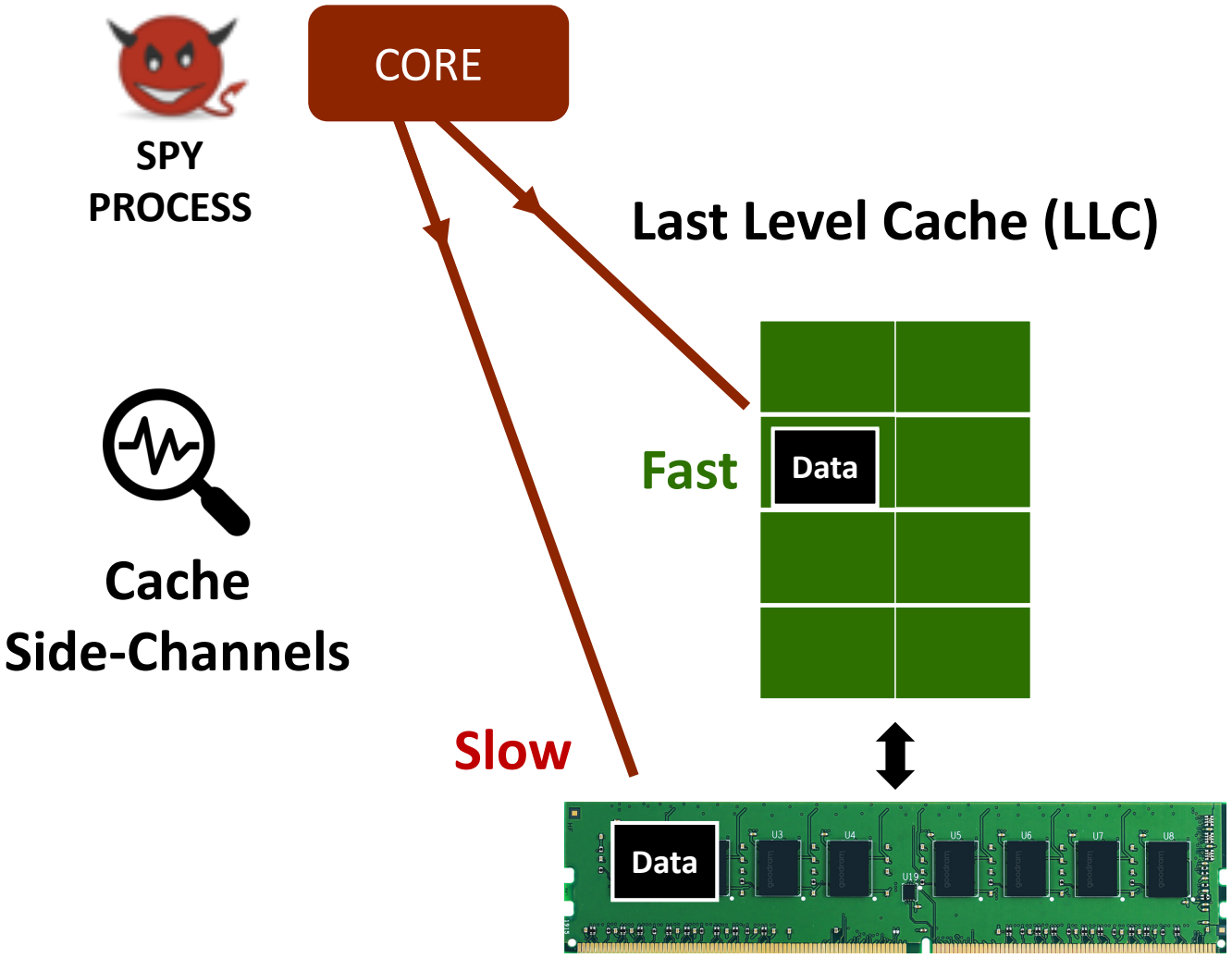
MIRAGE: Mitigating Cache Attacks with a Randomized Fully-Associative Cache

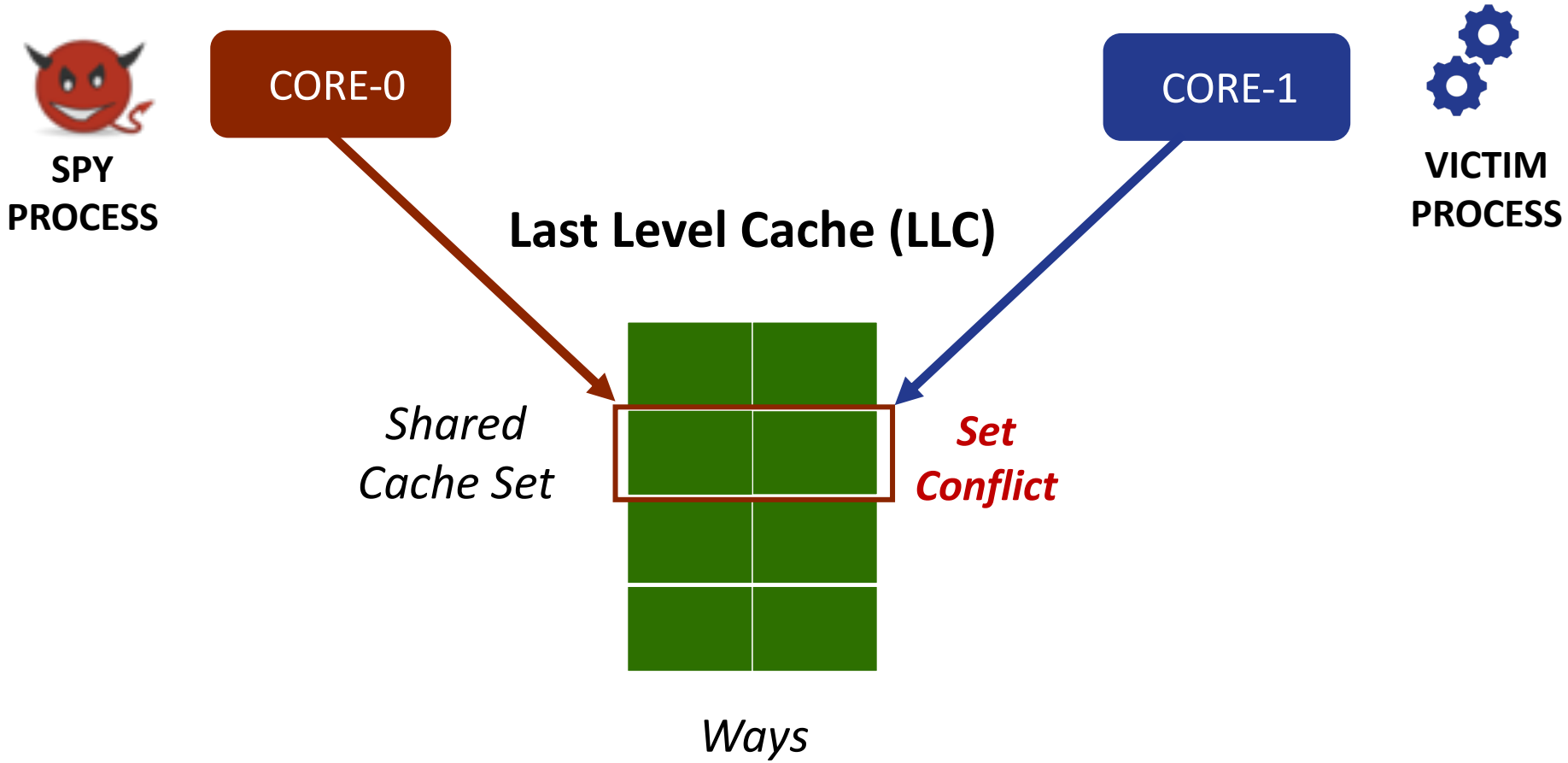
Published in USENIX Security 2021

Gururaj Saileshwar
NVIDIA Research / University of Toronto
&
Moinuddin Qureshi
Georgia Tech

