CaSA: End-to-end Quantitative Security Analysis of Randomly Mapped Caches

by

Jules Drean

B.S., Télécom Paris (2018)
S.M., Télécom Paris (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2020

Author . . . . . .
Department of Electrical Engineering and Computer Science
August 28, 2020

Certified by . . . . . .
Mengjia Yan
Assistant Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

CaSA: End-to-end Quantitative Security Analysis of Randomly Mapped Caches

by

Jules Drean

Submitted to the Department of Electrical Engineering and Computer Science
on August 28, 2020, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

It is well known that there are micro-architectural vulnerabilities that enable an attacker to use caches to exfiltrate secrets from a victim. These vulnerabilities exploit the fact that the attacker can detect cache lines that were accessed by the victim. Therefore, architects have looked at different forms of randomization to thwart the attacker's ability to communicate using the cache. The security analysis of those *randomly mapped caches* is based upon the increased difficulty for the attacker to determine the addresses that touch the same cache line that the victim has accessed.

In this paper, we show that the analyses used to evaluate those schemes were incomplete in various ways. For example, they were incomplete in only looking at one communication step, which is the step that the attacker uses to determine the set of addresses that can monitor the cache lines used by the transmitter address. Indeed, we generalize micro-architecture side channels to obtain the overall view of the communication process and identify that there exist other communication steps that can also affect the security of randomly mapped caches, but have been ignored by prior work.

We design an analysis framework, CaSA, to comprehensively and quantitatively analyze the security of these randomly mapped caches. We comprehensively consider the end-to-end communication steps and study the statistical relationship between different steps. In addition, to perform quantitative analysis, we leverage the concepts from the field of telecommunication to formulate the security analysis into a statistical problem. We use CaSA to evaluate a wide range of attack strategies and cache configurations. Our result shows that the randomization mechanisms used in the state-of-the-art randomly mapped caches are insecure.

Thesis Supervisor: Mengjia Yan
Title: Assistant Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to thank Professor Menjgia Yan for supervising me in the making of this thesis.

I would like to thank my other co-authors for this work that we integrally did together : Thomas Bourgeat, Yuheng Yang, Lillian Tsai and Professor Joel Emer.

I would also like to thank my research advisor Professor Srini Devadas for his support and his guidance in my PhD journey.

Finally, I have a thoughts for all my colleagues, my friends, my partner and my family who always support me and help me to fulfill myself.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The class of attacks that exploit micro-architectural vulnerabilities to breach processor security, generally referred to as *side-channel attacks*, have become a serious security threat. Using these attacks, an attacker can steal secrets from a victim application running on the same machine by monitoring the side effects of the victim's actions on various micro-architecture states. Such attacks are effective and have been used to leak encryption keys [1, 2]. Many of these attacks employ speculative execution to modify cache states [3, 4, 5] to completely bypass memory isolation and leak arbitrary data.

As described in [6], there is a series of elements common to most attacks that exploit micro-architectural vulnerabilities. These include either pre-existing or attacker generated code run in the victim's security domain that 1) accesses secret information and 2) transmits that information over a communication channel that 3) is received by an attacker. The signal received by the receiver leaks a secret that was supposed to be secured within the victim's security domain.

Focusing just on the communication phase of an attack, the *transmitter* is in the victim's code, and the *receiver* is in the attacker's code. The medium of the *communication channel* is the micro-architectural state that can be modified, i.e., *modulated*, by the activity of the transmitter. A communication channel may actually be composed of multiple *subchannels*, just as a radio transmission may use multiple frequencies.

For numerous contemporary attacks, the communication medium is the last-level cache,

and each cache line can be considered a communication subchannel. In the simple case of a directly mapped cache, modulating a subchannel involves the transmitter accessing a specific address, since that will change the state of exactly one well-defined cache line. A receiver can monitor the state of the same cache line (subchannel) for changes (modulation) by accessing an address to occupy that cache line (subchannel) and later time whether the re-access to the same address is a hit or miss. In general, the transmitters and receivers in an attack communicate via multiple subchannels, i.e. using several addresses. The addresses accessed by the transmitter and receiver are referred to as the *transmitter set* and the *receiver set* respectively.

Randomly Mapped Caches. Among various architectural solutions that address security vulnerabilities by disrupting communication via cache-based channels, randomly mapped caches [7, 8, 9, 10] are considered highly effective with plausible security properties and low performance overhead. Randomly mapped caches aim to significantly increase an attacker's efforts to find an effective receiver set. They leverage one of the two ideas: make cache behave non-deterministic by obfuscating the functions used to map memory addresses to cache lines (subchannels) [9, 8], and dynamically change these functions [7, 8, 10].

In such complex caches, the subchannels that the transmitter will modulate are not publicly known to attackers. Moreover, with non-deterministic caches, the attacker can only guess the probability of an address to be mapped to a given cache-line, and modulation can only be observed probabilistically. This uncertainty requires the attacker to use complex methods to generate receiver addresses that have a high probability to monitor to the same cache lines as the transmitter addresses.

Unfortunately, the security claims of randomly mapped caches are quite fragile. For example, a recent secure cache design, CEASER [7], which claimed to be able to tolerate years of attack, has been broken by more advanced eviction set construction algorithms [8, 11]. Similarly, another recently proposed randomly mapped cache design, ScatterCache [9], can be broken by a new eviction set construction algorithm [12] within a few seconds.

The reason behind the failures of those designs lies in their *limited security analyses*. In fact, those defense mechanisms were designed to block very specific eviction set construction techniques. For instance, some weak security analyses [9, 8, 7, 12] only consider the case

where the attacker tries to obtain a receiver set that monitor the subchannels used by the transmitter with high probability. Such an analysis ignores the existence of alternative strategies where the attacker could spend a modest amount of resources on constructing a receiver set with a lower probability to monitor these subchannels. With such a weak receiver set, she would rely on repeatedly monitoring the modulations from the transmitter to ultimately leak the secret.

Communication Paradigm. In this paper, we introduce a generalized communication paradigm for micro-architecture side channels. The paradigm serves two purposes. First, it provides the overall view of the communication process and identifies the end-to-end communication steps a comprehensive security analysis has to consider. Second, it enables us to think of micro-architecture side-channel attacks using concepts from the filed of telecommunication, so we can formulate the security analysis into a statistical problem and perform quantitative analyses.

Figure 1-1: Communication paradigm.

The communication paradigm, shown in Figure 1-1, consists of three steps: *calibration*, *signaling* and *decode*.

First, the receiver often needs to perform a *calibration* step. Calibration is like a tuning process in a radio-based system, and aims to determine which subchannels will be modulated by the transmitter, and where to tune the receiver to monitor those subchannels. For a cache-based channel, the calibration step involves running an eviction set construction algorithm [11]. Prior analyses [12, 13, 8, 7, 9] have only focused on this step.

Second, the receiver performs a *signal transfer* step (*signaling* for short) to obtain the signal from the transmitter. The signal is embeded in the channel state. To obtain the signal, the receiver needs to detect the state changes of the channel caused by the transmitter. In cache-based side channels, the receiver can obtain the signal using various approaches, such as Prime+Probe [2]. In telecommunication, the signal is generally described using some

mathematical representation.

Finally, the receiver needs to perform a decode step to interpret the detected signals. The decode step can be straightforward if the detected signal directly corresponds to the secret value. In cache-based channels, this decode step can be complicated if it needs to cope with noise, and non-deterministic behaviors of the cache. In telecommunication, it is common to use sampling to increase decode success rate.

This Paper. We propose **Ca**che **S**ecurity **A**nalyzer (CaSA) to quantitatively analyze the security of randomly mapped caches. We aim to use CaSA to comprehensively evaluate a wide range of communication strategies and cache configurations.

The design of CaSA consists of three key novelties. First, instead of solely focusing on the calibration step, CaSA performs an end-to-end analysis on the three communication steps in Figure 1-1. Second, it leverages the telecommunication concept to formulate the security analysis into a statistical problem and quantify the security by measuring the end-to-end communication cost. Third, CaSA identifies the existence of a trade-off in distributing resources between the calibration step and the signaling step. It explores that trade-off to find the communication strategy that minimizes the overall communication cost.

We use CaSA to evaluate randomly mapped caches of different configurations and discover the *quantitative* impacts of different parameters on the communication cost (Findings 1-4 in Chapter 7). Furthermore, we learn new insights on the limitations and benefits of randomly mapped caches that refute several common beliefs. We highlight two insights here:

1. When communicating on randomly mapped caches, spending the maximum amount of resources on calibration is neither the only nor always the best strategy. This insight refutes the common belief [9, 12] that an effective receiver set must be able to achieve a high eviction rate.

2. In the case where dynamic changes in mapping functions is used, information can be leaked and accumulated across mapping function changes. This insight refutes the common belief that attacks must be completed during the life of a single mapping function [7, 8].

With those insights and quantitative results, we show that the randomization mechanisms used in the state-of-the-art randomly mapped caches [7, 8, 9] (except for NewCache [10]) are insecure.

The contributions of this paper are:

- A communication paradigm that generalizes cache-based side channels and identifies the end-to-end communication steps.
- Formulating the security analysis into a statistical problem to enable quantitative analysis.
- CaSA, an end-to-end quantitative security analysis framework of the communication procedures of side-channel attacks on randomly mapped caches.
- A thorough security evaluation and new insights to understand the benefits and limitations of randomly mapped caches.

Chapter 2

Background

2.1 Cache-based Side Channel Attacks

In a cache-based side channel attack, the transmitter and the receiver use the cache as the communication channel, and each cache line as a subchannel. Various such attacks exist [14, 15, 1, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26], and follow the procedure described in Figure 1-1.

In each attack, the receiver first performs a *calibration* step by finding a group of addresses called *receiver addresses*. The receiver uses the receiver addresses to monitor a set of subchannels, usually a cache set.

The receiver then performs the signaling step, which consists of two substeps: *precondition* and *detection*. The receiver *preconditions* a group of subchannels into a known state in order to optimize its chances of monitoring state changes in these subchannels by accessing the receiver addresses to fill the cache set. It waits for the transmitter to modulate some of the monitored subchannels by accessing some cache lines, and then *detects* the modulation of those subchannels by either measuring the time of re-accessing the receiver addresses (Prime+Probe [1, 16]), or measuring the time of accessing the transmitter addresses (Evict+Reload [15]), or measuring the execution time of the transmitter (Evict+Time).

