CASM: A Generalizable and Accessible Security Metric to Evaluate Security of Cache Architectures

Phaedra Curlin Tamara Lehman

University of Colorado Boulder

IISWC'25

Agenda

1. Background & Motivation
2. Cache Access Security Metric (CASM)
3. Results
4. Conclusion

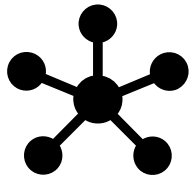
Agenda

1. **Background & Motivation**
2. Cache Access Security Metric (CASM)
3. Results
4. Conclusion

Cache Attacks

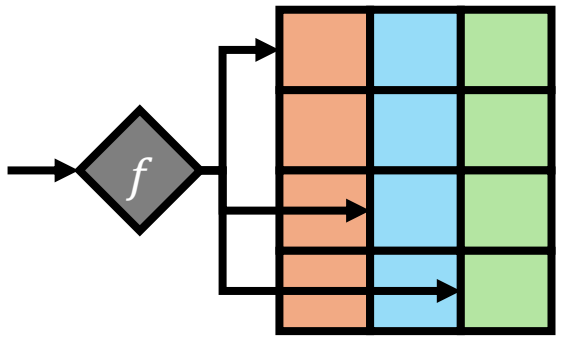
Website fingerprinting

Cryptographic key recovery

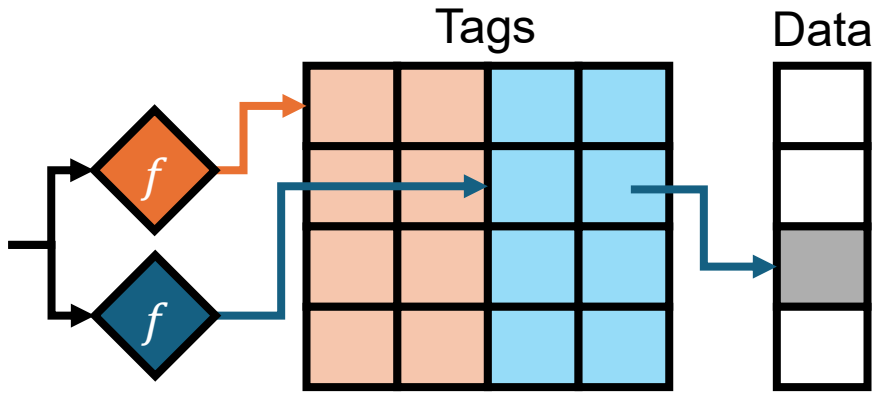


ML model stealing

Secure Cache Architectures



Randomization-based[1]

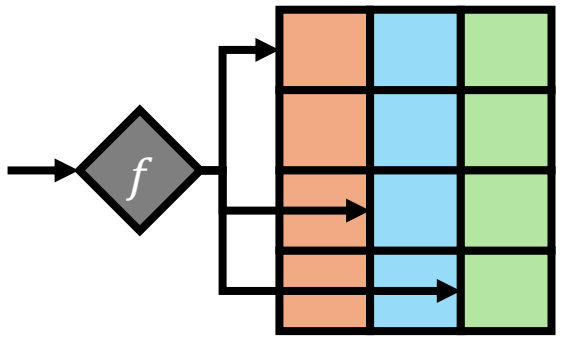


Pseudo-Fully Associative[2]

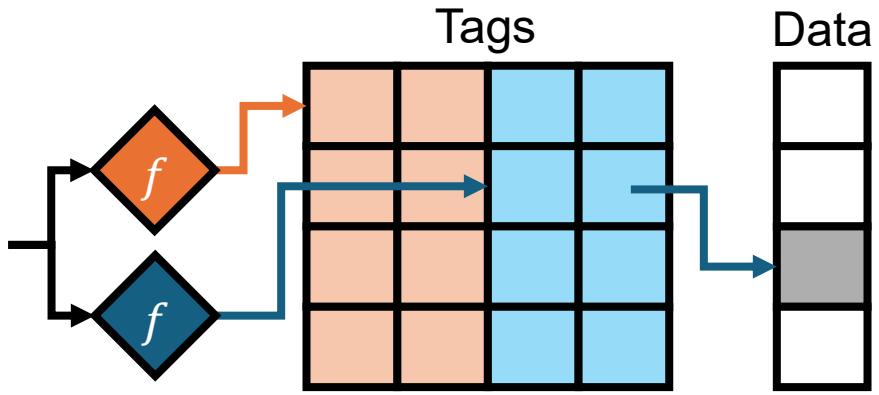
[1] Werner et al, “ScatterCache: Thwarting Cache Attacks via Cache Set Randomization”, *Usenix Security*, 2019

[2] Saileshwar and Qureshi, “MIRAGE: Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design”, *Usenix Security*, 2021

Secure Cache Architectures



Randomization-based[1]



Pseudo-Fully Associative[2]

Is cache X more secure than cache Y?