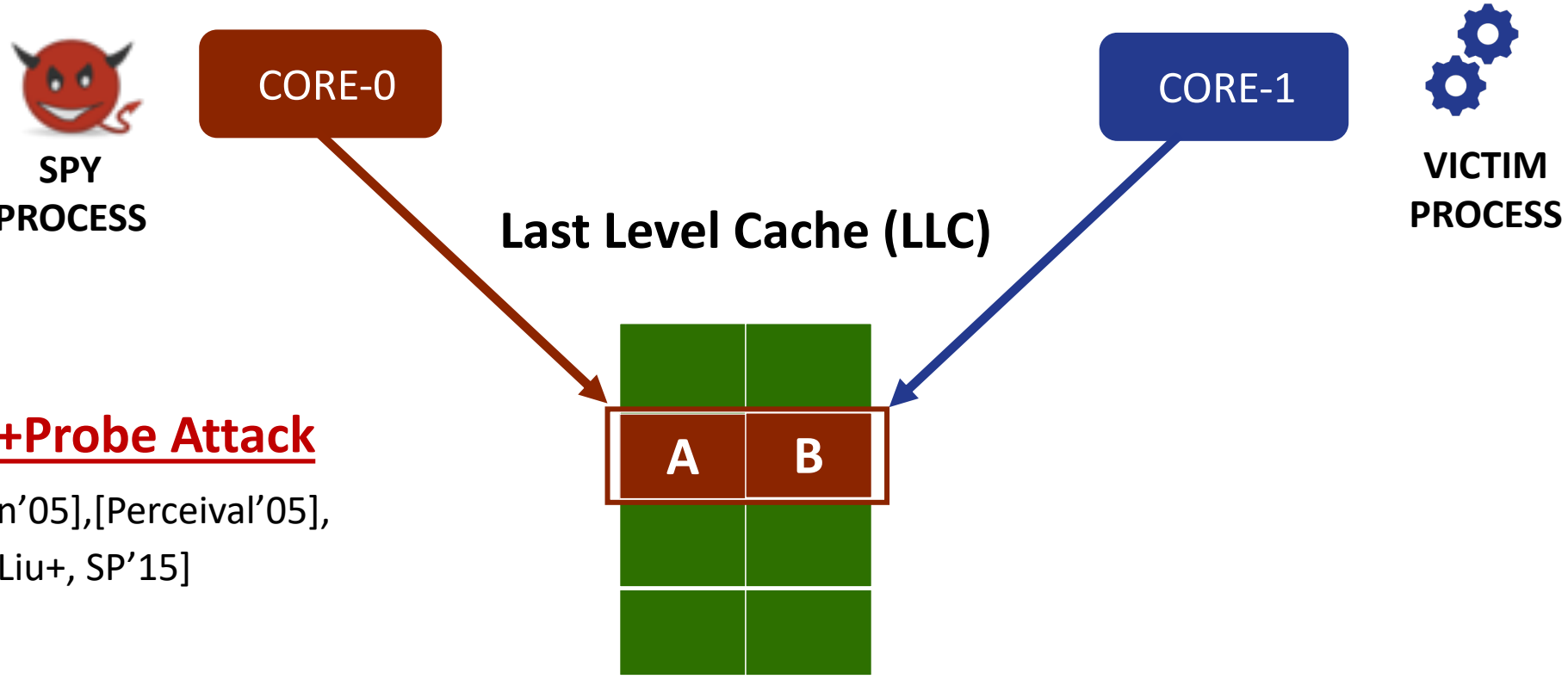
Problem: CPU Cache Side-Channels



Problem: CPU Cache Side-Channels



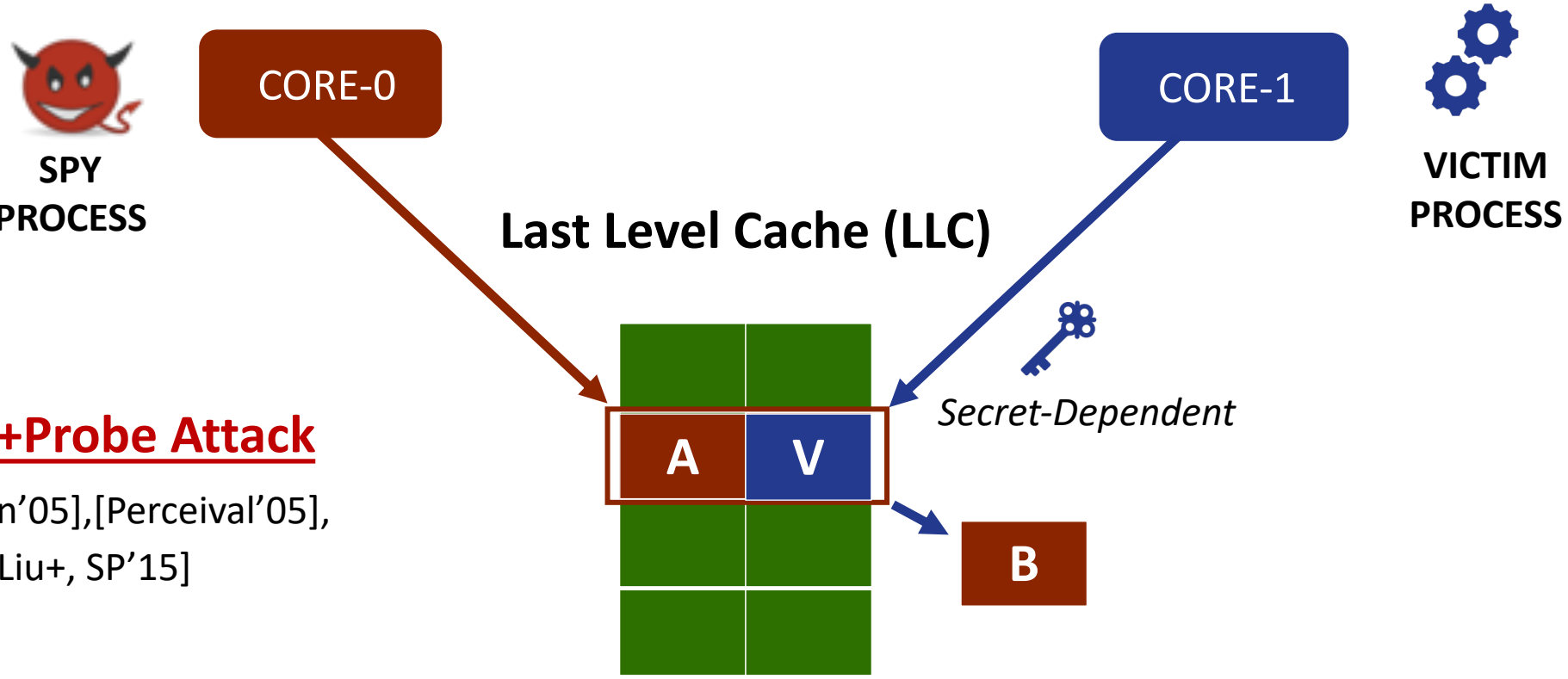
Problem: CPU Cache Side-Channels



Prime+Probe Attack

[Bernstein'05],[Perceival'05],
[Liu+, SP'15]

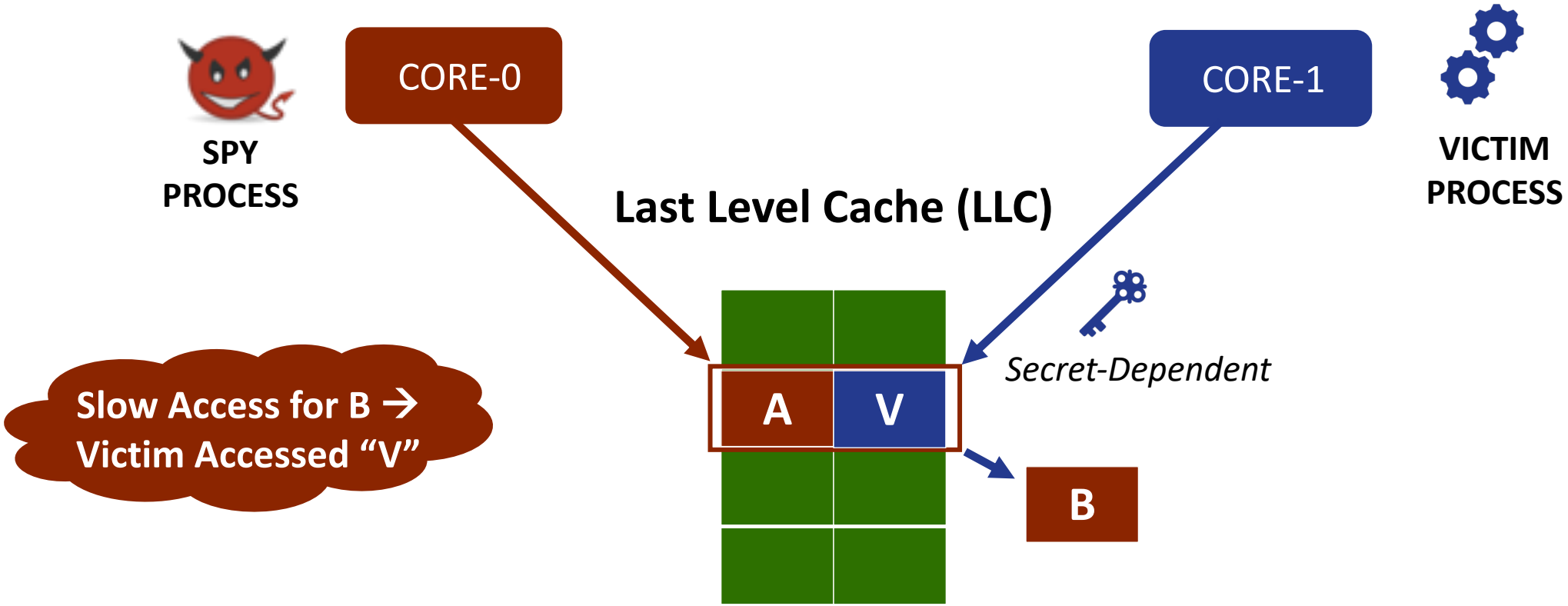
Problem: CPU Cache Side-Channels



Prime+Probe Attack

[Bernstein'05],[Perceival'05],
[Liu+, SP'15]

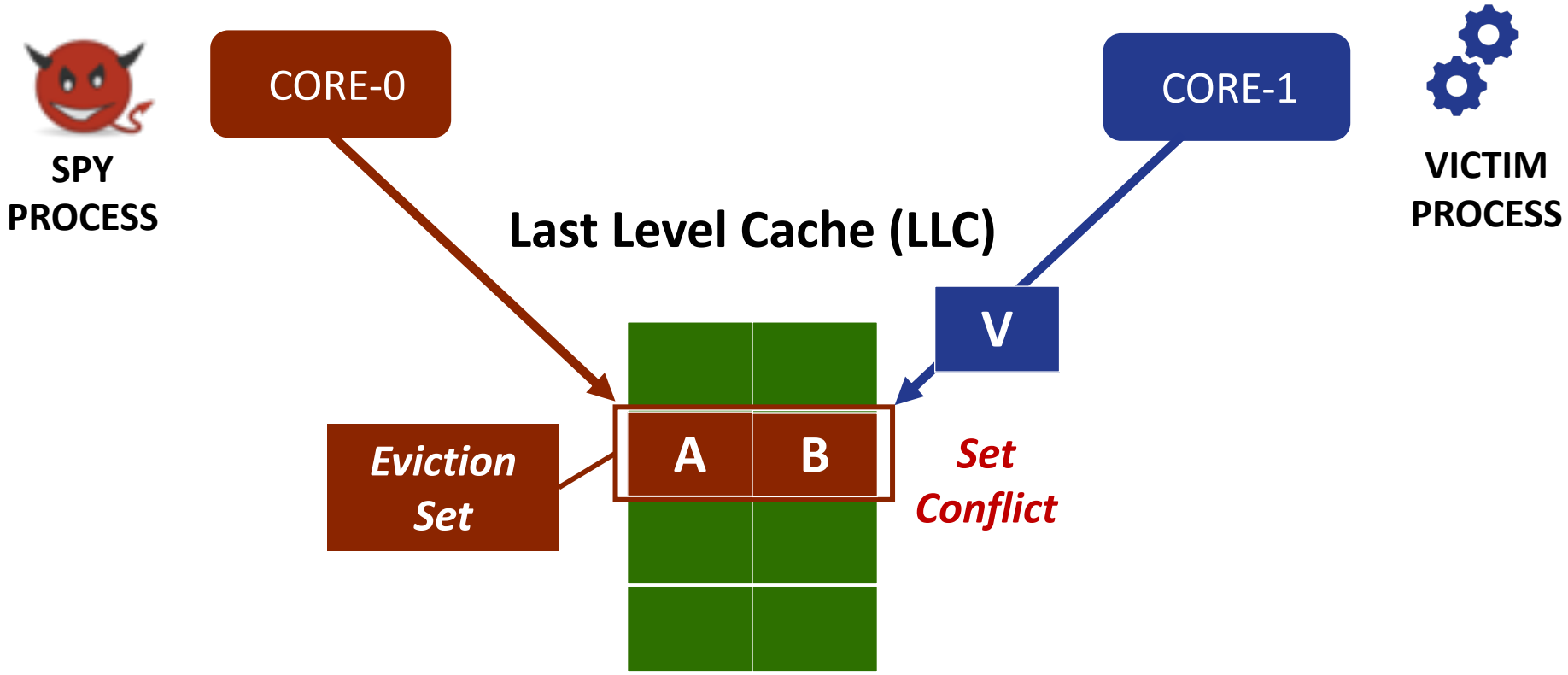
Problem: CPU Cache Side-Channels



Spy Observing Victim's Accesses Can Infer Sensitive Data
(e.g. AES Keys[1], Fingerprint Websites in Browsers[2], ML Model Architecture[3])

1 - [Bernstein'05], 2 - [Shusterman+, SEC'20], 3 - [Yan+, SEC'20], [Hong+, ICLR'20]

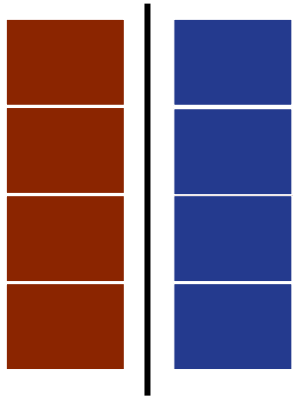
Key Requirement for Attack: Set Conflicts



Prior Defense: Partitioning and Randomization

Partitioned Cache Defense

[MICRO'18], [MICRO'19]