Figure 2-1 visualizes an example of a Prime+Probe attack on a two-way cache. It corresponds to the signaling step in our communication paradigm, which in this example,

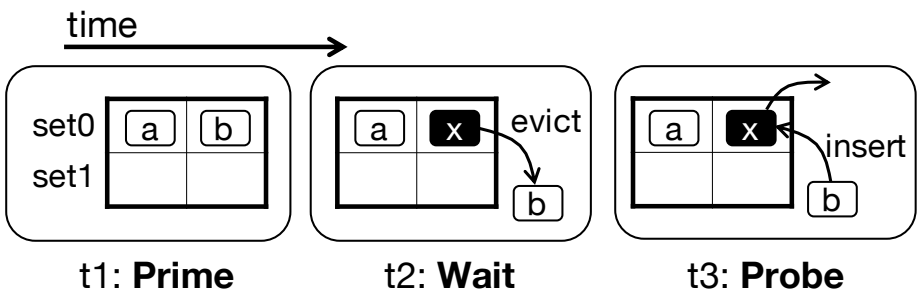


Figure 2-1: An example of Prime+Probe attacks.

Line a and b are receiver addresses; line x is the transmitter address.

contains three steps: Prime, Wait, and Probe. The receiver preconditions two subchannels in set0 by accessing lines a and b (Prime). It then waits for the transmitter to modulate a subchannel in set0 by accessing line x, which evicts line b from that subchannel (Wait). At a later time, the receiver checks the state of the subchannels in set0 by re-accessing lines a and b, and measuring the access latency (Probe). Based on the long access latency, the receiver knows that line b missed in the cache and the state of a subchannel in set0 has been modified (modulated) by the transmitter.

2.2 Randomly Mapped Cache Designs

The mapping function in a cache decides how memory addresses are mapped to cache sets. Randomly mapped caches obfuscate the mapping functions to make it harder for a receiver to know which subchannels will be modulated by a transmitter, and which subchannels are preconditioned or monitored by the receiver. It aims to mitigate cache attacks by significantly increasing the difficulty of the receiver's calibration step.

There are various flavors of randomly mapped cache designs [7, 8, 9, 10], each with different performance and security characteristics. To better understand their differences, we distinguish these designs based on three characteristics of the mapping function, namely whether:

1) It uses public or secret hash functions;

2) It is static or can be dynamically changed over time;

3) It uses a single or multiple hash functions at a point in time.

Table 2.1 categorizes each design by mapping strategy.

	Static	**Dynamic**
Single Hash Group	Set-associative cache* Intel sliced LLC [27]	CEASER [7] NewCache [10]
Multiple Hash Groups	ScatterCache [9]	Skewed-CEASER [8]

Table 2.1: Classification of cache mapping strategies.

* Uses public hash functions.

1) Public vs. Secret hash functions. Traditional set-associative caches use a public hash function, which simply extracts bits from the physical address and uses them as the set index. The other caches in Table 2.1 use secret hash functions. For example, the last-level caches in Intel processors are organized into multiple slices. The mapping function includes an undocumented slice hash function to decide which slice an address should map to. NewCache [10] uses a table-based hash function, while CEASER [7] and ScatterCache [9] use encryption-based hash functions. Even though using a secret mapping function could be thought to make calibration more difficult, it alone cannot thwart cache attacks. It has been demonstrated that there exist efficient algorithms for attackers to reverse engineer the hash function [28, 29] or even to directly construct effective receiver sets [11, 8] without needing to know anything about the mapping function.

2) Static v.s. Dynamic hash functions. To further secure the cache, researchers proposed to periodically change the hash function instead of using a static hash function. A cache with a dynamic mapping function uses one hash function in each *epoch*, and switches to a different hash function at the end of an epoch.

The length of an epoch has significant impact on the performance and security of the design. To be secure, the epoch should be short enough so that the receiver cannot both calibrate and detect signals within one epoch. However, upon epoch switching, every line in the cache has to be remapped, and using small epochs thus incurs serious performance overhead.

NewCache [10] uses extremely small epochs—changing the hash function every time

a cache conflict occurs. CEASER [7] and Skewed-CEASER [8] change the hash function when the number of cache accesses reaches a threshold. The threshold is configured to be smaller than the number of accesses required by the state-of-the-art calibration algorithm based on constructing eviction sets [11]. Skewed-CEASER [8] claims years of security when setting the threshold as $100 \times \mathsf{L}$, where L is the total number of lines in the cache.

3) Single vs. Multiple hash functions. Researchers have proposed more advanced secure cache designs, namely *multi-hash caches* such as ScatterCache [9] and Skewed-CEASER [8], which use multiple hash functions at any point in time. These designs contrast with *single-hash caches*, which only ever use a single hash function.

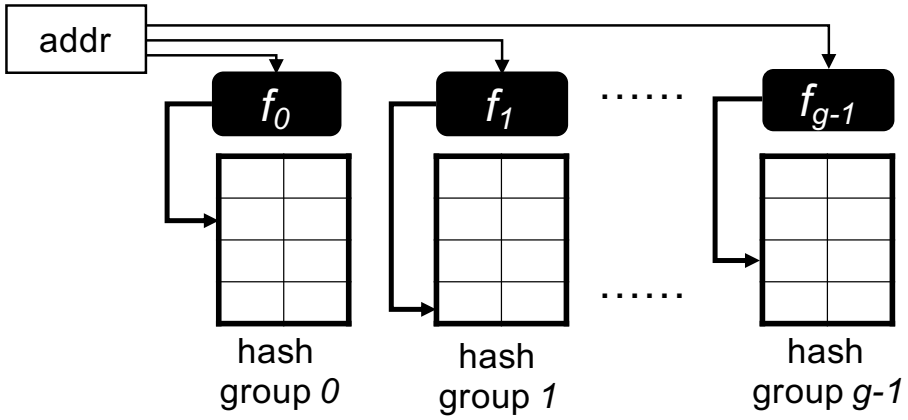


Figure 2-3: A cache with multiple hash groups.

As shown in Figure 2-3, a multi-hash cache is organized as multiple hash groups. Each hash group is organized as a normal set-associative cache, and uses a distinct hash function. To lookup the cache, all the hash-group are looked up, with at most one of them being a cache hit. On a cache miss, the cache first picks one of the hash groups using a uniformly random policy and then uses the corresponding hash function to generate the set index for that hash-group.

As a result, the mapping function becomes *non-deterministic*, since an address can end up in different hash groups even within the same epoch.

ScatterCache [9] uses a single-way per hash group design. Skewed-CEASER [8] makes the number of ways per hash group a configurable parameter.

Chapter 3

Threat Model and Scope

We follow the standard thread model of cross-core cache-based side channel attacks. We assume the attacker and the victim are co-located on the same processor chip, but reside on different cores. A transmitter embedded in the victim and a receiver controlled by the attacker communicate via channels in a shared last-level cache. Even though we focus on the last-level cache, our analysis and our tool, CaSA, can be easily extended to other levels of cache.

The attacker can reside in a user-level process or in a malicious operating system in a secure enclave context, such as SGX [30]. Like previous work [21], we assume the receiver can use a single thread or multiple threads to control multiple cores on the chip. The transmitter may be latent in the code of the victim and execute as part of the victim's normal processing, or the attacker can leverage speculative execution [3] to provoke the execution of the transmitter.

Scope. Our analysis focuses on investigating the fundamental problems in the randomization schemes used by randomly mapped caches. Prior work [31] has shown that the mapping function used in CEASER [7] and Skewed-CEASER [8] only consists of linear functions and has a key invariant vulnerability, that is, changing the key used in the mapping function cannot change the collisions between addresses. This vulnerability can be fixed using non-linear hash functions. Note that, our analysis is independent from what hash functions are used and studies new vulnerabilities that have not been explored in prior work [31].

Indeed, we focus on analyzing the fundamental problem that is intrinsic to randomly mapped caches.

Besides, we consider the analysis of the following two types of attacks orthogonal to the analysis of randomly mapped caches: flush-based attacks [14, 19] and occupancy attacks [32]. The reason is that randomly mapped caches are not designed to and thus are unable to mitigate these attacks.

Chapter 4

Motivation

Correctly reasoning about the security of randomly mapped caches is challenging. Prior security analysis have made incorrect security claims by narrowly considering two communication strategies by the attacker. We claim that correct security analysis should provide an end-to-end quantitative analysis of a broad range of attack scenarios.

4.1 Limitations of Prior Work

Prior security analysis only targeted specific calibration strategies that require a huge amount of resources and are unlikely to be completed within one epoch, and have led to incorrect security claims. We use the following simple example in Figure 4-1 to illustrate the intuitions of their analysis and clearly point out their limitations. Figure 4-1 compares the results of three different calibration strategies on a cache with 2 hash groups and 1 way in each group. Each figure shows the hash group 0 on top, hash group 1 at the bottom, and how the transmitter and receiver addresses are mapped to corresponding subchannels. The subchannels that can be used by the transmitter address are marked in grey.

The security analysis in Skewed-CEASER [8] only considered using "hard-conflict" receiver addresses for signaling. A "hard-conflict" receiver address maps to the same cache set as the transmitter address in *every* hash group, shown in Figure 4-1(a). The reason to only consider such addresses is that once the attacker has enough hard-conflict addresses (2

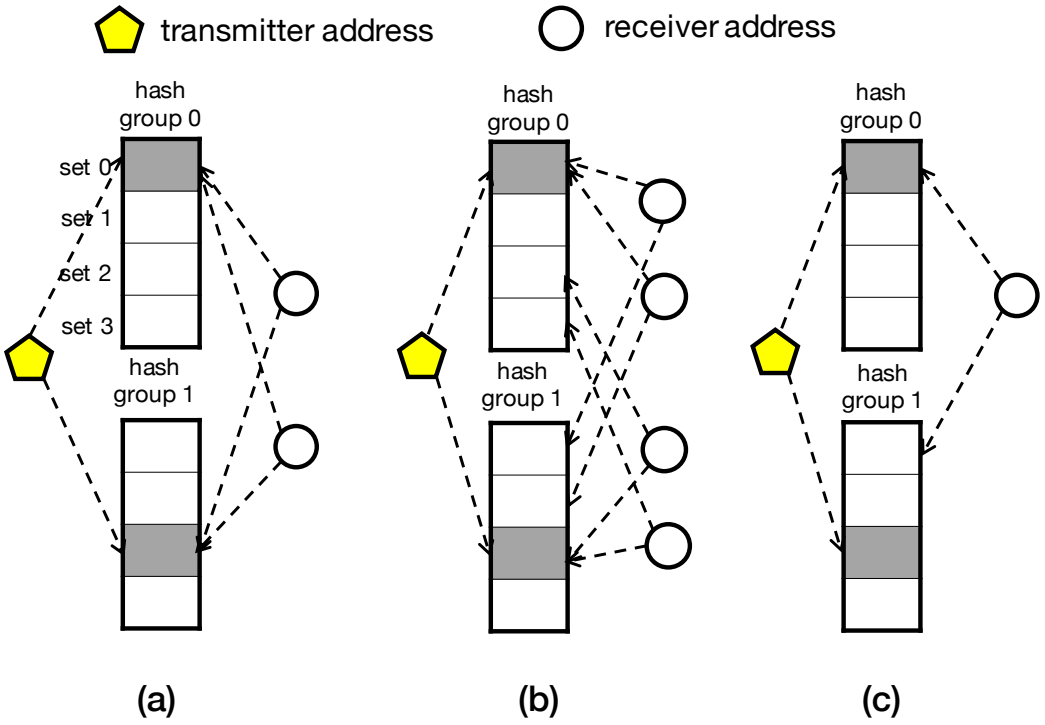


Figure 4-1: An illustrative example of different calibration strategies.

