CASPER

CS773-2025-Spring: Computer Architecture for Performance and Security

Lecture 6: Let’s save the cache ☺

https://www.cse.iitb.ac.in/~biswa/

ON SILENT MODE PLEASE

Quiz-1 coming in few days

- Monday, 7 PM
- Please go through all that we cover till tonight
- PA1, final part will be released next Thursday. Vivas and all will be in February (1 week after the PA1, final part).
- February 13 is the deadline
- Folks who were absent on hands-on, and did not email, we wont be able to evaluate your assignment-1.

Three Pertinent Attacks

- Flush-based
- Conflict-based
- Occupancy based

Let’s mitigate all: One Step at a Time

Flush the clflush

- Make it privileged
- Restrict it to private data only, but not for shared data ☺
- For shared data, make it privileged, it will make it persistent memory programmers happy

Let’s do some more, before going deep

- Let’s fudge the timer

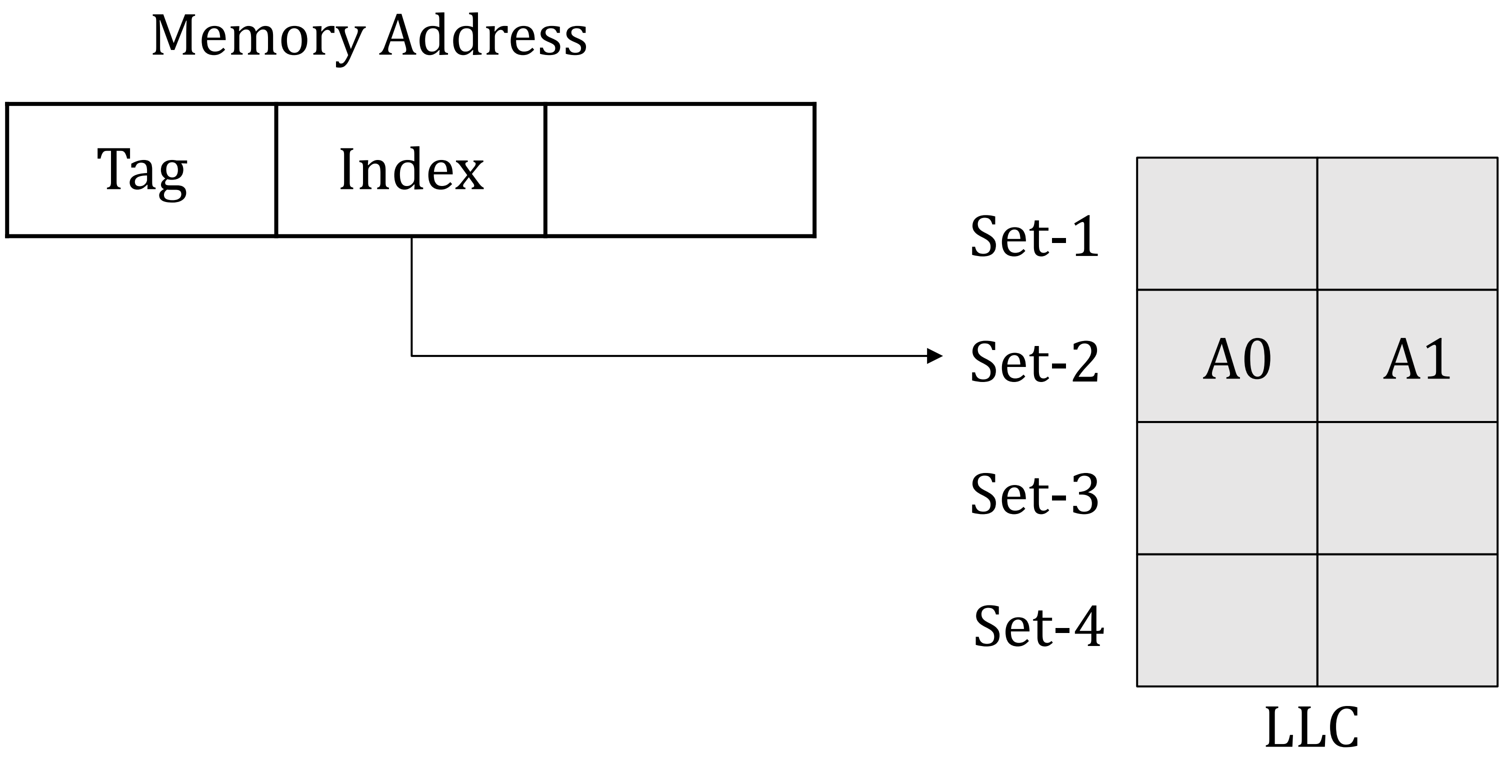
- Let rdtsc returns noisy values ☺

- How to do it? Add an epsilon to rdtsc ☺ ☺

Conflict Based Attacks (Evict+Reload, Prime+Probe)