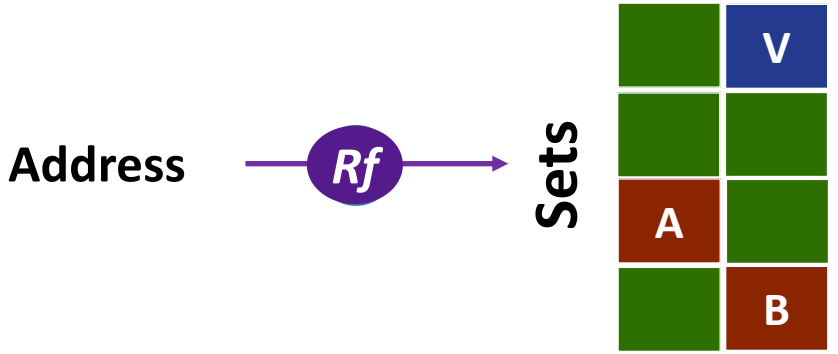


Insulate Cache Usage of Different Processes

Limited Scalability or Practicality

Randomized Cache Defenses

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

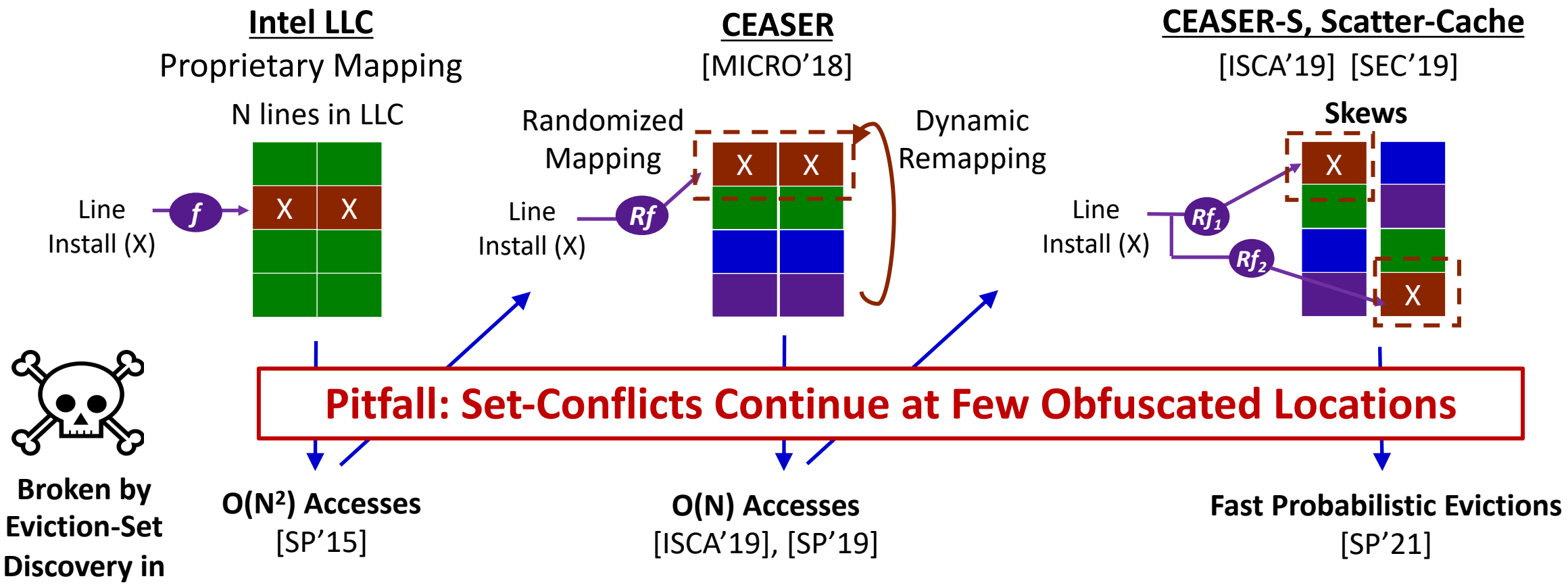


Randomized Mapping Obfuscates Set-Conflicts

Practical To Adopt, But Successive Defenses Broken

Can We Design Principled Randomization?

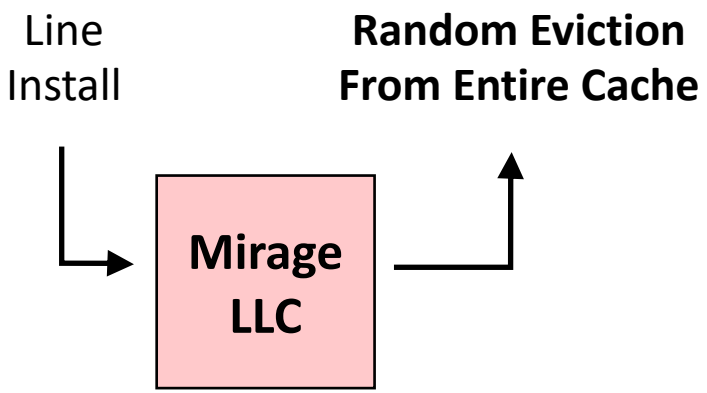
Arms Race Between Attacks & Defenses



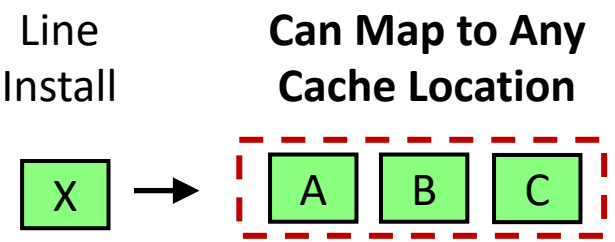
Goal: Need to Eliminate Set-Associative Evictions (Set-Conflicts)

Our Solution MIRAGE: A Fully-Associative Randomized LLC

Abstraction to SW: *Fully-Associative Randomized Cache*



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

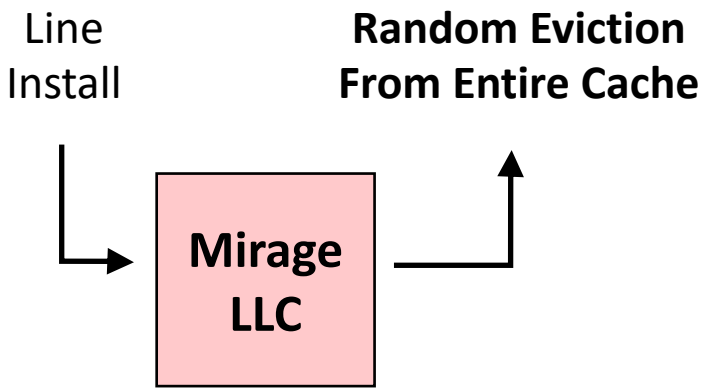


Principled Security

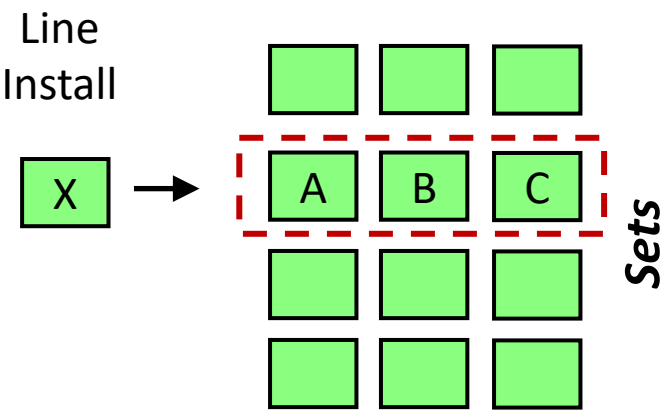
Impractical Lookup Latency & Power

Our Solution MIRAGE: A Fully-Associative Randomized LLC

Abstraction to SW: *Fully-Associative Randomized Cache*



Set-Associative Cache



Principled Security

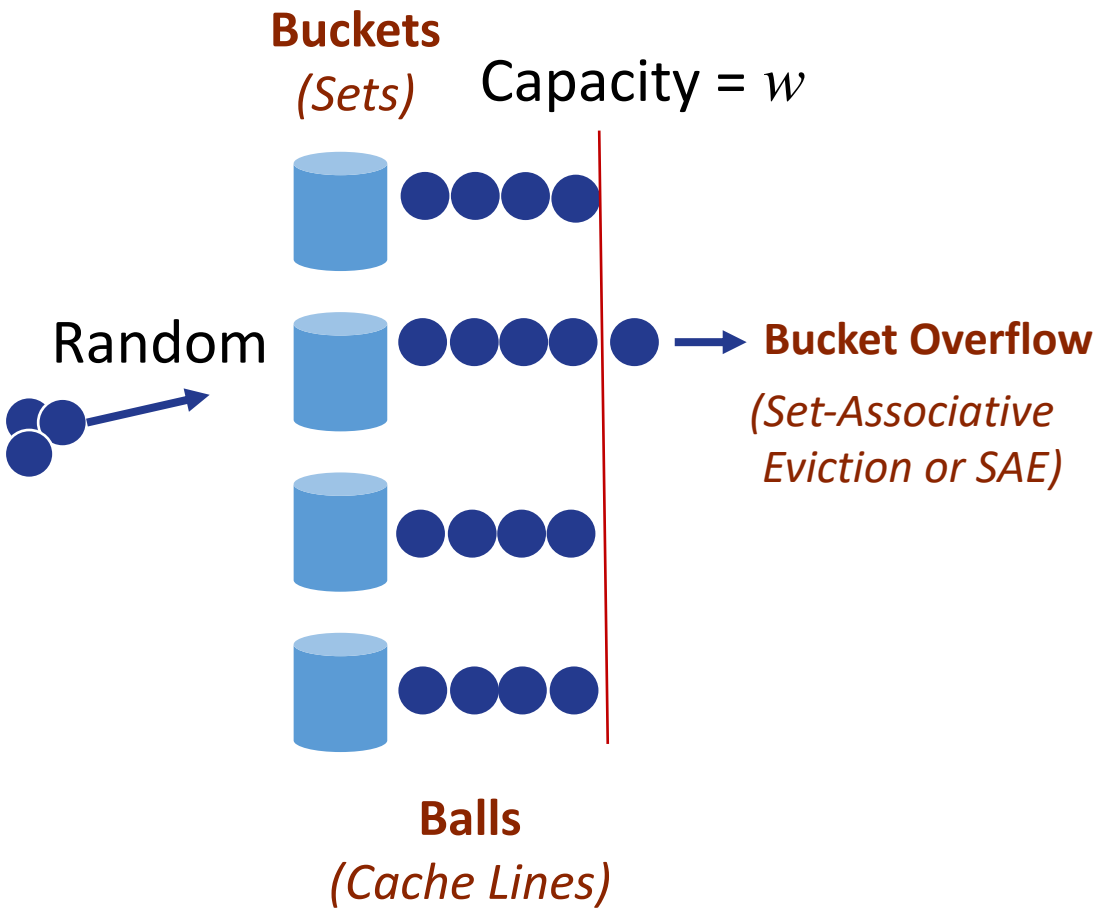
Practical Lookup within Set

(16-32 Locations)

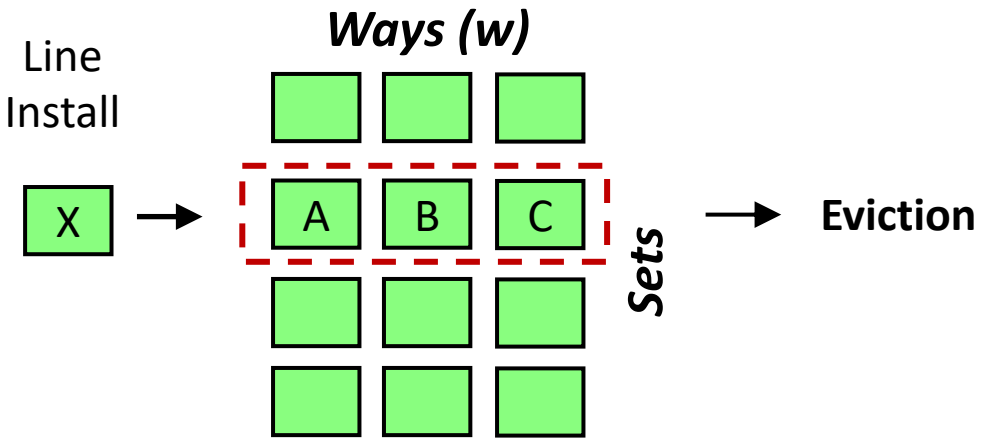
Key Challenge: How to get Security of Fully-Associative Design with Set-Associative Lookups?

