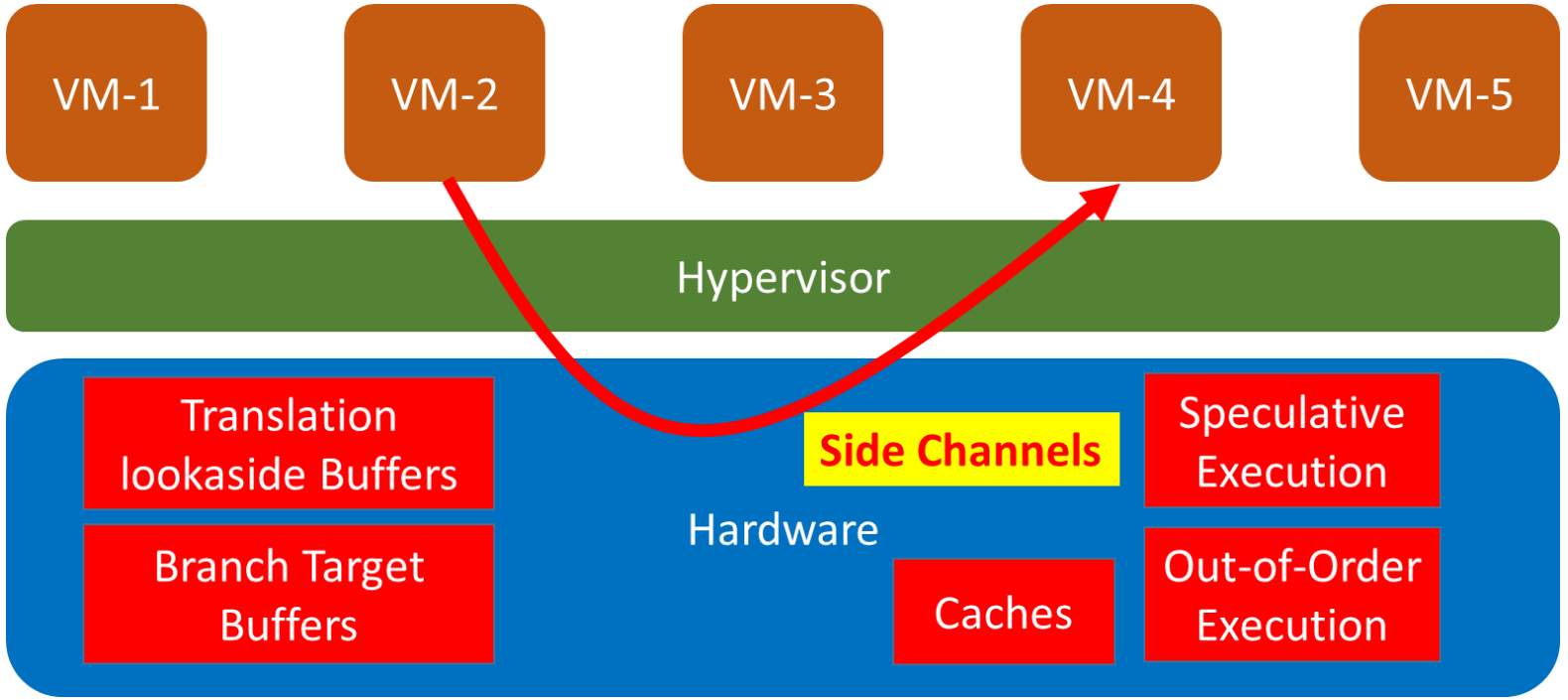
PCache: Permutation-based Cache to Counter Eviction-based Cache-based Side-Channel Attacks

M. Asim Mukhtar [1] M. Khurram Bhatti [1] Guy Gogniat [2]

1 Information Technology University, Lahore, Pakistan

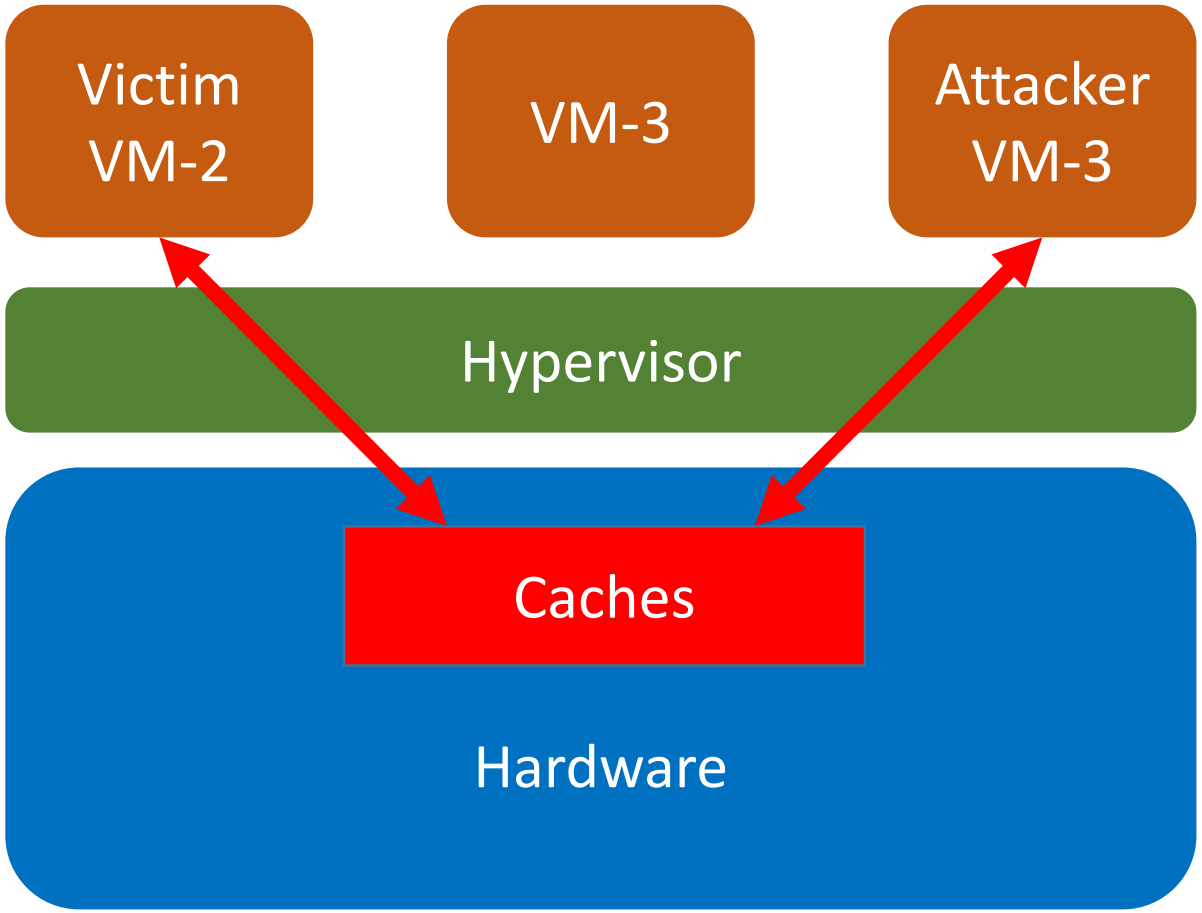
2 University of South Brittany, Lorient, France

