UCLA Samueli School of Engineering

MIRAGE

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

Original Authors:

Gururaj Saileshwar and
Moinuddin Qureshi
Georgia Tech

Parangat Mittal
Electrical and Computer Engineering

Background

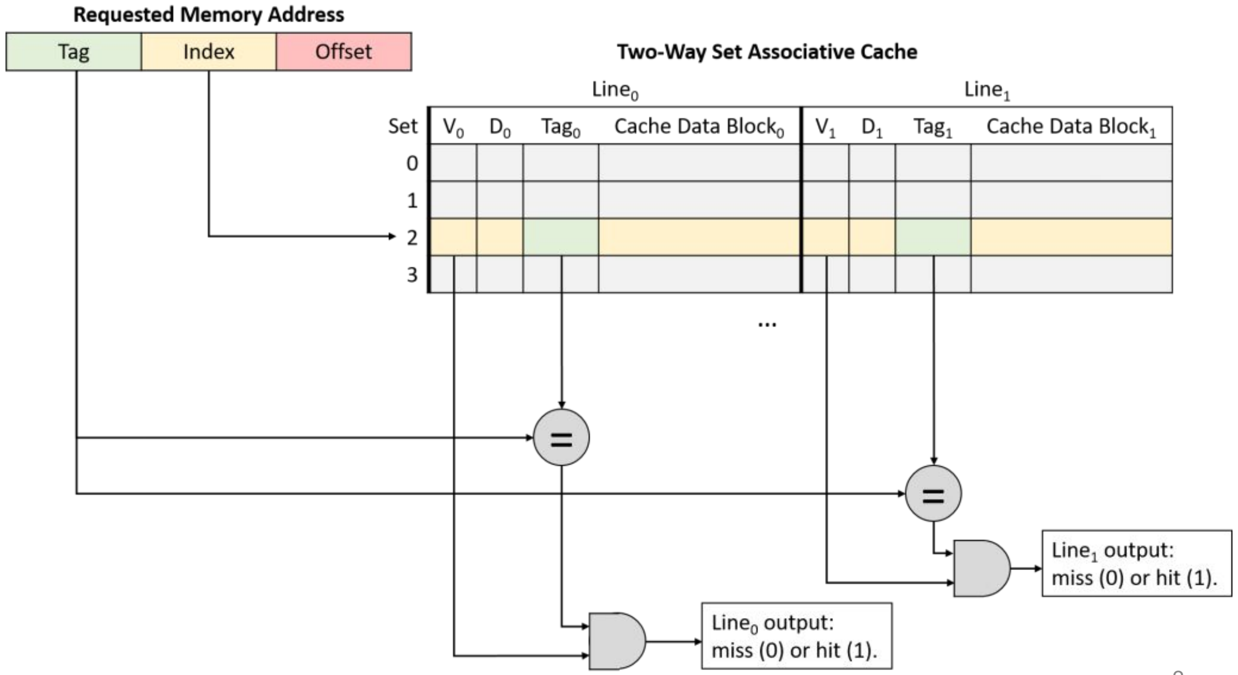
Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design

