MIRAGE: Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design

A design to eliminate eviction-sets & cache attacks

USENIX Security 2021

Gururaj Saileshwar and Moinuddin Qureshi

Georgia Tech
Problem: CPU Cache Side-Channels

Shared LLC helps improve performance, but timing difference (LLC Hit vs Miss) leads to side-channels!
Problem: CPU Cache Side-Channels

- **L1 Cache**
- **L2 Cache**
- **CORE-0**
- **CORE-1**
- **SPY PROCESS**
- **VICTIM PROCESS**
- **Ways**
- **Shared Cache Set**
- **Last Level Cache (Set-Associative)**
- **Shared Channel**
- **Problem: CPU Cache Side-Channels**
Problem: CPU Cache Side-Channels

Prime+Probe Attack
[Bernstein’05], [Perceival’05],
Liu et al. [S&P’15]

Shared Cache Set

Last Level Cache
(Set-Associative)
Problem: CPU Cache Side-Channels

Prime+Probe Attack
[Bernstein’05],[Perceival’05], Liu et al. [S&P’15]
Problem: CPU Cache Side-Channels

Prime+Probe Attack
[Bernstein’05],[Perceival’05],
Liu et al. [S&P’15]

Spy can Leak Victim Secrets like AES/RSA Keys, User Key-Strokes, etc.
Randomized Cache Defenses

Prime+Probe Attack

Address $f$

Eviction-Set ($V$)

Set-Conflict

CORE-0 (Spy)

CORE-1 (Victim)

LLC
Randomized Cache Defenses

Prime+Probe Attack

Core-0 (Spy)

Core-1 (Victim)

LLC

Address

f

Set-Conflict

Eviction-Set (V)

Randomized Cache Defenses

[MICRO’18], [ISCA’19], [SEC’19], [NDSS’20], [S&P’21]

Address

Rf

Sets

A

B

V

Randomized Set-Indexing
Obfuscates Set-Conflicts

Eviction-Sets harder to discover
Randomized Cache Defenses

Prime+Probe Attack

Address $f$ → $f$

Eviction-Set $(V)$

CORE-0 (Spy)

CORE-1 (Victim)

LLC

A B

Set-Conflict

Randomized Cache Defenses

[MICRO’18], [ISCA’19], [SEC’19], [NDSS’20], [S&P’21]

Eviction-Set $(V)$

Randomized Set-Indexing
Obfuscates Set-Conflicts

Successive Defenses Broken By Newer Attacks
Arms Race Between Attacks & Defenses

Intel LLC
Proprietary Mapping
N lines in LLC

CEASER [MICRO’18]
Randomized Mapping
Dynamic Remapping

CEASER-S, Scatter-Cache
[ISCA’19] [USENIX-SEC’19]

Broken by
Eviction-Set
Discovery in

O(N²) Accesses
[S&P’15]

O(N) Accesses
[ISCA’19], [S&P’19]

Fast Probabilistic Evictions
[S&P’21]

Skews

Intel LLC
Proprietary Mapping
N lines in LLC

Line Install (X)

CEASER
Randomized Mapping
Dynamic Remapping

CEASER-S, Scatter-Cache
[ISCA’19] [USENIX-SEC’19]

Broken by
Eviction-Set
Discovery in

O(N²) Accesses
[S&P’15]

O(N) Accesses
[ISCA’19], [S&P’19]

Fast Probabilistic Evictions
[S&P’21]
Arms Race Between Attacks & Defenses

**Intel LLC**
Proprietary Mapping
N lines in LLC

**CEASER** [MICRO’18]
Randomized Mapping
Dynamic Remapping

**CEASER-S, Scatter-Cache**
[ISCA’19] [USENIX-SEC’19]

Broken by Eviction-Set Discovery in

O(N²) Accesses
[S&P’15]

O(N) Accesses
[ISCA’19], [S&P’19]

Fast Probabilistic Evictions
[S&P’21]

Pitfall: Set-Conflicts Continue at Few Obfuscated Locations

Our Goal: Eliminate Set-Conflicts to End the Arms Race
Goal: Fully-associative Randomized LLC

Abstraction to SW

Fully-Associative: No Set Conflicts

- Line Install
- Random Eviction From Entire Cache

Mirage LLC

Principled Security

Challenge: Naive Fully-Associative Lookup Requires Checking 100,000+ LLC Locations

Can Map to Any Cache Location

Line-Install

A B C

X

Impractical Lookup Latency & Power
Goal: Fully-associative Randomized LLC

**Key Challenge:** How to get Security of Fully-Associative Design (No Set-Conflicts) with Set-Associative Lookups?

**Abstraction to SW**

*Fully-Associative: No Set Conflicts*

- Line Install
- Random Eviction From Entire Cache

**Principled Security**

**Traditional Set-Associative Lookup**

- Line-Install
- Sets

**Practical Lookup within Set (16-32 Locations)**

- Extra Tags & Indirection
- Global Replacement

**Mirage**

- Any line evicted
- Maps to any line in cache
- Abstraction to SW
- Overview of Mirage
- Tag-Store