Insight: Use Load-Balancing to Eliminate Set-Conflicts

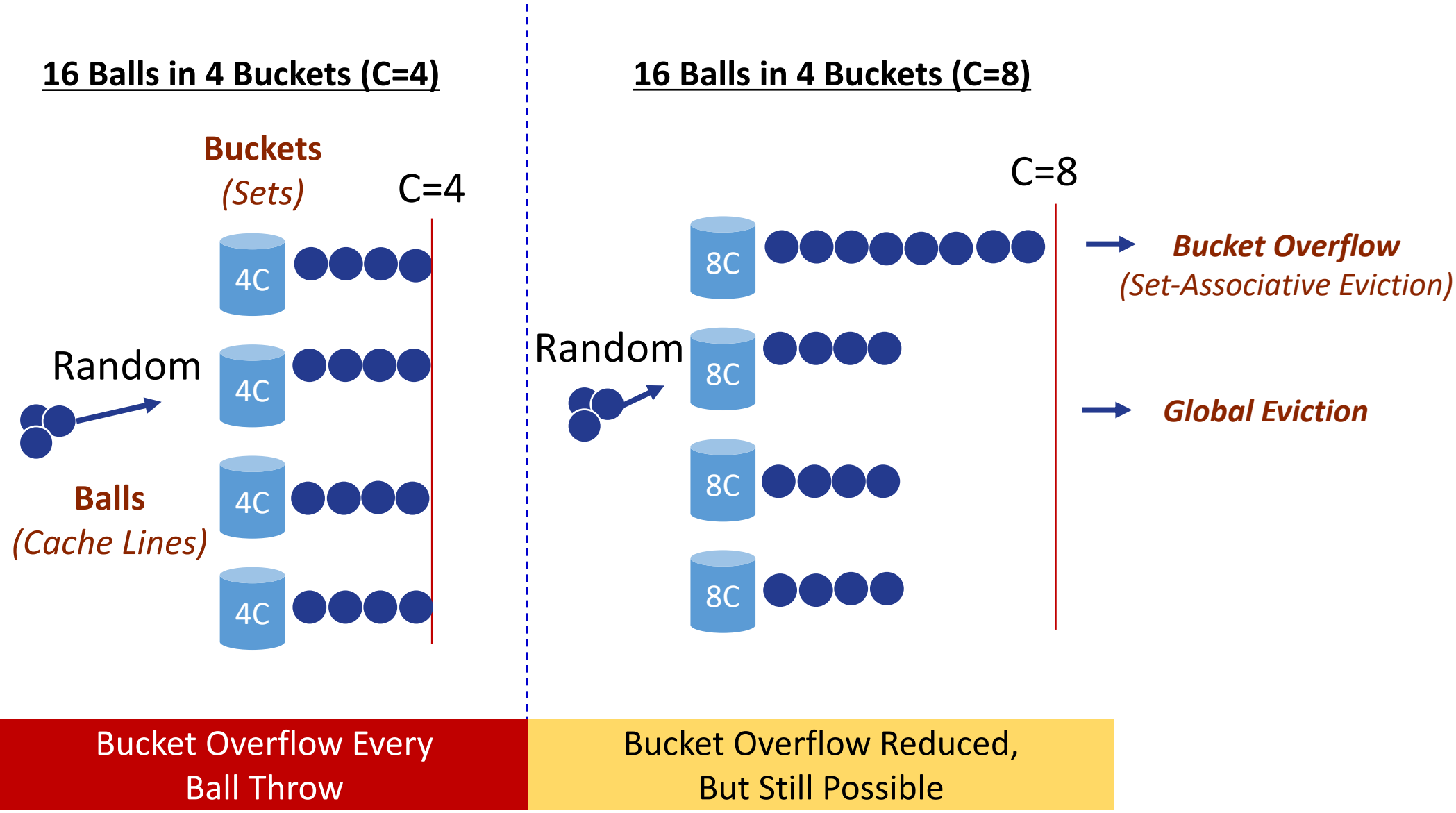
Buckets & Balls Problem



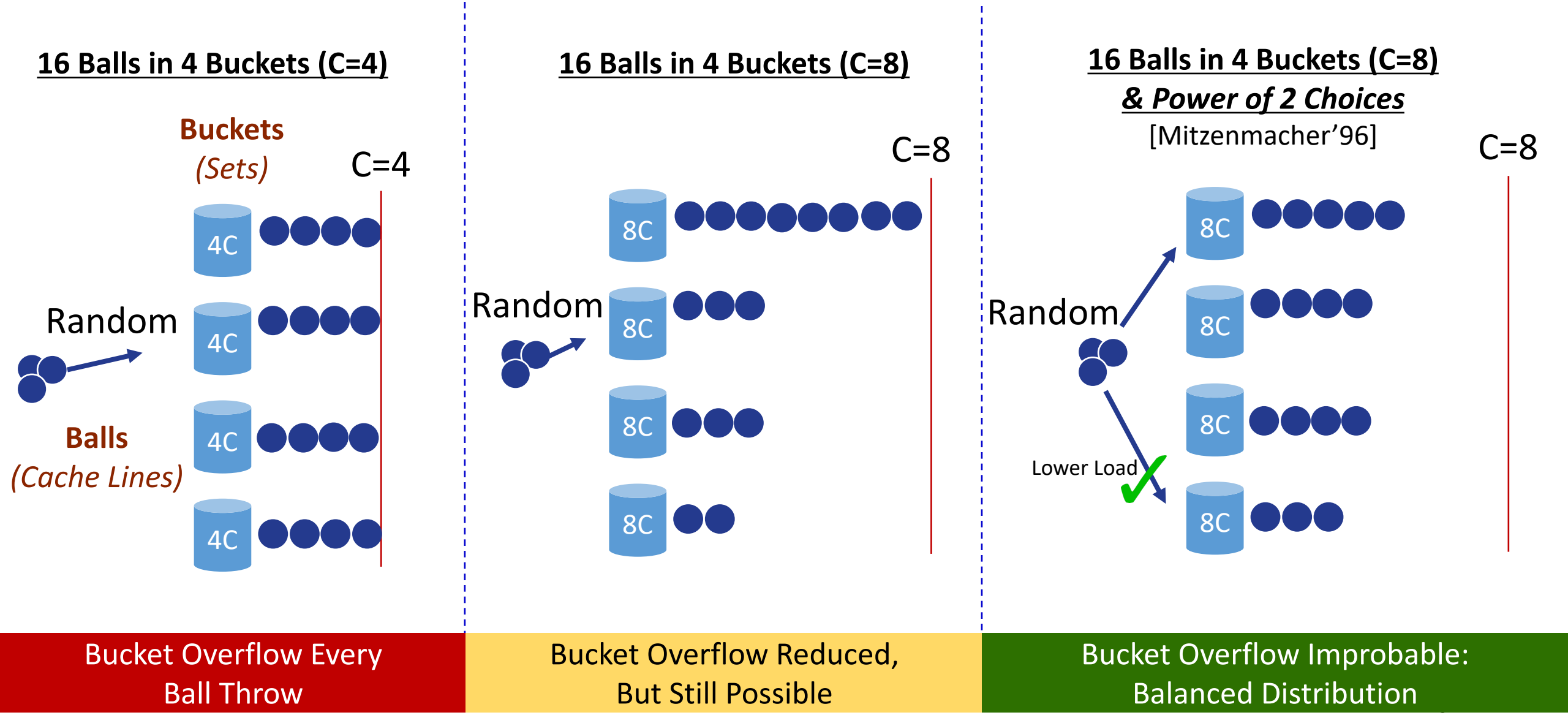
Set-Associative Randomized Cache



Insight: Use Load-Balancing to Eliminate Set-Conflicts

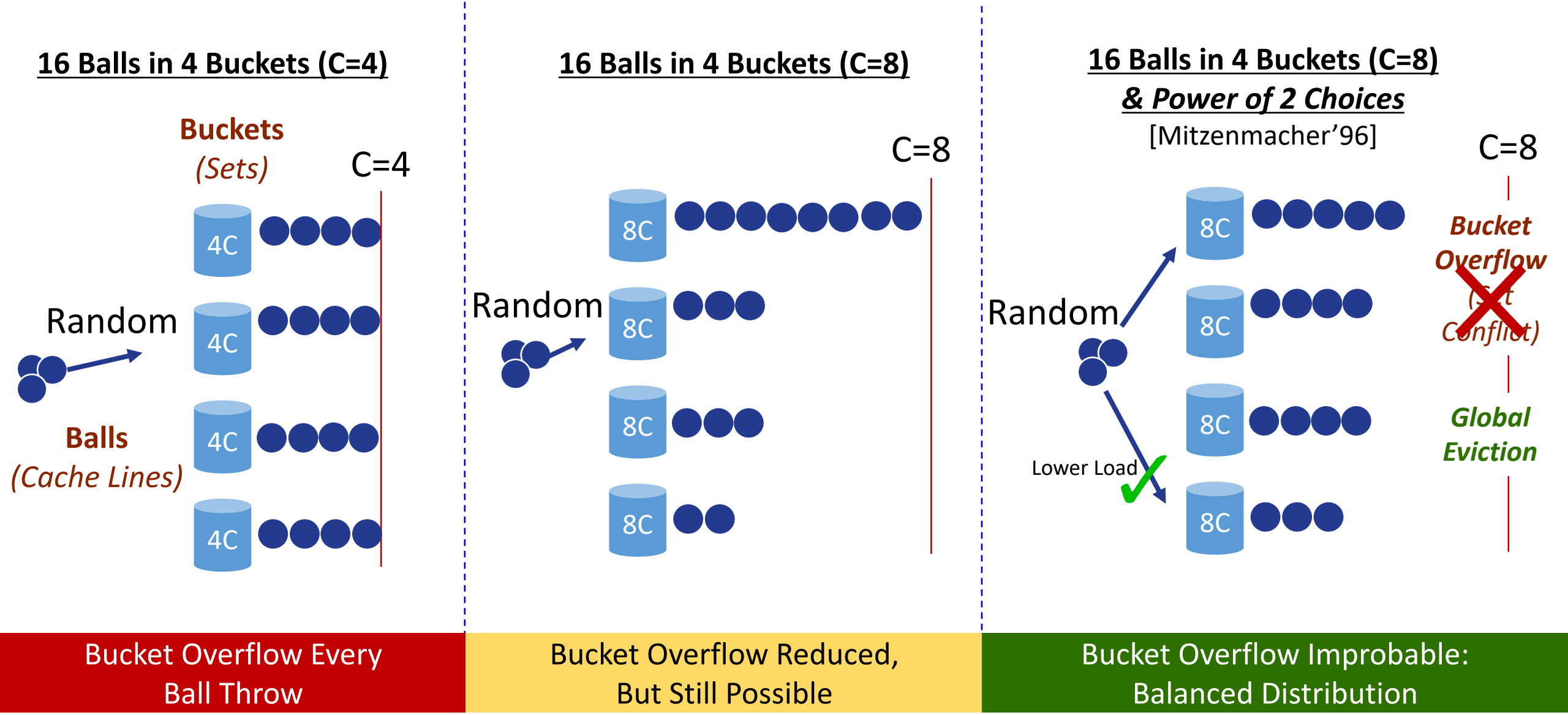


Insight: Use Load-Balancing to Eliminate Set-Conflicts

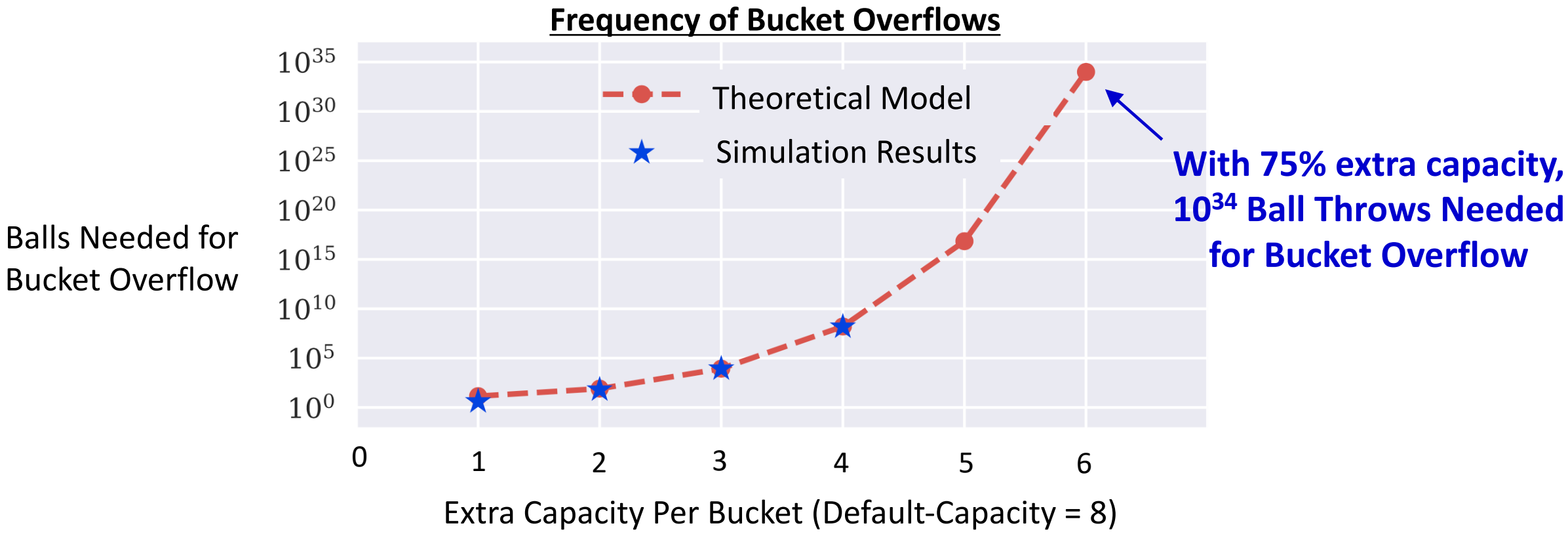


Bucket Overflow Every Ball Throw	Bucket Overflow Reduced, But Still Possible	Bucket Overflow Improbable: Balanced Distribution