[1] Werner et al, “ScatterCache: Thwarting Cache Attacks via Cache Set Randomization”, *Usenix Security*, 2019

[2] Saileshwar and Qureshi, “MIRAGE: Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design”, *Usenix Security*, 2021

Cache Security Metrics

- Generalizability
  - CacheAudit[1] requires description of cache in custom analyzer
- Accessibility
  - Probability of Attack Success[2] results depend on precise description of attack flow

[1] Doychev et al, “CacheAudit: A Tool for the Static Analysis of Cache Side Channels”, *Usenix Security*, 2013.

[2] He and Lee, “How secure is your cache against side-channel attacks?”, *MICRO*, 2017.

Cache Security Metrics

- Generalizability – attack-independence, framework agnostic
  - CacheAudit[1] requires description of cache in custom analyzer
- Accessibility – security and non-security experts can reason about cache's security
  - Probability of Attack Success[2] results depend on precise description of attack flow

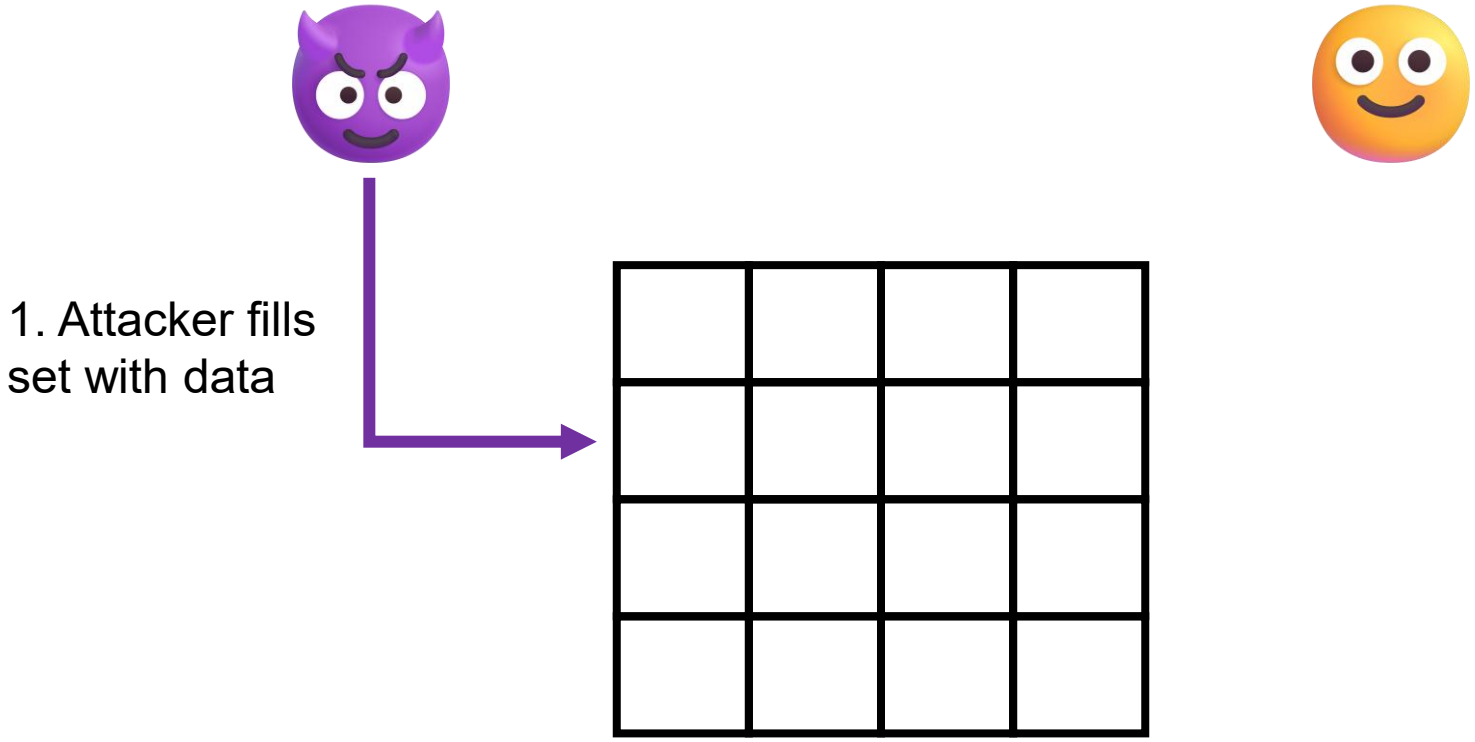
How can we create a generalizable *and* accessible metric?

