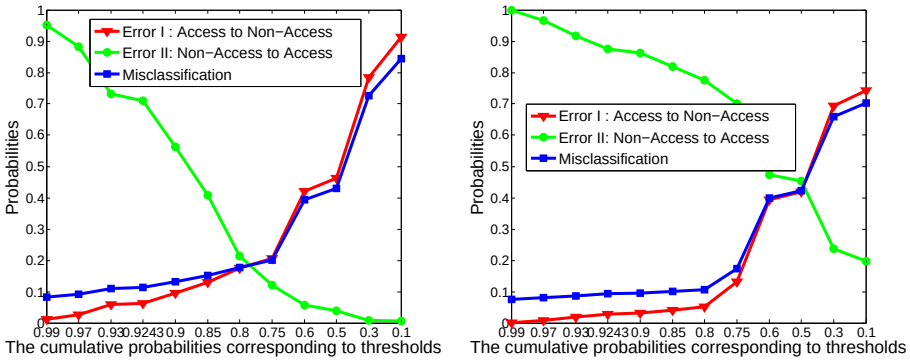


(a) The first cache line (b) The last cache line

Fig. 1: The probability density functions of timing corresponding to accessed and non-accessed cache lines for the first cache line and last cache line

the two types of cache line behavior (access vs non-access). Typically a timing threshold is used to separate the behaviors (shown as red vertical solid lines in the figure, where timing falling to the left of the line is taken as non-access, and timing falling to the right of the line is taken as access). Type I error rate accounts for the area under the solid blue line to the left of the threshold line (mistaking access as non-access), and type II error rate is for the area under the dash green line to the right of the threshold line. As the timing threshold increases,

Type I error rate increases but Type II error rate decreases. Fig. 2 shows the classification error rate corresponding to different thresholds. Comparing the pdf of the first and last cache lines in Fig. 1a and Fig. 1b, they have similar but different patterns which corresponds to different type I and II errors.



(a) The first cache line (b) The last cache line

Fig. 2: The misclassification error on real data for the first cache line and last cache line

From equation (7), on average a proportion $(1-1/n_L)^{n_A}$ of times a cache line should be classified as being non-access. Let $F(t)$ denote the cumulative density distribution (CDF) of probing time. Hence the threshold t_0, which satisfies

$$F(t_0) = (1 - 1/n_L)^{n_A}, \tag{9}$$

will yield the expected number of observed cache accesses agreeing to the expected number of algorithm-based cache accesses. If the algorithm-based cache line accesses and non-accesses are well separated, this threshold would result in 100% correct classification. However, this threshold t_0 value may not be optimal for one attack when there are indeed confusion errors. The adversary can use a threshold value t_g different from t_0. Abstractly, for the chosen threshold t_g, we can consider that there is one (observed) number of cache lines accesses $n_A^g(t_g) = \log(F(t_g))/\log(1 - 1/n_L)$. When $t_g = t_0$, we have $n_A^g = n_A$. In what follows, instead of n_A accesses by the algorithm, we focus on the n_A^g abstract observed accesses to derive the success rate formulas in next Section 4. The n_A^g is determined by the Type I and Type II error rates.

For a cache line $\boldsymbol{l}_i$, $i = 0, \cdots, n_L - 1$, using a threshold value t_g, let $p_1(t_g)$ and $p_2(t_g)$ denote the Type I error rate and the Type II error rate, respectively. That is, p_1 is the probability that one algorithm-based access to $\boldsymbol{l}_i$ is observed as non-access by the adversary, p_2 is the probability that one algorithm-based non-access to $\boldsymbol{l}_i$ is observed as access by the adversary. Let r_1 be the average proportion of algorithm-based cache accesses, then the average proportion of

algorithm-based cache non-accesses is $1 - r_1$. Then the adversary observes an average number of cache accesses as

$$r_0(t_g) = r_1 \cdot (1 - p_1(t_g)) + (1 - r_1) \cdot p_2(t_g). \tag{10}$$

On the other hand, $r_0(t_g) = 1-(1-1/n_L)^{n_A^g}$. Match this to the above equation, we get $n_A^g = \log(1-r_1+r_1p_1(t_g)-(1-r_1)p_2(t_g))/\log(1-1/n_L)$. For simplicity, in what follows, we use r_0, p_1, p_2 and n_A^g instead of $r_0(t_g)$, $p_1(t_g)$,$p_2(t_g)$ and $n_A^g(t_g)$. Smaller p_1 or larger p_2 values would lead to larger n_A^g values.

Recall that $\boldsymbol{f}(x, k_c)$ denotes the cache line accessed in one targeted SBox operation, which is calculated from the secret key k_c and input x. This access can be missed by the adversary, and we denote the probability by $1 - p_0$. In the next section, we derive the success rate formulas in terms of n_A^g and p_0 instead of p_1 and p_2. This leads to somewhat simpler formula for the probability of observing the targeted operation $\boldsymbol{f}(x, k_c)$ access below. For the adversary to report that the line $\boldsymbol{f}(x, k_c)$ is not accessed after one execution of AES, it has to be observed as non-access for the key-sensitive access, also for the other $n_A^g - 1$ operations.

$$\mathbb{P}(\boldsymbol{f}(x, k_c) \notin \boldsymbol{S}^{\mathrm{ac}}) = (1 - p_0)(1 - 1/n_L)^{n_A^g-1}. \tag{11}$$

On the other hand, we have $\mathbb{P}(\boldsymbol{f}(x, k_c) \notin \boldsymbol{S}^{\mathrm{ac}}) = p_1$. So, we get $p_0 = 1 - p_1(1 - n_L)/(1 - r_0)$.

4 Success Rate Formulas

In this section, we consider two kinds of attacks, one utilizing the cache access and the other utilizing cache non-access. We present the success rate formulas for these attacks on the first round and the last round of AES, typically conducted for cache timing attacks.

The attacks based on the number of cache accesses (or non-access) over n_E executions are in fact additive distinguisher as defined in [33]. Hence the central limit theorem yields their success rate described by multivariate Gaussian distribution [33, 34]. We shall find explicit formulas for the mean and variance of the Gaussian distribution.

For cache analysis, after the wth execution, we would have three groups of cache lines. The first group contains the predicted accessed cache lines which depend on the algorithm; the second group $\boldsymbol{S}_w^{\mathrm{ac}}$ contains all observed (in abstraction) accessed cache lines; and the last one $\boldsymbol{S}^{\mathrm{mo}}$ being the cache lines which are monitored by the adversary. The adversary only observes the cache lines in $\boldsymbol{S}^{\mathrm{mo}}$ and concludes these cache lines accessed or non-accessed. For the wth execution, the adversary observes a set of cache accesses $\boldsymbol{S}_w^{\mathrm{ac}} \bigcap \boldsymbol{S}^{\mathrm{mo}}$ and a set of cache non-accesses $\boldsymbol{S}_w^{\mathrm{na}} \bigcap \boldsymbol{S}^{\mathrm{mo}}$.

4.1 Success Rate for Attacks on the Last Round

For the last round of AES algorithm, the cache accesses leak key information through the operation *add_round_key*, i.e.,

$$c = \mathrm{SBox}(s) \oplus k_c \tag{12}$$

where c is the ciphertext byte, and s is some intermediate state that will lead to the predicted cache line index. Since there are d_L T-table elements in one cache line (corresponding to d_L continuous s values), given a ciphertext c_w, for one observed accessed cache lines $\boldsymbol{l}_i \in \boldsymbol{L}$, a set of d_L key values $\{k \mid \boldsymbol{f}_{\mathrm{last}}(c_w, k) = \boldsymbol{l}_i\}$ can cause such an access. Here, $\boldsymbol{f}_{\mathrm{last}}(c_w, k)$ is the cache line which includes $\mathrm{InvSBox}(c_w, k)$, where $\mathrm{InvSBox}(\cdot, \cdot)$ is the operation of *inverse_*SBox. We call all k values in the set as *observed cache-access keys*. Since the operation (inverse SBox) is non-linear, the adversary can recovery the secret key, while based the first round the adversary only can recover the secret key-line explained in Section (4.2).