N-Way Set Associative Cache

One-to-one mapping between Tag Store and Data store

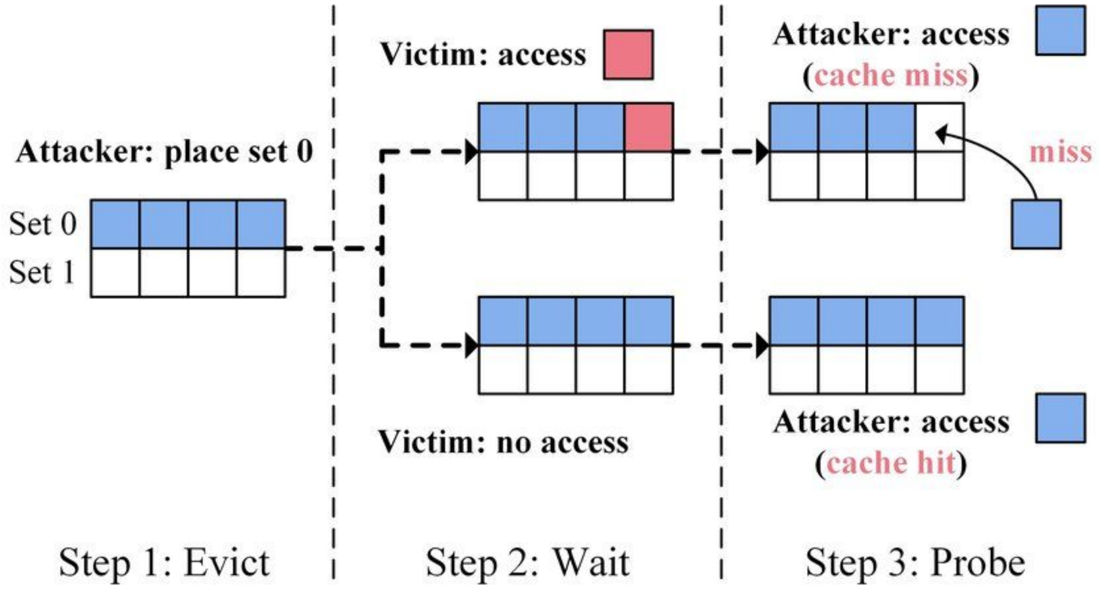
Blocks with same index can be placed in one of only N locations

On cache full, the new block must replace an older block from one of the N locations



Background

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



Prime: Attacker occupies cache sets to evict victim's data and set up a known cache state.

Wait: Victim uses the cache, potentially altering its primed state without attacker interference.

Probe: Attacker measures access times to identify cache hits or misses, revealing victim's cache use.

UCLA Samueli School of Engineering

Proposed Idea

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

One-to-one mapping between Tag Store and Data store

Blocks with same index can be placed in one of only N locations

On cache full, the new block must replace an older block from one of the N locations

Multi-**I**ndex **Ra**ndomized Cache with **G**lobal **E**victions

MIRAGE

Proposed Idea

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

Main Idea -

Decouple tag-store and data-store

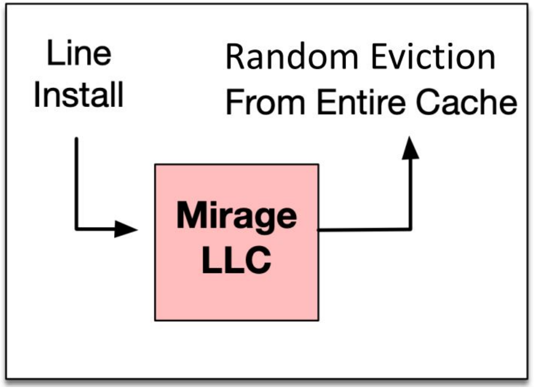
Decouple line install and line replacement decisions