Quantify behavior of well-known cache mechanisms

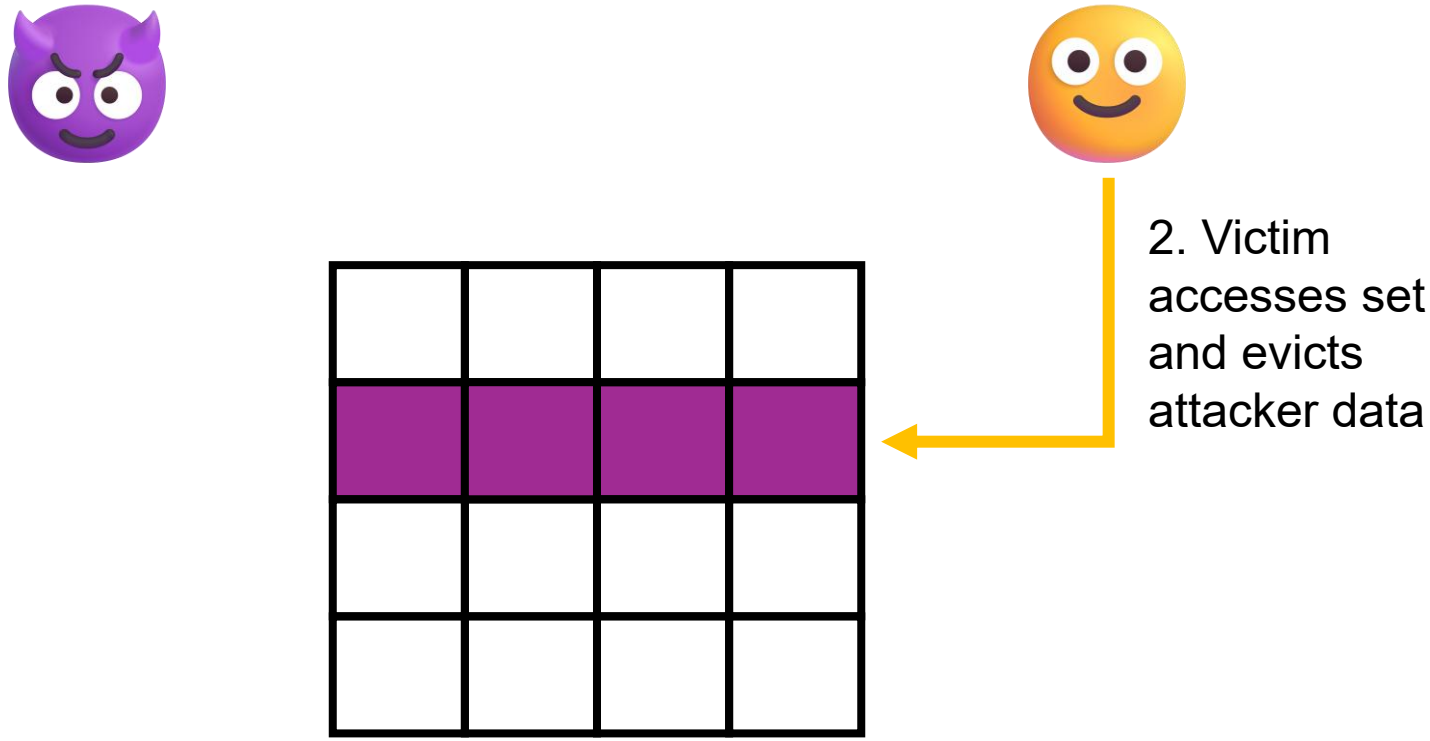
[1] Doychev et al, "CacheAudit: A Tool for the Static Analysis of Cache Side Channels", *Usenix Security*, 2013.

[2] He and Lee, "How secure is your cache against side-channel attacks?", *MICRO*, 2017.