(a) hard-conflict receiver addresses; (b) many soft-conflict receiver addresses; (c) one soft-conflict receiver address.

in this example), she can perform the rest of the communication (e.g., Prime+Probe) in the same way as on a conventional cache. Randomly mapped caches are designed to make it extremely difficult to obtain hard-conflict addresses. In fact, for a given transmitter address, when there are 8 or 16 hash groups, there may not exist enough hard-conflict addresses given the limited size of the address space in the state-of-the-art systems [9]. But while it is correct that "hard-conflict" are required to build eviction sets that are guaranteed to be functional, if we are ready to tolerate imperfect receivers - that is eviction set that sometimes fail to evict - we can make use of the much more common "soft-conflict" addresses. Since the analysis only considered hard-conflict addresses, they made the incorrect security claim that these caches can tolerate years of attacks.

The security analysis in ScatterCache [9] considered using a high number of "soft-conflict" receiver addresses. A "soft-conflict" receiver address maps to the same set as the transmitter address in *at least one* hash group, shown in Figure 4-1(b), and can only be used

to monitor the transmitter with some probability. The assumption behind their analysis is that the attacker needs to get a large receiver set (e.g., 256 addresses on an 8MB LLC) in order to monitor the transmitter with 99% probability. Crafting such a large receiver set is expensive and unlikely to be completed within one epoch. Consequently, strong security claims were made under such assumptions.

There exists a key problem with these analysis: they overlooked a broad range of communication strategies that are available to the attacker. In addition to the prior analysis where the receiver spends a huge amount of resources on calibration to achieve a high monitoring probability, other effective communication strategies are also possible, such as, using a small amount of resources on calibration to obtain a receiver set with low monitoring probability, and relying on repeating signaling transfer to decode secrets with a high success rate. A comprehensive analysis should explore the trade-off in distributing resources between calibration and signaling.

Figure 4-1(c) shows an example of using 1 receiver address that soft-conflicts with the transmitter on a single subchannel. Such a receiver set is fairly cheap to construct. In this example, the receiver has a probability of 0.5 to monitor that subchannel and the transmitter also has a probability of 0.5 to modulate that subchannel. As a result, when the transmitter address is accessed, the probability of the receiver observing a modulation is only $0.5 \times 0.5 = 0.25$. When the transmitter is not accessed, this probability is 0. Even though the probability to observe a modulation is low, the receiver can repeat the signal transfer to accumulate samples. Those samples are then used to infer if the transmitter was accessed (observing some modulation) or not (observing no modulation). This last phase is the decoding step and increasing the number of samples will increase the decode success rate.

In our example, by accumulating 16 samples, an attacker can know if the transmitter was accessed or not with 99% confidence, based on whether it detects modulations in at least one of the signaling samples or it detects no modulations at all across all samples. Alternatively, the attacker could spend more resources on calibration to obtain two receiver addresses instead of one. In this situation, she would only need to accumulate 7 samples to decode the secret with the same level of confidence. The examples above clearly demonstrate the

existence of a trade-off in distributing resources between calibration and signaling.

4.2 The Need for End-to-end Quantitative Analysis

In addition to the trade-off between calibration and signal transfer, we find it is necessary to perform an end-to-end quantitative analysis of randomly mapped caches as there exists multiple other factors that can affect the security of these designs. We provide the intuitions of how these factors can impact the attack efficiency as below.

First, we need to consider the effects of having multiple transmitter addresses. Intuitively, having more transmitter addresses can make communication easier, because the number of subchannels associated with the transmitter increases and the communication can work as long as the receiver can successfully monitor at least one modulation from the transmitter. Note that, in practice, multi-address transmitters do occur in many security-sensitive applications. For example, the square-and-multiply exponentiation algorithm used in RSA encryption [33] acts as a multi-address transmitter: both the `square` and `multiply` functions are composed of instructions residing in multiple cache lines.

Secondly, we need to consider the effects of noise. Intuitively, the presence of noise can make attacks more difficult, because the receiver often cannot distinguish the modulations generated by the transmitter or by the noise. CaSA quantitatively measures the impacts of noise and we discovered a new finding : that noise can have a positive impact on communication.

Finally, for caches that periodically change the mapping functions, we investigate the feasibility of performing the communication across epochs. Prior work assumed that communication must complete within one epoch and no information can be carried across epochs. In this paper, we challenge this assumption. It is true that, a receiver set constructed in an epoch can only be used for the signaling steps in the same epoch. However, we observe that different receiver sets from different epochs generate signals that are similar to each other, since the signals are mainly determined by the numbers of addresses in the receiver sets. Intuitively, if the same secret bit is transmitted, the samples obtained from different epochs can be combined to increase decoding accuracy.

CaSA is designed to quantitatively analyze the impacts of the above factors on the security of randomly mapped caches. Specifically, CaSA can answer the following questions.

- Given a cache configuration, such as the one in Figure 4-1, and the number of transmitter addresses, how should an attacker distribute resources between calibration and signal transfer to exfiltrate the maximum amount of information?
- Considering background noise, how much more difficult is it for an attacker to mount a successful attack?
- Among different cache configurations (e.g., 1-way per hash group and 2-way per hash group), which one is more difficult to attack, measured by the number of attacker's cache accesses to leak one secret bit?

Chapter 5

CaSA Overview

The goal of CaSA is to measure the security of different configurations of randomly mapped caches. We strive to comprehensively evaluate how various communication parameters quantitatively affect the amount of information leakage on a given cache configuration. To enable quantitative analysis, we innovatively leverage concepts from the field of telecommunication (Chapter 1) and formulate the signals in cache-based side channels into a statistical representation. In this chapter, we first describe the full security analysis space, and then describe the statistical representation of signals, followed by the security metric used in CaSA.

5.1 The Security Analysis Space

The paper strives to comprehensively evaluate the choices available with respect to the three components used in cache-based side channel communication, i.e., transmitter, receiver and channel (i.e., cache), as well as parameters related to noise.

Transmitter. An important parameter related to the transmitter is the number of transmitter addresses. We expect program developers to set that sole transmitter parameter based on their knowledge of the applications or using program analysis tools [34, 35, 36].

Receiver. The receiver can choose from a wide range of calibration, signaling and decoding strategies, and it can accumulate information *across epochs* on caches periodically changing

their hash functions. We investigate various possible combinations of calibration, signaling and decoding strategies, especially considering the case that the receiver spends medium to low resources on calibration.

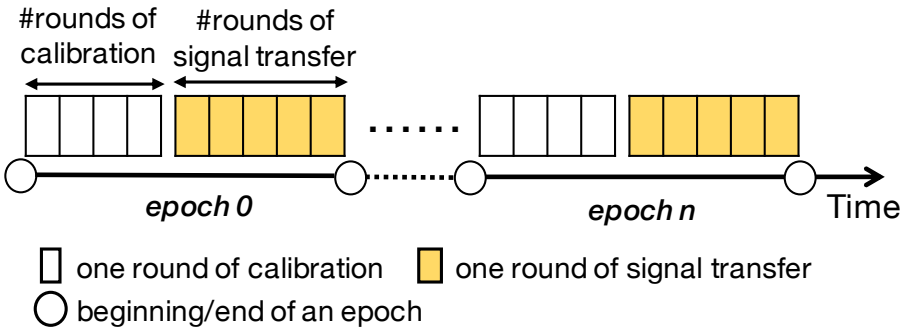


Figure 5-1: Attack procedure on a multi-hash cache that periodically changes hash functions.

Figure 5-1 visualizes the communication process on a multi-hash cache that periodically changes hash functions and indicates receiver parameters. The cache changes the hash functions at the end of each epoch (marked as circles). Within each epoch, the receiver generates a receiver set via multiple rounds of calibration (white boxes), and then uses the receiver set to perform signal transfer and collect signal samples once or multiple times before the epoch ends (highlighted boxes). The receiver strategy decides how to distribute efforts between calibration and signal transfer

If enough samples have been acquired within one epoch to allow decoding of the secret value with sufficient certainty, the communication of the secret is complete. If the number of samples acquired is insufficient to decode the secret, the attacker will start a new epoch with a new calibration step, acquiring more samples.

Our analysis assumes that the epoch size is constant and epoch changes are public to the receiver. It is useful to study the security of randomly mapped caches independently of complicated epoch parameters, especially as the security of those caches is not believed to rely on hiding epoch parameters. We discuss how CaSA can be extended to analyze detecting epoch changes and handling variable-size epoch in Chapter 8.

Channel (Cache). We consider randomly mapped caches with varied configurations, in terms of the number of hash groups, the number of ways in each hash group, the number of cache sets, and the epoch length. To make our analysis tractable, in this paper, we do

not consider other cache parameters such as the ones related to number of cache levels, directories, or MSHRs. However, note that CaSA can be extended to incorporate those parameters.

Noise. In a cache attack, noise can add spurious modulations and confuse the receiver. We identify and consider two types of noise. The first type is the background noise, which consists of random addresses and modulates random subchannels.

The second type is the *carrier* noise, which modulates a fixed set of subchannels independently from the secret. Taking the following victim code for example, `if (secret) {access A; access B;} else {access A;}`. Address `A` is a carrier address. If the receiver is calibrated on the subchannels mapped to A, it will waste resources monitoring subchannels that will not provide any useful information about the secret. As a result, carrier noise makes communication more difficult.

A summary of the parameters of all the communication components considered in this paper is shown in Table 5.1.

Component	Parameters
transmitter	number of transmitter addresses
receiver	number of rounds of calibration in one epoch number of rounds of signal transfer in one epoch
channel (cache)	number of hash groups number of ways per group number of sets epoch length size of upper-level caches
noise	background noise carrier noise

Table 5.1: The communication parameters considered in CaSA.

5.2 The Statistical Representation of Signals

We model the signal observed by a receiver as a random variable X, counting the number of modulations detected by the receiver during a signal transfer step. X follows a probability distribution that can be characterized by a *probability density function* (PDF for short)

$f(n) = P(X = n)$. We also note $F(n) = P(X \geq n)$ the cumulative density function (CDF for short), sometimes more convenient to use.