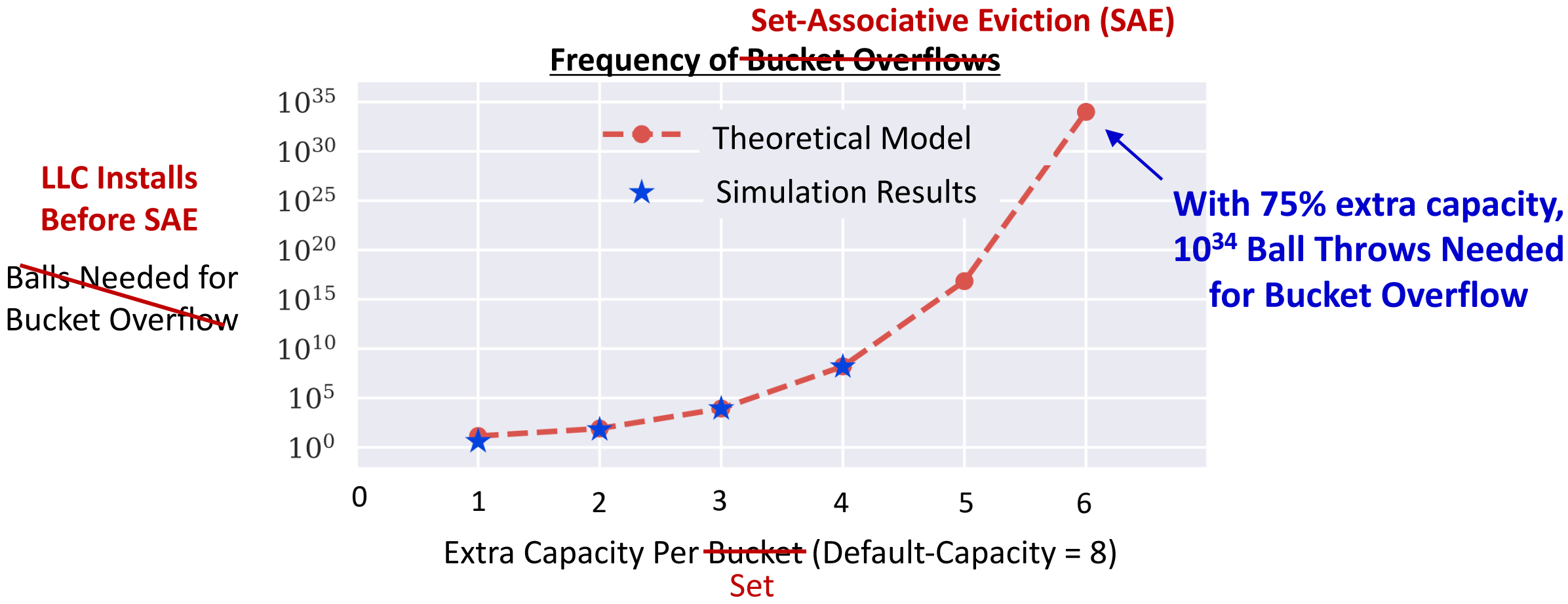
Insight: Use Load-Balancing to Eliminate Set-Conflicts



Security Guarantee With Power of 2 Choices



Security Guarantee With Power of 2 Choices

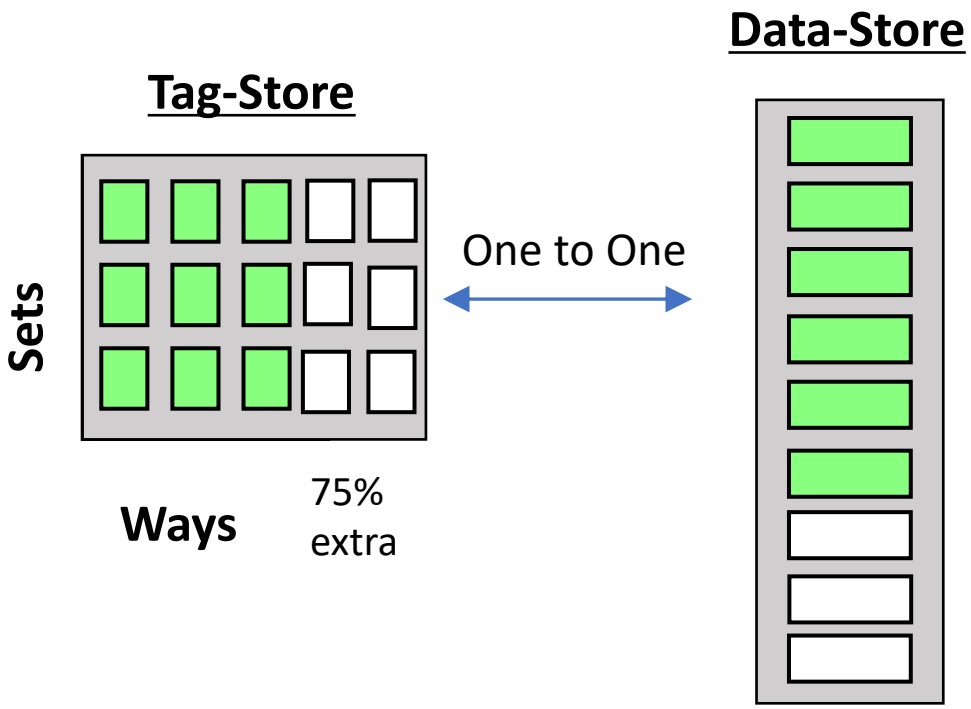


LLC with 75% extra capacity & Power of 2 Choices Indexing →

Security Guarantee: 1 SAE in 10^{34} LLC Installs (10^{17} years)

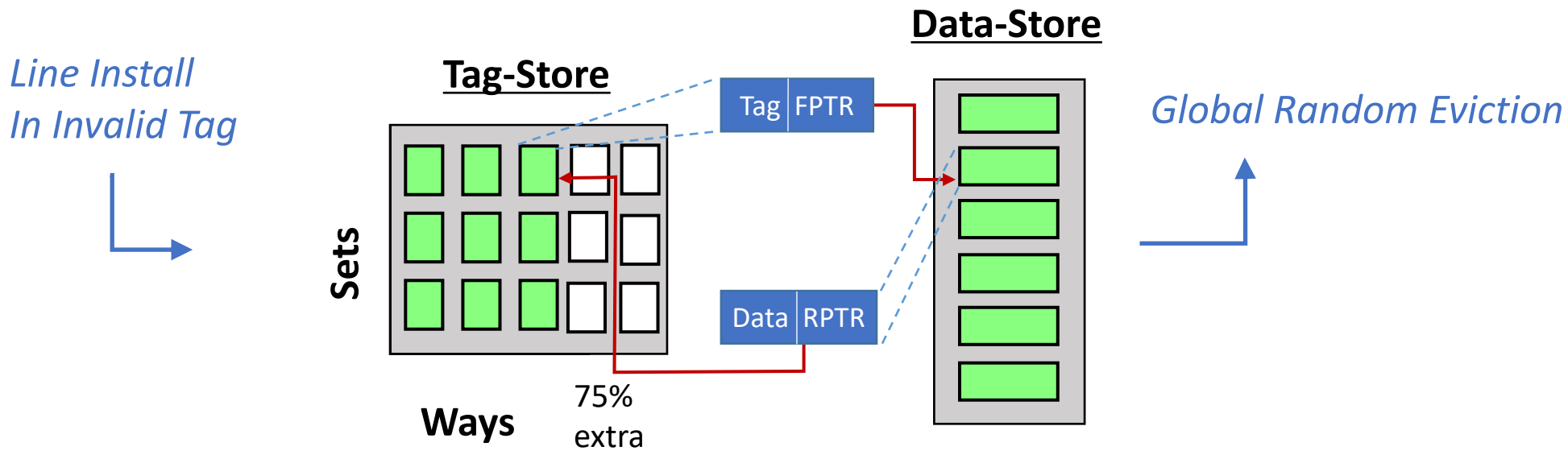
Implementing MIRAGE's Principled Randomization