We use the number of cache accesses $\boldsymbol{N}^{\mathrm{ac}}_{\mathrm{last}}$ (cache non-accesses $\boldsymbol{N}^{\mathrm{na}}_{\mathrm{last}}$) as the distinguisher, selecting the key as the secret key which is observed most (least) often in n_E observed sets among all key hypothesis, where

$$\boldsymbol{N}^{\mathrm{ac}}_{\mathrm{last}}(k) = \sum_{w=1}^{n_E} \boldsymbol{d}^{\mathrm{ac}}_w(k), \quad \boldsymbol{N}^{\mathrm{na}}_{\mathrm{last}}(k) = \sum_{w=1}^{n_E} \boldsymbol{d}^{\mathrm{na}}_w(k). \tag{13}$$

with $\boldsymbol{d}^{\mathrm{ac}}_w(k) = \mathbb{I}(\boldsymbol{f}_{\mathrm{last}}(c_w, k) \in \boldsymbol{S}^{\mathrm{ac}}_w, \boldsymbol{f}_{\mathrm{last}}(c_w, k) \in \boldsymbol{S}^{\mathrm{mo}})$ and $\boldsymbol{d}^{\mathrm{na}}_w(k) = \mathbb{I}(\boldsymbol{f}_{\mathrm{last}}(c_w, k) \in \boldsymbol{S}^{\mathrm{na}}_w, \boldsymbol{f}_{\mathrm{last}}(x_w, k) \in \boldsymbol{S}^{\mathrm{mo}})$ with input c_w known. For cache access and non-access based attacks, we denote the $(|\mathcal{K}| - 1)$-dimensional comparison vectors between the secret key k_c and false candidate keys as $\boldsymbol{\delta}^{\mathrm{ac}}_{\mathrm{last}}$ and $\boldsymbol{\delta}^{\mathrm{na}}_{\mathrm{last}}$ respectively, with the element corresponding to $k_g \in \mathcal{K} \backslash k_c$ as

$$\boldsymbol{\delta}^{\mathrm{ac}}_{\mathrm{last}}(k_g) = \frac{\boldsymbol{N}^{\mathrm{ac}}_{\mathrm{last}}(k_c) - \boldsymbol{N}^{\mathrm{ac}}_{\mathrm{last}}(k_g)}{n_E} = \frac{1}{n_E} \sum_{w=1}^{n_E} [\boldsymbol{d}^{\mathrm{ac}}_w(k_c) - \boldsymbol{d}^{\mathrm{ac}}_w(k_g)], \tag{14}$$

$$\boldsymbol{\delta}^{\mathrm{na}}_{\mathrm{last}}(k_g) = \frac{\boldsymbol{N}^{\mathrm{na}}_{\mathrm{last}}(k_c) - \boldsymbol{N}^{\mathrm{na}}_{\mathrm{last}}(k_g)}{n_E} = \frac{1}{n_E} \sum_{w=1}^{n_E} [\boldsymbol{d}^{\mathrm{na}}_w(k_c) - \boldsymbol{d}^{\mathrm{na}}_w(k_g)]. \tag{15}$$

An attack is successful if and only if all coordinates in the comparison vector $\boldsymbol{\delta}^{\mathrm{ac}}_{\mathrm{last}}$ ($\boldsymbol{\delta}^{\mathrm{na}}_{\mathrm{last}}$) are positive (negative). If there is only one access, i.e., $n_A = 1$, the distinguisher $\boldsymbol{\delta}^{\mathrm{ac}}_{\mathrm{last}}(k_g)$ ($\boldsymbol{\delta}^{\mathrm{na}}_{\mathrm{last}}(k_g)$) follows a binomial distribution $\binom{n_E}{p_0}$ ($\binom{n_E}{1-p_0}$), the exact success rate formula can be gotten. When $n_A \neq 1$, the access/non-access counts follow more complicated distributions from the Property 2. However, for $k_g \in \mathcal{K} \backslash k_c$, the distinguisher $\boldsymbol{\delta}^{\mathrm{ac}}_{\mathrm{last}}(k_g)$ ($\boldsymbol{\delta}^{\mathrm{na}}_{\mathrm{last}}(k_g)$) is the average of the difference between the indicator functions under k_c and k_g, then it is additive. So, we can apply the central limit theorem, and the success rate $SR^{\mathrm{ac}}_{\mathrm{last}}$ and $SR^{\mathrm{na}}_{\mathrm{last}}$ based on the cache accesses and non-accesses are

$$\begin{aligned} SR^{\mathrm{ac}}_{\mathrm{last}} &= \mathbb{P}(\boldsymbol{\delta}^{\mathrm{ac}}_{\mathrm{last}} > 0) = \Phi_{\boldsymbol{\Sigma}^{\mathrm{ac}}_{\mathrm{last}}}(\boldsymbol{\mu}^{\mathrm{ac}}_{\mathrm{last}}), \\ SR^{\mathrm{na}}_{\mathrm{last}} &= \mathbb{P}(\boldsymbol{\delta}^{\mathrm{na}}_{\mathrm{last}} < 0) = \Phi_{\boldsymbol{\Sigma}^{\mathrm{na}}_{\mathrm{last}}}(-\boldsymbol{\mu}^{\mathrm{na}}_{\mathrm{last}}), \end{aligned} \tag{16}$$

where $\Phi_{\boldsymbol{\Sigma}}(\cdot)$ denote the CDF of a mean zero Gaussian distribution with covariance matrix $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}^{\text{ac}}_{\text{last}}$ and $\boldsymbol{\Sigma}^{\text{ac}}_{\text{last}}$ (or $\boldsymbol{\mu}^{\text{na}}_{\text{last}}$ and $\boldsymbol{\Sigma}^{\text{na}}_{\text{last}}$) are mean-vector and covariance-matrix of $\boldsymbol{d}^{\text{ac}}_1(k_c) - \boldsymbol{d}^{\text{ac}}_1(k_g)$ (or $\boldsymbol{d}^{\text{na}}_1(k_c) - \boldsymbol{d}^{\text{na}}_1(k_g)$).

Theorem 1. *When the adversary monitors all $N_M = N_L$ cache lines, the success rate of the cache non-access/access based attack is given by (16). And*

- *each element in the mean $\boldsymbol{\mu}^{na}_{last}$ is:*

$$\frac{d_L - p_0 \cdot n_K}{n_K - 1}(1 - \frac{1}{n_L})^{n^g_A - 1}; \tag{17}$$

  and $\boldsymbol{\mu}^{ac}_{last} = -\boldsymbol{\mu}^{na}_{last}$;
- *each diagonal element of $\boldsymbol{\Sigma}^{na}_{last}/\boldsymbol{\Sigma}^{ac}_{last}$ is:*

$$\begin{array}{l}(2 - p_0 - \frac{d_L - p_0}{n_K - 1})(1 - \frac{1}{n_L})^{n^g_A - 1} - (\frac{p_0 \cdot n_K - d_L}{n_K - 1})^2(1 - \frac{1}{n_L})^{2(n^g_A - 1)} \\ -2(1 - p_0)\frac{n_K - d_L - 1}{n_K - 1}(\frac{(n_K - d_L)(n_K - d_L - 1)}{n_K(n_K - 1)})^{n^g_A - 1};\end{array} \tag{18}$$

- *each off-diagonal element in $\boldsymbol{\Sigma}^{na}_{last}/\boldsymbol{\Sigma}^{ac}_{last}$ is:*

$$\begin{array}{l}(1 - p_0)(1 - \frac{1}{n_L})^{n^g_A - 1} - (\frac{p_0 \cdot n_K - d_L}{n_K - 1})^2(1 - \frac{1}{n_L})^{2(n^g_A - 1)} \\ + \frac{(n_K - d_L - 1)(-n_K - d_L + 2 - 2p_0 + 2p_0 \cdot n_K)}{(n_K - 1)(n_K - 2)}(\frac{(n_K - d_L)(n_K - d_L - 1)}{n_K(n_K - 1)})^{n^g_A - 1},\end{array} \tag{19}$$

where n_K, n_L, d_L are the dimensions of key space, cache lines space, and one single cache line size, respectively. And $n_K = n_L \cdot d_L$.

Remark 1. When the adversary monitors all cache lines, the set of observed accessed cache lines is the complementary of the set of of observed non-accessed cache lines in the cache line space $\boldsymbol{\mathcal{L}}$. Hence the attack based on cache access is exactly the same as the attack based on cache non-access. Otherwise, these two attacks can differ. When the adversary monitors only a subset of $n_M < n_L$ cache lines, the attack based on cache non-access is usually better.

Remark 2. When the adversary monitors only a subset of $n_M < n_L$ cache lines, the mean and covariances formulas are more complicated. In fact, the proof in the Appendix 8.2 provides the formulas for this general case. The diagonal (off-diagonal) elements of $\boldsymbol{\Sigma}^{\text{na}}_{\text{last}}$ are not all the same there, and simplify to the same value in the special case of $n_M = n_L$. Formulas for the general cases are used in Section 5 for attacks monitoring a subset only.

Remark 3. When there is no confusion error so that $p_0 = 1$, the predicted cache lines by the secret key are always observed to be accessed. The mean elements in $\boldsymbol{\mu}^{\text{na}}_{\text{last}}$ are negative since $d_L - 1 \cdot n_K < 0$, so as the number of encryptions n_E increases, the success probability of the attack increases to one. As p_0 decreases, the confusion error increases, and the success rate of the attack decreases. When $p_0 = d_L/n_K = 1/n_L$, a cache line access by the victim becomes a totally random observation by the adversary. Hence no information is leaked in that case, reflected as mean zero in the Gaussian formula. For even smaller $p_0 < 1/n_L$,

the predicted cache lines corresponding to the secret key are more likely observed to be non-accessed than random observations. In other words, a cache access (non-access) by the victim tends to be observed by the adversary as the opposite: non-access (access). In that case, the adversary will need to reverse the attack: instead of selecting the k value that is most frequently observed in the cache-access-keys, select the least frequently observed. We call such a attack *reversed attack*. This is a very rare case for various architectures. In Section 5, we point out that the random-permutation cache architecture satisfies and the reversed attack can be used. The experimental results in Fig. 9b.

Remark 4. The Type II error rate p_2 affects the success rate through n_A^g, the apparent observed number of accesses to the adversary. The n_A^g increases as p_2 increases (when p_0 is fixed). This leads to increase in both the mean and variance. While the increases in the mean is due to the factor $(1-1/n_L)^{n_A^g-1}$, the increase in the variance is slower than corresponding rate $(1-1/n_L)^{2(n_A^g-1)}$. So the overall effect of increase in Type II error rate p_2 is to decrease the success rate of the attack.