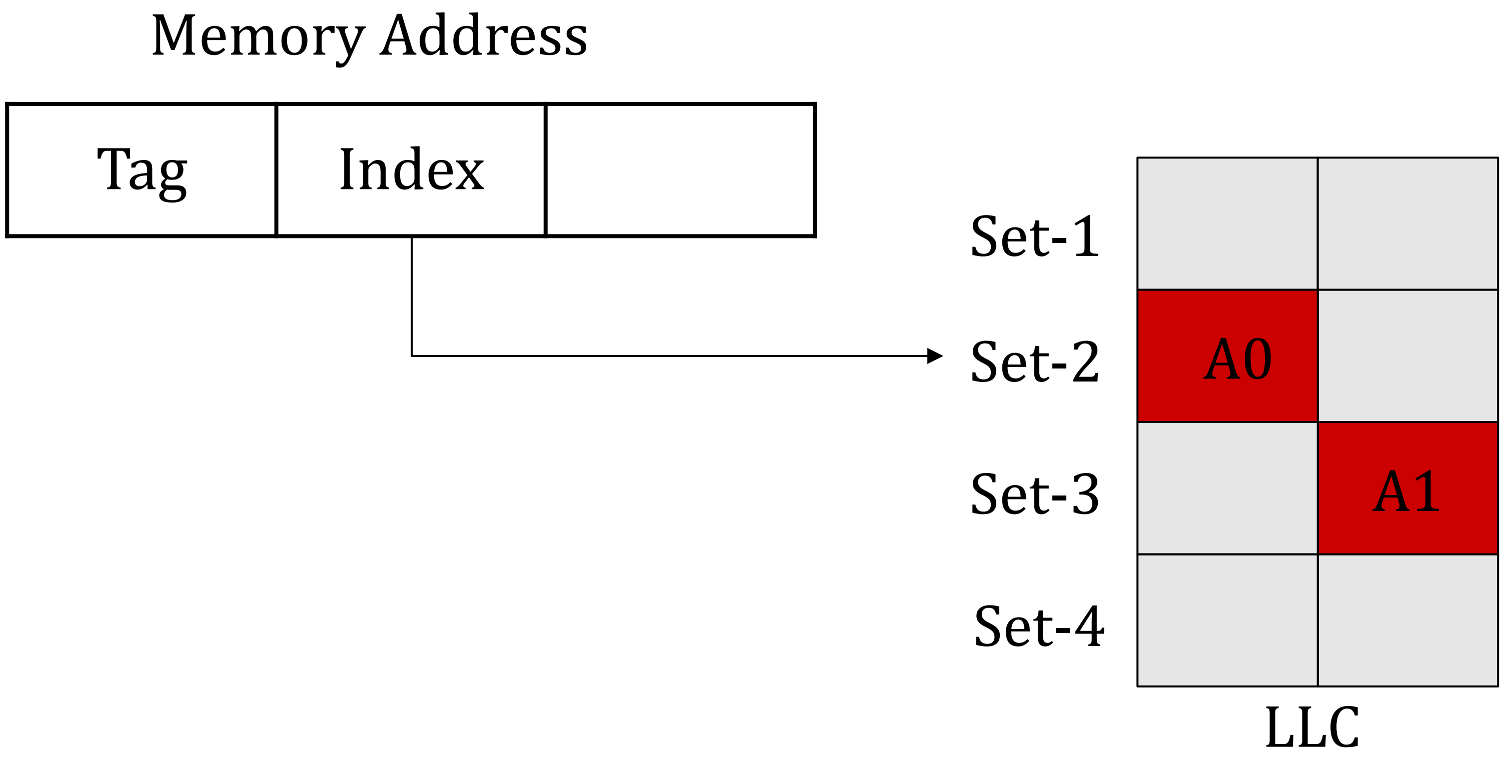
- Make creation of eviction set difficult
- How?
- Option-I: Randomized caches

Deterministic mapping



- A memory address is always mapped to the **same set**

Randomized mapping



Memory Address

Tag

Index

Set-1

Set-2

Set-3

Set-4

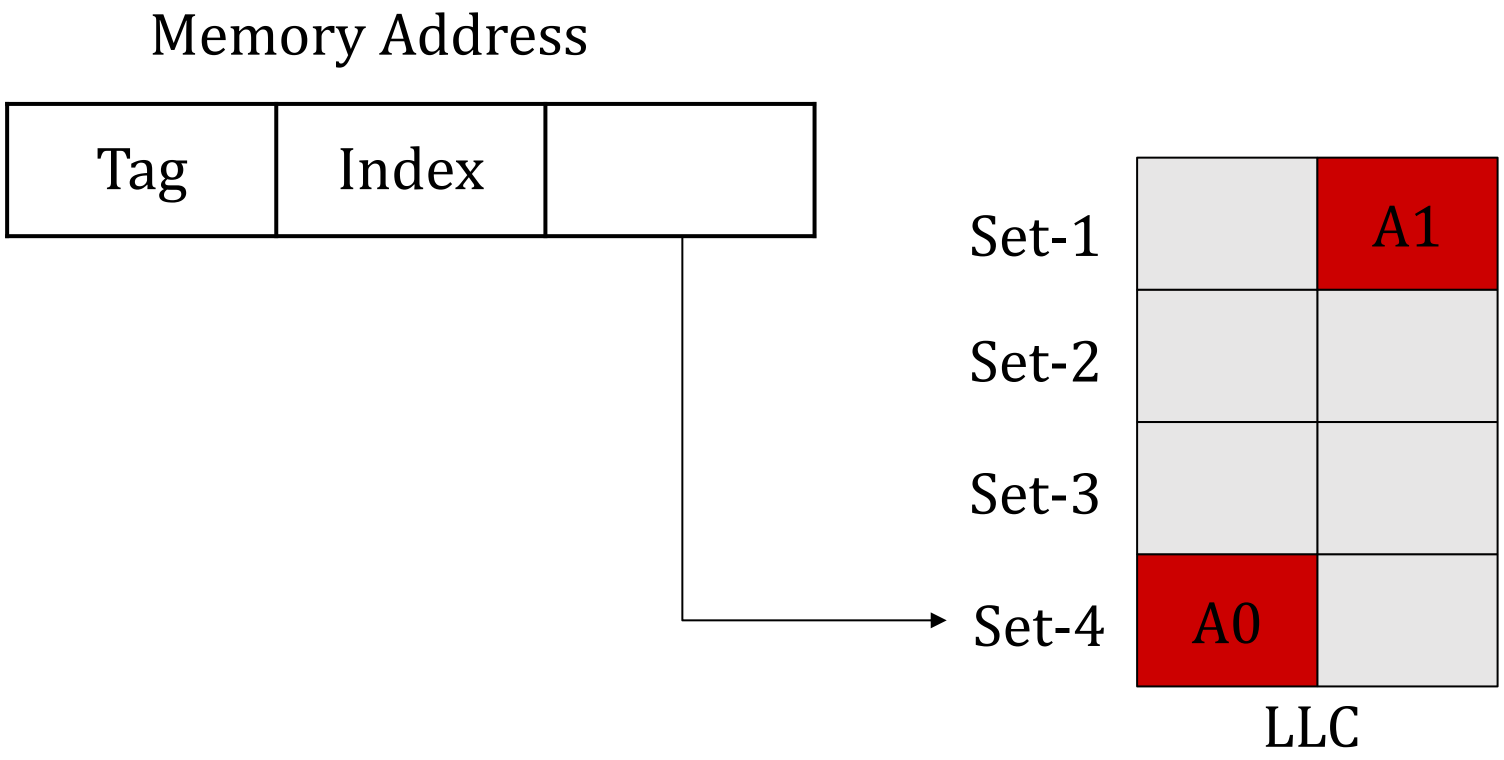
A0

A1

LLC

- A memory address is mapped to **random set**

Randomized mapping



- A memory address is mapped to **random set**
- **Non-deterministic mapping**

Randomized caches

CEASER [MICRO 2018]