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



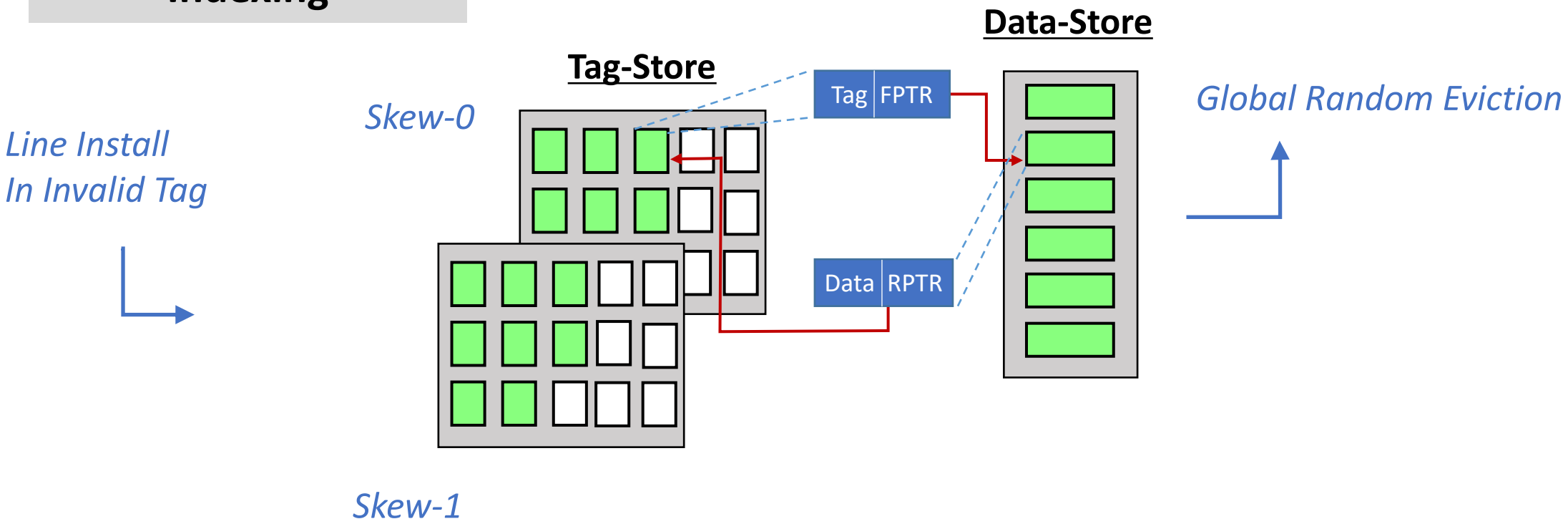
Implementing MIRAGE's Principled Randomization

MIRAGE (Decouples Tag and Data)



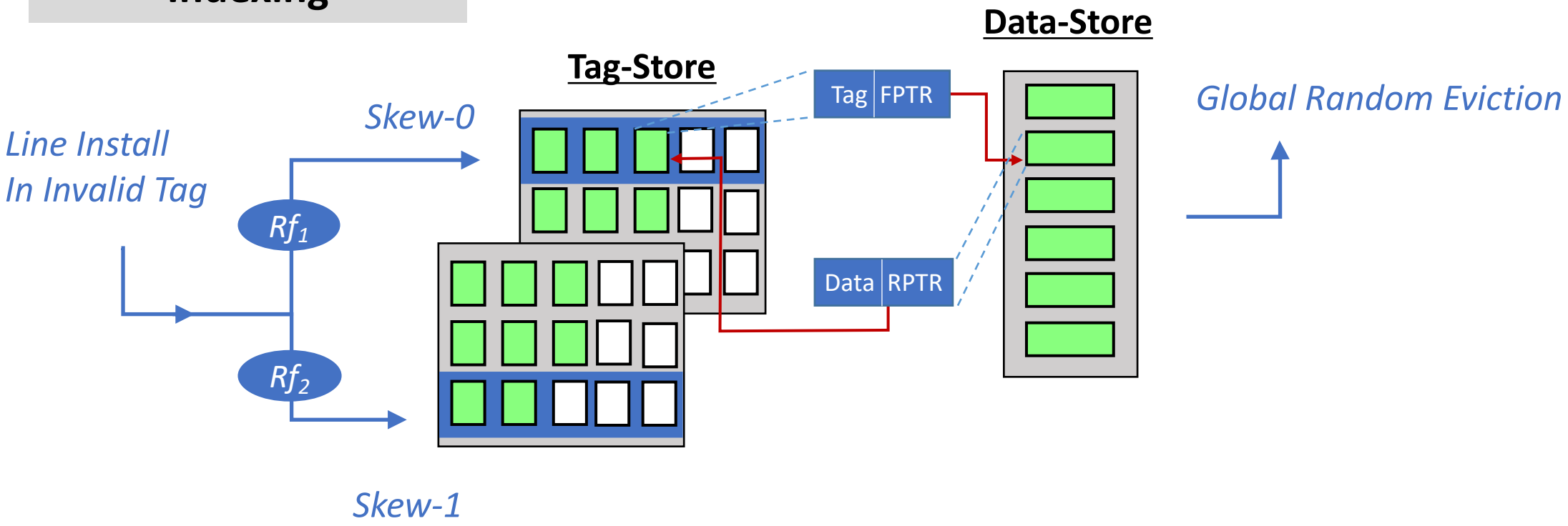
Implementing MIRAGE's Principled Randomization

Power-of-2-Choices Indexing



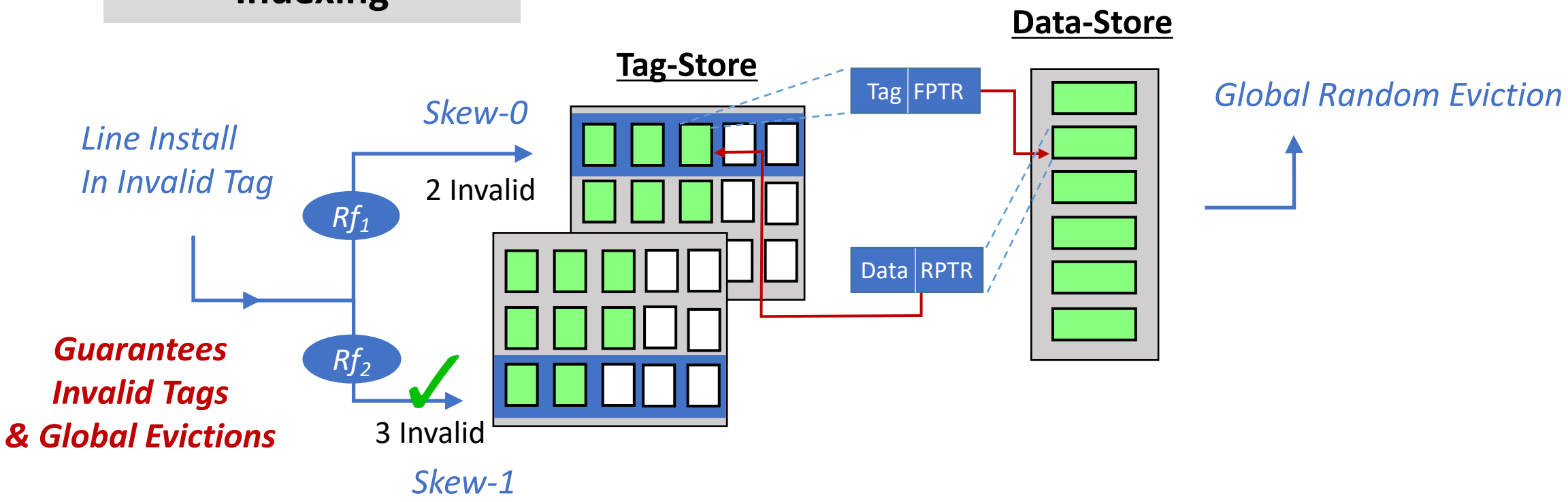
Implementing MIRAGE's Principled Randomization

Power-of-2-Choices Indexing



Implementing MIRAGE's Principled Randomization

Power-of-2-Choices Indexing

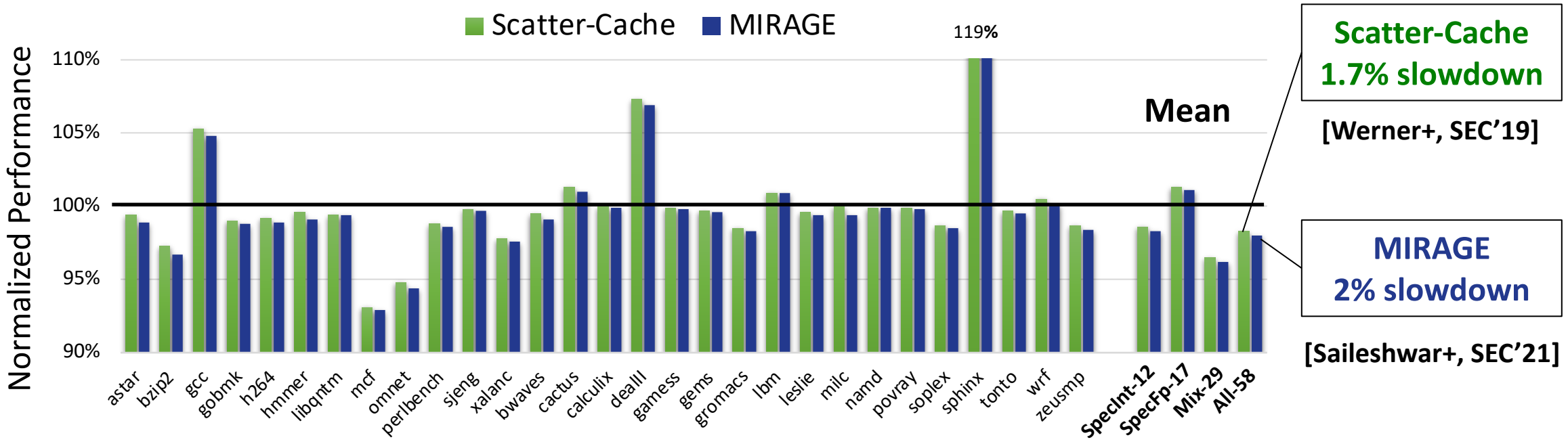


Security Guarantee: With 75% extra tags (~20% extra storage), MIRAGE ensures Set-Associative Eviction (that leaks info) occurs once in 10^{17} years

Note: MIRAGE also mitigates shared-memory attacks (like Flush+Reload) with duplication of shared-lines

Results – Performance

8-Cores, 16MB 16-way Last Level Cache, evaluated using a Trace-Based Simulator (using Intel Pin)



MIRAGE incurs slowdown of 2% (Storage-Neutral Slowdown of 3.5%) comparable to Scatter-Cache that got broken

Performance Validation with FireSim

- **Challenge: FireSim (as of 2020) only models the tag-store and not data-store for the last-level cache -** Timing model stalled till data functionally accessed from host DRAM
  - Cannot model global evictions in Mirage without the data-store & RPTR to tag-store

- **Still useful for performance validation: implemented randomized cache with 2 skews & increased access latency** (randomized evictions & access latency like MIRAGE)

4 x Rocket-Cores, 4MB /16-way L3 Cache)

Workload	Base	Randomized cache with increased lookup latency			
		+3 cycles	+4 cycles	+5 cycles	+6 cycles
perlbench	191	202	194	206	203
mcf	191	199	194	200	201
omnetpp	42	42	41	42	42
x264	699	707	702	696	707
deepsjeng	85	84	84	84	84
leela	44	44	45	45	45
exchange2	109	110	108	108	109
xz	119	114	114	115	115
MEAN	100%	100.6%	99.5%	100.9%	101.0%