Remark 5. For the last round attack, the cache line index is determined by the XOR result between the ciphertext and the SBox-inverse of the guessed key. In this paper, we assume that the joint distribution of the predicted caches lines corresponding to two different key candidates follows the distribution of randomly selecting two balls from $[0 : n_L - 1]$ without replacement. However, due to the SBox structure, the distribution actually slightly deviates from that which will cause a small discrepancy between the theoretical success rate and the empirical one, shown in Fig. 3.

4.2 The First Round Attack

For the first round of AES, the cache accesses leak information through the operation

$$p_w \oplus k_c \tag{20}$$

where p_w is one plaintext and k_c is the secret key. The operation is linear. Given plaintext p_w, the set of key values that will cause the SBox in the first round to access the line $\boldsymbol{l}_i \in \boldsymbol{\mathcal{L}}$ is $\{k \mid \boldsymbol{f}_{1\text{st}}(p_w, k) = \boldsymbol{l}_i\}$, where $\boldsymbol{f}_{1\text{st}}(p_w, k)$ is the cache line including $p \oplus k$. When n_L is a positive power of 2 ($n_L = 16$ in our AES example implementation), a cache line $\boldsymbol{l}_i = \{i \cdot d_L, i \cdot d_L + 1, ..., i \cdot d_L + d_L - 1\}$ (for $i = 0, ..., n_L - 1$) includes all the possible values with the same higher $\log_2(n_L)$ bits. Since the operation of $p_w \oplus k$ is linear, the set $\{k \mid \boldsymbol{f}_{1\text{st}}(p_w, k) = \boldsymbol{l}_i\}$ also contains all possible k values with the same higher $\log_2(n_L)$ bits. We denote such a set by $\langle k \rangle$, calling a key-line. Then there are n_L such key-lines, and we denote the set of these key-lines as $\boldsymbol{\mathcal{K}}$ corresponding to the set of cache lines $\boldsymbol{\mathcal{L}}$. For each plaintext p_w, $\boldsymbol{f}_{1\text{st}}(p_w, \cdot)$ is a bijective mapping between $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\mathcal{L}}$. Therefore the first round cache-based attack can only recover the key line $\langle k_c \rangle$ that contains the true secret key k_c. In contrast, due to the nonlinear SBox operation, the key-line $\{k \mid \boldsymbol{f}_{\text{last}}(c_w, k) = \boldsymbol{l}_i\}$ for last round is not restricted to

these n_L possible key-lines values only, which allows the adversary to recover k_c instead of only the key-line $\langle k_c \rangle$. In this paper, the success rate of the first round attack is defined as the probability to correctly retrieve the key-line $\langle k_c \rangle$.

Similarly to before, we define

$$\begin{aligned} \boldsymbol{N}_{1\text{st}}^{\text{ac}}(k) &= \tfrac{1}{n_E} \sum_{w=1}^{n_E} \mathbb{I}(\boldsymbol{f}_{1\text{st}}(p_w, k) \in \boldsymbol{S}_w^{\text{ac}}, \boldsymbol{f}_{1\text{st}}(p_w, k) \in \boldsymbol{S}^{\text{mo}}) \\ \boldsymbol{N}_{1\text{st}}^{\text{na}}(k) &= \tfrac{1}{n_E} \sum_{w=1}^{n_E} \mathbb{I}(\boldsymbol{f}_{1\text{st}}(p_w, k) \in \boldsymbol{S}_w^{\text{na}}, \boldsymbol{f}_{1\text{st}}(p_w, k) \in \boldsymbol{S}^{\text{mo}}) \end{aligned} \tag{21}$$

,

which have the same values for all d_L key values in the same key-line $\langle k \rangle$. So the success rates of revealing the secret-key line $\langle k^* \rangle$ are $SR_{1\text{st}}^{\text{ac}} = \Phi_{\boldsymbol{\Sigma}_{1\text{st}}^{\text{ac}}}(\boldsymbol{\mu}_{1\text{st}}^{\text{ac}})$ and $SR_{1\text{st}}^{\text{na}} = \Phi_{\boldsymbol{\Sigma}_{1\text{st}}^{\text{na}}}(-\boldsymbol{\mu}_{1\text{st}}^{\text{na}})$, with the mean-vector $\boldsymbol{\mu}_{1\text{st}}^{\text{ac}}(\boldsymbol{\mu}_{1\text{st}}^{\text{na}})$ and the covariance-matrix $\boldsymbol{\Sigma}_{1\text{st}}^{\text{ac}}(\boldsymbol{\Sigma}_{1\text{st}}^{\text{na}})$ given in the following Theorem 2.

Theorem 2. *When the adversary monitors all $n_M = n_L$ cache lines, then*

- *each element in the mean $\boldsymbol{\mu}_{1st}^{na}$ is*

$$\frac{1 - p_0 \cdot n_L}{n_L - 1}(1 - \frac{1}{n_L})^{n_A^g - 1}, \tag{22}$$

  and $\boldsymbol{\mu}_{1st}^{ac} = -\boldsymbol{\mu}_{1st}^{na}$;
- *each diagonal element of $\boldsymbol{\Sigma}_{1st}^{na}/\boldsymbol{\Sigma}_{1st}^{ac}$ is*

$$\begin{aligned} &(2 - p_0 - \tfrac{1-p_0}{N_L-1})(1 - \tfrac{1}{N_L})^{N_A^g - 1} - (\tfrac{p_0 \cdot N_L - 1}{N_L - 1})^2 (1 - \tfrac{1}{N_L})^{2(N_A^g - 1)} \\ &-2(1 - p_0)(1 - \tfrac{1}{N_L - 1})(1 - \tfrac{2}{N_L})^{N_A^g - 1}; \end{aligned} \tag{23}$$

- *each off-diagonal element of $\boldsymbol{\Sigma}_{1st}^{na}/\boldsymbol{\Sigma}_{1st}^{ac}$ is*

$$\begin{aligned} &(1 - p_0)(1 - \tfrac{1}{n_L})^{n_A^g - 1} - (\tfrac{p_0 \cdot n_L - 1}{n_L - 1})^2 (1 - \tfrac{1}{n_L})^{2(n_A^g - 1)} \\ &+(-1 + 2p_0 - \tfrac{(1-p_0)(p_0 n_L - 1)}{(n_L - 1)^3})(1 - \tfrac{2}{n_L})^{n_A^g - 1} \end{aligned} \tag{24}$$

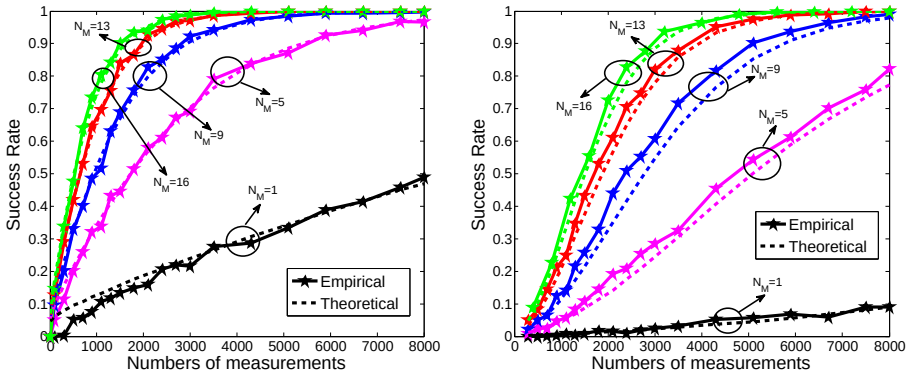
The derivation of Theorem 2 is very similar to that of Theorem 1, and omitted here. The first round attack generally has a success rate higher than that of the last round attack. Note that the first round attack only recovers a d_L-sized key-line $\langle k_c \rangle$ while the last round attack recovers the exact key value k_c.

5 Numerical Results

We evaluate the theoretical success rates formulas on synthetic data and the physical measurement data described in Section 2.2. We also use these formulas to evaluate effects of some secure cache architectures against timing attacks described in [31].

5.1 Success Rates on Synthetic Data

First we simulate prime+probe attack data on AES, with confusion error rate $p_0 = 0.30$ for the key-sensitive access. We simulate the attacks with varying number of monitored cache lines by the adversary. Fig. 3 plots the success rates of the first round and last round attacks. For the first round attack, there are very good agreements between the theoretical success rate curve and empirical one. For the last round attack, there are a small discrepancies as we mentioned in Remark 5. Also, we can see that more leakage information is extracted (higher success rate) when more cache lines are monitored. But the marginal incremental value of monitoring an extra cache line decreases when more cache lines are monitored.



(a) The first round attacks (b) The last round attacks

Fig. 3: Empirical and theoretical success rates based on cache non-access on simulated AES data.