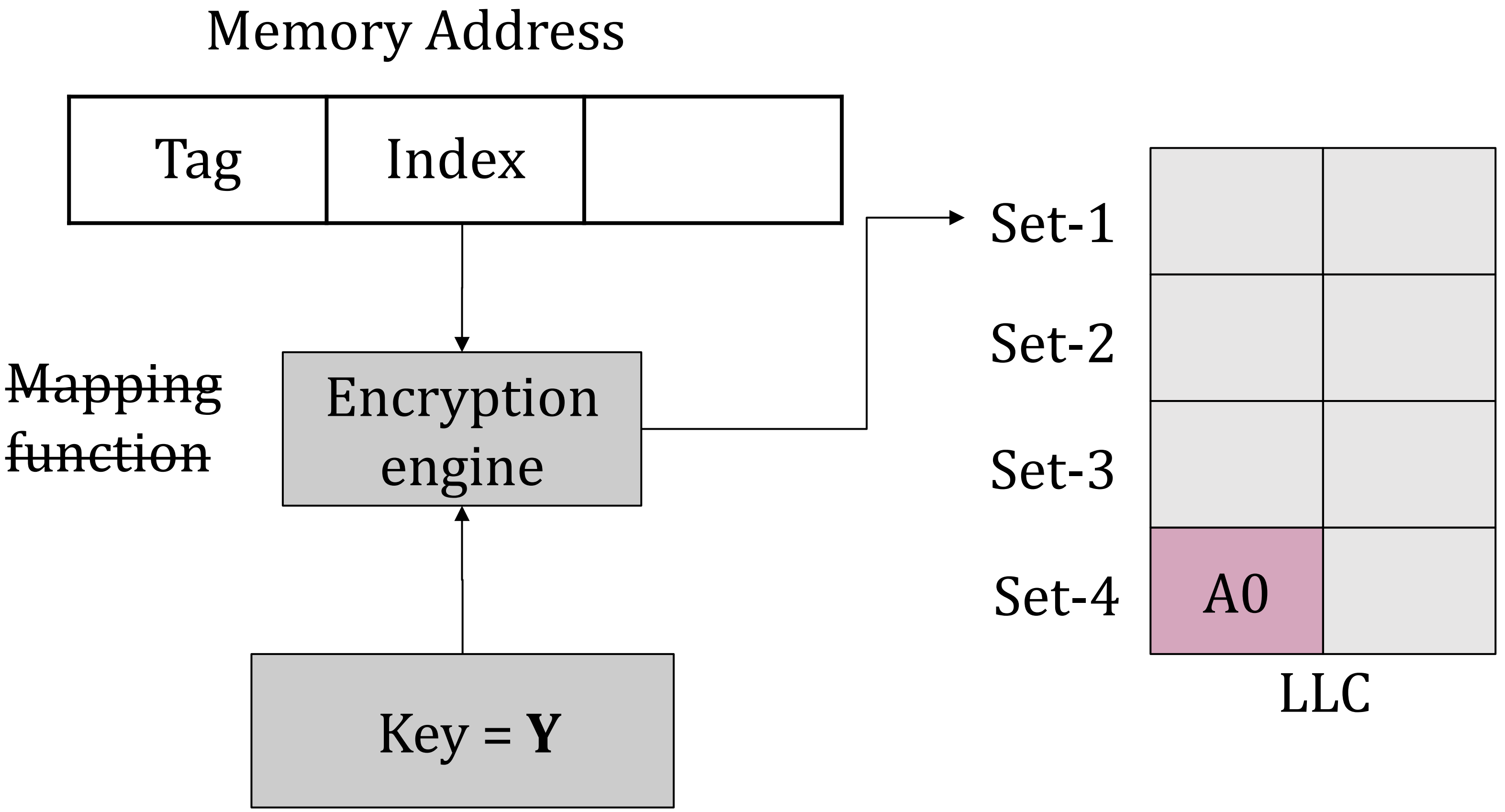
MIRAGE [USENIX Security 2020]

CEASER-S [ISCA 2019]