Introduction



Security Threat: In cloud computing, some components are used by attackers as a side-channel to steal secrets of co-running applications.

Introduction



Cache-based side-channel attacks [M. Werner, USENIX Security 2019]

- Extracting keys of cryptographic algorithms (RSA, AES, etc).
- Monitoring keystrokes.
- Reading unauthorized address space.

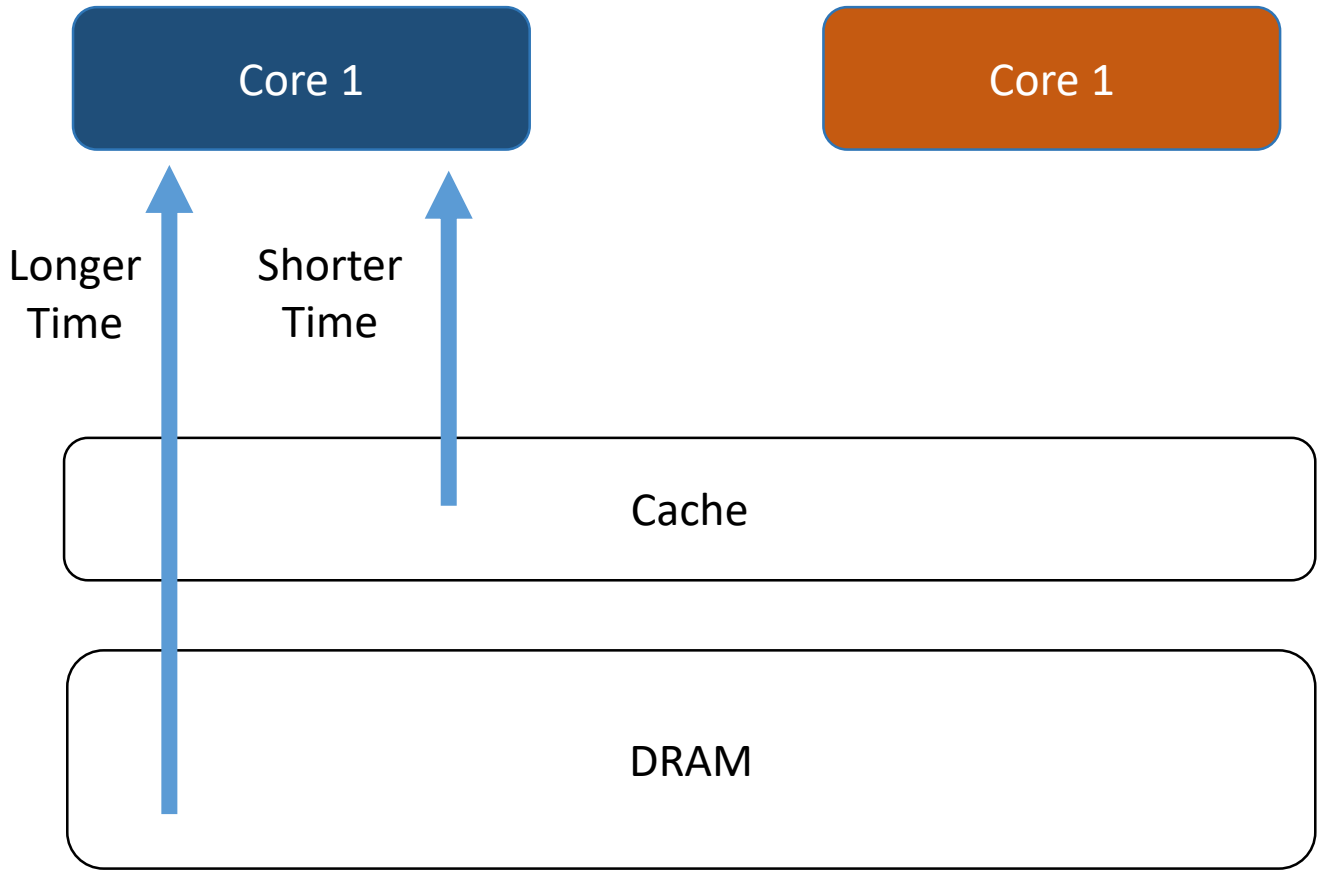
Introduction

- Unfortunately, existing countermeasures disable sharing, which incurs underutilization.
- Countermeasures retaining shared cache design are not secure.
- Our Goal: To achieve security against cache-based side-channel attacks while
  - Retaining the shared cache design
  - Negligible performance overhead
- Our Solution: Permutation based cache architecture