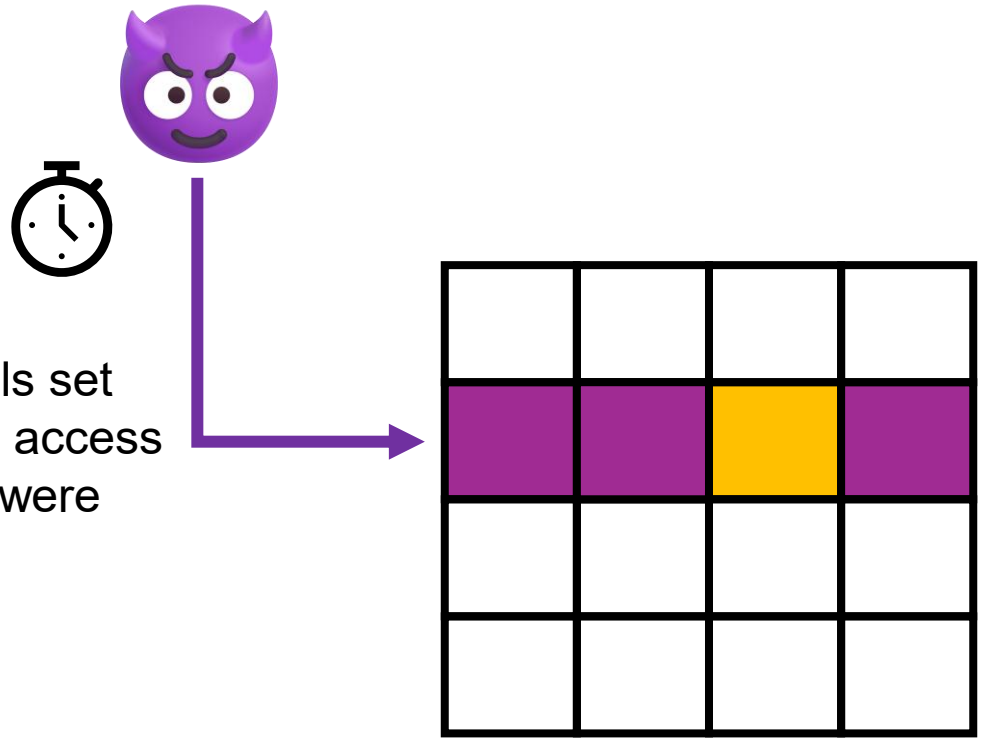
Cache Attack Behavior



Cache Attack Behavior



Cache Attack Behavior

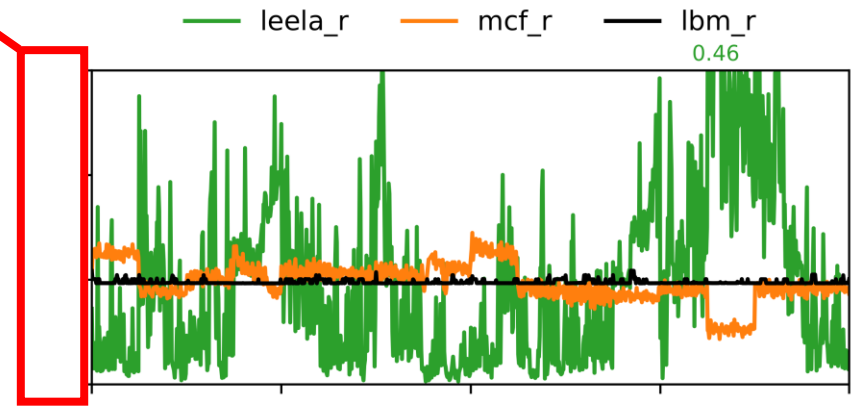


3. Attacker re-fills set with data, times access to see if blocks were evicted

Indexing Behavior

Benign programs

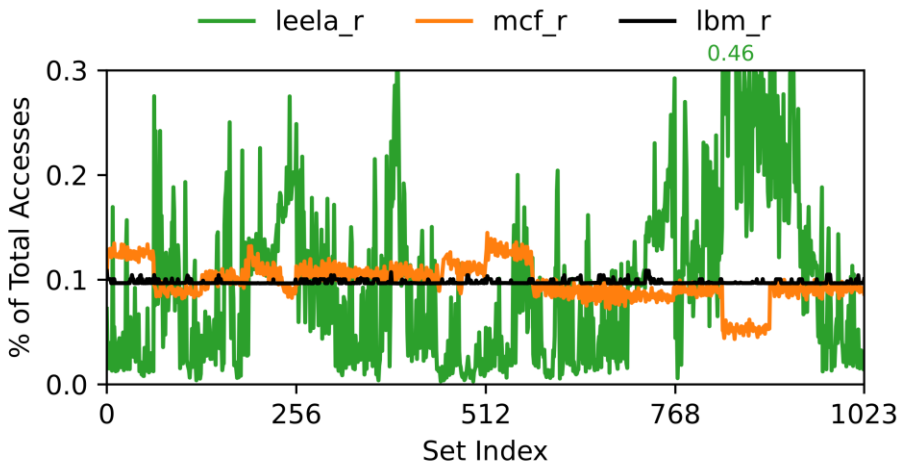
Scale <0.5%



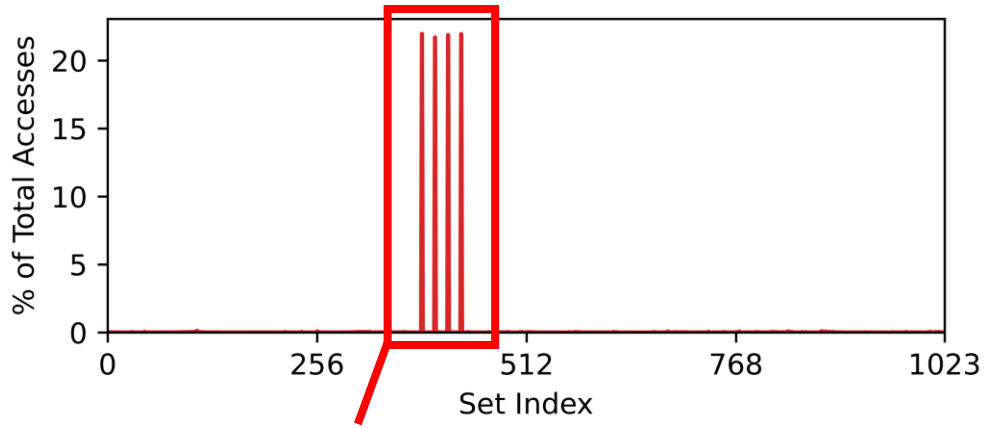
lbm_r most uniform

Indexing Behavior

Benign programs