MAYA [ISCA 2024] (IITB, CASPER) ☺

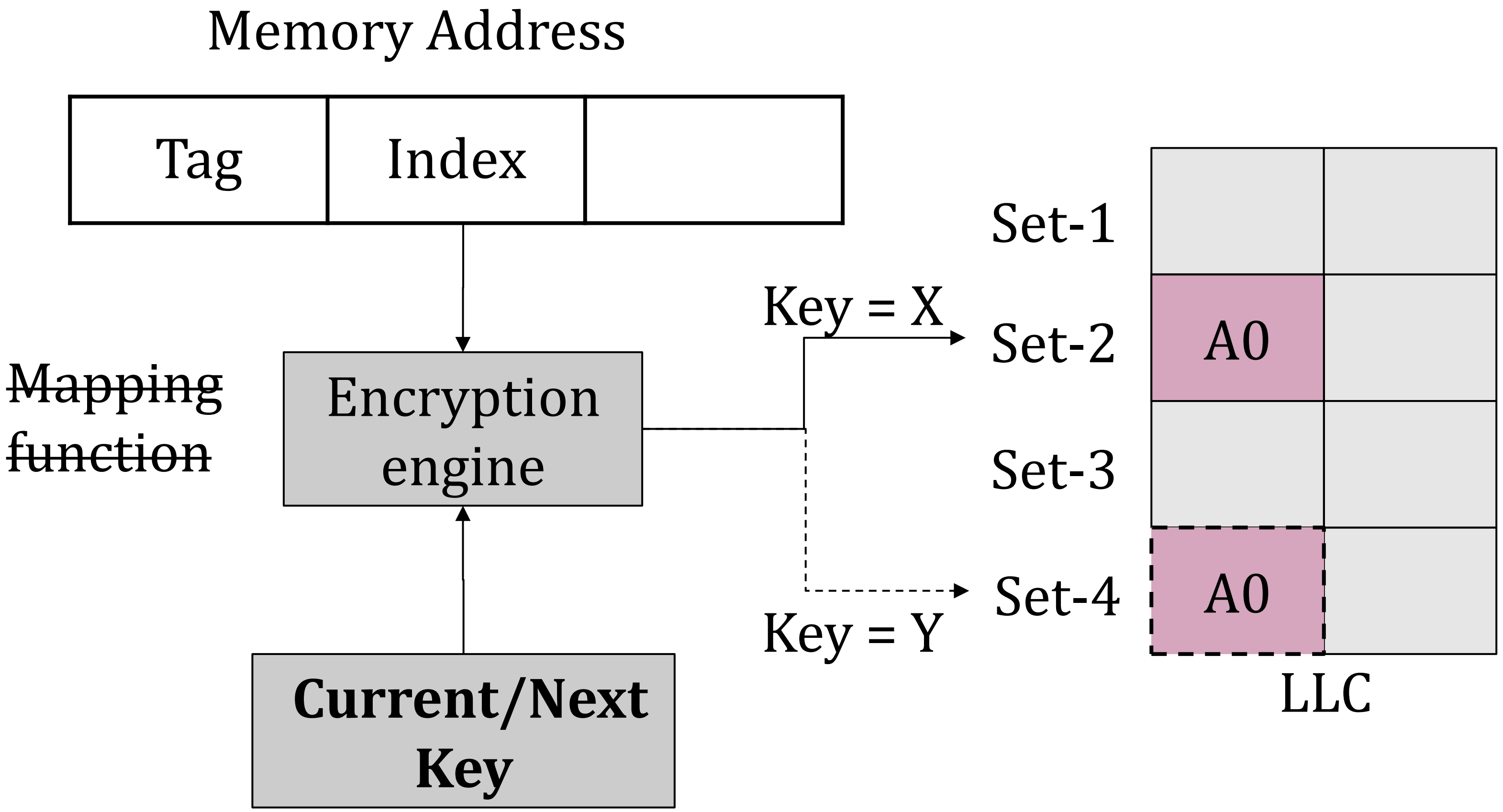
ScatterCache [USENIX Security 2019]