To give a concrete example, let's consider a transmitter that communicates a secret bit to a receiver by modulating one subchannel to send bit "1" and doing nothing to send bit "0". We use $f_0(n)$ and $f_1(n)$ to represent the density functions for X when the bit sent is "0" or "1" respectively. To decode the signal, the receiver samples X to determine whether X follows f_0 or f_1. We give one example of the density function. Note that, one of the key tasks of CaSA is to compute the PDFs or CDFs for a given communication configuration.

When communicating on a multi-hash cache, if the receiver uses soft-conflict addresses, such as in Figure 4-1(b) and (c), she will be only able to monitor the subchannels used by the transmitter with a certain probability. The corresponding PDFs are as below, with $p > 0$.

$$f_0(n) = \begin{cases} 1, & \text{if } n = 0 \\ 0, & \text{otherwise} \end{cases} \quad ; \quad f_1(n) = \begin{cases} p, & \text{if } n = 0 \\ 1-p, & \text{if } n = 1 \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

In a noiseless environment, both $f_0(n)$ and $f_1(n)$ have non-zero values at $n = 0$. Visually, the two functions partially overlap with each other. In the examples in Figure 4-1, when using one soft-conflict receiver address, $p = 0.25$, and when using a large number of soft-conflict receiver addresses, the value of p can decrease to 0.01 (i.e., $f_1(1) = 1-p = 0.99$). The smaller the value of p is, the easier the two distributions can be distinguished. Note that, in a noisy environment, even when the transmitter does nothing, the receiver can observe modulations which are generated by noise and the corresponding PDF of the received signal will have non-zero values at $n \geq 1$. Moreover, if the transmitter and the receiver are composed of multiple addresses, the PDFs can have non-zero values at $n \geq 2$.

With the two PDFs partially overlapped, the decoding step becomes complicated, but still feasible. As discussed in Chapter 4, if the receiver can collect enough samples of the signal, she can decode the secret bit with a high success rate, e.g., 99%.

5.3 The Security Metric

To evaluate randomly mapped caches and compare different cache configurations, we propose to use *end-to-end communication costs* as the quantitative security metric. Recall that, prior works [9, 7, 8, 12] analyze randomly mapped caches by quantifying the difficulty to perform the calibration step and have led to misleading security claims. Our end-to-end communication cost consists of the receiver's cost on the calibration step and the signaling step. Specifically, In CaSA, we use the number of cache accesses required by the receiver to decode the secret with a 99% confidence, where the cache accesses include the accesses performed by the receiver during the calibration step and the signal transfer step. Note that, other resources can also be used to express a cost, such as time or the number of times the victim is triggered.

Chapter 6

CaSA Implementation Details

CaSA is an end-to-end quantitative security analysis framework for communication via randomly mapped caches. Note that, even though we designed the framework for randomly mapped caches, it can be easily used to analyze other simpler cache designs.

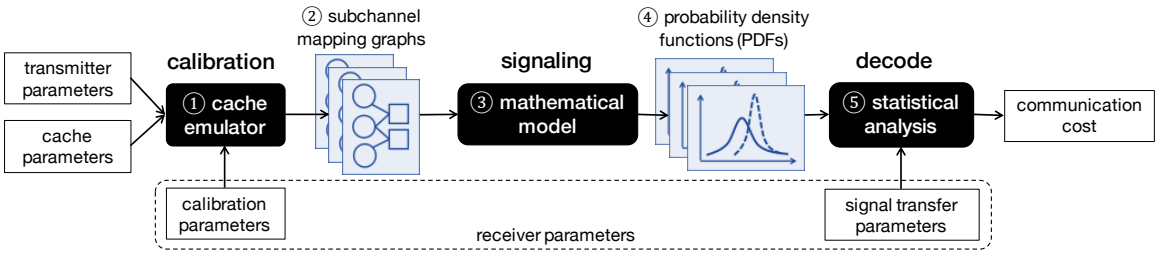


Figure 6-1: CaSA: end-to-end quantitative security analysis framework.

6.1 CaSA Work Flow

CaSA is composed of three modules analysing the three identified steps in the communication process: calibration, signaling and decode, shown in Figure 6-1. It can be used to explore a large security analysis space listed in Table 5.1 and compute the *communication cost* for various communication parameters.

The first module is the calibration module (①), which uses a cache emulator to simulate the cache's behavior during calibration. It takes the transmitter parameters and the cache parameters as input, and generates the calibration result. Due to the random behavior of the

cache, the module runs the calibration algorithm multiple times and each run generates a receiver set. We encode the calibration result (a group of receiver sets) as *subchannel mapping graph*s (②). The graph representation is to precisely capture the mapping relationship between addresses and subchannels (cache lines).

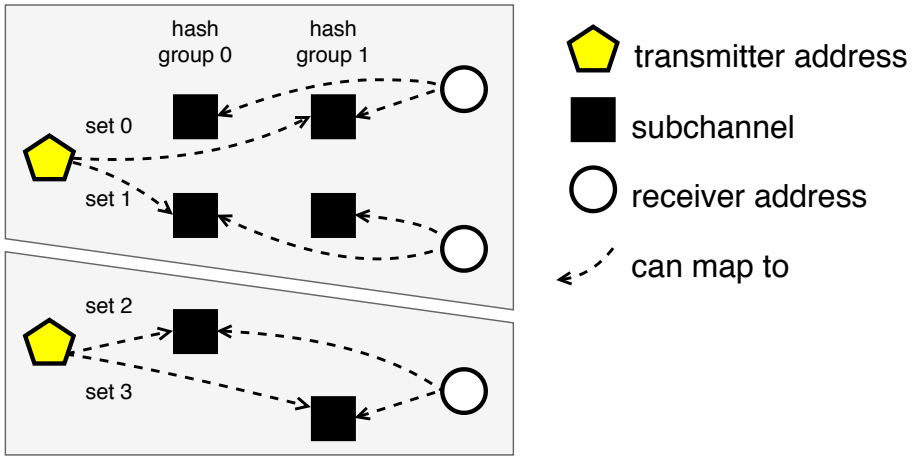


Figure 6-2: An example of subchannel mapping graph.

Figure 6-2 shows an example of a subchannel mapping graph on a cache with 4 sets and 2 hash groups. The graph is a directed bipartite graph [37] with two disjoint sets of vertices for addresses (pentagon and circle) and subchannels (square). An edge always connects an address vertex to a subchannel vertex, indicating the address can map to the subchannel. An address vertex can either be a transmitter address (pentagon) or a receiver address (circle). The graph does not include the subchannels which no address maps to, such as set 2 in hash group 1. There may exist multiple connected components in the graph, depending on the conflict relationships between addresses, such as the two shaded areas in Figure 6-2.

The second module is the signaling module (③), which takes a subchannel mapping graph as input and uses a mathematical model to compute the signal transfer result. For each value the secret can take, the signaling module outputs a signal probability density function (④), which describes the distribution of the number of modulations observed by the receiver.

The last module is the decode module (⑤), which takes the probability density functions (PDFs) as input and computes the end-to-end communication cost of the receiver (Section 5.3). It uses a statistical analysis method to compute the number of accesses needed

by the receiver to decode the secret with a given confidence value, e.g., $\geq 99\%$. Note that, it can also measure the cost for communicating across epochs.

To evaluate the security of a cache configuration, we use the above framework to compute the communication cost for different combinations of receiver parameters and find the one with the lowest cost.

We now provide details for each module.

6.2 The Calibration Module

The calibration module uses a cache emulator to model the state-of-the-art calibration algorithms. These algorithms are eviction set construction algorithms proposed by Qureshi et al. [8] and Purnal et al. [12]. We generalize the algorithms into three steps, shown in Figure 6-3.

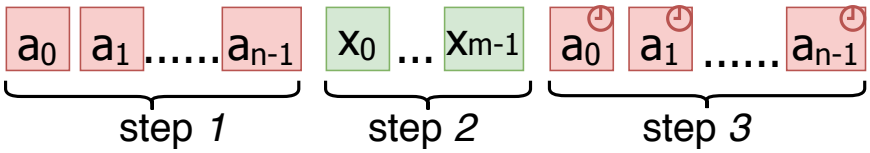


Figure 6-3: The calibration algorithm for multi-address transmitters.
a_0 to a_{n-1} are candidate addresses, and x_0 to x_{m-1} are transmitter addresses.

The calibration starts with a *candidate set* which is composed of many randomly chosen addresses. The candidate set should contain enough addresses so that some of them map to the subchannels used by the transmitter addresses.

1) The receiver accesses the addresses in the candidate set, making them all reside in the cache. This step requires multiple accesses to each candidate address to ensure every access hits in the cache. It also requires dropping some addresses if the candidate set cannot fit in the cache.

2) The transmitter addresses are accessed, which potentially evict some of the candidate addresses from the cache.

3) The receiver re-accesses the candidate set and measure the access latency of every address. Based on the latency, the algorithm decides whether a candidate address should be included in the receiver set or not.

In step 3, the algorithm can either add all the address that missed in the cache to the receiver set, or add only the first address that missed to the receiver set. We call the former one as a *greedy* calibration strategy and the latter one as a *non-greedy* strategy. We evaluate the cost of both calibration strategies in Section 7.1.

6.3 The Signaling Module

The signaling module takes the subchannel mapping graph as input, model the cache behaviors during the signal transfer step, and computes the distributions of the signals for different secret values. Recall that, the distributions of the signals are characterized by probability density functions (PDFs) or cumulative density functions (CDFs) of the number of modulations observed by the receiver (Section 5.2).

The signaling step consists of three operations: 1) the receiver performs precondition and monitors a group of subchannels; 2) the transmitter modules another group of subchannels; 3) the receiver performs detection and observes modulations on the subchannels that are both monitored by the receiver in step 1 and are modulated by the transmitter in step 2.

Correspondingly, given a set of subchannels, our module follows three steps to compute the probability of the receiver detecting modulations on these subchannels. Given a single subchannel s, we define the event D_s when a modulation is *detected* on the subchannel s. Similarly, the event $\bigcap_{i=1}^{n} D_{s_i}$ is the receiver detecting modulations on subchannels $\{s_1, \ldots, s_n\}$. The three steps are as follows.

1. Compute the probability that the given subchannels are monitored by the receiver, noted as $P_r(s_1, \ldots, s_n)$, using a Markov chain approach.

2. Compute the probability that these subchannels are modulated by the transmitter, noted as $P_t(s_1, \ldots, s_n)$, using a simple probability calculation.

3. Compute the probability that the receiver detects the modulations on these subchannels by calculating the joint probability from the above two steps. Since a subchannel being monitored and being modulated are *independent* events, their joint probability is the product of their individual probabilities: $P(\bigcap_{i=1}^{n} D_{s_i}) = P_r(s_1, \ldots, s_n) \times P_t(s_1, \ldots, s_n)$.