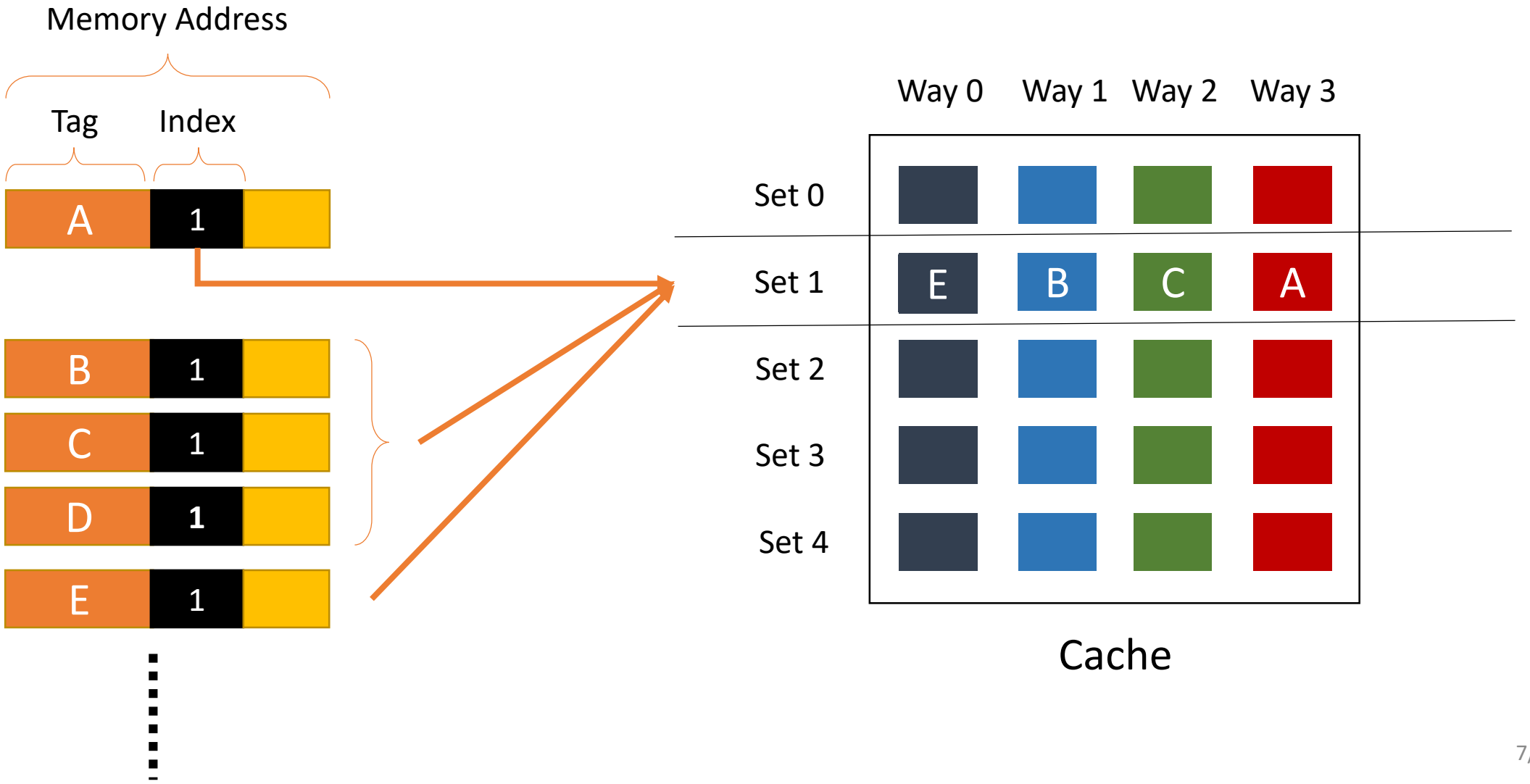
Outline

- Background
- Security Issue: Prime+Probe Attack
- Prior countermeasures & their limitations
- Direct Relation problem
- PCache
- Security and Performance Results