Randomized LLC: CEASER



- **Key is periodically changed** to provide randomized mapping

Randomized LLC: CEASER



Memory Address

Tag

Index

~~Mapping function~~

Encryption engine

Current/Next Key

Key = X

Key = Y

Set-1

Set-2

Set-3

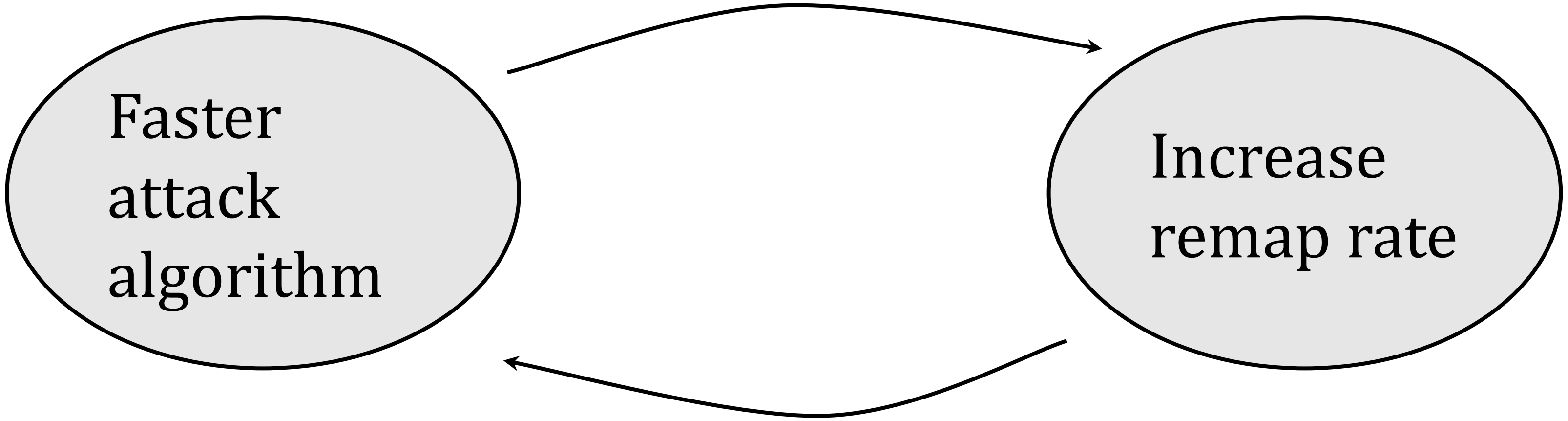
Set-4

A0

A0

LLC

Randomized caches: limitations

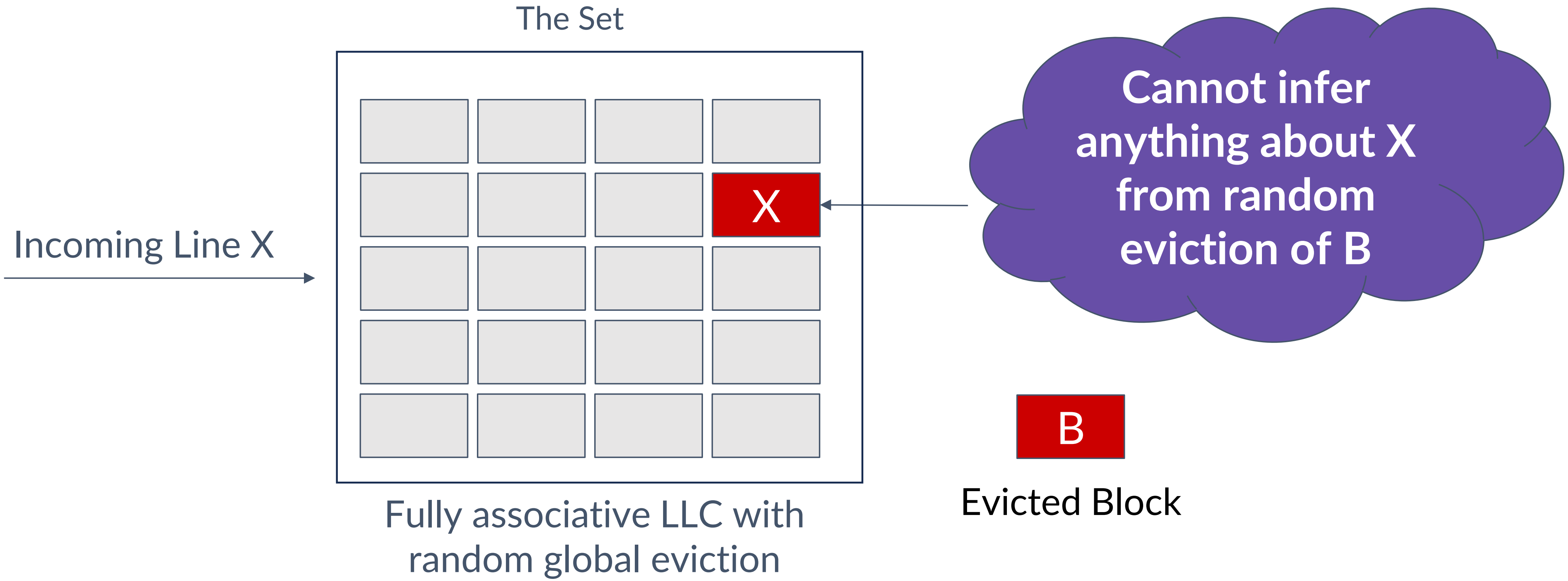


Faster attack algorithm

Increase remap rate

Randomized caches can not guarantee full security as applications contend for the same shared resource

Fully Associative Cache is the answer, but impractical


The Set
Incoming Line X
X
Cannot infer anything about X from random eviction of B
B
Evicted Block
Fully associative LLC with random global eviction

MIRAGE and MAYA provide an illusion of a fully associative cache, with randomization

Final Deal: Partitioning

- Summary so far:

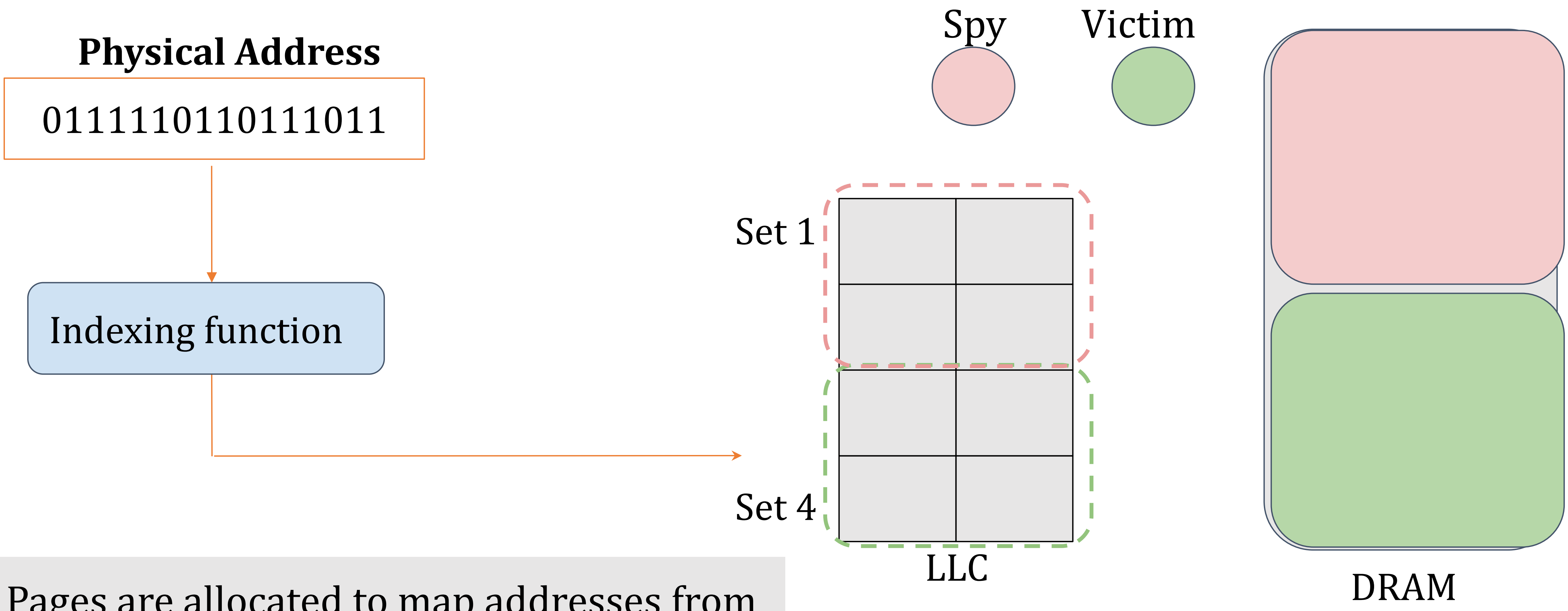
Randomization mitigates conflict-based attacks but not occupancy-based.

Performance overhead with randomization is small, as it does not affect the cache space.

Partitioning can provide complete isolation and hence security.