Once we obtain the detection probability for a given set of subchannels, we can compute the cumulative density functions (CDFs) of the signals. Basically, to compute $F(n)$, we enumerate all the sets of subchannels of size n and accumulate their detection probabilities.

Note that, for simplicity, the following discussion and formulas assume each hash group has a single way. The approach is applicable to caches with multi-way hash groups.

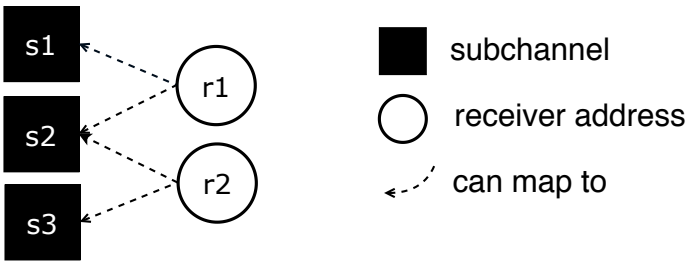
6.3.1 Compute Monitoring Probability

The precondition operation generally involves accessing the receiver addresses multiple times to *ensure* all the addresses are cached. Due to the random behavior of the cache, the precondition step is essentially a *stochastic process*.

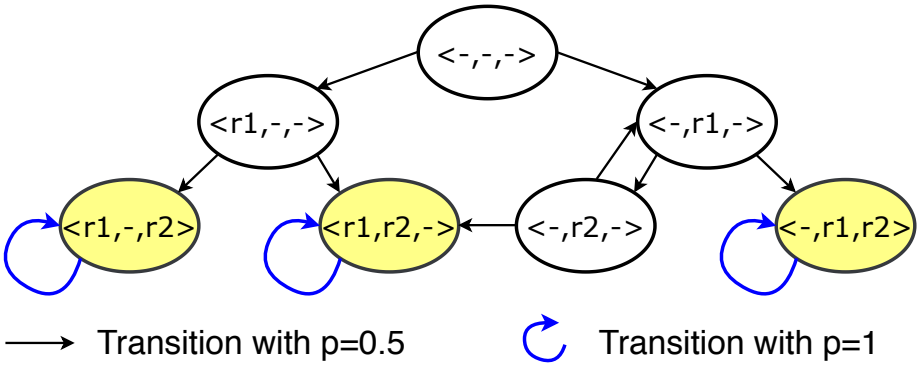
Given a subchannel mapping graph, several methods can be used to compute the monitoring probability (P_r). For instance, we could use a cache emulator to simulate the precondition process and empirically obtain the monitoring probability. An alternative approach is to model this process as a Markov chain [38] as below.

Figure 6-5(a) shows an example of the subchannel mapping graph without including transmitter addresses on a cache with 2 hash groups and 1 way in each hash group. Figure 6-5(b) shows the state transition graph of the corresponding Markov chain. Each state in the Markov chain summarizes the precondition states for all subchannels, denoted as a tuple. Each element in the tuple is for one subchannel, indicating which address monitors the subchannel. A dash (-) is used to indicate that the subchannel is not monitored.

The transition graph can be constructed as follows. The precondition process starts in the state <-, -, -> where none of the addresses monitor a subchannel. When address r1 is accessed, according to the subchannel mapping graph, it either preconditions s1, entering <r1, -, -> state with 0.5 probability, or preconditions s2, entering <-, r1, -> with the same probability. The other transitions can be constructed in a similar way. Conflicts between



(a) An example of subchannel mapping graph.



(b) The state transition graph of Markov chain.

Figure 6-5: Modeling the Signaling Step as a Markov Chain

receiver addresses are modeled by the transitions between states <-, r1, -> and <-, r2, ->, indicating address r1 and r2 conflict on subchannel s2 and keep evicting each other from the cache. A precondition operation completes when entering one of the absorbing states (highlighted). We can use standard approaches, such as power method [39], to compute the steady state of the Markov chain.

To compute the monitoring probability for a given set of subchannels P_r, we go through the absorbing states and accumulate the probabilities of all the states where these subchannels are monitored. For example, if we want to compute $P_r(s1)$, we will sum the probabilities to reach states <r1, -, -> and <r1, r2, ->.

Caches with multi-way hash groups. The Markov chain approach can be applied to caches with multi-way hash groups. We treat a cache set in a hash group as a subchannel. Instead of using a single element to represent the precondition state of a subchannel, we use a vector of elements and each element represents the state for one way within the subchannel. We can model random and LRU replacement policies.

6.3.2 Compute Modulation Probability

The modulation operation involves accessing the transmitter addresses for a fixed number of times. Considering the number of transmitter addresses is low, we assume that no subchannel is shared between each pair of transmitters. Therefore, we can simply compute the modulation probability (P_t) of each subchannel s that can be used by the transmitter as $P_t(s) = 1/g$, where g is the number of hash groups. In the case when we have a high number of transmitter addresses and these addresses share subchannels, other approaches like the Markov chain approach in Section 6.3.1 or simulation can be used.

Handling Noise. We model the impacts of two types of noise (Section 5.1): the background noise and the carrier noise. Both contribute to the modulation probability (P_t).

The background noise consists of accessing randomly chosen addresses which modulate random subchannels. We model a noise access as modulating each subchannel with a probability of $1/L$, where L is the number of cache lines. The carrier noise is a part of the transmitter. Therefore, we model the carrier noise in the same way as transmitter addresses.

6.3.3 Compute Density Functions

To compute the density functions, we first compute the detection probability of a given set of subchannels by calculating the joint probability of these subchannels being monitored and modulated. Next, we compute cumulative density functions ($F(n)$) by enumerating all the sets of subchannels at the size of n and summing their detection probabilities.

Consider $F(1) = P(X \geq 1)$, which gives the probability of the event "at least one modulation being detected". This event can be formulated as a union of multiple events and each event is "a modulation being detected on a specific subchannel", denoted as D_s. We note $\mathcal{S}$ of size N the set of all subchannels.

$$\text{Event } X \geq: \quad \bigcup_{s \in \mathcal{S}} D_s$$

We then apply the principle of inclusion-exclusion [40] to compute $F(1)$ as below.

$$F(1) = P(\bigcup_{s \in \mathcal{S}} D_s) = \sum_{k=1}^{N} \left((-1)^{k-1} \sum_{\{s_1,\ldots,s_k\} \subseteq \mathcal{S}} P(\bigcap_{i=1}^{k} D_{s_i}) \right) \tag{6.1}$$

We find it extremely expensive to compute the detection probability for every possible subset of $\mathcal{S}$. Indeed, it is incomputable for large subchannel mapping graphs. To greatly reduce the computation complexity, we use Bonferroni's inequalities [41] to solely compute bounds for the density functions. Specifically, to compute $F(1)$, we cut the sum at $k=1$ and $k=2$ to get the upper bound and the lower bound respectively.

$$\begin{aligned} F(1) &\leq \sum_{s \in \mathcal{S}} P(D_s) \\ F(1) &\geq \sum_{s \in \mathcal{S}} P(D_s) \; - \sum_{\{s,s'\} \subseteq \mathcal{S}} P(D_s \cap D_{s'}) \end{aligned} \tag{6.2}$$

With the same principle, we can derive bounds of $F(n)$ for any n.

Using the bounds makes the density functions imprecise, and can affect our estimation of the communication cost. We evaluate these impacts in Section 7.2. When the bound is too loose, we rely on simulation of the signaling step to empirically derive the CDFs.

6.4 The Decode Module

The decode module takes the density functions of the signal for each possible secret value as an input and computes the end-to-end communication cost of the receiver, which consists of the calibration cost and the signaling cost. The calibration cost can be directly derived from the calibration module by counting the number of accesses performed by the cache emulator. The signaling cost is computed by the decode module using a statistical method. Specifically, we compute the number of rounds of signal transfer that are needed by the receiver to decode the secret with 99% success rate and then convert it in number of cache accesses. We now describe how to compute the signaling cost, followed by the discussion on how to quantify the cost if communication spans across epochs.

The decode step consists in solving a statistical problem. Consider the transmitter communicates a secret bit $b \in \{0,1\}$ to the receiver by sending a signal X. The signal X is an *integer* random variable that follows its CDF $F_b(n)$, which is either equal to $F_0(n)$ or $F_1(n)$. The receiver decodes the secret by sampling X and deciding which one of the two distributions X follows. Intuitively, the more samples the receiver gets, the more accurately it can decode the secret. The problem that we need to solve is how many samples are needed to distinguish the two distributions with a certain confidence. This is a standard statistical problem with various solutions. We use an intuitive approach as follows.

To make the mathematical analysis simple, we convert the integer random variable X to a simpler signal, a Boolean variable Y. We define Y, equal to 1 if $X \geq 1$ (that is observing at least 1 modulation) and equal to 0 otherwise (observing no modulation). Hence the problem becomes: how many samples are needed to distinguish between two Boolean distributions of mean $F_0(1)$ and $F_1(1)$.

As a result, the decode strategy is straightforward. The attacker will simply perform the signal transfer several times, observe if the empirical average of Y is closer to $F_0(1)$ or $F_1(1)$, and guess the value of b accordingly.

Note that we could look at Y' equal to 1 if $X \geq 2$ (that is observing at least 2 modulation) and equal to 0 otherwise (observing 0 or 1 modulation). In some cases, Y' can be a better distinguisher than Y. In practice, we look for the value of n that maximizes the distance $|F_0(n) - F_1(n)|$, denoted as n_{max} and define Y as equal to 1 if $X \geq n_{max}$, 0 otherwise.

Intuitively, the more samples we get, the closer our empirical mean will get to $F_b(n_{max})$, and the more confidence we will have for the decode result. The Chernoff-Hoeffding bound [42] provides the relationship between the number of samples and the upper bound of the decode error rate (i.e., the lower bound of the certainty) as below.

$$P\left(|\frac{1}{N}\sum_{i=0}^{N} y_i - F_b(n_{max})| > \delta\right) \leq e^{-2\delta^2 N} \tag{6.3}$$

where N is the number of samples, y_i is a sample of the Boolean variable Y. The above formula shows that the empirical mean of Y gets closer to $F_b(n_{max})$ exponentially fast with the number of samples N.

To compute the signaling cost, we compute the number of samples (N) needed to achieve 1% error rate, which can be obtained by setting $\delta = |F_1(n_{max}) - F_0(n_{max})|/2$ and $e^{-2\delta^2 N} = 1\%$ in Equation (6.3).

Communication spanning across epochs. If the receiver cannot collect enough samples to achieve the required error rate within one epoch, the receiver has to decode based on samples gathered from multiple epochs. As shown in Figure 5-1, since the mapping function used by the cache is changed upon switching epochs, the receiver needs to redo the calibration in each epoch to generate a different receiver set, which leads to a different distribution of the signal. This is one of the reasons that prior work [7, 8] considers it infeasible to communicate across epochs.