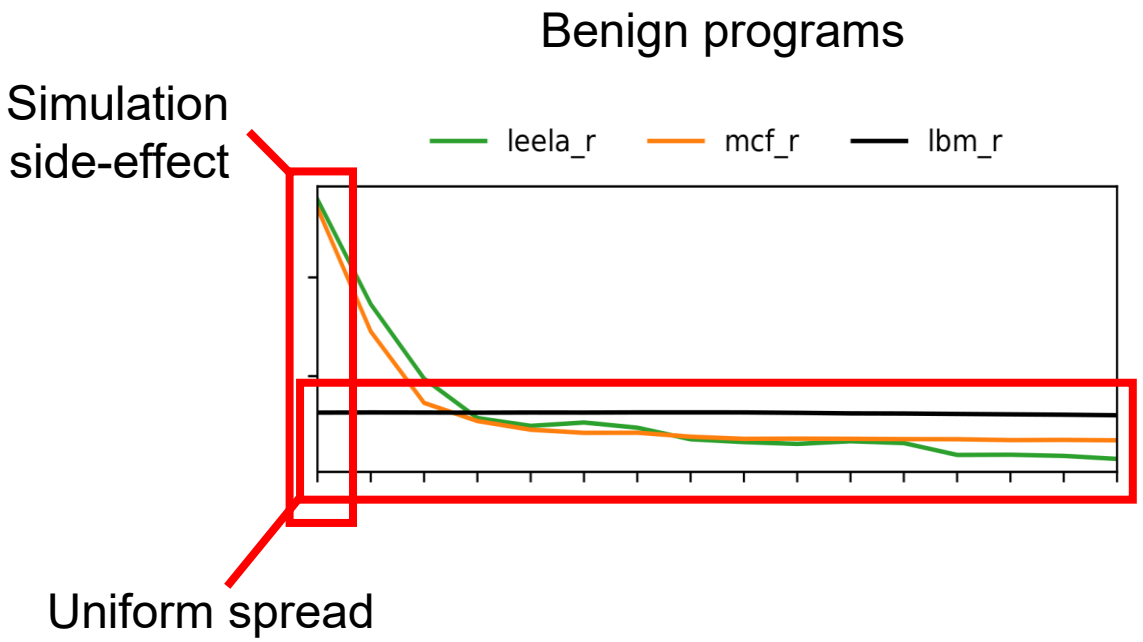


`Flush+Reload` T-Table Attack



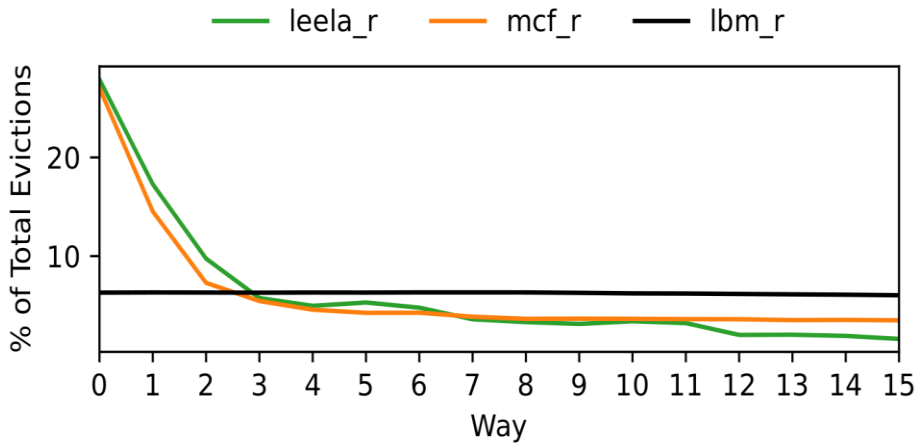
~25% accesses to each T-Table address

Eviction Behavior

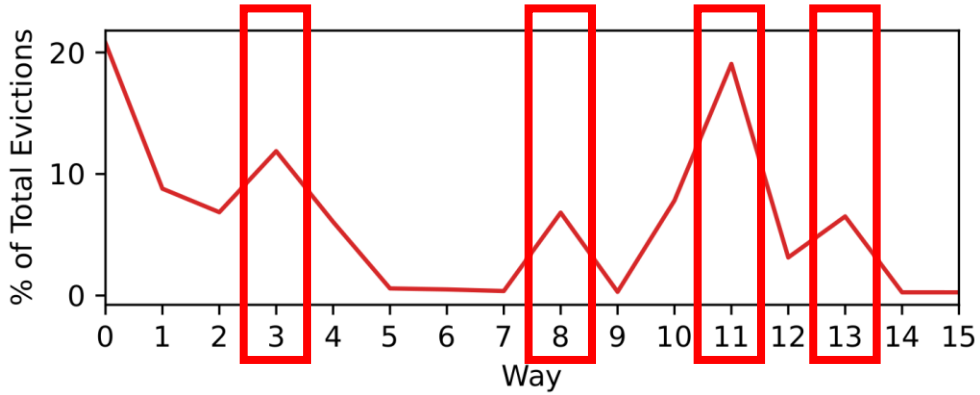


Eviction Behavior

Benign programs



`Flush+Reload` T-Table Attack



Cache Attack Behavior

- Benign programs spread uniformly across all sets and ways
- Cache attack repeatedly re-accesses small number of sets/blocks

