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

Thomas Bourgeat, Jules Drean, Yuheng Yang, Lillian Tsai, Joel Emer, Mengjia Yan

Presented by Peter Deutsch
MIT 6.888 - Secure Hardware Design
Motivation

It is well known that caches can be used to exfiltrate secrets through *timing side channels* such as Prime + Probe.

Micro-architects have attempted to mitigate side-channel leakage through the use of *randomly mapped caches*, which aim to increase the difficulty of an attack.

Many of these mitigation schemes make bold (*and ultimately quite fragile*) security claims based on varying attack strategies.

It is apparent that a unified framework is required to thoroughly evaluate cache security across proposed designs!
Threat Model

CaSA assumes that an attacker can:

- Observe the latency of its own memory accesses
- Reside in a user-level process or secure enclave
- Use more than one thread to control multiple cores
- Leverage speculative execution to provoke the victim

CaSA does not reason about:

- Attacks mounted in an SMT context
- Flush and occupancy based cache attacks
Overview - Primary Contributions

CaSA (Cache Security Analyzer) provides the following contributions:

1. **Demonstrates** a three-step, end-to-end communication paradigm which better evaluates the security properties of caches *beyond eviction set generation*

2. **Formulates** the security analysis of randomized caches into a statistical problem, allowing quantitative analysis through a novel framework

3. **Evaluates** existing randomly mapped caches and provides new insights regarding noise and communicating across cache epochs
Any Initial Thoughts? Strengths? Weaknesses?
My Thoughts

Strengths

- Provides the first framework which allows for a fair comparison of the security of contemporary secure/randomized caches
- Is very flexible, and can analyze a wide variety of potential cache configurations, allowing for design space exploration
- Clearly expresses and justifies surprising results (such as the impact of noise)

Weaknesses

- Doesn’t provide a tool to determine upper bounds for side-channel \textit{bandwidth}
- Fails to formulate statistical representations for multi-way caches
- Doesn’t consider communications schemes which use multi-bit symbols
Background - Cache-Based Side Channel Attacks

In cache-based side channel attacks, the cache is used as a communication channel, where each line can be viewed as a sub-channel.

**Takeaway:** We would like a cache where it is difficult to concretely know which channels are pre-conditioned by an attacker, and which channels are modulated by a victim.
**Background - Randomly Mapped Caches**

By introducing randomness into mapping functions, we can significantly increase the difficulty for an attacker to create an eviction set.

**Single Hash Group - Static Mapping** (ex. Standard Set-Associative Cache)

**Multiple Hash Groups - Dynamic Mapping** (ex. Skewed CEASAR\textsuperscript{1})

**Q:** Do randomized caches protect against Flush + Reload attacks? Why or why not?
Background - Hard and Soft Conflicts

In prior work, signalling is accomplished through abusing set conflicts with the victim.

**Hard Conflict Eviction**
- Receiver maps to the same set as the transmitter in **every** hash group.

**Soft Conflict Eviction**
- Receiver maps to the same set as the transmitter in **at least one** hash group.
Motivation - Limitations of Prior Work

Prior work makes *differing assumptions* on attacker strategies!

**Skewed-CEASAR**\(^1\) assumes the attacker uses hard-conflict receivers

**ScatterCache**\(^2\) assumes the attacker uses a large number of soft-conflict receivers

Which of these assumptions are valid?

What is the optimal attacker strategy?

\(^1\) New Attacks and Defense for Encrypted-Address Cache - Qureshi et al.

\(^2\) ScatterCache: Thwarting Cache Attacks via Cache Set Randomization - Werner et al.
There exists a *tradeoff* between communications steps. An attacker can either:

- *Spend more time on calibration*, obtaining a large eviction set which can be used to detect modulations with a higher probability
- *Spend more time on signalling*, taking more measurements in order to better filter out noise and obtain a higher success rate

**Q:** How does this tradeoff relate to the epoch length of a randomized cache?
The Calibration Module attempts to establish a relationship between transmitter/receiver addresses and the subchannels to which they map to.

**Cache Parameters**
- (# of ways, hash groups, etc)

**Transmitter Parameters**
- (# of transmitter addresses)

**Calibration Parameters**
- (# of calibration rounds)

**Sub-channel Mapping Graphs**
- (analogous to a receiver set!)

**Q:** How do we know how many transmitter addresses there are?
The **Signalling Module** attempts to model the distribution of the number of modulations observed by the receiver *for each possible value of the secret*.

**Q:** Where is noise considered?
The **Decode Module** computes the number of signal transfer rounds required to achieve a 99% success rate, then determines the *total communication cost*.
Key Insights

CaSA makes the following novel observations:

1. Spending the maximum amount of time in the calibration phase is not always the best strategy.

2. Noise can actually reduce our signalling cost in some cases!

3. Information can be leaked and accumulated across epochs, even when the mapping functions are changed.
**Evaluation - Signalling Cost + Noise**

**Q:** Can noise be beneficial when there is only one way per hash group?  

**A:** No.
**Evaluation - Communications Costs**

**Q:** Why is spending 20% of epoch units on calibration so much more productive in the “1 Way per Hash Group” case?
Discussion Questions
Discussion Questions - Cache Hardening

- Can hash mechanisms be devised to minimize collisions between programs and provide better results than random mapping?
- It's important during the calibration step to only choose addresses from the candidate set that are useful - how does this factor into the calibration efficiency?
- How can the attacker determine when a new epoch has started? Is intermittently randomizing the epoch length a viable option to improve security?
- This is a side channel and not a covert channel - what's the guarantee that the transmitter will access the same specific address as many times as you need?
Discussion Questions - Future Work

- What can be done in the future to avoid making the same mistakes as the previous security analyses and making incorrect security guarantees?
- Can an analysis framework similar to CaSA be applied to other structures within the CPU? Could it be applied to multi-level caches in an SMT context?
- How would CaSA need to be adapted in order to consider multi-bit symbol transmissions?
- Is it feasible (or worth attempting) to determine lower bounds for communications costs?
- Are we “doomed” to a future where caches must have tunable parameters (such as epoch lengths and hash groups) to remain secure?