The success rate formulas in Section 4 quantify the leakage resilience of cache architectures against AES through the confusion error rate p_0 value. Fig. 4 plots the number of traces required to achieve SR $= 0.8$ on simulated data for different p_0 values. Without confusion errors being taken into account, it is equivalent to assuming $p_0 = 1$, thus needing only around 100 traces to break AES as shown in Fig. 4. A perfect protection against cache leakage means that $p_0 = 1/N_L$. In practice, the confusion rate p_0 lies between those two extreme values, the practitioners can decide if a system meets security specifications based on the quantitative metrics in Fig. 4. Some advanced cache architectures do increase security by decreasing p_0. Their effect can then be quantitatively assessed through our formulas as done in Section 5.3 later.

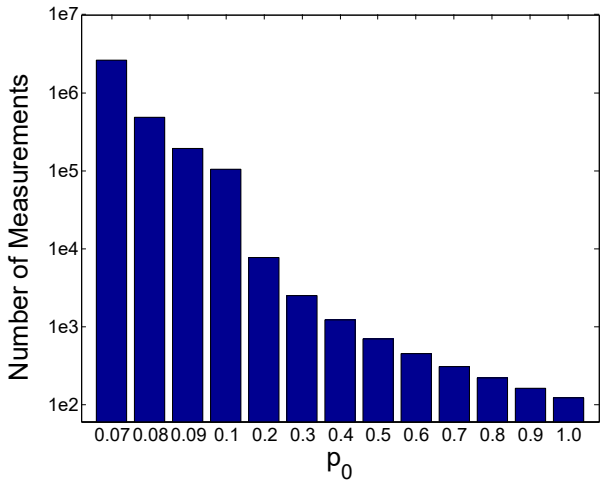


Fig. 4: Sample size for achieving SR $= 0.8$ for different values of p_0.

5.2 Experimental Results on Physical Implementations

We now apply the statistical model on the physical measurement data for the AES implementation described in Section 2.2. As shown in Fig. 1, the misclassification error of accesses and non-accesses are different for different cache lines. In Table 1, we present the probability estimation value of one predicated cache

Table 1: The values of p_0 for n_L cache lines

L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7
0.5385	0.0766	0.4642	0.1182	0.2707	0.1134	0.2473	0.0871
L_8	L_9	L_{10}	L_{11}	L_{12}	L_{13}	L_{14}	L_{15}
0.2909	0.0753	0.4395	0.0938	0.4265	0.0648	0.4399	0.4611

line being observed due to this access for each cache line L_i, $i = 0, \cdots, N_L - 1$. Large value of p_0 means less confusion error, according to the success rate formulas in Section 4. Then more secret information leaks through the first cache line than others. Since the values of p_0 for cache lines L_1, L_3, L_5, L_7, L_9 and L_{11} and L_{13} are small, near the random probability $1/N_L$, there are a little information leaked through monitoring these lines here. Notice that the modern last-level cache is sliced into two slices, and depending on the physical address, it can be stored into one of slices. Since we only monitor one of cache slices, we cannot observe the information leaking from cache sets of the other slice. Therefore, we are seeing low p_0 values for those cache lines. Fig. 5 draw the empirical success rate curves of last round attacks using cache misses with all cache lines versus

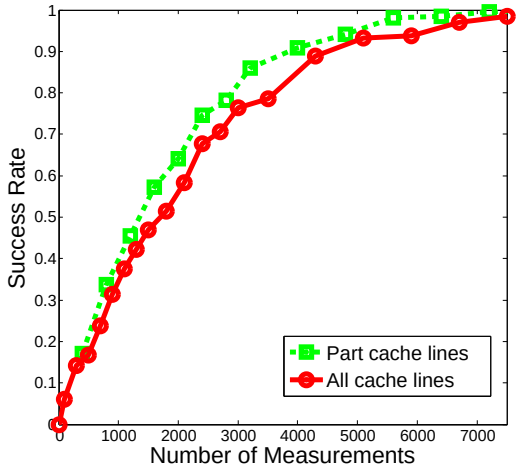


Fig. 5: Success rates of last round attacks using all or part cache lines on the AES implementation.

without these low leakage cache lines. The attack success rate is improved when using only the high leakage lines.

Now, we verify the success rate formulas on the first round attack. As discussed in Section 3.3, we need to select the threshold value to classify cache access and cache non-access using the probing time. First, we set the N_A^g to the theoretical AES value $N_A = 40$ for each cache line, which corresponds to $r_0 = 0.9243$. This is the ideal threshold value assuming no confusion errors. Fig. 6 draws the theoretical and empirical success rate curves for attacks monitoring the first cache line or monitoring all $N_L = 16$ caches lines. When all $N_L = 16$ caches lines are monitored, the cache accesses and non-accesses provide the equivalent information for adversaries. For the single cache line monitoring, the cache non-access attack is stronger than the cache access attack.

The first round attacks and the last round attacks are compared assuming all cache lines are monitored. Since the first-round attack can only retrieve the higher nibble of a key byte, with each nibble corresponding to 16 key byte values (enumerating the lower 4-bit nibble) while the last-round attack recovers the entire key byte. For a fair comparison, in Fig. 7(b), in addition to the common used first order success rates curves, we also add the 16-th success rate (the probability that the true key value is in the top 16 key candidates) for the last-round attack. The 16-th order success rate curve of the last round attack is close to the first order success rate of the first round attack. Thus, some method of disrupting aligned T-tables would improve the first round attack to achieve the performance of last round attacks.

The above threshold selection assumes no confusion errors, which is not true for this data set. Using the above threshold value, we find the probability to correctly observe a cache access is $p_0 = 0.5385$ when monitoring only the first

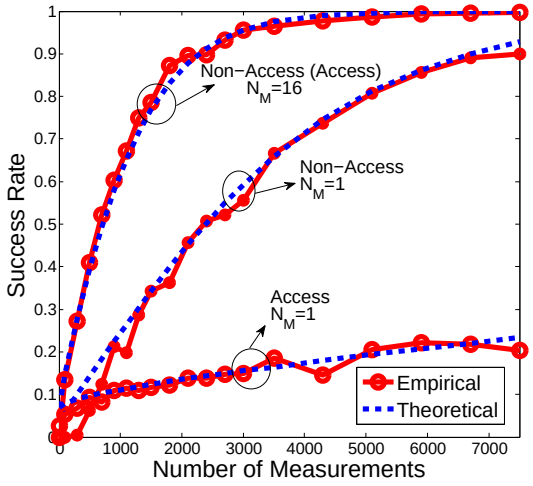


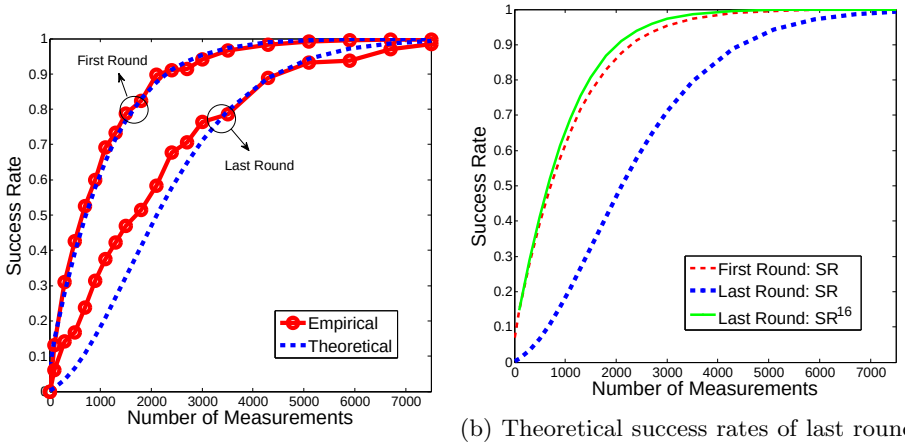
Fig. 6: Success rates of first round attacks for different numbers of cache lines monitored on the AES implementation.

cache line. When monitoring all cache lines, since plaintext follows an discrete uniform distribution, so does the predicated cache lines, so we estimate p_0 by the average of all p_0 values, approximating to 0.2630. These probabilities still exceeds $1/N_L = 1/16 = 0.0625$, therefore our formulas predict that the attacks succeed for large number of executions q. But different threshold values can be used in the attacks. We consider the threshold values shown in Fig. 1 for the attacks monitoring all cache lines. The numbers of measurements needed to achieve the 80% success rate are plotted in Fig 8. It shows that the attack does better with a threshold value of $r_0 = 0.93$ and $N_A^g = 41.2$, slightly different from the threshold $r_0 = 0.9243$ value which corresponds to assuming no confusion errors.

5.3 Quantification of Effects on SCA by Cache Architectures

Different cache architectures protect against side-channel attacks through interference occurred between the input (victims') and the output (adversaries'). No matter how these cache architectures work, only two types of error in the observations of adversaries can affect the efficiency of revealing the secret information, one is taking access as non-access, the other is taking non-access as access. Zhang and Lee [31] considered several advanced architectures and used mutual information to compare them. They showed several example interference probability tables, and it would decide the type I and type II error rates when the adversary classifies the cache accesses versus cache non-accesses.

In Table 2, we give the notations of general interference probabilities for some Cache. Now, we apply the abstract statistical model to evaluate effects of different architectures on the access-driven SCA. For the example abstract cache architecture described in Table 2, there are three normal cache lines $N_L = 3$, and



(a) Comparison of success rates between the first round and the last round attacks on the AES implementation.

(b) Theoretical success rates of last round attacks including the 16-th order success rate, where SR denotes the common used first order success rate, while SR^{16} is the 16-th order success rate.