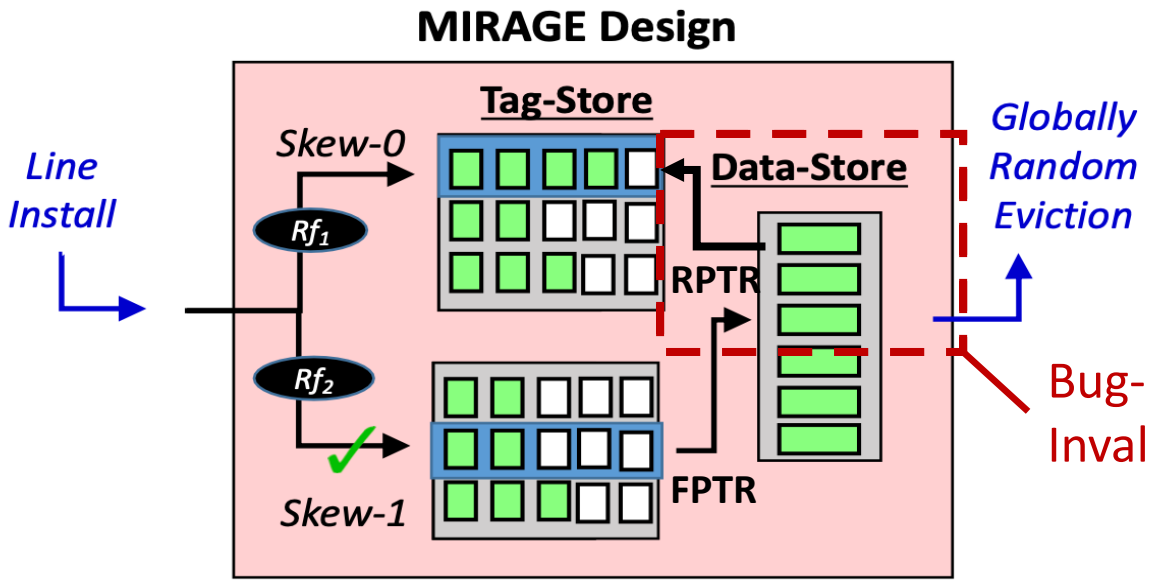
Randomized Cache with 3 - 6 cycles extra access latency → limited slowdown of <1%

Pitfalls of Inaccurate Modeling of Mirage

Case Study of the HPCA'23 Paper "Are Randomized Caches Truly Random"

Claim: **MIRAGE has set-conflicts within 100K cache accesses & is broken.**

A. Chakraborty, S. Bhattacharya, S. Saha, and D. Mukhopadhyay, "Are Randomized Caches Truly Random? Formal Analysis of Randomized-Partitioned Caches". Published In HPCA'23.



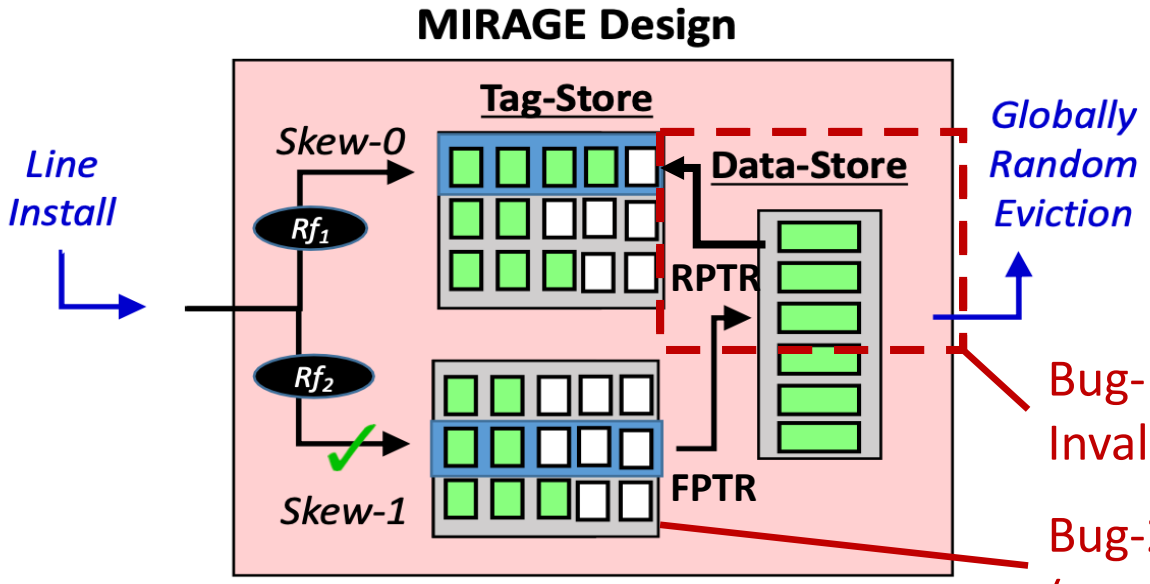

```
▶ python3 main.py
262144 16384 229376
valid eviction
ASSERT FAILURE: assert(Valid Tags <= Cache Capacity). Valid Tags : 295339, Cache Capacity : 262144
valid eviction
ASSERT FAILURE: assert(Valid Tags <= Cache Capacity). Valid Tags : 301414, Cache Capacity : 262144
valid eviction
```

Pitfalls of Inaccurate Modeling of Mirage

Case Study of the HPCA'23 Paper "Are Randomized Caches Truly Random"

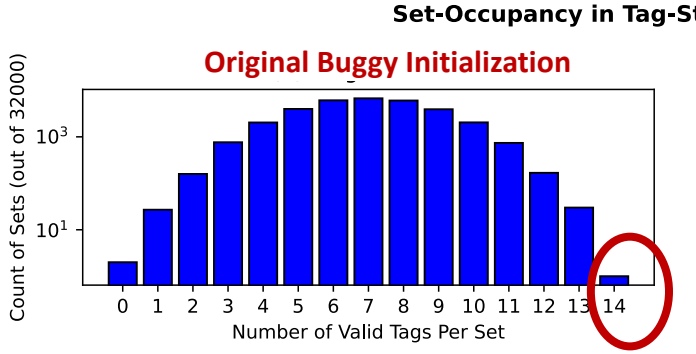
Claim: **MIRAGE has set-conflicts within 100K cache accesses & is broken.**

A. Chakraborty, S. Bhattacharya, S. Saha, and D. Mukhopadhyay, "Are Randomized Caches Truly Random? Formal Analysis of Randomized-Partitioned Caches". Published In HPCA'23.



Bug-1: Data-Store and RPTR Invalidation not modeled

Bug-2: Buggy Initialization of Tags (p_{valid} =0.5) → Starts with Full Sets

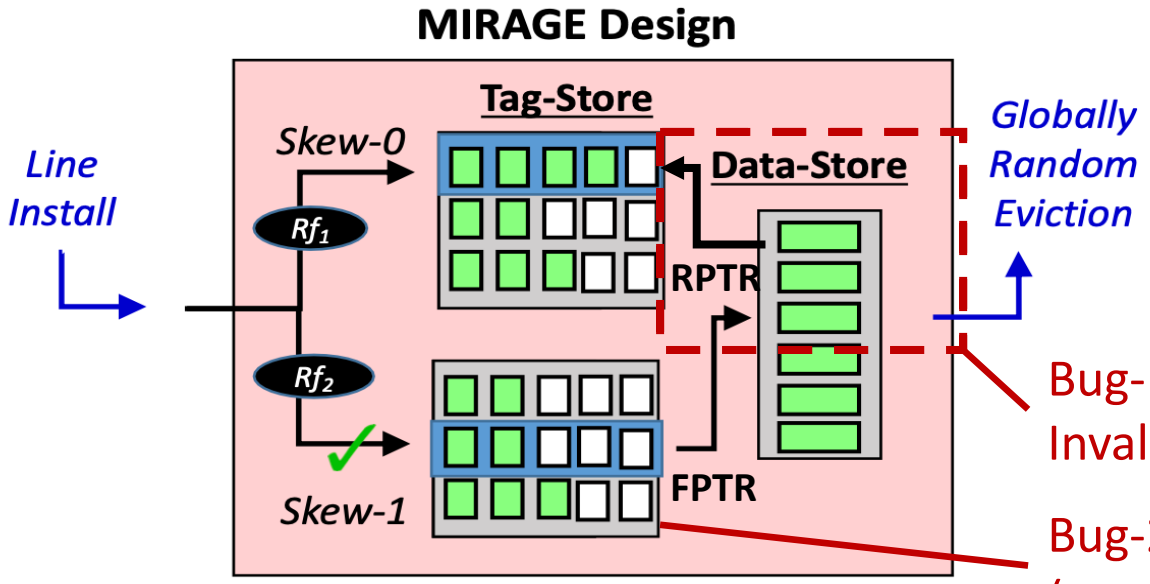


Pitfalls of Inaccurate Modeling of Mirage

Case Study of the HPCA'23 Paper "Are Randomized Caches Truly Random"

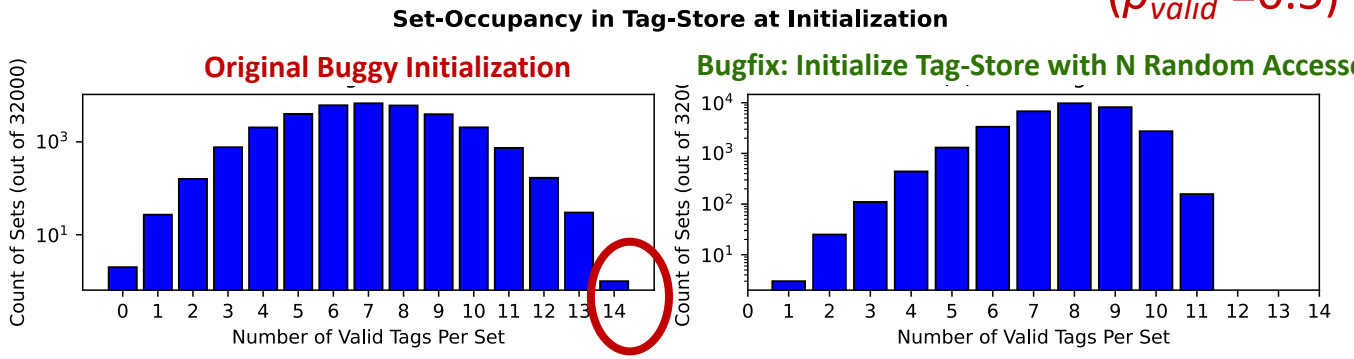
Claim: **MIRAGE has set-conflicts within 100K cache accesses & is broken.**

A. Chakraborty, S. Bhattacharya, S. Saha, and D. Mukhopadhyay, "Are Randomized Caches Truly Random? Formal Analysis of Randomized-Partitioned Caches". Published In HPCA'23.