Agenda

1. Background & Motivation
2. **Cache Access Security Metric (CASM)**
3. Findings
4. Conclusion

Cache Indexing Behavior

- Indexing function should spread accesses uniformly across sets
  - Optimal performance[1]
  - Decouple set index from access providing stronger security
- Relative entropy between uniform and empirically indexing
  - Bounded by $[0, \infty)$

$$Ent_i(n) = \sum_{s \in S} i_u(s) \log_2 \frac{i_u(s)}{i_e(s)}$$

n: epoch
s: cache set
i_u: uniform prob.
i_e: empirical prob. of accessing index

[1] Hill and Smith, “Evaluating associativity in CPU caches”, *IEEE Transactions on Computers*,1989

Cache Eviction Behavior

- Eviction policy should consider all blocks set as an equally likely candidate for eviction
- Average normalized entropy for evictions over all sets
  - Bounded by $[0,1]$

$$Ent_e(n) = 1 - \frac{1}{S}\sum_{s \in S}\sum_{w \in W_s} \frac{e_e(s)\log_2 e_e(s)}{\log_2(W_s)}$$

n: epoch
s: cache set
w: way
W_s: associativity
e_e: empirical prob. of evicting way

Cache Access Security Metric (CASM)

- Combine indexing and eviction entropy into single metric
  - Linear equation to provide equal weight to each entropy type
- Lower CASM score is better
  - Fully associative cache/uniform indexing, $Ent_i = 0$
  - Uniform evictions, $Ent_e = 0$

$$CASM = \widetilde{Ente_i} + \widetilde{Ent_e}$$

Implementation

- Generalizability of metric means it can be implemented in common processor simulators
- Implemented as C++ extension to gem5[1]
  - Results outputted at end of simulation with performance metrics

[1] Binkert et al, “The gem5 simulator”, *ACM SIGARCH Computer Architecture News*, 2011.

Agenda

1. Background & Motivation
2. Cache Access Security Metric (CASM)
3. **Results**
4. Conclusion

Evaluation

- SPEC 2017 benchmarks, 2x flush-based attacks, occupancy
- 4 cache architectures:
  - Set-Associative, Skewed Associative[1], ScatterCache[2], Mirage[3]
- 6 eviction policies:
  - BIP[4], BRRIP[5], FIFO, LRU, MRU, PR
- Varying associativity
- Effect of weights

[1] Seznec and Bodin, “Skewed-associative Caches”, *PARLE*, 1993

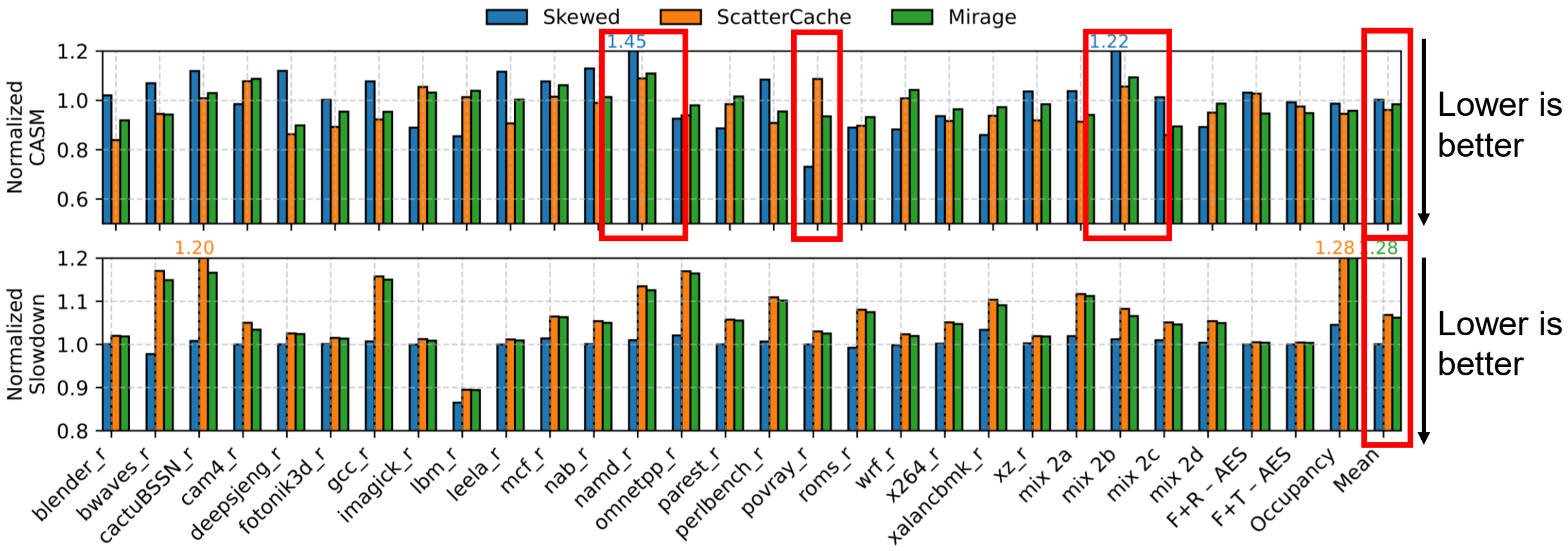
[2] Werner et al, “ScatterCache: Thwarting Cache Attacks via Cache Set Randomization”, *Usenix Security*, 2019