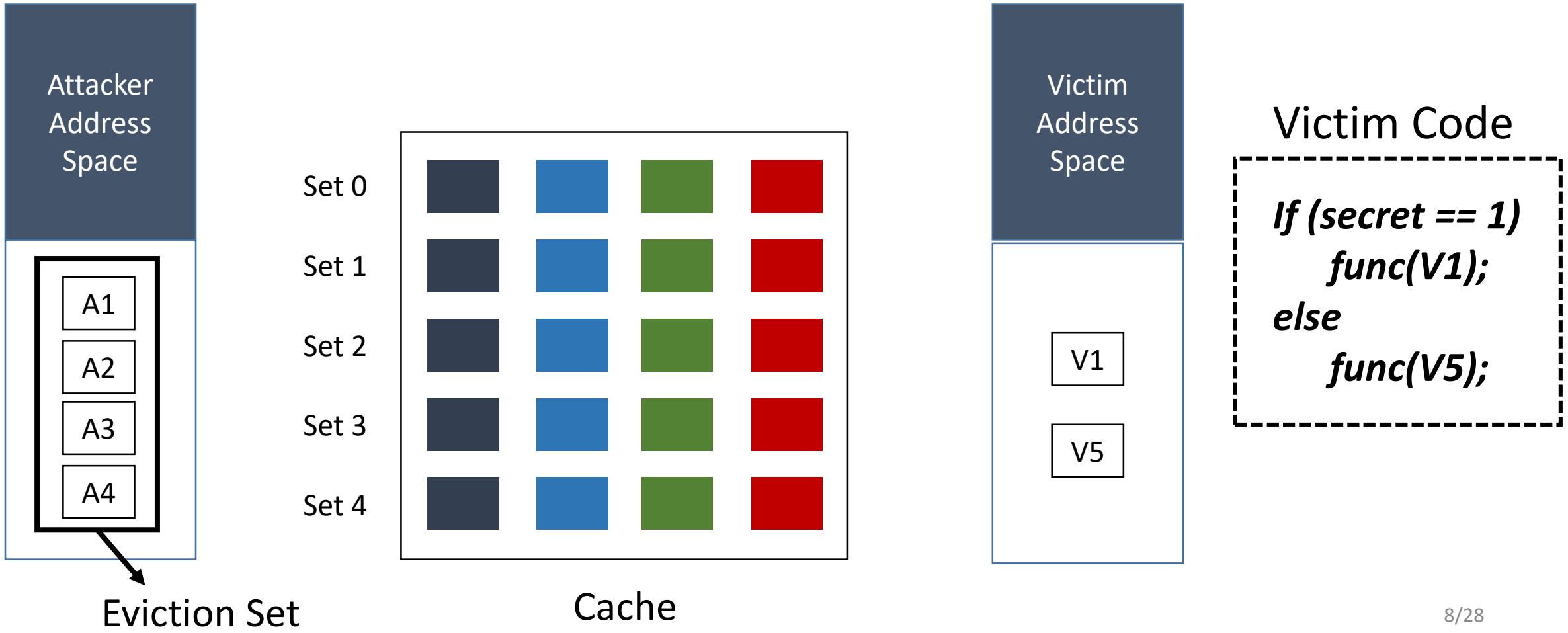
Caches



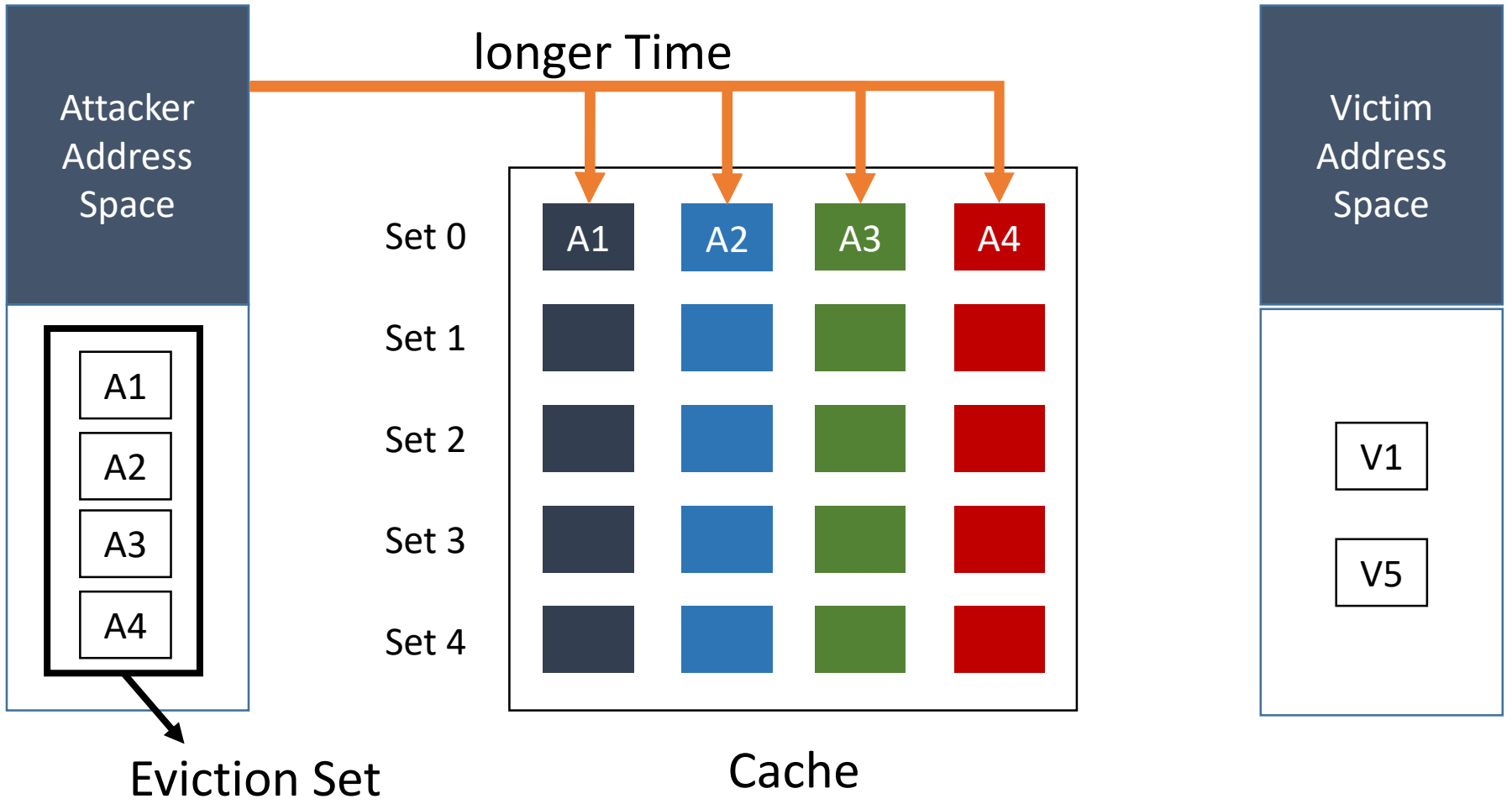
Background: Cache structure and Mapping



Eviction based Cache Side Channel Attacks: Prime+Probe Attack

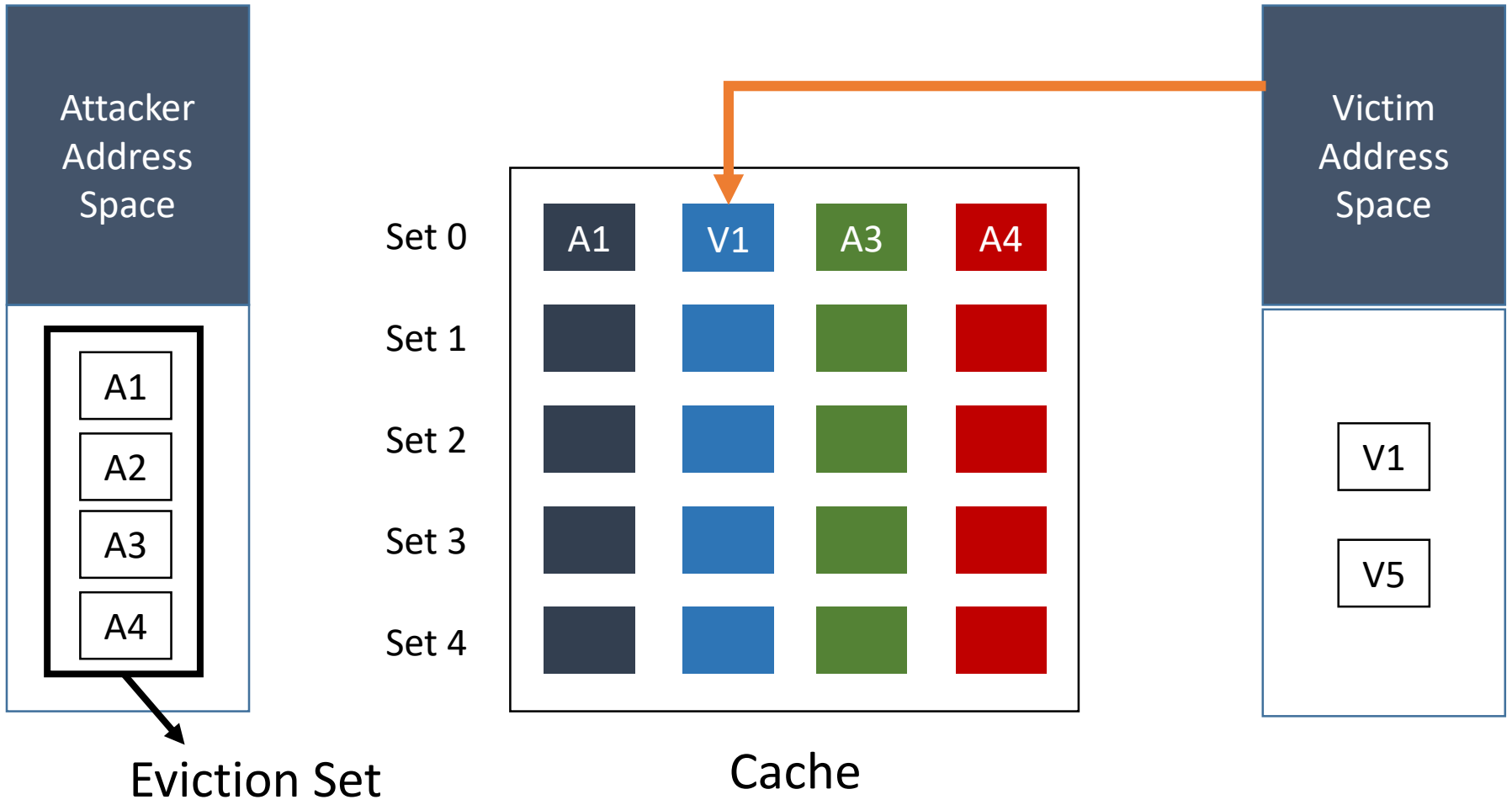