Bug-1: Data-Store and RPTR Invalidation not modeled

Bug-2: Buggy Initialization of Tags (p_{valid} =0.5) → Starts with Full Sets

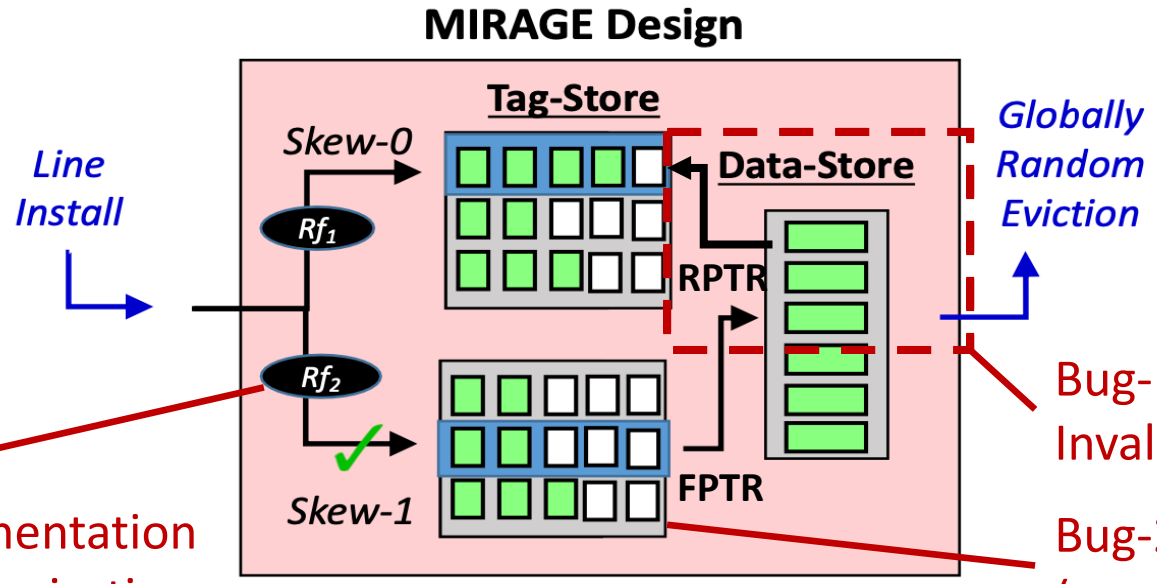


Pitfalls of Inaccurate Modeling of Mirage

Case Study of the HPCA'23 Paper "Are Randomized Caches Truly Random"

Claim: **MIRAGE has set-conflicts within 100K cache accesses & is broken.**

A. Chakraborty, S. Bhattacharya, S. Saha, and D. Mukhopadhyay, "Are Randomized Caches Truly Random? Formal Analysis of Randomized-Partitioned Caches". Published In HPCA'23.



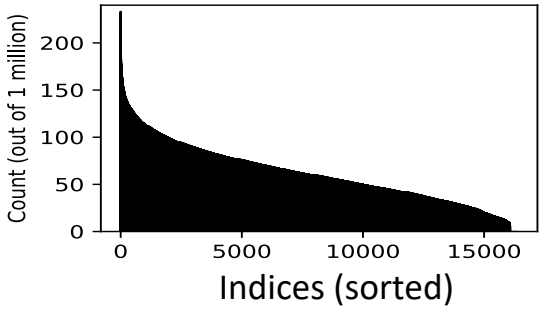
Bug-1: Data-Store and RPTR Invalidation not modeled

Bug-2: Buggy Initialization of Tags (p_{valid} =0.5) → Starts with Full Sets

Bug-3: Buggy Cipher Implementation Caused Non-Uniform Randomization

Distribution of Set-Indices for 1 Million Addresses

Original Buggy PRESENT Cipher

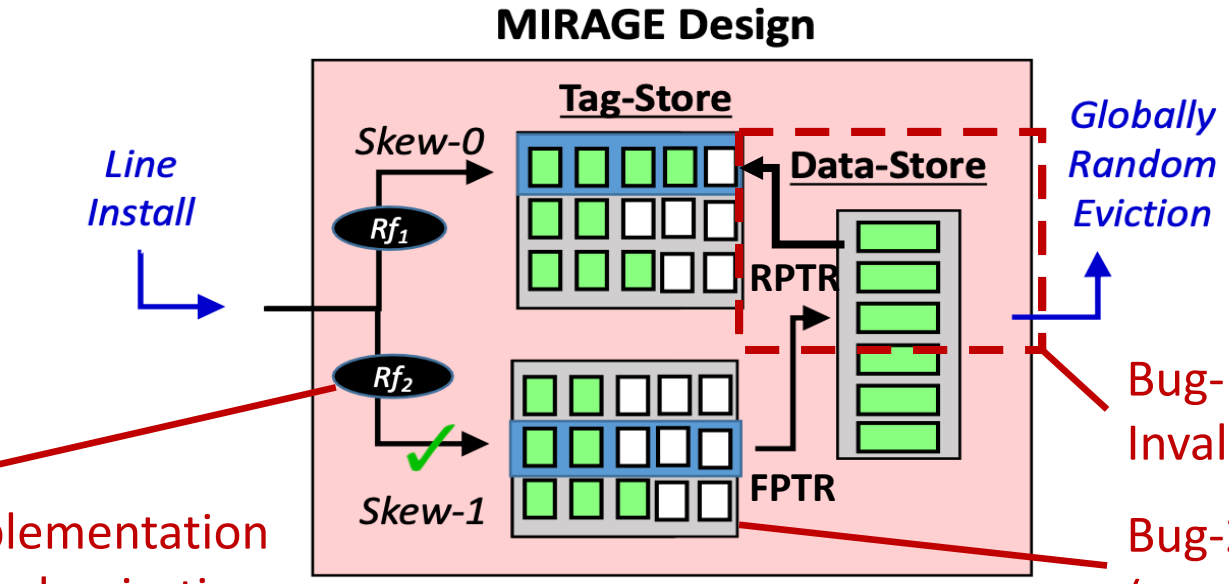


Pitfalls of Inaccurate Modeling of Mirage

Case Study of the HPCA'23 Paper "Are Randomized Caches Truly Random"

Claim: **MIRAGE has set-conflicts within 100K cache accesses & is broken.**

A. Chakraborty, S. Bhattacharya, S. Saha, and D. Mukhopadhyay, "Are Randomized Caches Truly Random? Formal Analysis of Randomized-Partitioned Caches". Published In HPCA'23.

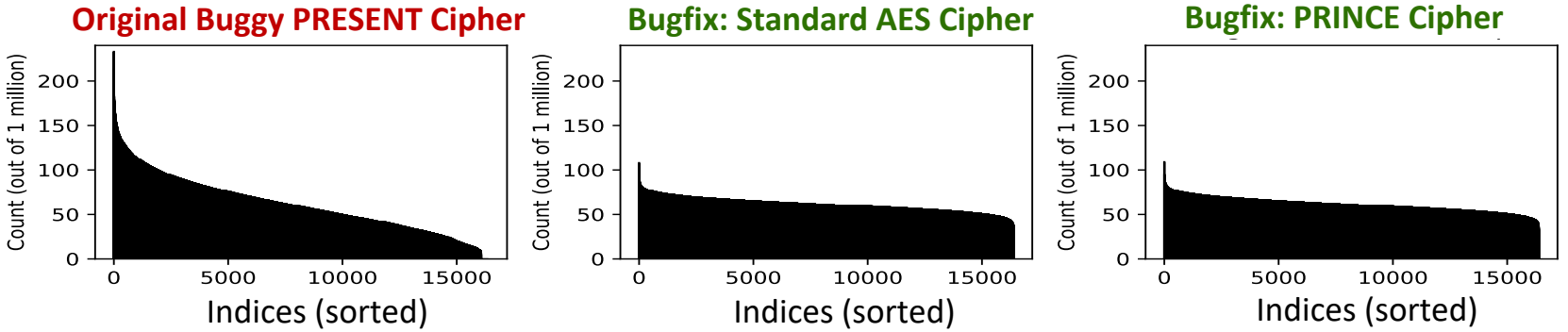


Bug-1: Data-Store and RPTR Invalidation not modeled

Bug-2: Buggy Initialization of Tags (p_{valid} =0.5) → Starts with Full Sets

Bug-3: Buggy Cipher Implementation Caused Non-Uniform Randomization

Distribution of Set-Indices for 1 Million Addresses

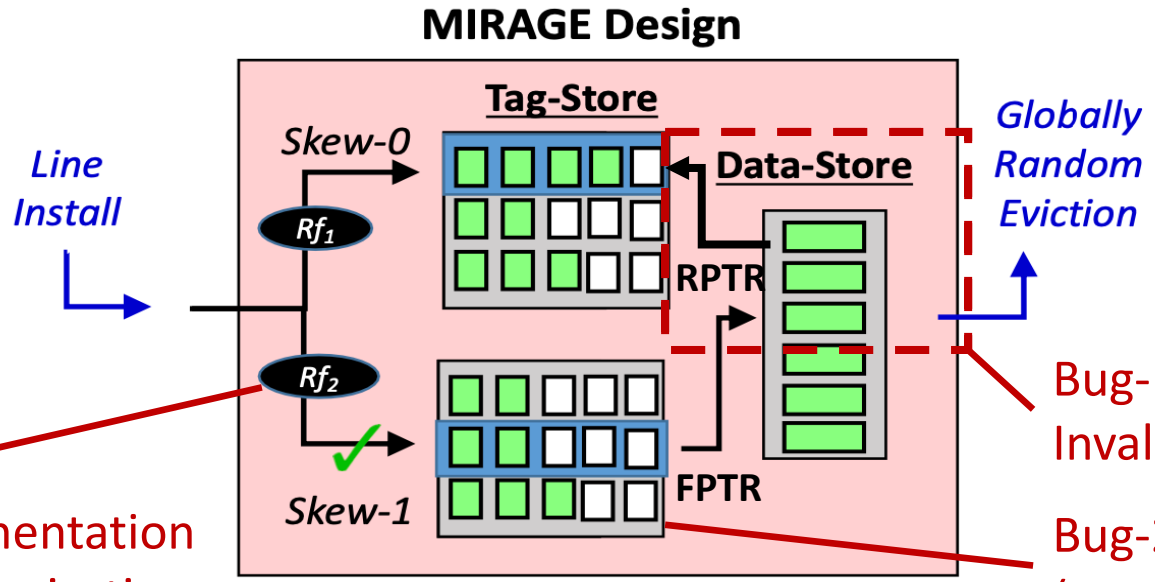


Pitfalls of Inaccurate Modeling of Mirage

Case Study of the HPCA'23 Paper "Are Randomized Caches Truly Random"

Claim: **MIRAGE has set-conflicts within 100K cache accesses & is broken.**

A. Chakraborty, S. Bhattacharya, S. Saha, and D. Mukhopadhyay, "Are Randomized Caches Truly Random? Formal Analysis of Randomized-Partitioned Caches". Published In HPCA'23.



Bug-1: Data-Store and RPTR Invalidation not modeled

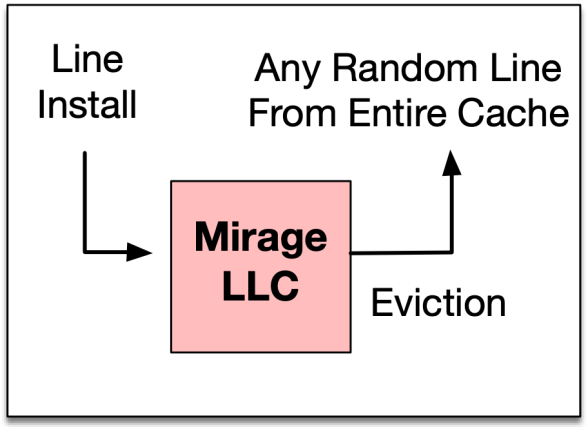
Bug-2: Buggy Initialization of Tags (p_{valid} =0.5) → Starts with Full Sets

Bug-3: Buggy Cipher Implementation Caused Non-Uniform Randomization

After Fixing Bugs in Authors' Simulator, No Set-Conflicts observed in MIRAGE (as expected)

More details: https://github.com/gururaj-s/refuting_HPCA23_randCache

Takeways from MIRAGE



Code: https://github.com/gururaj-s/mirage

Principled Randomized Cache → Future-Proof Security

MIRAGE enables fully-associative evictions (leaking no address information) practically

Impact: MIRAGE Promises an End to the Arms Race

Between 2018 - 2020, 5 defenses were broken by 6 attacks. MIRAGE has been unbroken since 2020