[3] Saileshwar and Qureshi, “MIRAGE: Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design”, *Usenix Security*, 2021

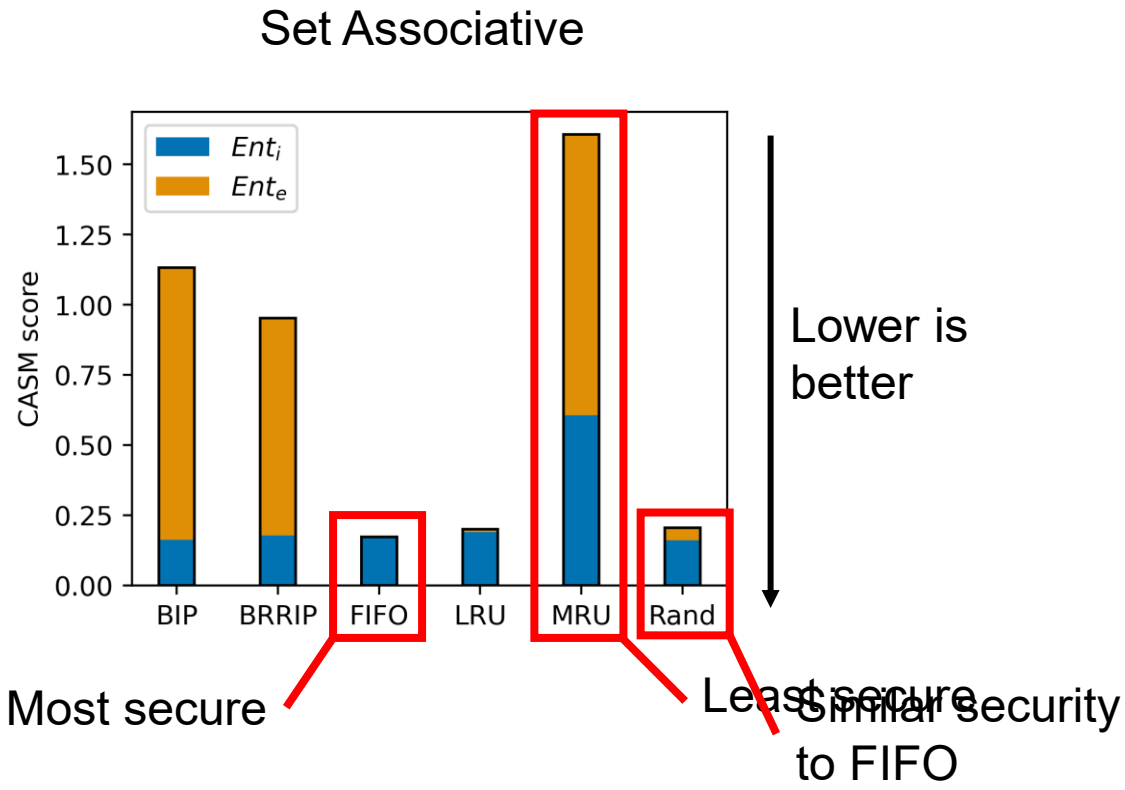
[4] Qureshi et al, “Adaptive insertion policies for high performance caching”, *ISCA,* 2007

[5] Jaleel et al, “High performance cache replacement using re-reference interval prediction”, *ISCA*, 2010