Prime+Probe Attack: Prime Step



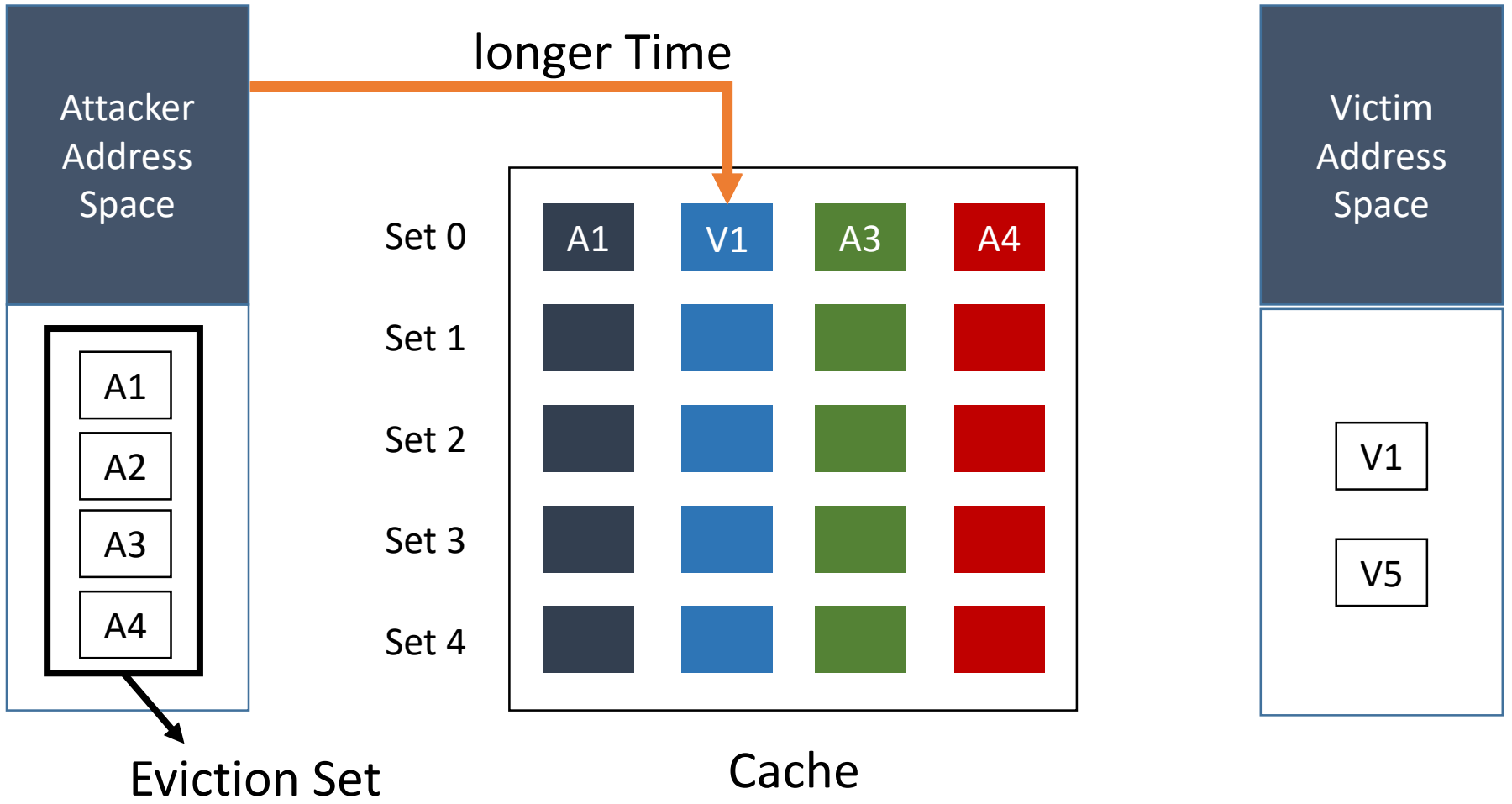
- Attacker finds the eviction set, which are the memory addresses that collide with the V1.
- Attacker fills the cache with the eviction set memory addresses by accessing them