Introduce randomness to reduce predictability of access patterns

Main Challenge -

A practical cache lookup for all cache lines

Optimizing additional storage and logic overheads



UCLA Samueli School of Engineering

Implementation

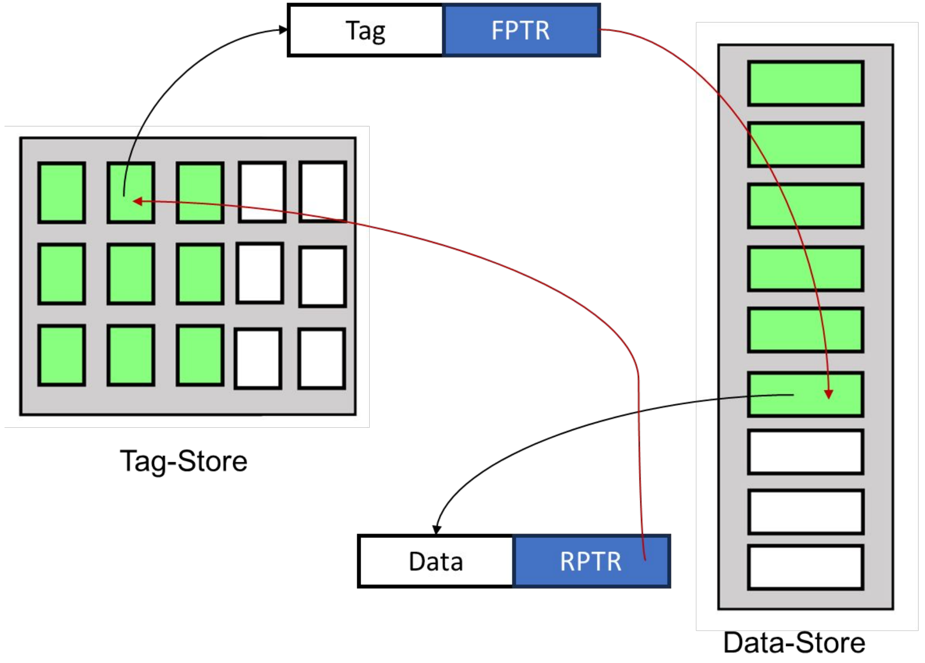
Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design

Decouple tag-store and data-store

Each tag-store entry has the **Forward Pointer (FPTR)** to the location in data-store where corresponding data is stored.

Each data-store entry has the **Return Pointer (RPTR)** back to the tag-store where its metadata is stored.

Each tag entry can be arbitrarily mapped to any available data location, breaking off the *always-mapped-to-one* approach.



Implementation

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

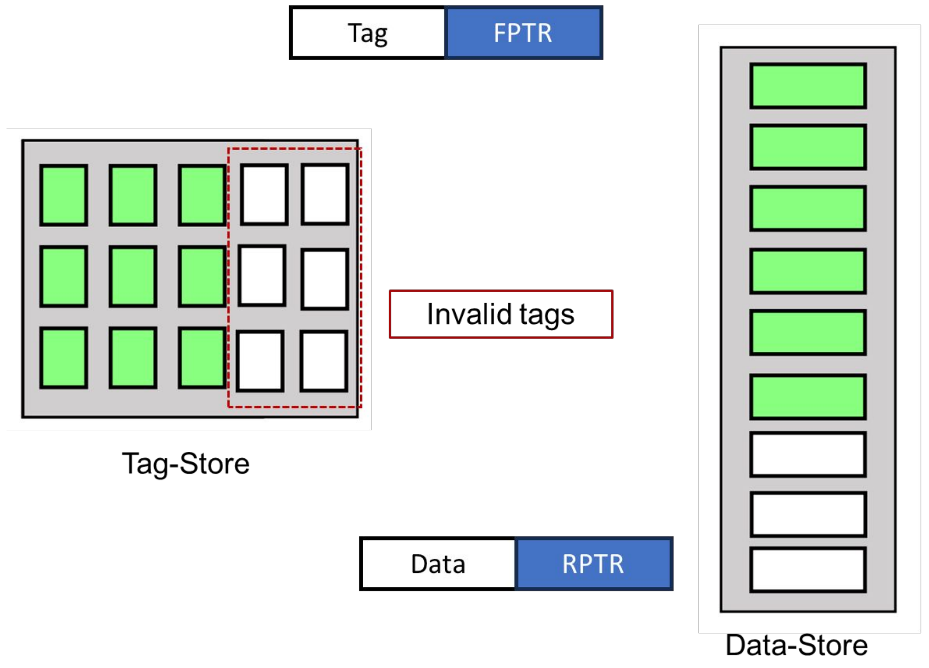
Decouple line install and line replacement

Inexpensive to have more tag store entries than data store entries.

Tag-Store increased in size to have extra *invalid tags*, to hold metadata of new lines.

On installing a new line, if invalid tag is available (most of the times), then new line is installed in that tag-store eliminating the need for an eviction.

Data-store entries that have been invalidated on previous evictions are pointed by invalid tags.