The performance will be a concern though ☹

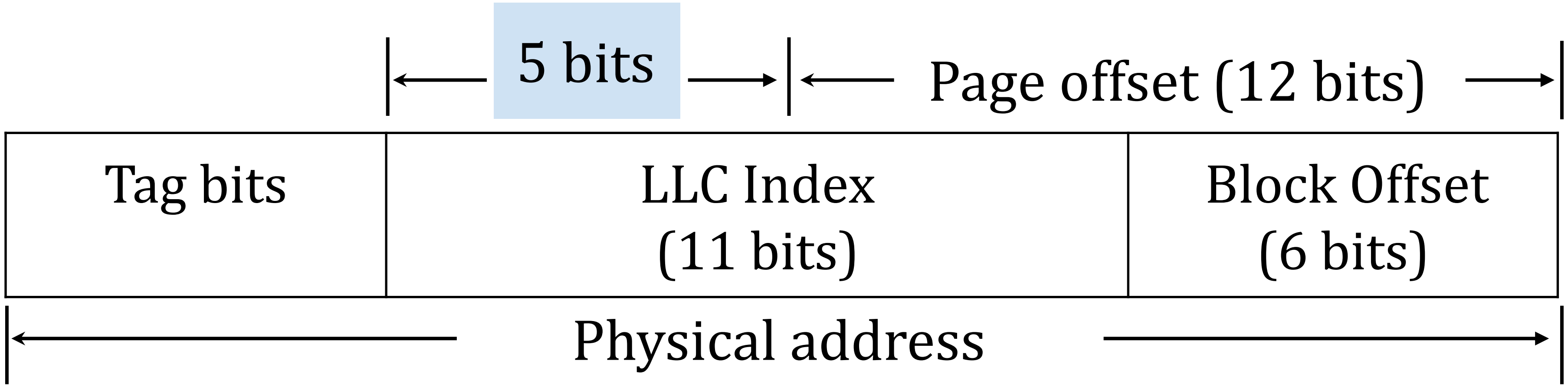
Approach-I: Page-Coloring



Pages are allocated to map addresses from different cores to disjoint sets in LLC

LLC: Color bits

- LLC: 2MB, 16 way (2048 sets)
- Page size: 4KB
- Cache line size: 64B



5 bits

Page offset (12 bits)

Tag bits

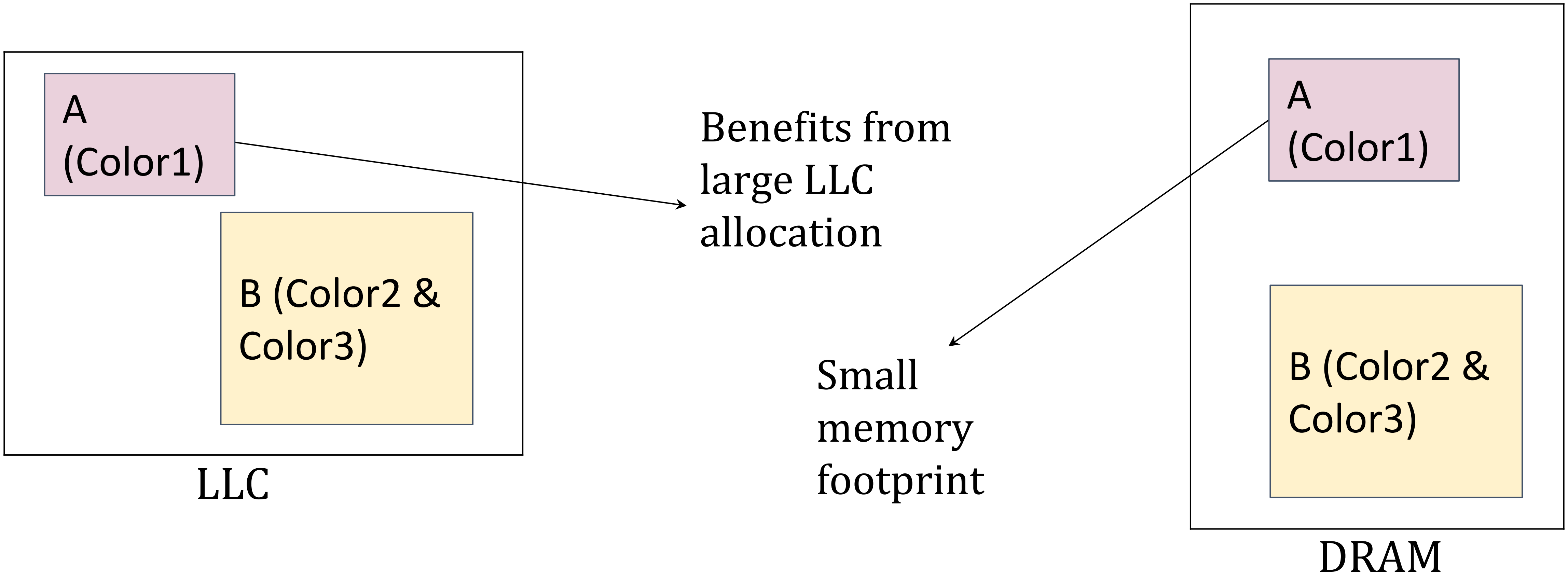
LLC Index
(11 bits)

Block Offset
(6 bits)

Physical address

32 regions can be created in LLC of 64KB each

Page-Coloring: limitation



A (Color1)

B (Color2 & Color3)

LLC

Benefits from large LLC allocation

Small memory footprint

A (Color1)

B (Color2 & Color3)

DRAM

- Not beneficial if application’s memory footprint is not in proportion with its cache utilization