Fig. 7: Comparison of success rates when monitoring all cache lines.

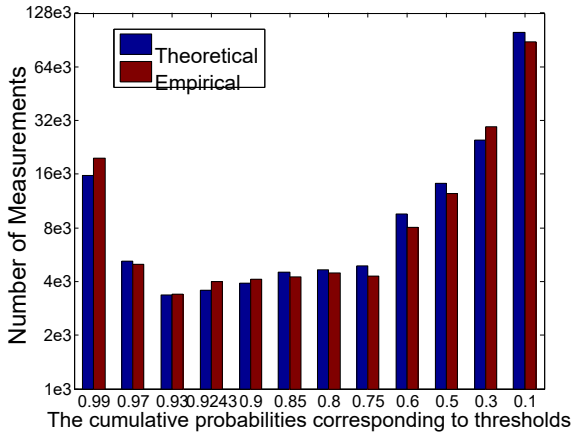


Fig. 8: Sample sizes (on log-scale) for achieving 80% success rates for attacks using the last round on real AES data.

an extra input cache line I_{-1}. Here, we focus on the effects caused by the cache architecture, ignoring other noise. When attackers use this cache architecture to launch implementations, they can distinguish accesses and non-accesses, and the classification errors only come from the cache architecture. In Table 2, there is $p_{0,0} + p_{1,0} + p_{2,0}$ probability of input cache line I_0 occuring, and the input cache line I_0 and the output cache line O_0 coincide with the probability $p_{0,0}$. So, when the victim input the cache line I_0, the corresponding output O_0 is observed

Table 2: Interference Probability for cache

$P_{\mathcal{I},\mathcal{O}}(I,O)$	I_0	I_1	I_2	I_{-1}
O_0	$p_{0,0}$	$p_{0,1}$	$p_{0,2}$	$p_{0,-1}$
O_1	$p_{1,0}$	$p_{1,1}$	$p_{1,2}$	$p_{1,-1}$
O_2	$p_{2,0}$	$p_{2,1}$	$p_{1,3}$	$p_{2,-1}$

with the probability $p_{0,0}/(p_{1,0}+p_{1,0}+p_{2,0})$, and then the rate of Type I error (p_1), taking accesses as non-accesses, is $1-p_{0,0}/(p_{0,0}+p_{2,0}+p_{2,0})$. When the victim does not input the cache line I_0, O_0 should not be observed, so there is $1-p_{0,0}+p_{1,0}+p_{2,0}$ possibility of cache line I_0 non-accessed. Hence, the rate of Type II error (p_2), taking non-accesses as accesses, is $(\sum_{j\neq 0} p_{0,j} - p_{0,0})/(1-\sum_j p_{j,0})$. Further more, by the equation (10) in Section 3.3, with the values of p_1 and p_2, we have $r_0 = r_1(1-p_1)+r_2p_2$ and $p_0 = 1-p_1(1-1/N_L)/(1-r_0)$, where N_L is the number of cache lines, r_1 and r_2 are determined by the algorithm.

We investigate the performance of five example cache architectures in [31]: conventional cache, partition-locked cache without preload (PL-w/o preload), random-eviction cache (RE), random-permutation cache (RP), newcache (New). Letting $N_A = 3$, then $r_1 = 0.7037$ and $r_2 = 0.2963$, we show the values of some intermediate quantities of our formulas for these cache architectures in Table 3. From this table, the conventional cache and partition-locked cache

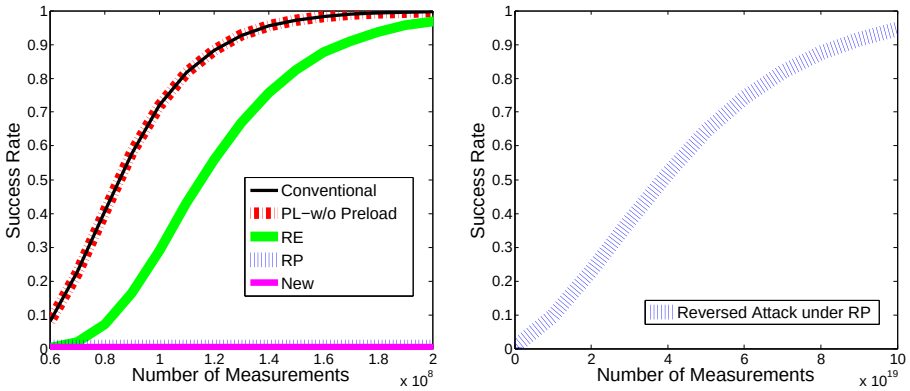
Table 3: Quantities for cache architectures when $N_A = 3$

Cache Architectures	p_1	p_2	p_0	r_0
Conventional	0.00	0.00	1.00	0.7037
PL-w/o preload	0.00	0.00	1.00	0.7037
RE	0.00	0.2614	1.00	0.7811
RP	0.6687	0.3337	0.3326	0.3320
New	0.6667	0.3333	0.3333	0.3333

without preload share the same values of quantities, no Type I and Type II error occurring, so both of these caches lead the largest leakage information and they have no difference from the view of attackers. Using random-eviction cache, $p_1 = 0$ but $p_2 = 0.2614$, then the predicted accesses can be observed correctly, but more non-accesses would be taken as accesses. So more traces are required to detect the secret key than under the first two cache architectures. For the newcache, since p_0 equals to the probability $1/N_L$ corresponding to perfectly random distribution on all cache lines, so theoretically there is no information leaked. For the random-permutation cache, $p_0 = 0.3326$ which is less than $1/N_L$,

then there is a little secret information leaked. However, since $p_0 < 1/N_L$, if the adversaries want to recover the secret key, they need take the observed access (non-access) data set as non-access (access) data to do attack, as in Remark 3 in Section 4.1.

Here, we consider AES with $N_A = 40$ and compare the vulnerabilities of these cache architectures. Fig. 9 shows the success rate curves of the last round attack on AES from our theoretical formula on the five example architectures. From this figure, we can see that conventional and PL-w/o preload caches leak most information, and cache attack is equally effective on these two architectures. The random-eviction cache architecture provides some protection, lowering the success rate of the attack significantly. Since $p_0 < 1/N_L$ for random-permutation cache, it can protect against the normal attack, but the reversed attack can help to recovery the secret key with larger number of traces than the first three cache architectures, as shown in Fig. 9b. The new cache architecture is most effective against the SCA, eliminate the leakage almost entirely, corresponding to its $p_0 = 1/N_L$.



(a) Normal Attacks (b) Reversed attack under RP

Fig. 9: Theoretical success rates for attacks using the last rounds under some advanced cache architectures.

6 Discussions

6.1 Related Work

How to evaluate an advanced cache architecture vulnerability comprehensively and accurately is an important issue. Previous work can be classified into two groups based on the metrics studied.

Success rate: such work predicts the probability that an attack succeeds given the number of side-channel measurements. This is indeed the ultimate practical measure to reflect the architecture's vulnerability. However, for cache-driven attacks, the previous success rate model [28] bases on the conventional cache architecture and perfect implementation environment, i.e., no confusion between accesses and non-accesses. The model can not differentiate the advanced architecture, and often overestimates the side-channel attacks. Savas and Yilmaz [11] showed the average number of keys recovered on simulated data for different overall noise level which did not separate further the effect of the two types of errors on the side-channel leakage.

Intermediate security related quantities: those work study some quantities related to the cache architectures. One is the probability distribution of some events, such as correctly identifying the predicted cache access for different cache architectures [30], a cache hit (miss) occurring by the qth random access to one table for cache memories with prefetching [32]. The other one is the mutual information, serving as a measure to evaluate effects of misclassification, and compare different architectures [31]. To some extend, these quantities measure the effects caused by misclassification. However, the probability quantities and mutual information are intermediate measures. There is still a gap between the intermediate quantity and the ultimate security measure (success rate). The relationship between these probability distributions and success rate is unclear, while the relationship between mutual information and success rate is not always monotone.

Our success rate model considers two types of confusion error as impact factors. The explicit success rate formulas well indicate the architecture security, and well evaluate the effects caused by the algorithm, the system architecture and the specific implementation.

6.2 Extension to Time-Driven and Trace-Driven Attacks

Trace-driven attacks: suppose that the adversary can observe the cache activity that results from running the algorithm. The resulting traces record whether a particular cache access is a hit or miss. With n_A accesses for the wth encryption, the adversary gets a observed vector with the elements being cache hit or miss, denoting by $\boldsymbol{O}^{\mathrm{tr}}$:

$$\boldsymbol{O}^{\mathrm{tr}} = (\boldsymbol{o}_{w,1}^{\mathrm{tr}}, \cdots, \boldsymbol{o}_{w,n_A}^{\mathrm{tr}}). \tag{25}$$

and $\boldsymbol{o}_{w,i}^{\mathrm{tr}} \in \{\mathcal{H}, \mathcal{M}\}$, for $i = 1, \cdots, n_A$, with $\mathcal{H}$ and $\mathcal{M}$ denoting the cache hit and cache miss respectively. Since the development of cache architectures, the cache hits and misses can not perfectly coincide with the calculated results from algorithm any more. Similar with the analysis for the access-driven attacks, there also exist these two kinds of confusion errors: type I error, taking cache access as cache non-access, type II error, taking cache non-access as cache access. We denote the two types of error rates by p_1 and p_2 respectively.