UCLA Samueli School of Engineering

Implementation

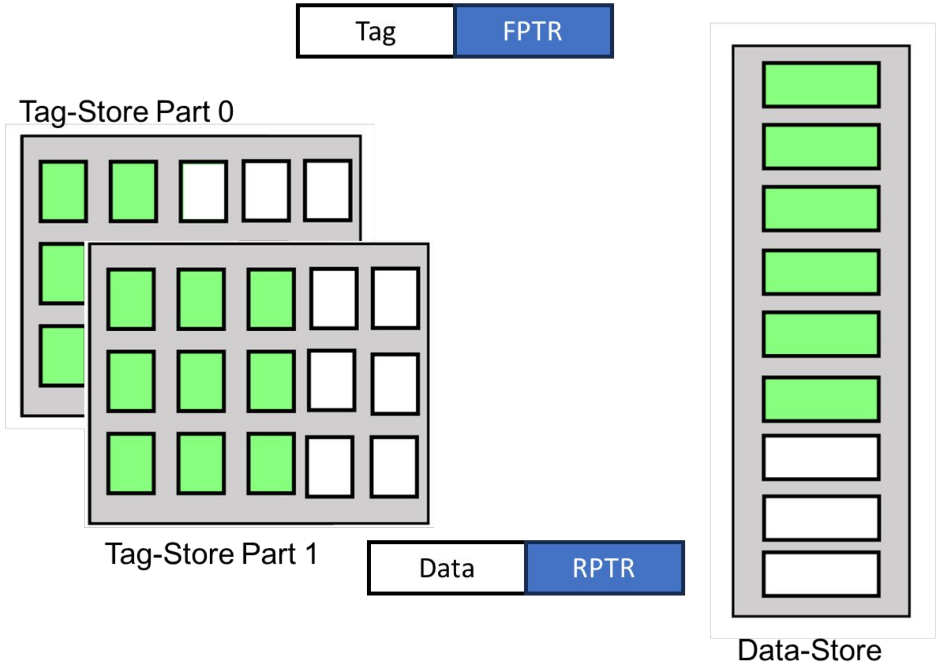
Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design

Introduction of Randomness

Tag-Store is split into two equal partitions (or skews), allowing a new line to map to two sets in the cache completely randomly.

Hash functions are used to map addresses to sets in each partition.

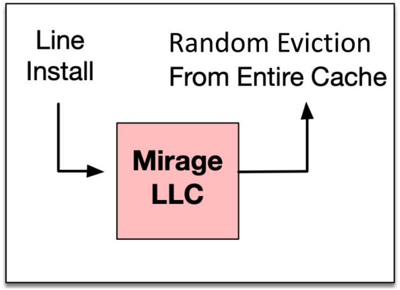
Both skews have invalid tags present, the one with more invalid tags is selected to have the new line installed in, maximizing chances of no eviction.