Approach-II: Way-partitioning

	Way1	Way2	Way3	Way4
Set1				
Set2				
Set3				
Set4				
Set5				
Set6				

Way-partitioning

	Way1	Way2	Way3	Way4
Set1				
Set2				
Set3				
Set4				
Set5				
Set6				

Spy	Victim
●	○

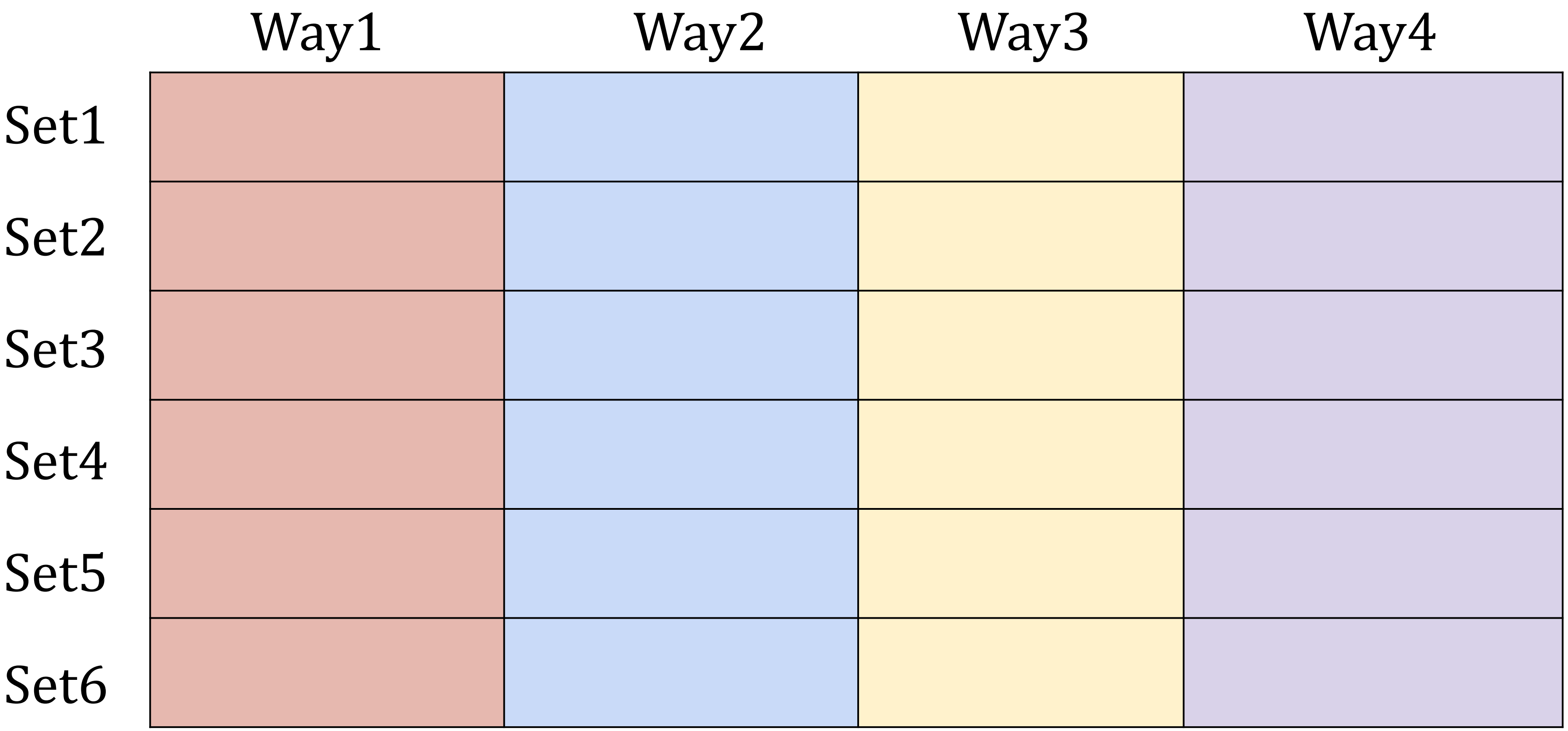
Way-partitioning

Spy 🟥 Victim 🟨

Partitioning across cache ways

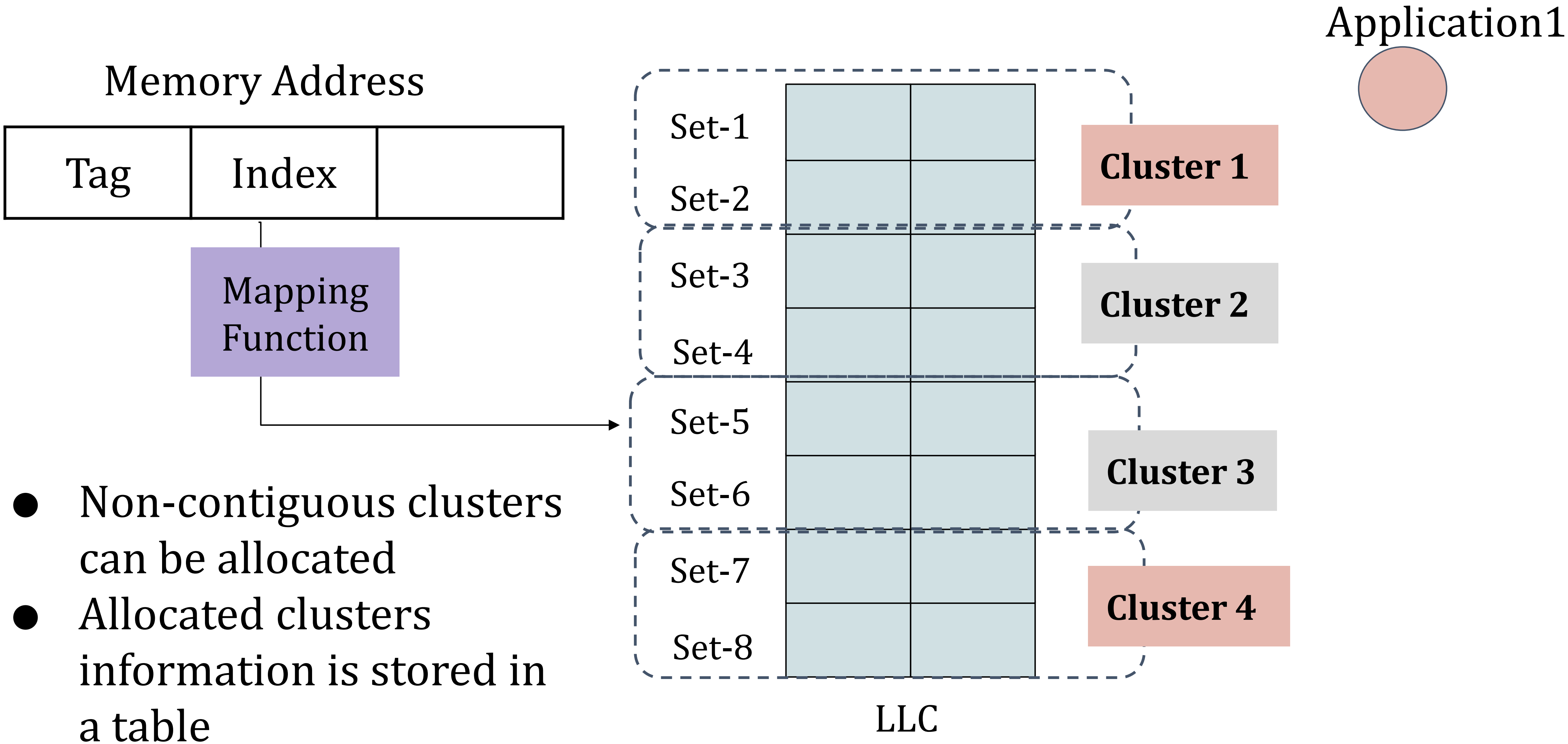
	Way1	Way2	Way3	Way4
Set1	🟥	🟥	🟨	🟨
Set2	🟥	🟥	🟨	🟨
Set3	🟥	🟥	🟨	🟨
Set4	🟥	🟥	🟨	🟨
Set5	🟥	🟥	🟨	🟨
Set6	🟥	🟥	🟨	🟨