Take $n_A = 2$ as an example, suppose the first access is cache miss and there are n_L cache lines, we have

$$\begin{aligned} \mathbb{P}(\boldsymbol{o}_{w,1} = \mathcal{M}, \boldsymbol{o}_{w,2}^{\text{tr}} = \mathcal{M}) &= \tfrac{1}{n_L} p_1 + \tfrac{n_L - 1}{n_L}(1 - p_2) \\ \mathbb{P}(\boldsymbol{o}_{w,1} = \mathcal{M}, \boldsymbol{o}_{w,2}^{\text{tr}} = \mathcal{H}) &= \tfrac{n_L - 1}{n_L} p_2 + \tfrac{1}{n_L}(1 - p_1). \end{aligned} \tag{26}$$

Based on the principal, further analysis with confusion errors can be done. The probability of obtained trace with any pattern can be calculated.

Time-driven attacks: the adversary utilizes the total execution time in a cryptographic operation which is related to the numbers of cache misses and hits. Instead of a trace for the trace-driven attack, here, the adversary can only get two numbers: the number of cache hits $\boldsymbol{N}^{\text{hi}}$ and the number of cache misses $\boldsymbol{N}^{\text{mi}}$

$$\begin{aligned} \boldsymbol{N}^{\text{hi}} &= \sum_{j=2}^{n_A} \mathbb{I}(\mathcal{H} = \boldsymbol{o}_{w,j}), \\ \boldsymbol{N}^{\text{mi}} &= \sum_{j=2}^{n_A} \mathbb{I}(\mathcal{M} = \boldsymbol{o}_{w,j}) + 1. \end{aligned} \tag{27}$$

Based on the probabilities gotten for trace-driven attack, the properties of time-driven attacks can be achieved.

6.3 Relationship to Other New Cache Attacks

In this paper, we just use last-level cache Prime+Probe attack with dis-aligned T-tables to illustrate our model. Recently some new variants of cache-drive attacks, such as Flush+Reload, CacheBleed, attracted more interest, including to applications to cloud-based environment and new cache systems. Meanwhile, misaligned T-tables are often employed to improve the first round attack. Our abstract mathematical model can be applied to analyze these attacks also.

Flush+Reload attack [35, 36] relies on the memory duplication feature, and uses *clflush* instruction to flush all cache lines that contains the target from the cache system. While the last-level Prime+Probe attack uses an eviction set and cache inclusiveness property to evict the target from the cache system. Although Flush+Reload and last-level Prime+Probe attacks differ on how to evict a target from the cache, they have the similar methodology. To launch either of these two cache-driven attacks, adversaries need to find out the access and non-access data sets based on collected timing values. Obtaining the corresponding parameters (misclassification rates, etc), our abstract model can predict the success rate of Flush+Reload attack, facilitating the security evaluation without extensive empirical assessment of success rate.

CacheBleed [37, 38] relies on the same principle with the Prime+Probe attack, which monitors the cache bank instead of the entire cache line (which contains multiple banks). It thus can detect more detailed leakage that the whole line does not provide. Tackling with cache banks and using the corresponding misclassification rates, our model can also be applied to the CacheBleed attack.

Misaligned T-tables [39–41] can improve the first round attack to recover the whole secret key (rather than just half) through randomizing the locations

of the first element of Sbox table and expanding the cache-access patterns. By considering the probability distribution of the modified cache lines, our model can be extended to cover the misaligned T-tables attacks. However, the last round attack model remain the same for misaligned T-tables, whose success rate dominates the first round attack.

6.4 Other New Techniques

Our model is based on access-driven attacks which exploit the memory access on the cache. The techniques of circumventing cache leaks, such as bitslice, AES-NI, can completely resist against the access-driven attacks, so thoroughly vitiate our model.

Bitslice implementations [42–44] convert an algorithm into a series of logical bit operations and implement different encryptions in parallel. Such implementations do not use any lookup tables whose address is dependent on secret information, thus they are inherently immune to cache-access attacks, so to our model.

Finally, Intel has announced a new AES-NI instruction set [45] that will provide dedicated hardware support for AES encryption and thus avoiding cache leaks on future CPUs. The AES-NI option makes cache attacks on AES infeasible. However, there are still a lot of platforms that do not have such option, and existing software libraries (like OpenSSL) are also based on table look-up, especially in cloud-environment. So cache-timing attacks will continue to be a threat to AES.

7 Conclusions

In this paper, we present a detailed statistical analysis for cache access-driven SCA, taking two types of errors as impact factors. We derive an explicit success rate formulas for the first time, accounting for the adversary's confusion errors between cache access and non-access. The model accuracy is confirmed through simulations and physical experimental data sets. These formulas provide clear quantification of cache architecture vulnerability to access-driven cache timing attacks. Therefore, the proposed model provides insight on important security-cognizant design parameters. By providing accurate quantitative success rate formulas, for the first time, our model enables exploration of quantitative security-performance tradeoff consideration.

8 Appendix

8.1 Proof of Property 2

Proof. We find the probability from counting the combinations of possible n_A accessed lines. The total number of possible combinations is $n_L^{n_A}$. Consider a subset $B \subseteq \boldsymbol{\mathcal{L}}$ with $|B| = b$. There are $(n_L - b)^{n_A}$ possible combinations such

that no line in B was accessed. By the inclusion-exclusion rule of combinators, the number of possible combinations with at least one element in B not being accessed is

$$\sum_{j=1}^{b}(-1)^{j-1}\binom{b}{j}(n_L-j)^{n_A}.$$

So the number of combinations with all cache lines in B being accessed is

$$n_L^{n_A}-\sum_{j=1}^{b}(-1)^{j-1}\binom{b}{j}(n_L-j)^{n_A}=\sum_{j=0}^{b}(-1)^{j}\binom{b}{j}(n_L-j)^{n_A}.$$

Since there are $\binom{n_L}{r}$ subsets with exactly r cache lines, we get the probability that exactly r cache lines not being accessed as

$$\mathbb{P}(n^{\mathrm{na}}=r)=\frac{\binom{n_L}{r}\sum\limits_{j=0}^{r}(-1)^{j}\binom{r}{j}(n_L-j)^{n_A}}{n_L^{n_A}}.$$

8.2 The Proof of Theorem 1

Attack Based on Cache Non-access: Since the procedure of execution is independent with the monitoring procedure,

$$\mathbb{P}(\boldsymbol{l}\notin\boldsymbol{S}_w^{\mathrm{ac}},\boldsymbol{l}\in\boldsymbol{S}^{\mathrm{mo}})=\mathbb{P}(\boldsymbol{l}\notin\boldsymbol{S}_w^{\mathrm{ac}})\,\mathbb{P}(\boldsymbol{l}\in\boldsymbol{S}^{\mathrm{mo}})=\mathbb{P}(\boldsymbol{l}\notin\boldsymbol{S}_w^{\mathrm{ac}})\frac{n_M}{n_L}. \tag{28}$$

For the first access, $\mathbb{P}(\boldsymbol{f}_{\mathrm{last}}(c_w,k_c)\neq\boldsymbol{o}_{w,1})=1-p_0$, and for other access $i=2,\cdots,n_A^g$, $\mathbb{P}(\boldsymbol{f}_{\mathrm{last}}(c_w,k_c)\neq\boldsymbol{o}_{w,i})=1-1/n_L$ due to the uniform distribution of $\boldsymbol{L}$. By the independence of n_A^g accesses,

$$\mathbb{P}(\boldsymbol{f}_{\mathrm{last}}(c_w,k_c)\notin\boldsymbol{S}_w^{\mathrm{ac}})=(1-p_0)(1-\frac{1}{n_L})^{n_A^g-1}. \tag{29}$$

Consider a key $k_g\neq k_c$. If $\boldsymbol{f}_{\mathrm{last}}(c_w,k_c)\neq\boldsymbol{o}_{w,1}$, then for $\boldsymbol{f}_{\mathrm{last}}(c_w,k_g)\neq\boldsymbol{o}_{w,1}$, k_g has to be one of the other n_K-d_L-1 keys not in the set $\{k\mid\boldsymbol{f}_{\mathrm{last}}(c_w,k)=\boldsymbol{o}_{w,1}\}$. Hence $\mathbb{P}(\boldsymbol{f}_{\mathrm{last}}(c_w,k_g)\neq\boldsymbol{o}_{w,1},\boldsymbol{f}_{\mathrm{last}}(c_w,k_c)\neq\boldsymbol{o}_{w,1})=(1-p_0)(n_K-d_L-1)/(n_K-1)$. Similarly $\mathbb{P}(\boldsymbol{f}_{\mathrm{last}}(c_w,k_g)\neq\boldsymbol{o}_{w,1},\boldsymbol{f}_{\mathrm{last}}(c_w,k_c)=\boldsymbol{o}_{w,1})=p_0(n_K-d_L)/(n_K-1)$. So

$$\mathbb{P}(\boldsymbol{f}_{\mathrm{last}}(c_w,k_g)\neq\boldsymbol{o}_{w,1})=(1-p_0)\frac{n_K-d_L-1}{n_K-1}+p_0\frac{n_K-d_L}{n_K-1}=1-\frac{d_L-p_0}{n_K-1}.$$