**Skewed-Indexing**

**Load-Aware Skew Selection**

**Skewed-Indexing +/− Global Random Eviction**

**Extra-tags & Indirection**

- Data-Store
Insight: Use Load-Balancing to Eliminate Set-Conflicts

Buckets & Balls Problem

Buckets

\( \text{(Sets)} \)

Capacity = \( w \)

Random

Balls

\( \text{(Cache Lines)} \)

\[ \rightarrow \text{Bucket Overflow (Set-Associative Eviction or SAE)} \]

Set-Associative Randomized LLC

\( w \text{ ways} \)

Line-Install

\[ \rightarrow \text{Eviction} \]
Insight: Use Load-Balancing to Eliminate Set-Conflicts

16 Balls in 4 Buckets (C=4)

- Buckets (Sets) C=4
  - Bucket Overflow Every Ball Throw

16 Balls in 4 Buckets (C=8)

- Buckets (Sets) C=8
  - Bucket Overflow Likelihood Reduced, But Still Possible

16 Balls in 4 Buckets (C=8) & Power of 2 Choices
  [Mitzenmacher’96]

- Bucket Overflow Improbable: Balanced Distribution
Security Guarantee With Power of 2 Choices

Frequency of Bucket Overflows

- Theoretical Model
- Simulation Results

With 75% extra capacity, 1 Bucket Overflow per $10^{34}$ Ball Throws

Ball Throws per Bucket Overflow

Extra Capacity Per Bucket (Default-Capacity = 8)
Security Guarantee With Power of 2 Choices

- **Strong Security:** SAE unlikely in lifetime of universe
- **Modest Costs:** 2% Slowdown, 18% Storage Overhead

**Set-Associative Eviction (SAE)**

LLC with 75% extra capacity & Power of 2 Choices Indexing has Security Guarantee of 1 SAE Per $10^{34}$ LLC Installs ($10^{17}$ years)

**Simulation Results**

- With 75% extra capacity, 1 Bucket Overflow per $10^{34}$ Ball Throws

**Frequency of Bucket Overflows**

- **Theoretical Model**
- **Simulation Results**

**Extra Capacity Per Bucket (Default-Capacity = 8)**

- 0
- 1
- 2
- 3
- 4
- 5
- 6

**Line Installs per SAE**

**Ball Throws per Bucket Overflow**
Extra Tags Cheap, Extra Data Expensive (1:10)

Tag-Store

Sets

Ways 75% extra

Data-Store

One to One
MIRAGE Design

MIRAGE (Decouples Tag and Data)

Install in Invalid-Tag

Global Random Eviction

Tag-Store

Data-Store

Sets

Ways

75% extra
MIRAGE Design

Power-of-2-Choices Indexing

Line Install In Invalid Tag

Skew-0

Skew-1

Global Random Eviction

[Diagram showing MIRAGE Design with Tag-Store and Data-Store with tags and data pointers]
MIRAGE Design

Power-of-2-Choices Indexing

Line Install In Invalid Tag

Skew-0

Rf\textsubscript{1}

Skew-1

Rf\textsubscript{2}

Global Random Eviction

MIRAGE

Tag-Store

Data-Store

Extra Tags Cheap, Extra Data Expensive (1:10)

Tag FPTR

Data RPTR

Sets Ways

Power -of-2 -Choices

Indexing

In Invalid Tag

Skew-0

Global Random Eviction

MIRAGE

Security Guarantee: With 75% extra tags, MIRAGE ensures 1 Set-Associative Eviction that can leak information every $10^{34}$ LLC Installs (once in $10^{17}$ years)

Eliminates Conflict-Based Attacks
Randomization Alone Cannot Mitigate
Shared-Memory Attacks
(e.g. Flush+Reload, Flush+Flush)

MIRAGE uses Domain-ID for duplication
of shared cache lines

Eliminates Shared-Memory Based Attacks
Results - Performance

Simulation of 8-Core 16MB/16-way LLC system on Trace-Based Simulator

- Scatter-Cache: 2% slowdown, 20% storage overhead (75% extra tags).
- Mirage: Neutral slowdown → 3.5%
- ScatterCache: 1.7% slowdown
- MIRAGE: 2% slowdown

2% Slowdown, 20% Storage Overhead (75% extra tags). Storage-Neutral Slowdown → 3.5%.

Paper includes MIRAGE-Lite with lower storage overheads (50% extra tags & similar security)

Additional Results in Paper: LLC Misses, Lookup Latency, Logic Overhead, RISC-V, Gem5 etc.
Principled Security that Eliminates Cache-Attacks Leaking Victim Addresses

- **Strong Benefits:** Security of 1 SAE per $10^{17}$ Years
- **Modest Costs:** 2% Slowdown, 17% - 20% Storage Overhead
Thanks!

Code (Gem5 Artifact): https://github.com/gururaj-s/mirage
Slides: http://memlab.ece.gatech.edu/slides/SEC_2021_1_slides.pptx

Gururaj Saileshwar
gururaj.s@gatech.edu

Moinuddin Qureshi
moin@gatech.edu