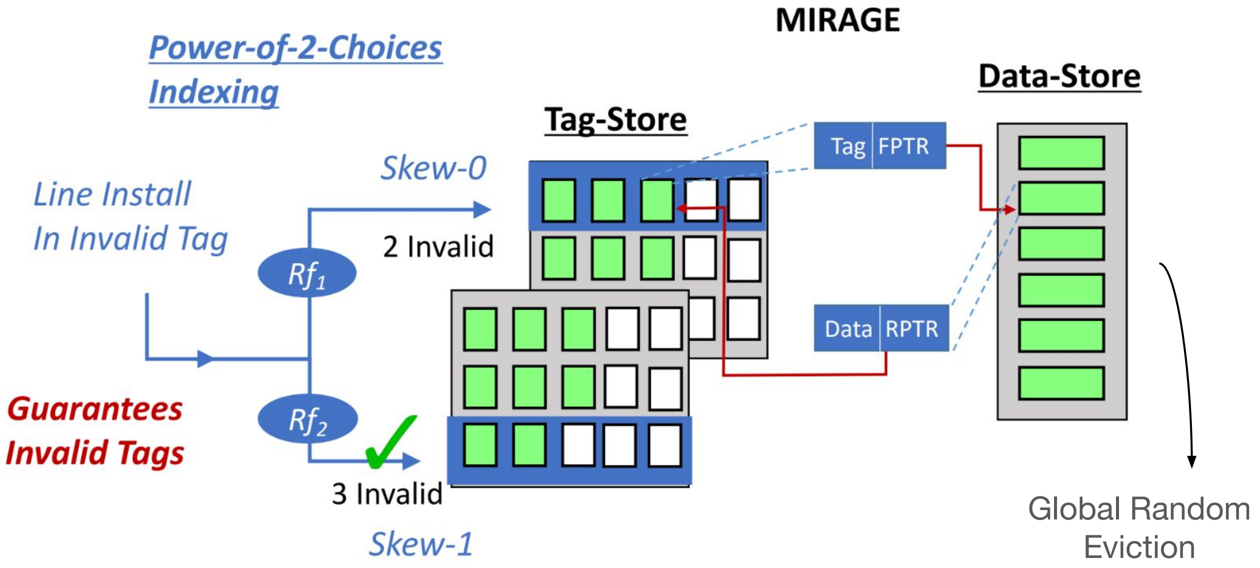


UCLA Samueli School of Engineering

Implementation

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





Performance Analysis

Performance	Impact	Analysis
Attacks	**~ Never**	**1 Eviction per 10^{17} years (lifetime)**
Latency	3 extra cycles	1. Hash computation for index 2. Cache lookup for extra ways 3. Accessing data store at FPTR
Cache Misses	2.4% more	Baseline LRU preserves addresses likely to be reused soon
Storage	20% more	Storage overheads due to extra invalid tags and FPTR/RPTR pointers in each entry
Power	18 - 20% more	Static Leakage power increases due to increased storage; minimal increase in dynamic
Hardware	35k GEs	Extra logic for set-index hash computation, skew selection, pointers and overall FSM

UCLA Samueli School of Engineering

Problems

So many assumptions.

1. Hash function is truly random
2. Secret keys are privately and perfectly managed
3. Even a single eviction is security vulnerability

Too high of a cost.

1. High Latency and more misses in cache accesses
2. More storage, power, area
3. Not efficient for smaller speed-critical systems

Suggestions

1. **Independence from Quality of Randomness**

   The security of *Mirage* largely relies on the degree of randomness and the quality of hash functions used in mapping addresses to tags in the two skews. Better randomness would be achieved by advanced encryption mechanisms which would add to the overhead, and easier-to-implement mechanisms carry the risk of being broken into by the adversaries, defeating the whole purpose.

2. **Scalability and Compatibility**

   Integrating *Mirage* into existing hardware architectures is challenging due to its unique tag-store and data-store requirements. Moreover, secure key management might not be possible in all devices due to different constraints.

3. **Isolation and Randomization**

   Further research on defending against similar attacks have suggested randomization and isolation as two directions of research. The existing Mirage design can be made simpler to implement with less overheads if certain isolation strategies are implemented, perhaps it might be easier to integrate with existing systems.

References

G. Saileshwar and M. Qureshi, “MIRAGE: Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design,” 30th USENIX Security Symposium (USENIX Security 21), pp. 1379-1396. 2021.

K. Wang, F. Yuan, L. Zhao, R. Hou, Z. Ji, and D. Meng, “Secure hybrid replacement policy: Mitigating conflict-based cache side channel attacks,” Microprocessors and Microsystems, vol. 89, p. 104420, Mar. 2022, doi: 10.1016/j.micpro.2021.104420.

W. Song, B. Li, Z. Xue, Z. Li, W. Wang, and P. Liu, “Randomized Last-Level Caches Are Still Vulnerable to Cache Side-Channel Attacks! But We Can Fix It,” in 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA: IEEE, May 2021, pp. 955–969. doi: 10.1109/SP40001.2021.00050.

A. Purnal, F. Turan, and I. Verbauwhede, “Prime+Scope: Overcoming the Observer Effect for High-Precision Cache Contention Attacks,” in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 2906–2920. doi: 10.1145/3460120.3484816.

Q&A

UCLA Samueli School of Engineering