Hence for $k_{g_1}\neq k_{g_2}\in\mathcal{K}\backslash k_c$, we have the following equations:

$$\mathbb{P}(\boldsymbol{f}_{\mathrm{last}}(c_w,k_{g_1})\notin\boldsymbol{S}_w^{\mathrm{ac}})=(1-\frac{d_L-p_0}{n_K-1})(1-\frac{1}{n_L})^{n_A^g-1}, \tag{30}$$

$$\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\notin \boldsymbol{S}_w^{\text{ac}},\boldsymbol{f}_{\text{last}}(c_w,k_{g_1})\notin \boldsymbol{S}_w^{\text{ac}})=(1-p_0)\frac{n_K-d_L-1}{n_K-1}\cdot\left(\frac{\binom{n_K-d_L}{2}}{\binom{n_K}{2}}\right)^{n_A^g-1}, \tag{31}$$

$$\begin{aligned}&\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_{g_1})\notin \boldsymbol{S}_w^{\text{ac}},\boldsymbol{f}_{\text{last}}(c_w,k_{g_2})\notin \boldsymbol{S}_w^{\text{ac}})\\ =&\left(p_0\frac{\binom{n_K-d_L}{2}}{\binom{n_K-1}{2}}+(1-p_0)\frac{\binom{n_K-d_L-1}{2}}{\binom{n_K-1}{2}}\right)\cdot\left(\frac{\binom{n_K-d_L}{2}}{\binom{n_K}{2}}\right)^{n_A^g-1}.\end{aligned} \tag{32}$$

By equations (28), (29) and (30), the element of $\mu_{\text{last}}^{\text{na}}$ is

$$\begin{aligned}&\boldsymbol{\mu}_{\text{last}}^{\text{na}}(k_g)=\mathbb{E}[\boldsymbol{d}_w^{\text{na}}(k_c)-\boldsymbol{d}_w^{\text{na}}(k_g)]\\ =&[\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\notin \boldsymbol{S}_w^{\text{ac}})-\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_g)\notin \boldsymbol{S}_w^{\text{ac}})]\frac{n_M}{n_L}\\ =&\frac{d_L-p_0\cdot n_K}{n_K-1}(1-\frac{1}{n_L})^{n_A^g-1}\frac{n_M}{n_L}\end{aligned} \tag{33}$$

The diagonal element of $\boldsymbol{\Sigma}_{\text{last}}^{\text{na}}$ corresponding to $k_g\in\mathcal{K}\backslash k_c$, by equations (28), (29), (30) and (31), is

$$\begin{aligned}&\Sigma_{\text{last}}^{\text{na}}(k,k)\\ =&\mathbb{E}[\{\mathbb{I}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\notin \boldsymbol{S}_w^{\text{ac}},\boldsymbol{f}_{\text{last}}(c_w,k_c)\in \boldsymbol{S}^{\text{mo}})-\mathbb{I}(\boldsymbol{f}_{\text{last}}(c_w,k_g)\notin \boldsymbol{S}_w^{\text{ac}},\boldsymbol{f}_{\text{last}}(c_w,k_g)\in \boldsymbol{S}^{\text{mo}})\}^2]\\ &-[\boldsymbol{\mu}_{\text{last}}^{\text{na}}(k_g)]^2\\ =&\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\notin \boldsymbol{S}_w^{\text{ac}})\,\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\in \boldsymbol{S}^{\text{mo}})+\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_g)\notin \boldsymbol{S}_w^{\text{ac}})\,\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_g)\in \boldsymbol{S}^{\text{mo}})\\ &-2\,\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\notin \boldsymbol{S}_w^{\text{ac}},\boldsymbol{f}_{\text{last}}(c_w,k_g)\notin \boldsymbol{S}_w^{\text{ac}})\,\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\in \boldsymbol{S}^{\text{mo}},\boldsymbol{f}_{\text{last}}(c_w,k_g)\in \boldsymbol{S}^{\text{mo}})\\ &-\boldsymbol{\mu}_{\text{last}}^{\text{na}}(k_g)^2\\ =&(2-p_0-\frac{d_L-p_0}{n_K-1})(1-\frac{1}{n_L})^{n_A^g-1}\frac{n_M}{n_L}\\ &-2(1-p_0)\frac{n_K-d_L-1}{n_K-1}\left(\frac{(n_K-d_L)(n_K-d_L-1)}{n_K(n_K-1)}\right)^{n_A^g-1}\cdot\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\in \boldsymbol{S}^{\text{mo}},\boldsymbol{f}_{\text{last}}(c_w,k_g)\in \boldsymbol{S}^{\text{mo}})\\ &-\left(\frac{p_0\cdot n_K-d_L}{n_K-1}(1-\frac{1}{n_L})^{2n_A^g-2}\right)^2\left(\frac{n_M}{n_L}\right)^2.\end{aligned} \tag{34}$$

This is the formula for the general case, and the $\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\in \boldsymbol{S}^{\text{mo}},\boldsymbol{f}_{\text{last}}(c_w,k_g)\in \boldsymbol{S}^{\text{mo}})$ depends on the values of k_c and k_g and also the SBox property. When all $n_M=n_L$ cache lines are monitored, $\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w,k_c)\in \boldsymbol{S}^{\text{mo}},\boldsymbol{f}_{\text{last}}(c_w,k_g)\in \boldsymbol{S}^{\text{mo}})=1$, and the formula simplifies to equation (18).

The element of $\boldsymbol{\Sigma}^{\text{na}}_{\text{last}}$ corresponding to $k_{g_1}, k_{g_2} \in \mathcal{K}\backslash k_c$, by equations (28), (29), (30), (31) and (32), is

$$
\begin{aligned}
&\Sigma^{\text{na}}_{\text{last}}(k_{g_1}, k_{g_2}) \\
&= \mathbb{E}\{[\mathbb{I}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \notin \boldsymbol{S}^{\text{ac}}_w, \boldsymbol{f}_{\text{last}}(c_w, k_c) \in \boldsymbol{S}^{\text{mo}}) - \mathbb{I}(\boldsymbol{f}_{\text{last}}(c_w, k_{g_1}) \notin \boldsymbol{S}^{\text{ac}}_w, \boldsymbol{f}_{\text{last}}(c_w, k_{g_1}) \in \boldsymbol{S}^{\text{mo}})] \\
&\quad \cdot[\mathbb{I}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \notin \boldsymbol{S}^{\text{ac}}_w, \boldsymbol{f}_{\text{last}}(c_w, k_c) \in \boldsymbol{S}^{\text{mo}}) - \mathbb{I}(\boldsymbol{f}_{\text{last}}(c_w, k_{g_2}) \notin \boldsymbol{S}^{\text{ac}}_w, \boldsymbol{f}_{\text{last}}(c_w, k_{g_2}) \in \boldsymbol{S}^{\text{mo}})]\} \\
&\quad -[\mu^{\text{na}}_{\text{last}}(k_g)]^2 \\
&= \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \notin \boldsymbol{S}^{\text{ac}}_w)\, \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \in \boldsymbol{S}^{\text{mo}}) \\
&\quad - \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \notin \boldsymbol{S}^{\text{ac}}_w, \boldsymbol{f}_{\text{last}}(c_w, k_1) \notin \boldsymbol{S}^{\text{ac}}_w)\, \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \in \boldsymbol{S}^{\text{mo}}, \boldsymbol{f}_{\text{last}}(c_w, k_1) \in \boldsymbol{S}^{\text{mo}}) \\
&\quad - \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \notin \boldsymbol{S}^{\text{ac}}_w, \boldsymbol{f}_{\text{last}}(c_w, k_2) \notin \boldsymbol{S}^{\text{ac}}_w)\, \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \in \boldsymbol{S}^{\text{mo}}, \boldsymbol{f}_{\text{last}}(c_w, k_2) \in \boldsymbol{S}^{\text{mo}}) \\
&\quad + \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_1) \notin \boldsymbol{S}^{\text{ac}}_w, \boldsymbol{f}_{\text{last}}(c_w, k_2) \notin \boldsymbol{S}^{\text{ac}}_w)\, \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_1) \in \boldsymbol{S}^{\text{mo}}, \boldsymbol{f}_{\text{last}}(c_w, k_2) \in \boldsymbol{S}^{\text{mo}}) \\
&\quad -[\boldsymbol{\mu}^{\text{na}}_{\text{last}}(k_g)]^2 \\
&= (1-p_0)(1-\tfrac{1}{n_L})^{n^g_A-1}\tfrac{n_M}{n_L} \\
&\quad -(1-p_0)\tfrac{n_K-d_L-1}{n_K-1}\big[\tfrac{(n_K-d_L)(n_K-d_L-1)}{n_K(n_K-1)}\big]^{n^g_A-1} \\
&\quad \cdot[\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \in \boldsymbol{S}^{\text{mo}}, \boldsymbol{f}_{\text{last}}(c_w, k_{g_1}) \in \boldsymbol{S}^{\text{mo}}) + \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_c) \in \boldsymbol{S}^{\text{mo}}, \boldsymbol{f}_{\text{last}}(c_w, k_{g_2}) \in \boldsymbol{S}^{\text{mo}})] \\
&\quad +\tfrac{(n_K-d_L-1)(n_K-d_L-2+2p_0)}{(n_K-1)(n_K-2)} \cdot \big[\tfrac{(n_K-d_L)(n_K-d_L-1)}{n_K(n_K-1)}\big]^{n^g_A-1} \\
&\quad \cdot \mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_{g_1}) \in \boldsymbol{S}^{\text{mo}}, \boldsymbol{f}_{\text{last}}(c_w, k_{g_2}) \in \boldsymbol{S}^{\text{mo}}) \\
&\quad -\left(\tfrac{d_L-p_0\cdot n_K}{n_K-1}\right)^2 (1-\tfrac{1}{n_L})^{2n^g_A-2}\left(\tfrac{n_M}{n_L}\right)^2.
\end{aligned}
\tag{35}
$$

The off-diagonal elements, in the general case, are different from different k_{g_1} and k_{g_2} values. Similarly, when all $n_M = n_L$ cache lines are monitored, $\mathbb{P}(\boldsymbol{f}_{\text{last}}(c_w, k_{g_1}) \in \boldsymbol{S}^{\text{mo}}, \boldsymbol{f}_{\text{last}}(c_w, k_{g_2}) \in \boldsymbol{S}^{\text{mo}}) = 1$, and the formula simplifies to equation (19) for all off-diagonal elements.