Prime+Probe Attack: Waiting steps



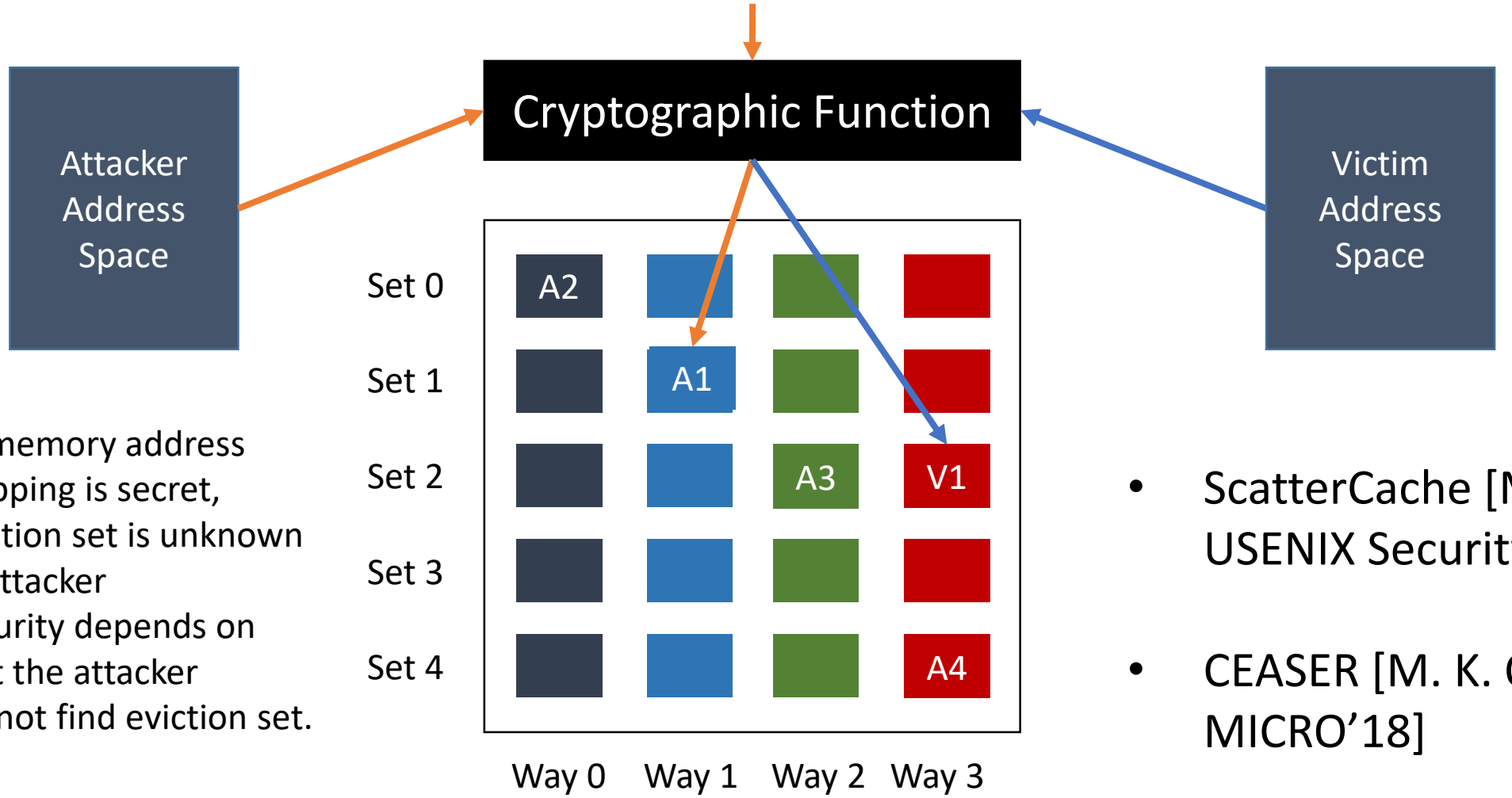
- Attacker waits for the prefixed time and call the victim to execute.
- Let say while execution victim evicts A2.

Prime+Probe Attack: Probe Step



- Attacker again check the state of cache by accessing all members of eviction set
- If attacker finds any member of eviction set evicted from the cache, he gets the information that victim has accessed the Set 0.

Prior Countermeasures



- As memory address mapping is secret, eviction set is unknown to attacker
- Security depends on that the attacker cannot find eviction set.

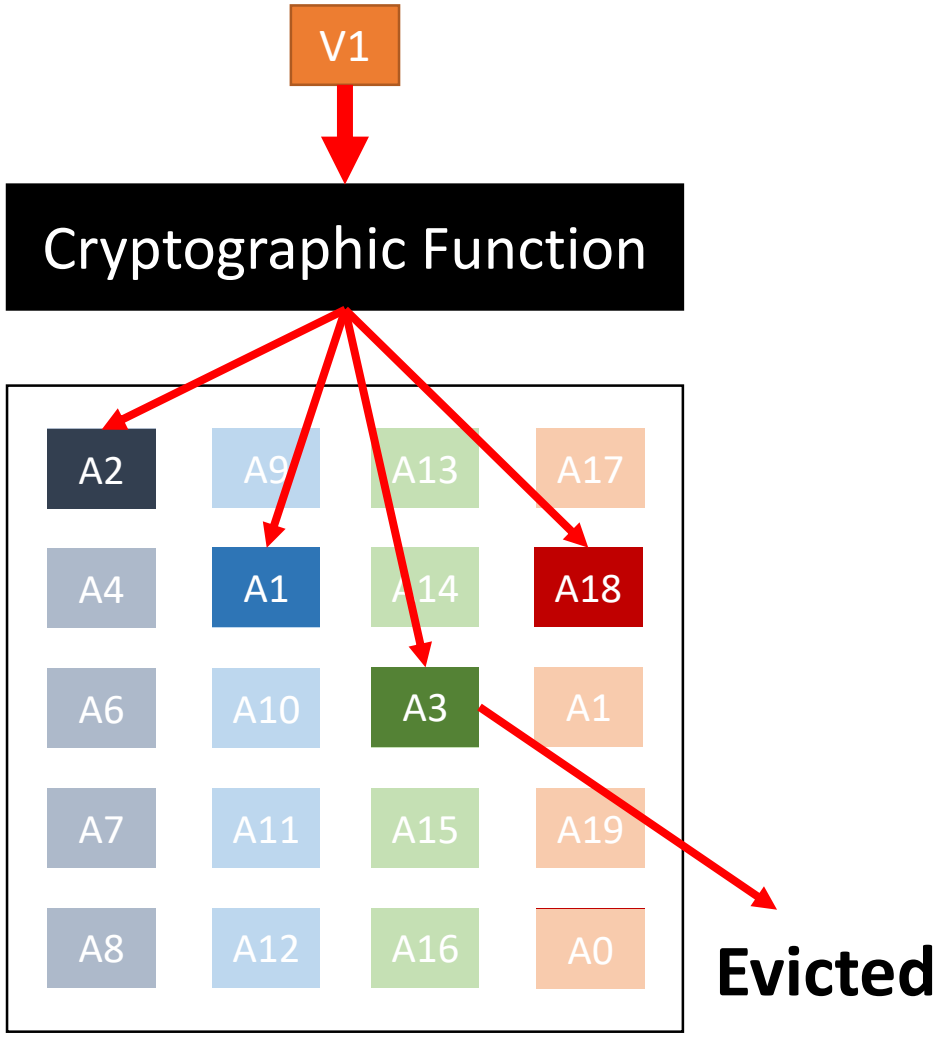
- ScatterCache [M. Werner, USENIX Security 2019]
- CEASER [M. K. Qureshi, MICRO'18]