We claim that it is viable to communicate across epochs, because the signals from different epochs follow a global distribution that can be leveraged for the decode step. Let's use g_i to denote the subchannel mapping graph for the ith epoch and X_i for the corresponding signal. For specific calibration and cache parameters, we define the space of all subchannel mapping graphs as G and the space of the corresponding signals as $\mathcal{X}$. Intuitively, when the receiver performs the calibration, no matter in which epoch, it obtains a sample from the subchannel mapping graph space G. Similarly, when the receiver collects a sample from any of the epochs, it is sampling the signal space $\mathcal{X}$. We call the distribution that $\mathcal{X}$ follows the global distribution. CaSA computes the global distribution of signals and use Hoeffding bound [42] to compute the signaling cost.

We now describe how to compute the global distribution. Until now, we have computed the distribution of the signal X_i conditioned on the subchannel mapping graph g_i, denoted as $P(X_i \geq n | g_i)$. Theoretically, given the probability of generating each subchannel mapping graph $P(g)$, we can compute the global distribution of signal $\mathcal{X}$ as below.

$$P(\mathcal{X} \geq n) = \sum_{g \in G} P(g) \times P(X \geq n | g) \tag{6.4}$$

With the global distribution, we can transform the signal $\mathcal{X}$ into a Boolean random variable as before. We then apply the Hoeffding bound [42] to compute the signaling cost, i.e., the number of samples required to decode the secret with 1% error rate.

In practice, we do not directly compute Equation (6.4), because we are unable to obtain the probability of generating each subchannel mapping graph $P(g)$ due to the extremely large space of G. Instead, we use an empirical approach. We have already obtained samples of subchannel mapping graphs in the calibration module and computed the conditional distributions in the signaling module. We found that if using the same calibration strategy, the conditional distributions $P(X \geq n|g)$ are fairly similar across epochs. Therefore, we can obtain a useful approximation of the global distribution using a small number of samples, e.g., using 30 subchannel mapping graph samples for 15 transmitter addresses. We show communication across epochs is feasible in Section 7.4.

Chapter 7

Evaluation

We use CaSA to evaluate 1MB caches with 1024 sets and 16 ways. We evaluate 3 cache configurations: a) 16 hash groups with 1 way per group, b) 8 hash groups with 2 ways per group, and c) 4 hash groups with 4 ways per group. Within each hash group, if there exists multiple ways, Last Recent Used (LRU) replacement policy is used. For caches that dynamically change the hash functions, we define the length of an epoch using the number of *epoch units*. In each epoch unit, the cache is accessed for L times where L is the total number of LLC lines. We evaluate the state-of-the-art calibration strategies [8, 12] and the classical signaling strategy, i.e., Prime+Probe.

CaSA measures and compares the communication cost of different attack strategies on randomly mapped caches. In this chapter, we first compare different calibration strategies. Second, we show how calibration parameters, cache parameters and noise parameters quantitatively affect the signaling cost. Finally, we show evaluation results of communication spanning across epochs.

7.1 Comparing Calibration Strategies

We compare *calibration efficiency* of using different calibration strategies in Figure 7-1. The calibration efficiency is measured by the number of receiver addresses generated per epoch unit. We assume the attacker chooses the size of the candidate set that is smaller than the

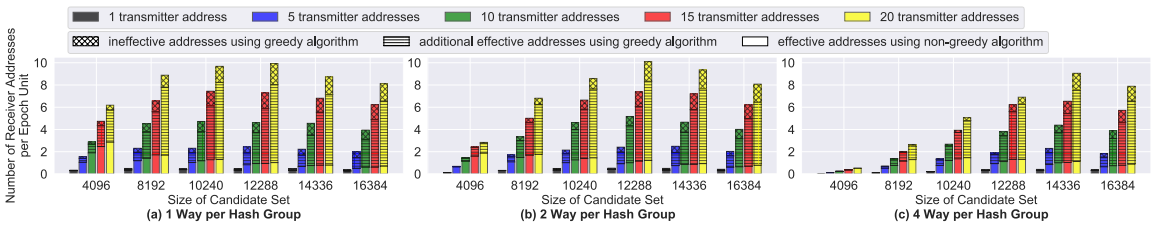


Figure 7-1: Comparing calibration efficiency of different calibration strategies.

private caches under its control. For example, if the receiver attacker can launch two threads and use two 256KB private caches, it could use 8192 candidate addresses.

For each candidate set size, we show from left to right, the calibration efficiency when the transmitter is composed of 1, 5, 10, 15 and 20 addresses. Each bar is broken into three categories from bottom to top: the number of receiver addresses obtained using the non-greedy calibration algorithm, the additional number of effective addresses obtained using the greedy algorithm, and the number of ineffective addresses obtained by the greedy algorithm. Ineffective addresses do not map to any of the subchannels associated with the transmitters.

On the 3 cache configurations, the greedy algorithm consistently obtains more receiver addresses than the non-greedy algorithm when there is more than 1 transmitter address. However, the greedy algorithm introduces 5% to 20% ineffective addresses into the receiver sets. The calibration efficiency of the greedy algorithm increases almost *linearly* with the number of transmitter addresses. In the following evaluation, we use the greedy calibration algorithm.

Finding 1: *Calibration efficiency increases almost linearly as the number of transmitter addresses increases.*

7.2 Validating the Signaling Module and the Decode Module

Validating the signaling module. We evaluate the precision of the signaling module by comparing the CDFs calculated by CaSA and the ones obtained empirically via sampling.

Figure 7-2 shows the comparison results of $F_1(1)$ on a cache with 1 way per hash group, i.e., $P(X \geq 1)$: the probability of the receiver detecting at least 1 modulation when the transmitter addresses are accessed. For each data-point, the theoretical bound is computed using Equation (6.2) and the empirical probability is computed by averaging 20k signaling samples.

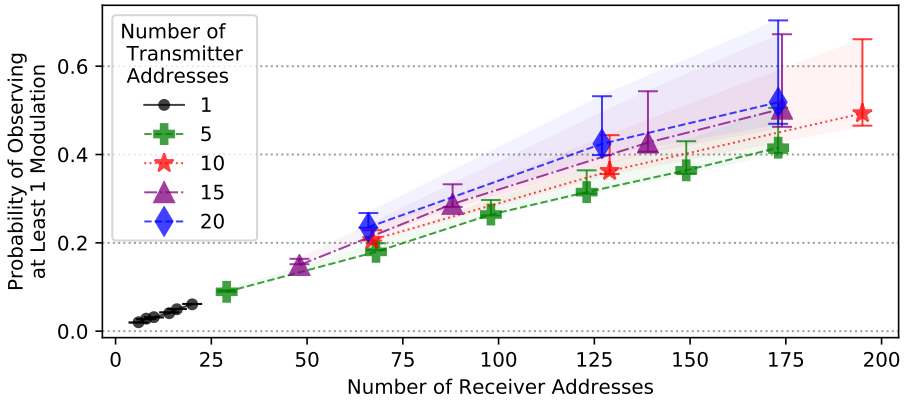


Figure 7-2: Validating CaSA's signaling module.

The probability of detecting at least one modulation increases almost linearly with the number of receiver addresses, which matches our intuition (Chapter 4) that using more receiver addresses has a higher probability to monitor subchannels that can be used by transmitters.

The empirical probability always falls within the theoretical bounds found by our model. The theoretical bound gets looser as the number of receiver addresses increases. When computing the probability of observing more modulations, e.g., $F(2)$, the empirical probability can be out of the theoretical bound, when the number of receiver addresses is high. This imprecision is caused by the approximation method (Section 6.3) that we used for computing large Markov chains.When the bounds are invalid, we rely on using empirical CDFs for the end-to-end analysis.

Validating the decode module. We further validate the decode module by comparing the signaling cost computed by CaSA and the cost obtained via sampling. The signaling cost is measured by the number of samples needed by the receiver to achieve 1% decode error rate (Section 6.4). Figure 7-3 shows the comparison results of on a cache with 1 way per hash group in a noisy environment, where the noise is arbitrarily modeled by accessing

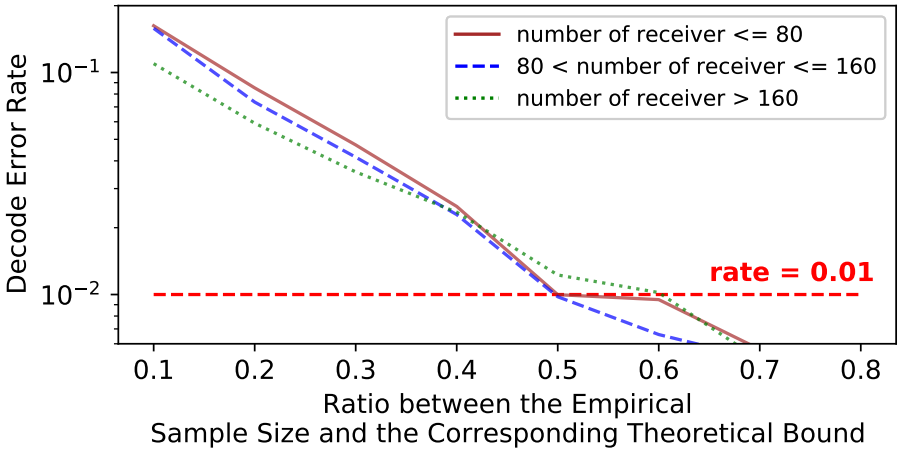


Figure 7-3: Validating CaSA's decode module.

300 random addresses between the receiver's precondition and detection. The theoretical numbers are computed using Equation (6.3). To obtain the empirical numbers, we first gather 600k samples for each calibration result. We then divide these samples into groups, perform the decode step in each group and compute the decode error rate.

Figure 7-3 shows the empirical decode error rate decreases when the number of samples increases. The x axis shows the ratio between the empirical sample size and the corresponding theoretical bound. In all cases, our bound is valid, as the error rates get below 1% when the sample sizes are around 50%-60% of the bound. The gaps between the empirical numbers and the theoretical bounds come from the imprecision in computing monitoring probabilities and the Chernoff bound.

7.3 Measuring Signaling Cost

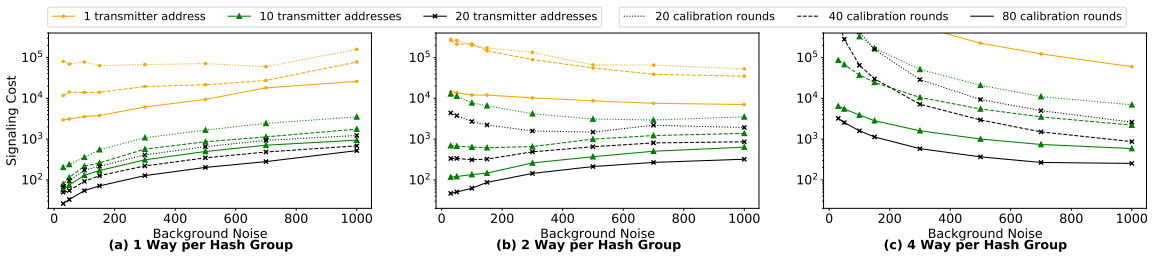


Figure 7-4: Impacts of communication parameters on signaling cost.