Results – Secure Architectures

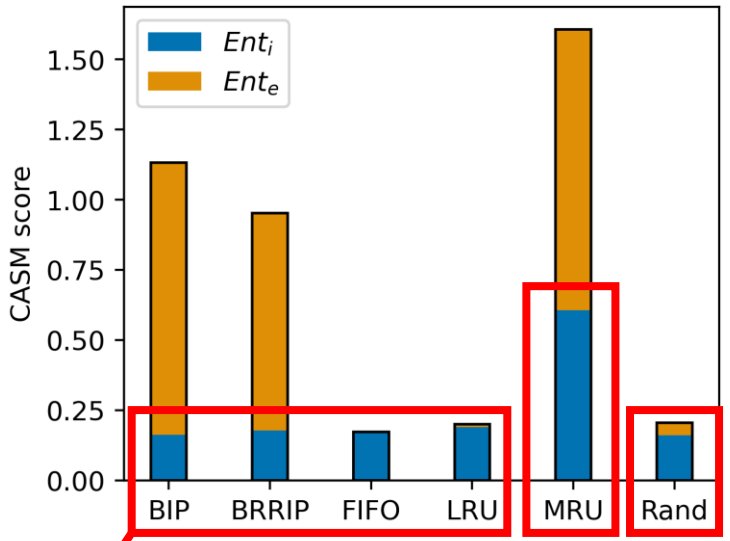


Results – Eviction Policies



Results – Eviction Policies

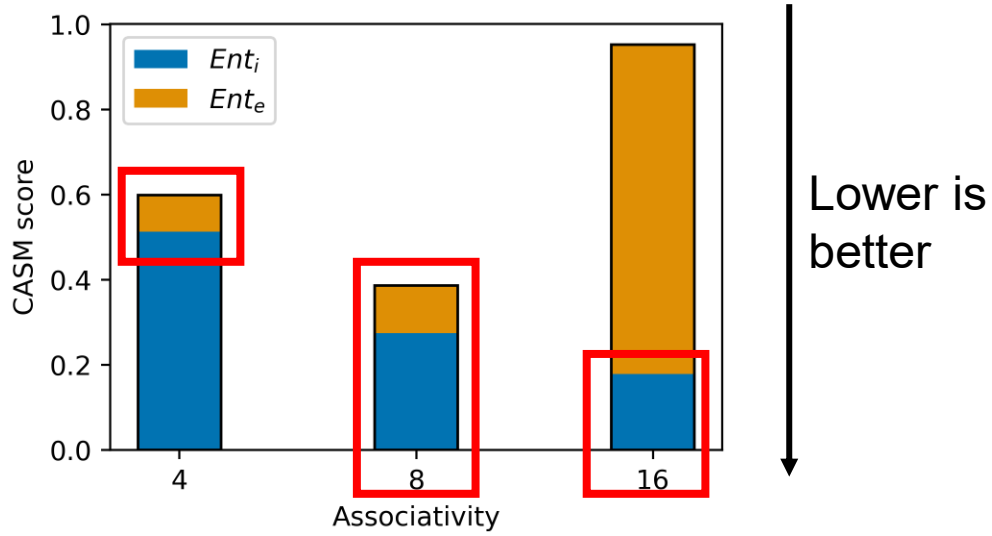
Set Associative



Lower is better

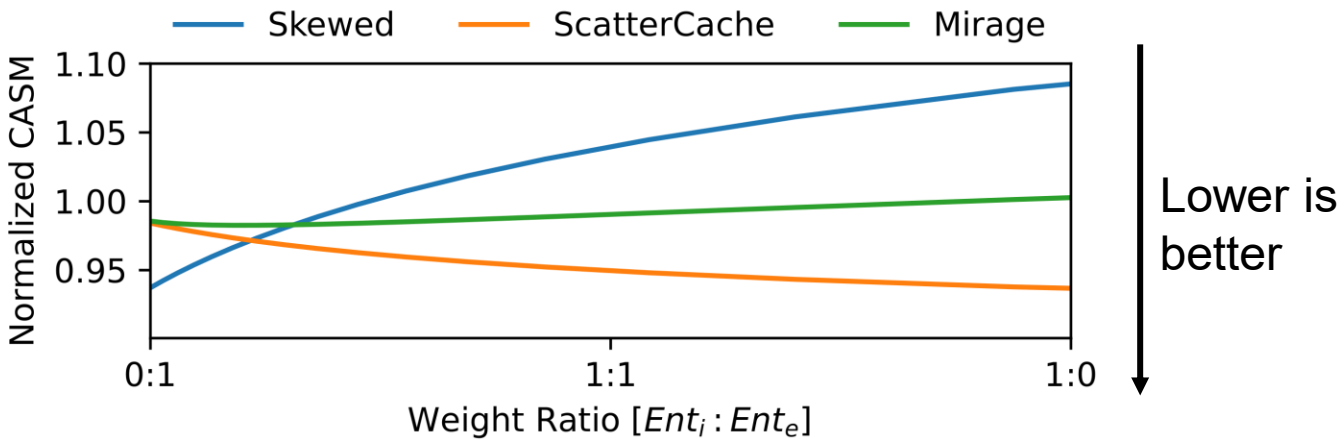
Similar indexing entropy

Results – Associativity



Results – Weights

$$CASM = \widetilde{Ent_i} + \widetilde{Ent_e}$$



Agenda

1. Background & Motivation
2. Cache Access Security Metric (CASM)
3. Results
4. **Conclusion**

Conclusion

- Quantifying behavior of well-understood cache mechanisms can provide a generalizable and accessible metric
- ScatterCache provides a better performance-security trade-off than Mirage
- FIFO eviction policy provides favorable security
- Higher associativity does not always provide best security guarantees

CASM: A Generalizable and Accessible Security Metric to Evaluate Security of Cache Architectures

CASM is a **generalizable** and **accessible** cache security metric that can be implemented into common processor simulators. CASM is composed of two metrics: the **indexing** and **eviction entropy**. The indexing entropy evaluates the **uniformity of cache indexing**. The eviction entropy evaluates the average **uniformity of evictions** for each block.

Thank you!

Questions?

phaedra.curlin@colorado.edu