Limitation in ScatterCache and CEASER

Prime+Prune+Probe technique can reveal eviction sets [A. Purnal et al. , S&P' 2020]

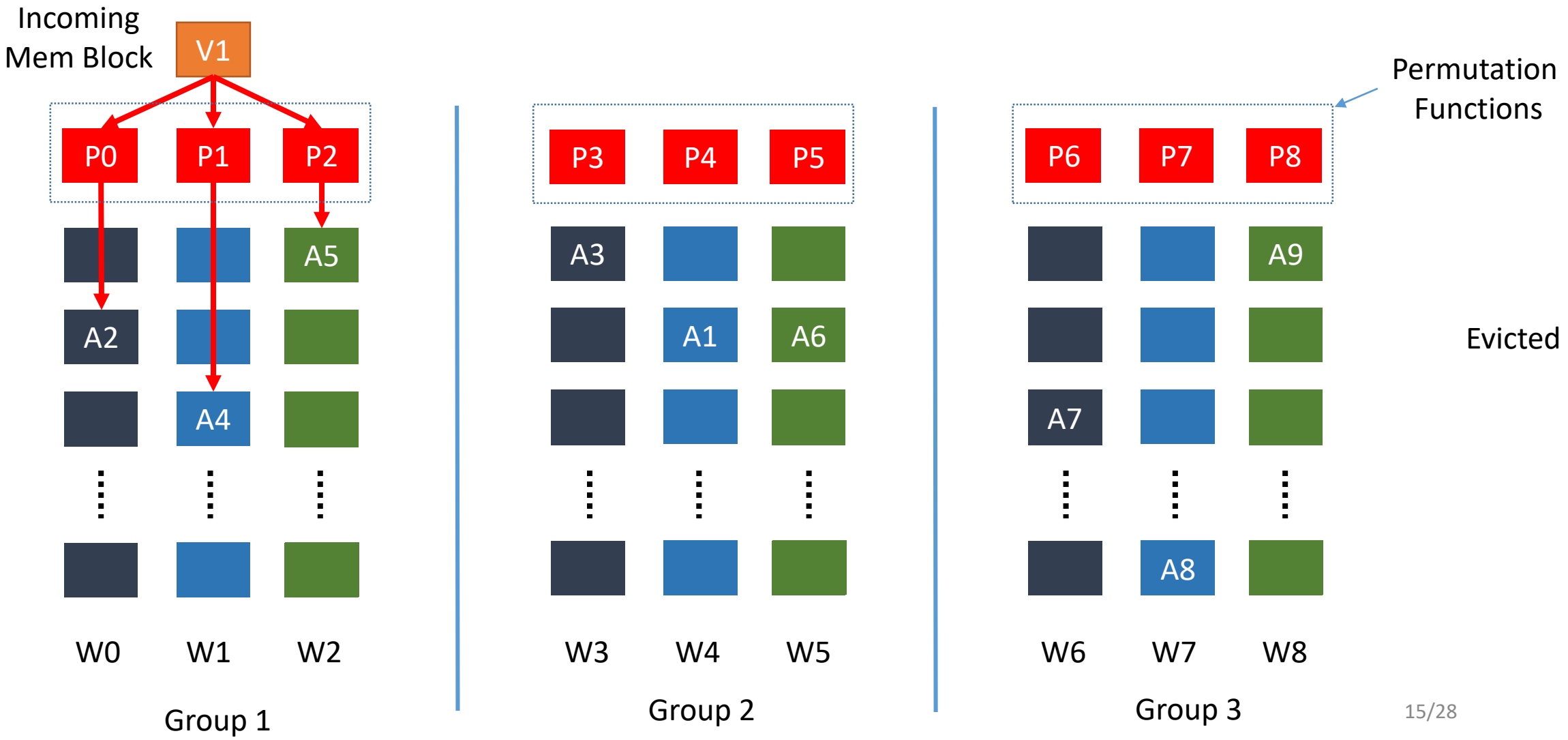
1) Prime - Attacker randomly chooses memory addresses and places them in cache.
2) Prune - Attacker ensures that all accessed addresses are in cache by re-accessing.
3) Call the victim to execute.
4) Probe - Attacker accesses again all addresses and observes access latency.