We evaluate the impacts of communication parameters on signaling cost, including transmitter parameters, calibration parameters, cache configurations and background noise. Figure 7-4 compares the signaling cost for achieving 1% error rate on 3 different cache configurations. The signaling cost is the number of samples computed using the Chernoff bound (Equation (6.3)) on the empirical density functions which we obtain via sampling. In each plot, we show how the signaling cost changes with the background noise, which is modeled as accessing a certain number of random addresses. We also compare the signaling cost for different numbers of transmitters and calibration parameters.

Across the three cache configurations, more transmitter addresses and more calibration efforts both lead to lower signaling cost. On a cache with 1 way per hash group in Figure 7-4(a), the signaling cost increases almost exponentially as the noise increases when the transmitter is composed of 1 address. When there are more transmitter addresses and noise is low, the signaling cost increases sub-exponentially. However, an interesting finding from our evaluation results is that noise does not always have negative impacts on communication. On caches with multiple ways per hash group in Figure 7-4(b) and (c), increasing noise sometimes help decrease signaling cost. This phenomenon can be explained intuitively using the following example. Consider a cache with 2 ways per hash group and both the transmitter where the receiver use one address to communicate. Without noise, the receiver cannot detect any modulation no matter whether the transmitter address is accessed or not. With a single noise access, the receiver has a chance to observe a modulation when both the transmitter address and the noise address modulate the subchannel that it monitors, and still no chance to observe a modulation when the transmitter is not accessed. Hence, adding noise in this example has a positive impact on communication.

Comparing the three cache configurations, we do not see a particular cache configuration has a clear advantage of security over others. When there is light background noise, the signaling cost is higher when the cache has more ways per hash group. However, communication on such caches can tolerate more noise. For example, when using 20 transmitter addresses, 80 rounds of calibration and 1000 accesses as background noise, the signaling cost on the cache with 2 ways per hash group (321 samples) is lower than the cost on the cache with 1 way per hash group (521 samples).

Finding 2: *Signaling cost on caches with 1 way per hash group increases exponentially or sub-exponentially as the noise increases.*

Finding 3: *Signaling costs on different cache configurations are mostly at the same order of magnitude.*

Insight: Noise does not always have negative impacts on communication on randomly mapped caches.

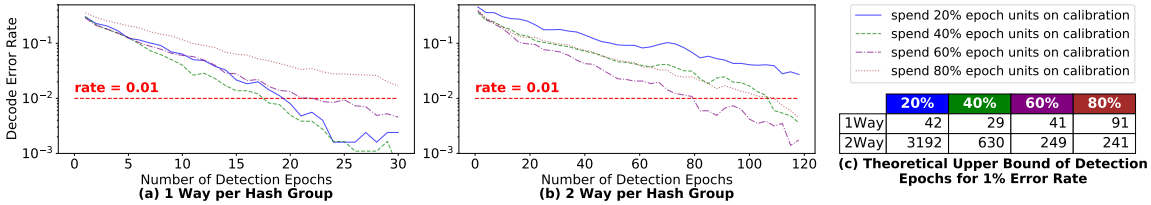
Insight: There does not exist a cache configuration that has a clear advantage of security over others.

7.4 Communication Across Epochs when Attacking RSA

We show the feasibility of communicating across epochs using transmitter parameters from a real victim application, the square-and-multiply exponentiation function [33] in the RSA encryption algorithm. The receiver tries to distinguish whether the transmitter executes the `square` or `multiply` function. We use Pin [36] to identify 16 transmitter addresses (at cache line granularity) exclusively used by the `square` function and 10 carrier addresses shared by the two functions.

The end-to-end communication cost is the number of LLC accessed by the receiver during calibration and signaling. The calibration cost is directly derived from the decode module (results in Figure 7-1). The signaling cost is computed by multiplying the number of signaling samples and the number of LLC accesses needed to obtain each sample. The number of LLC accesses per sample can be very different depending on whether the receiver can use the `clflush` instruction. To repeat the signaling step within one epoch, which we call *contiguous signaling*, the receiver needs to evict the receiver addresses from the cache before the next signaling round begins. This *self-eviction* operation can be completed using the `clflush` instruction with negligible cost and according to our evaluation, the communication can *always* complete within one epoch with 1% error rate. However, in the case that `clflush` is unavailable, the self-eviction operation requires accessing many random addresses. Specifically, for n receiver addresses, we additionally count $(\ln(n)+1) \times 16k$

LLC accesses in each signaling round. Note that, this self-eviction operation on traditional set-associative caches only require the number of accesses equal to the associativity.



	20%	40%	60%	80%
1Way	42	29	41	91
2Way	3192	630	249	241

(c) Theoretical Upper Bound of Detection Epochs for 1% Error Rate

Figure 7-5: Empirical decode error rate (a, b) and theoretical bound of communication cost (c) when communicating across epochs.

In Figure 7-5, we show evaluation results of using different receiver parameters to communicate on two cache configurations whose epoch sizes equal to 100 epoch units [8]. Figure 7-5(a) and (b) shows the empirical number of epochs needed to achieve 1% decode error rate, and Figure 7-5(c) shows the theoretical bounds. The error rate decreases as the number of epochs increases, confirming the effectiveness of the multi-epoch strategy.

We observe that spending more resources on calibration does not always help communication. For example, on a cache with 1 way per hash group, spending 40% of the epoch on calibration achieves the highest communication efficiency, and on a cache with 2 ways per hash group, the best efficiency is achieved when spending 60% of the epoch on calibration.

Finding 4: ***Contiguous*** *signaling requires evicting receiver addresses, which increases the signaling cost by multiple thousand times on randomly mapped caches compared to traditional set-associative caches.*

Insight: Information can be leaked and accumulated across epochs even when mapping functions are changed.

Insight: Spending the maximum amount of resources on calibration is neither the only nor always the best strategy.

7.5 Varying Epoch Sizes

We analyze how varying epoch sizes can affect the communication cost. We repeat the experiment on attacking RSA in a cache with 1 way per hash group and vary the epoch size

from 5 units to 100 units and an infinite size. The communication cost is the number of LLC accesses to send 1 secret bit across epochs. For each epoch size, Figure 7-6 shows the lowest communication cost and the corresponding communication strategy, which is represented using the number of calibration rounds per epoch. The signaling cost is the number of samples computed using Equation (6.3) on the empirical density functions which we obtain via sampling.

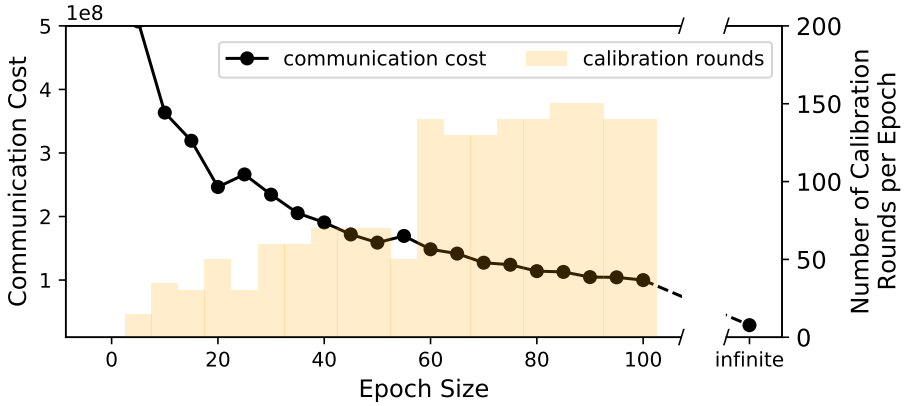


Figure 7-6: The impacts of varying epoch sizes on communication cost.

Overall, the communication cost decreases as the epoch size increases from 5 units to 100 units. When the epoch size is equal to or below 1 unit, such as the configuration in NewCache [10], our communication strategy becomes infeasible. However, as demonstrated in prior work [7, 8], using such a small epoch size can introduce high performance overhead. The number of calibration rounds increases as the epoch size increases from 5 to 60 units. When the epoch size is larger than 60 units, spending more resources on calibration does not help decrease the overall communication cost.

Chapter 8

Discussion

We briefly discuss how CaSA can be extended for the following cases.

Handling Complex Encoding Schemes

We have evaluated a simple encoding scheme of one Boolean secret bit so far. When multiple transmitters are used to encode multiple secret bits using a more complex encoding scheme, we identify two potential decode strategies and the corresponding analysis that can be supported by CaSA.

First, multiple receiver sets are used for signaling. If n transmitters are used to encode n secret bits, the receiver can decode these bits using n different receiver sets. Specifically, it calibrates one receiver set for each transmitter. When perform a signaling step, the receiver uses the n receiver sets in parallel. CaSA could be easily extended to handle this decode strategy as follows. In this case, for each receiver set, the modulations from the other transmitters and receivers are modeled as a new type of noise during signaling, as this receiver set is not calibrated for those addresses. CaSA could be easily extended to handle this type of noise.

Second, a single receiver set is used for signaling. In the case that each transmitter is composed of a different number of addresses, the receiver can calibrate to obtain a receiver set for a union of these transmitters. Accessing different combinations of the transmitters may result in 2^n different distributions of the signal. The mathematical problem that needs

to be solve in the decode module is how many samples are needed to distinguish a group of distributions, instead of two. To compute bounds for this problem, CaSA will need to be extended to use more advanced mathematical methods.

Computing Upper Bounds of Communication Bandwidth

CaSA does not compute lower bounds of communication cost (i.e., upper bounds of side-channel bandwidth) for a given cache configuration. However, from the an information leakage perspective, the upper bound of the communication bandwidth can be more useful, as it makes it possible to reason about the the maximum number of bits that can be leaked per epoch. We think it is possible to derive a *probabilistic upper bound* using an approach similar to the one proposed by Purnal et al. [13], where they computed a probabilistic upper bound for the calibration step only.

Chapter 9

Related Work

We have discussed the limitations of prior attempts to analyze randomly mapped caches [8, 9, 12] in Chapter 4. We now cover related work on analyzing and measuring side channel vulnerabilities.