References

1. P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '96, 1996, pp. 104–113.
2. J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side channel cryptanalysis of product ciphers," *J. Comput. Secur.*, vol. 8, no. 2,3, pp. 141–158, Aug. 2000.
3. Y. Tsunoo, T. Saito, T. Suzaki, and M. Shigeri, "Cryptanalysis of des implemented on computers with cache," in *Proc. of CHES 2003, Springer LNCS.* Springer-Verlag, 2003, pp. 62–76.
4. D. J. Bernstein, "Cache-timing attacks on aes," 2005.
5. G. Bertoni, V. Zaccaria, L. Breveglieri, M. Monchiero, and G. Palermo, "Aes power attack based on induced cache miss and countermeasure," in *Proceedings*

of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I - Volume 01, ser. ITCC '05, 2005, pp. 586–591.
6. C. Lauradoux, "Collision attacks on processors with cache and countermeasures." *WEWoRC*, vol. 5, pp. 76–85, 2005.
7. O. Acıiçmez and c. K. Koç, "Trace-driven cache attacks on aes (short paper)," in *Proceedings of the 8th International Conference on Information and Communications Security*, ser. ICICS'06, 2006, pp. 112–121.
8. J. Fournier and M. Tunstall, "Cache based power analysis attacks on aes," in *Information Security and Privacy*. Springer, 2006, pp. 17–28.
9. X.-J. Zhao and T. Wang, "Improved cache trace attack on aes and clefia by considering cache miss and s-box misalignment." *IACR Cryptology ePrint Archive*, vol. 2010, p. 56, 2010.
10. J.-F. Gallais, I. Kizhvatov, and M. Tunstall, "Improved trace-driven cache-collision attacks against embedded aes implementations," in *Information Security Applications*. Springer, 2010, pp. 243–257.
11. E. Savaş and C. Yılmaz, "A generic method for the analysis of a class of cache attacks: A case study for aes," *The Computer Journal*, p. bxv027, 2015.
12. J. Bonneau and I. Mironov, "Cache-collision timing attacks against aes," in *Cryptographic Hardware and Embedded Systems-CHES 2006*. Springer, 2006, pp. 201–215.
13. O. Acıiçmez, W. Schindler, and Ç. K. Koç, "Cache based remote timing attack on the aes," in *Topics in Cryptology–CT-RSA 2007*. Springer, 2007, pp. 271–286.
14. C. Percival, "Cache missing for fun and profit," *BSDCan 2005*, 2005.
15. D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: The case of aes," in *Proceedings of the 2006 The Cryptographers' Track at the RSA Conference on Topics in Cryptology*, ser. CT-RSA'06, 2006, pp. 1–20.
16. M. Neve and J.-P. Seifert, "Advances on access-driven cache attacks on aes," in *Proceedings of the 13th International Conference on Selected Areas in Cryptography*, ser. SAC'06, 2007, pp. 147–162.
17. E. Tromer, D. A. Osvik, and A. Shamir, "Efficient cache attacks on aes, and countermeasures," *J. Cryptol.*, vol. 23, no. 2, pp. 37–71, Jan. 2010.
18. Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in *Proc. ACM Conf. on Computer & Communications Security*, 2012, pp. 305–316.
19. G. I. Apecechea, M. S. Inci, T. Eisenbarth, and B. Sunar, "Wait a minute! A fast, Cross-VM attack on AES," in *Research in Attacks, Intrusions and Defenses (RAID)*, 2014, pp. 299–319.
20. ——, "Fine grain Cross-VM Attacks on Xen and VMware are possible!" in *Int. Symp. on Privacy & Security in Cloud & Big Data*, Dec. 2014.
21. G. Irazoqui, T. Eisenbarth, and B. Sunar, "S$A: A shared cache attack that works across cores and defies VM sandboxing - and its application to AES," in *IEEE Int. Symp. on Security & Privacy*, 2015, pp. 591–604.
22. F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *IEEE Symposium on Security and Privacy*, 2015, pp. 605–622.
23. Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07, 2007, pp. 494–505.
24. ——, "A novel cache architecture with enhanced performance and security," in *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*. IEEE, 2008, pp. 83–93.

25. L. Domnitser, A. Jaleel, J. Loew, N. Abu-Ghazaleh, and D. Ponomarev, "Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, no. 4, p. 35, 2012.
26. D. Page, "Theoretical use of cache memory as a cryptanalytic side-channel," Department of Computer Science, University of Bristol, Tech. Rep. CSTR-02-003, June 2002.
27. ——, "Defending against cache-based side-channel attacks," *Information Security Technical Report*, vol. 8, no. 1, pp. 30 – 44, 2003.
28. K. Tiri, O. Aciiçmez, M. Neve, and F. Andersen, "An analytical model for time-driven cache attacks," in *Proceedings of the 14th International Conference on Fast Software Encryption*, ser. FSE'07, 2007, pp. 399–413.
29. Z. Wang and R. B. Lee, "Covert and side channels due to processor architecture," in *null*. IEEE, 2006, pp. 473–482.
30. L. Domnitser, N. Abu-Ghazaleh, and D. Ponomarev, "A predictive model for cache-based side channels in multicore and multithreaded microprocessors," in *Computer Network Security*. Springer, 2010, pp. 70–85.
31. T. Zhang and R. B. Lee, "New models of cache architectures characterizing information leakage from cache side channels," in *Proceedings of the 30th Annual Computer Security Applications Conference*, 2014, pp. 96–105.
32. C. Rebeiro and D. Mukhopadhyay, "A formal analysis of prefetching in profiled cache-timing attacks on block ciphers," Cryptology ePrint Archive, Report 2015/1191, 2015.
33. V. Lomné, E. Prouff, M. Rivain, T. Roche, and A. Thillard, "How to estimate the success rate of higher-order side-channel attacks," in *Proceedings of the 16th International Workshop on Cryptographic Hardware and Embedded Systems — CHES 2014 - Volume 8731*, 2014, pp. 35–54.
34. Y. Fei, A. A. Ding, J. Lao, and L. Zhang, "A statistics-based success rate model for dpa and cpa," *Journal of Cryptographic Engineering*, vol. 5, no. 4, pp. 227–243, 2015.
35. Y. Yarom and K. Falkner, "Flush+ reload: a high resolution, low noise, l3 cache side-channel attack," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 719–732.
36. B. Gülmezoğlu, M. S. Inci, G. Irazoqui, T. Eisenbarth, and B. Sunar, "A faster and more realistic flush+ reload attack on aes," in *Constructive Side-Channel Analysis and Secure Design*. Springer, 2015, pp. 111–126.
37. Y. Yarom, D. Genkin, and N. Heninger, "Cachebleed: A timing attack on openssl constant time rsa," Cryptology ePrint Archive, Report 2016/224, Tech. Rep., 2016.
38. G. Doychev and B. Köpf, "Rigorous analysis of software countermeasures against cache attacks," *arXiv preprint arXiv:1603.02187*, 2016.
39. Z. Xinjie, W. Tao, M. Dong, Z. Yuanyuan, and L. Zhaoyang, "Robust first two rounds access driven cache timing attack on aes," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 3. IEEE, 2008, pp. 785–788.
40. J. Takahashi, T. Fukunaga, K. Aoki, and H. Fuji, "Highly accurate key extraction method for access-driven cache attacks using correlation coefficient," in *Australasian Conference on Information Security and Privacy*. Springer, 2013, pp. 286–301.
41. R. Spreitzer and T. Plos, "Cache-access pattern attack on disaligned aes t-tables," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2013, pp. 200–214.

42. C. Rebeiro, D. Selvakumar, and A. Devi, "Bitslice implementation of aes," in *International Conference on Cryptology and Network Security*. Springer, 2006, pp. 203–212.
43. M. Matsui and J. Nakajima, "On the power of bitslice implementation on intel core2 processor," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 121–134.
44. E. Käsper and P. Schwabe, "Faster and timing-attack resistant aes-gcm," in *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 1–17.
45. S. Gueron, "Advanced encryption standard (aes) instructions set," *Intel, http://softwarecommunity. intel. com/articles/eng/3788. htm, accessed*, vol. 25, 2008.