Way-partitioning: limitation



Maximum number of isolated regions supported in LLC are #NUM_WAYS

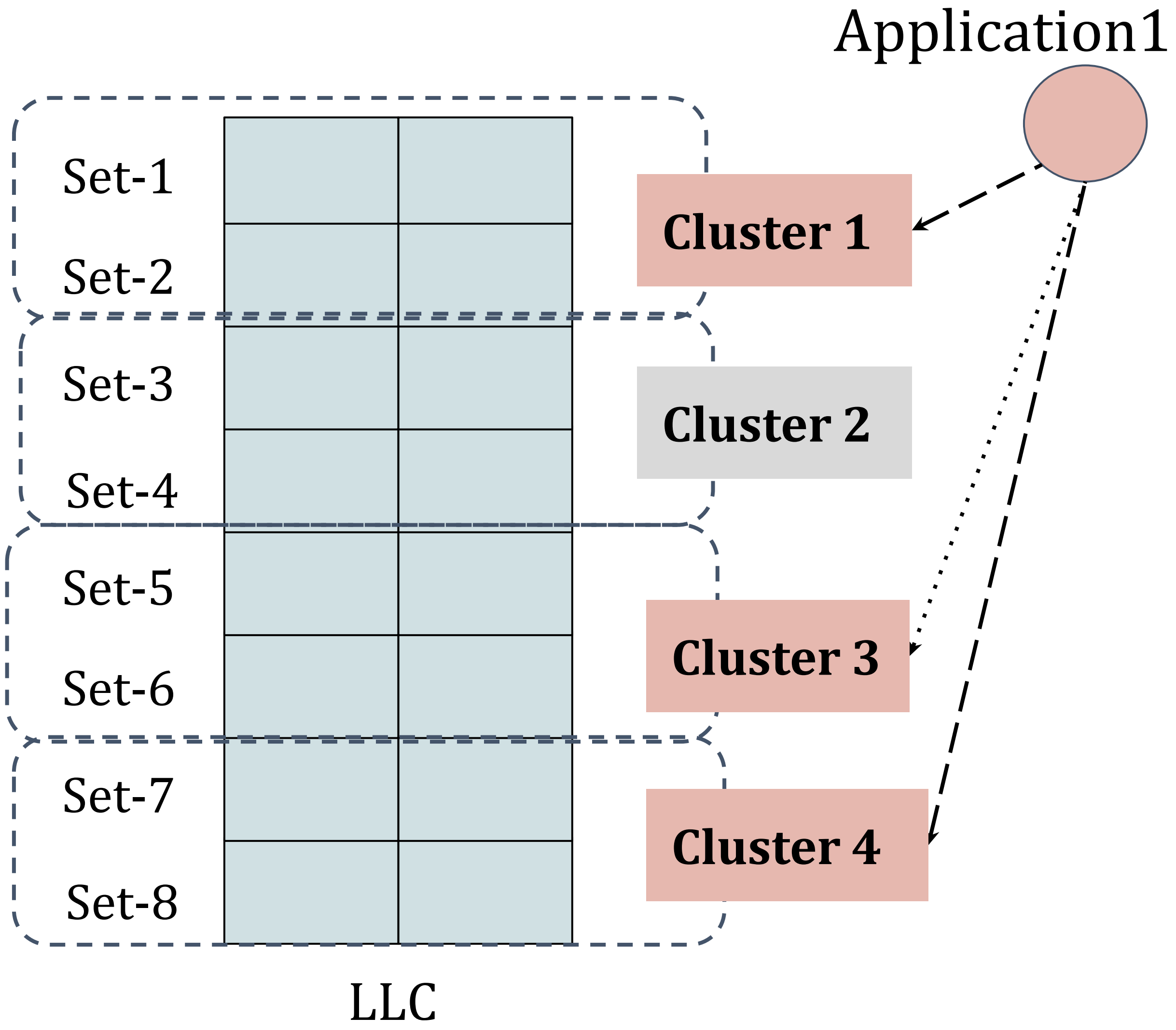
Approach III: Set-partitioning



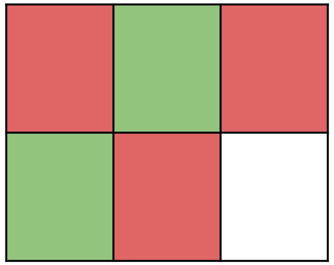
- Non-contiguous clusters can be allocated
- Allocated clusters information is stored in a table

Set-partitioning: limitations

- How many clusters to allocate ?
- For change in allocation, all allocated clusters need to be flushed.

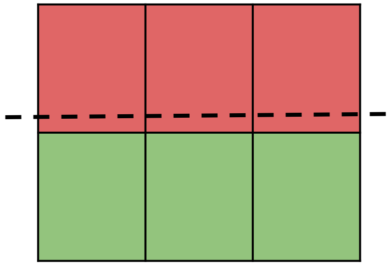


State-of-the-art mitigations and their limitations



Cache-randomization

not fully secure



Cache-partitioning

degrades performance

Summary

- Partitioning is the solution for all: Performance degradation is huge when we increase core count.

- Randomization is good but not for occupancy-based attacks.

- Flush-based attacks can be mitigated easily ☺