The closest related work concurrently published is by Purnal et al. [13]. They also aims to quantitatively analyze the security of randomly mapped caches. Similar to us, they consider communication using smaller receiver sets and multi-address transmitters. There are two key differences. First, one of the key contributions of CaSA is to identify the existence of a trade-off between signaling and calibration and to quantify the end-to-end communication cost. However, they solely focus on the calibration step. Their contributions lie in optimizations for the attacker to reduce calibration cost. Second, one of the key findings of CaSA is that communication can happen across epochs. However, their analysis still assumes communication needs to complete within one epoch and for a given epoch size, they compute the upper bound of the success rate to obtain a receiver set with 95% eviction rate.

He et al. [43] proposed an approach to quantitatively evaluate a cache's resilience against multiple classes of attacks on traditional set-associative caches. They build a probabilistic information flow graph for steps in an attack, compute the probability of success for each step, and then compute the probability of success for the whole attack. The key difference from CaSA is that their approach only focuses on the signaling step, and works assuming

a specific calibration result. If the calibration result is unknown, they cannot compute the probability for each attack step. Thus, their approach cannot be used to analyze randomly mapped caches.

SVF [44] and CSV [45] are metrics used to quantitatively measure side-channel leakage in processors by compute statistical correlation between transmitter and receiver's execution traces. CaSA is different from these works. First, they use empirical approaches to obtain traces, while CaSA builds a mathematical model to obtain the communication signal. Second, they compute the metric for given attack traces, while CaSA performs space exploration to find the attack parameters that achieve the best communication efficiency.

Several tools, such as CacheAudit [46], cacheD [35] and CaSym [34], have been proposed to detect side channel vulnerabilities in software. These tools are effective in locating data-dependent memory accesses. They generally focus on analyzing software and use very simple cache models. We find these tools complementary to CaSA, and it could be promising to extend these tools to generate the transmitter parameters for CaSA.

Statistic-based analysis has been widely adopted in analyzing power side channel attacks [47, 48, 49]. To the best of our knowledge, we are the *first* to formulate the signals in cache-based side-channel attacks into a statistical representation.

Chapter 10

Conclusion

In this paper, we comprehensively analyze the security of randomly mapped caches. Our result shows that the randomization mechanisms used in the state-of-the-art randomly mapped caches are insecure.

We have made key contributions in identifying the end-to-end communication procedure of microarchitecture side channels. Additionally, we leverage concepts from the field of telecommunication to formulate security analysis of randomly mapped caches into a statistical problem. It is promising to apply our approach to analyze side channels on other types of micro-architecture structures.

References

[1] D. A. Osvik, A. Shamir, and E. Tromer, "Cache Attacks and Countermeasures: The Case of AES," in *Cryptographers' Track at the RSA conference*, Springer, 2006.

[2] C. Percival, *Cache Missing for Fun and Profit*, `http://www.daemonology.net/papers/htt.pdf`, 2005.

[3] C. Canella, J. Van Bulck, M. Schwarz, M. Lipp, B. Von Berg, P. Ortner, F. Piessens, D. Evtyushkin, and D. Gruss, "A Systematic Evaluation of Transient Execution Attacks and Defenses," in *Proceedings of the 28th USENIX Conference on Security Symposium*, 2019.

[4] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre Attacks: Exploiting Speculative Execution," in *IEEE Symposium on Security and Privacy (SP)*, 2019.

[5] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading Kernel Memory from User Space," in *USENIX Security Symposium*, 2018.

[6] V. Kiriansky, I. Lebedev, S. Amarasinghe, S. Devadas, and J. Emer, "DAWG: A Defense Against Cache Timing Attacks in Speculative Execution Processors," in *51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE, 2018.

[7] M. K. Qureshi, "CEASER: Mitigating Conflict-Based Cache Attacks via Encrypted-Address and Remapping," in *51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018.

[8] ——, "New Attacks and Defense for Encrypted-address Cache," in *Proceedings of the 46th International Symposium on Computer Architecture (ISCA)*, 2019.

[9] M. Werner, T. Unterluggauer, L. Giner, M. Schwarz, D. Gruss, and S. Mangard, "ScatterCache: Thwarting Cache Attacks via Cache Set Randomization," in *28th USENIX Security Symposium*, 2019.

[10] Z. Wang and R. B. Lee, "New Cache Designs for Thwarting Software Cache-Based Side Channel Attacks," *SIGARCH Comput. Archit. News*, 2007.

[11] P. Vila, B. Köpf, and J. F. Morales, "Theory and Practice of Finding Eviction Sets," in *IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019.

[12] A. Purnal and I. Verbauwhede, "Advanced Profiling for Probabilistic Prime+Probe Attacks and Covert channels in ScatterCache," *arXiv preprint arXiv:1908.03383*, 2019.

[13] A. Purnal, L. Giner, D. Gruß, and I. Verbauwhede, "Systematic analysis of randomization-based protected cache architectures," in *42th IEEE Symposium on Security and Privacy*, May 2021.

[14] Y. Yarom and K. Falkner, "Flush+Reload: A High Resolution, Low Noise, L3 Cache Side-channel Attack," in *23rd USENIX Security Symposium*, 2014.

[15] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, "ARMageddon: Cache Attacks on Mobile Devices," in *25th USENIX Security Symposium*, 2016.

[16] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-Level Cache Side-Channel Attacks are Practical," in *Proceedings of the 2015 IEEE Symposium on Security and Privacy (SP)*, 2015.

[17] C. Disselkoen, D. Kohlbrenner, L. Porter, and D. Tullsen, "Prime+Abort: A Timer-Free High-Precision L3 Cache Attack Using Intel TSX," in *26th USENIX Security Symposium*, 2017.

[18] J. Bonneau and I. Mironov, "Cache-collision Timing Attacks against AES," in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2006.

[19] D. Gruss, C. Maurice, and K. Wagner, "Flush+Flush: A Stealthier Last-Level Cache Attack," in *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2016.

[20] D. Gullasch, E. Bangerter, and S. Krenn, "Cache Games—Bringing Access-based Cache Attacks on AES to Practice," in *IEEE Symposium on Security and Privacy (SP)*, IEEE, 2011.

[21] M. Yan, R. Sprabery, B. Gopireddy, C. W. Fletcher, R. Campbell, and J. Torrellas, "Attack Directories, Not Caches: Side Channel Attacks in a Non-Inclusive World," in *IEEE Symposium on Security and Privacy (SP)*, 2019.

[22] D. Gruss, C. Maurice, A. Fogh, M. Lipp, and S. Mangard, "Prefetch Side-channel Attacks: Bypassing SMAP and Kernel ASLR," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, ACM, 2016.

[23] T. Hornby, "Side-Channel Attacks on Everyday Applications: Distinguishing Inputs with FLUSH+RELOAD," in *BackHat*, 2016.

[24] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*, ACM, 2009.

[25] Y. Xu, M. Bailey, F. Jahanian, K. Joshi, M. Hiltunen, and R. Schlichting, "An Exploration of L2 Cache Covert Channels in Virtualized Environments," in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, 2011.

[26] D. Gruss, R. Spreitzer, and S. Mangard, "Cache Template Attacks: Automating Attacks on Inclusive Last-level Caches," in *24th USENIX Security Symposium*, 2015.

[27] Intel, *6th Gen Intel Core X-Series Processor Family Datasheet - 7800X, 7820X, 7900X*, `https://www.intel.com/content/www/us/en/products/processors/core/6th-gen-x-series-datasheet-vol-1.html`, 2017.

[28] C. Maurice, N. Le Scouarnec, C. Neumann, O. Heen, and A. Francillon, "Reverse Engineering Intel Last-level Cache Complex Addressing Using Performance Counters," in *Research in Attacks, Intrusions, and Defenses*, Springer, 2015.

[29] G. Irazoqui, T. Eisenbarth, and B. Sunar, "Systematic Reverse Engineering of Cache Slice Selection in Intel Processors," in *Proceedings of the 2015 Euromicro Conference on Digital System Design (DSD)*, 2015.

[30] Intel, *Intel Software Guard Extensions Programming Reference*, `https://software.intel.com/en-us/sgx/sdk`, 2013.

[31] R. Bodduna, V. Ganesan, P. Slpsk, C. Rebeiro, and V. Kamakoti, "BRUTUS: Refuting the Security Claims of the Cache Timing Randomization Countermeasure proposed in CEASER," *IEEE Computer Architecture Letters (CAL)*, 2020.

[32] A. Shusterman, L. Kang, Y. Haskal, Y. Meltser, P. Mittal, Y. Oren, and Y. Yarom, "Robust Website Fingerprinting Through the Cache Occupancy Channel," in *28th USENIX Security Symposium*, 2019.

[33] D. M. Gordon, "A Survey of Fast Exponentiation Methods," *Journal of Algorithms*, 1998.

[34] R. Brotzman, S. Liu, D. Zhang, G. Tan, and M. Kandemir, "CaSym: Cache Aware Symbolic Execution for Side Channel Detection and Mitigation," in *IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019.

[35] S. Wang, P. Wang, X. Liu, D. Zhang, and D. Wu, "CacheD: Identifying Cache-based Timing Channels in Production Software," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017.

[36] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," *Acm Sigplan Notices*, 2005.

[37] Wikipedia, *Bipartite Graph*, `https://en.wikipedia.org/wiki/Bipartite_graph`, 2020.

[38] ——, *Markov Chain*, `https://en.wikipedia.org/wiki/Markov_chain`, 2020.

[39] ——, *Power Iteration*, `https://en.wikipedia.org/wiki/Power_iteration`, 2020.

[40] ——, *Inclusion-exclusion Principle*, `https://en.wikipedia.org/wiki/Inclusion-exclusion_principle`, 2020.

[41] J. Galambos, "Bonferroni Inequalities," *The Annals of Probability*, 1977.

[42] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, Mar. 1963. [Online]. Available: `http://www.jstor.org/stable/2282952?`.

[43] Z. He and R. B. Lee, "How Secure is Your Cache Against Side-channel Attacks?" In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ACM, 2017.

[44] J. Demme, R. Martin, A. Waksman, and S. Sethumadhavan, "Side-channel Vulnerability Factor: A Metric for Measuring Information Leakage," in *39th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2012.

[45] T. Zhang, F. Liu, S. Chen, and R. B. Lee, "Side Channel Vulnerability Metrics: the Promise and the Pitfalls," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, 2013.

[46] G. Doychev, B. Köpf, L. Mauborgne, and J. Reineke, "CacheAudit: A Tool for the Static Analysis of Cache Side Channels," *ACM Transactions on Information and System Security (TISSEC)*, 2015.

[47] Y. Fei, A. A. Ding, J. Lao, and L. Zhang, "A Statistics-based Fundamental Model for Side-channel Attack Analysis," *IACR Cryptology ePrint Archive*, 2014.

[48] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer Science & Business Media, 2008.

[49] F.-X. Standaert, T. G. Malkin, and M. Yung, "A Unified Framework for the Analysis of Side-channel Key Recovery Attacks," in *Annual international conference on the theory and applications of cryptographic techniques*, Springer, 2009.