Our Perspective About Problem



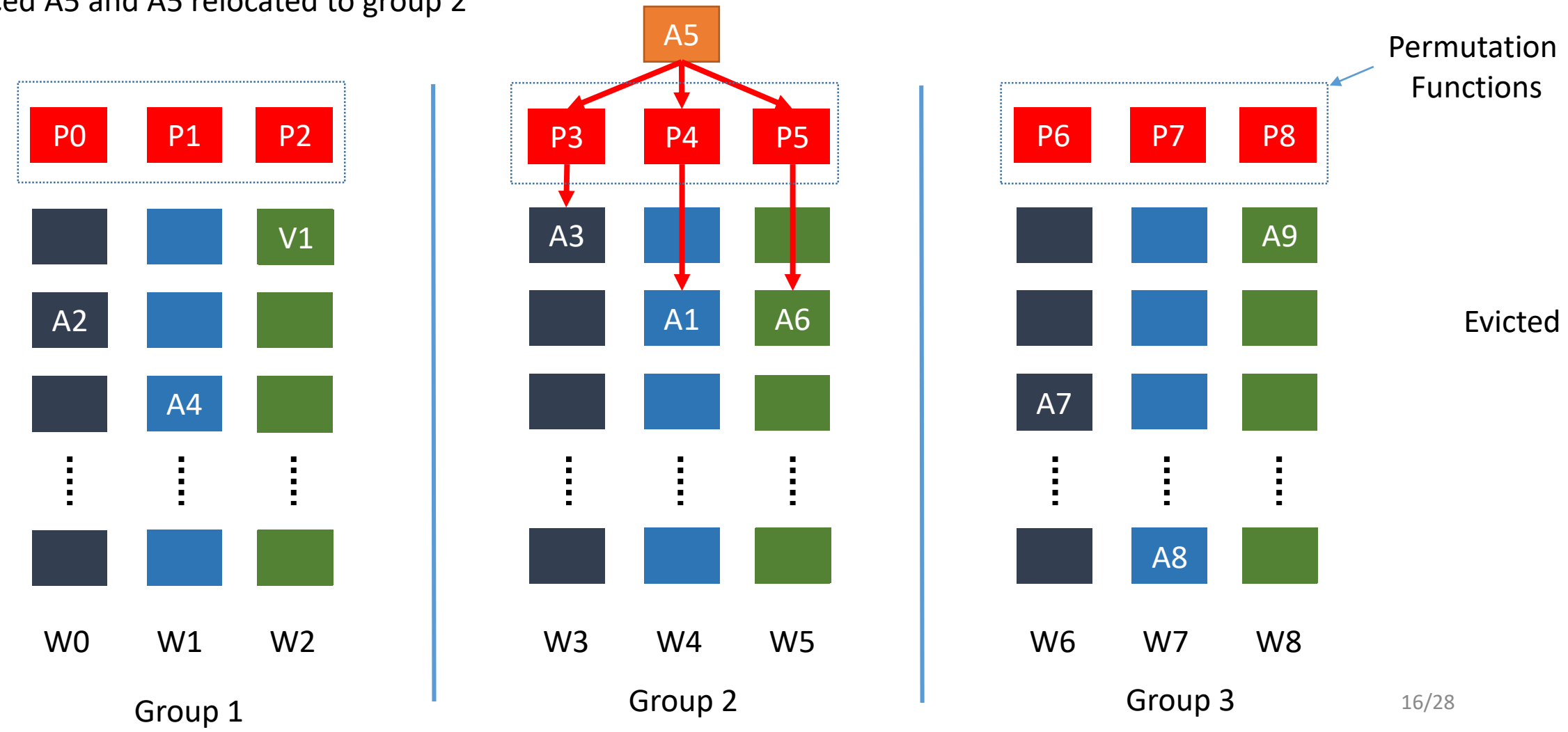
- Direct Relation Problem
- Our Hypothesis
  - Eliminating direct relation between incoming memory address (V1) and evicted cache line (A3) can make impractical for attacker to find eviction set by generating random collisions in cache.

Pcache: Realization of Direct Relation Elimination

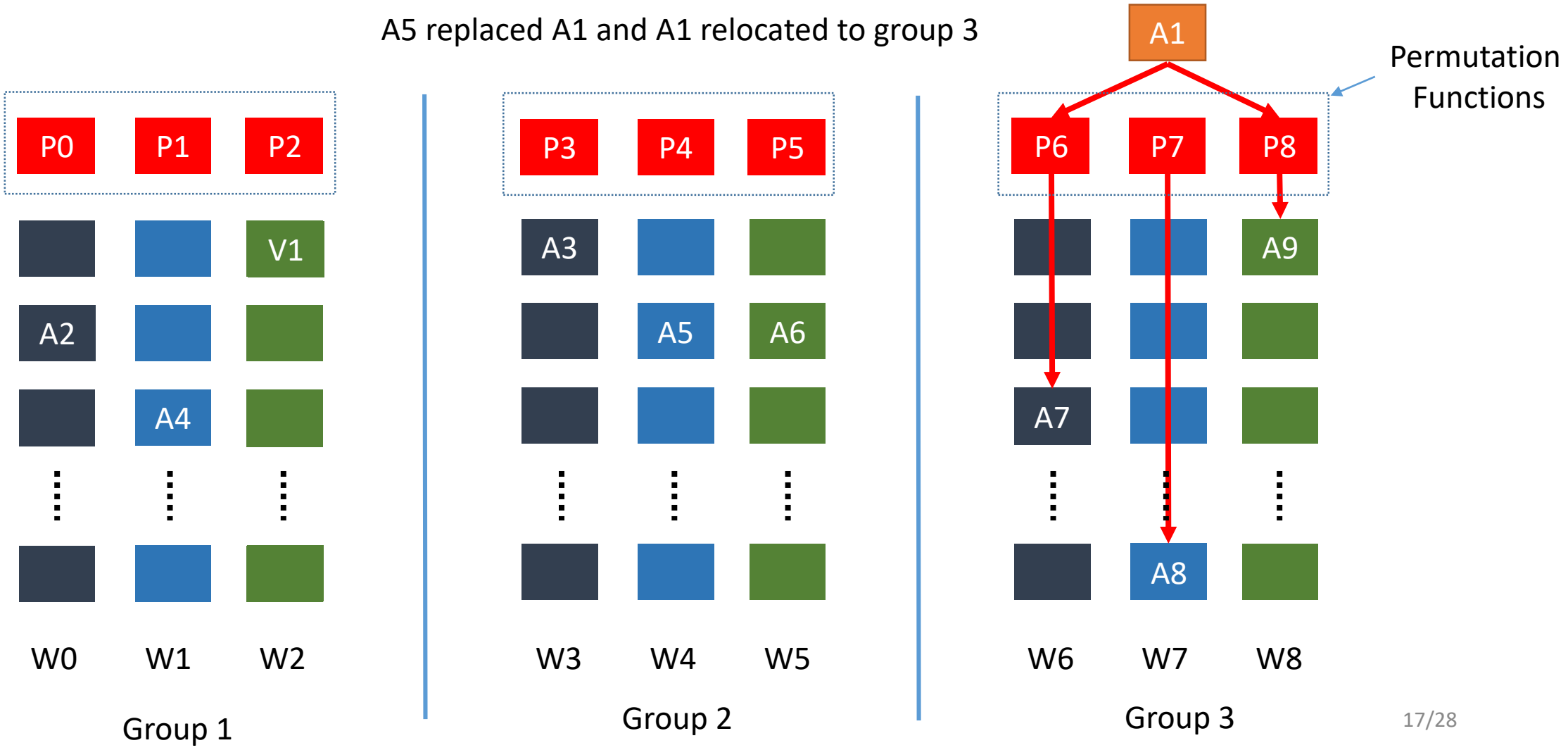


Pcache: Realization of Direct Relation Elimination

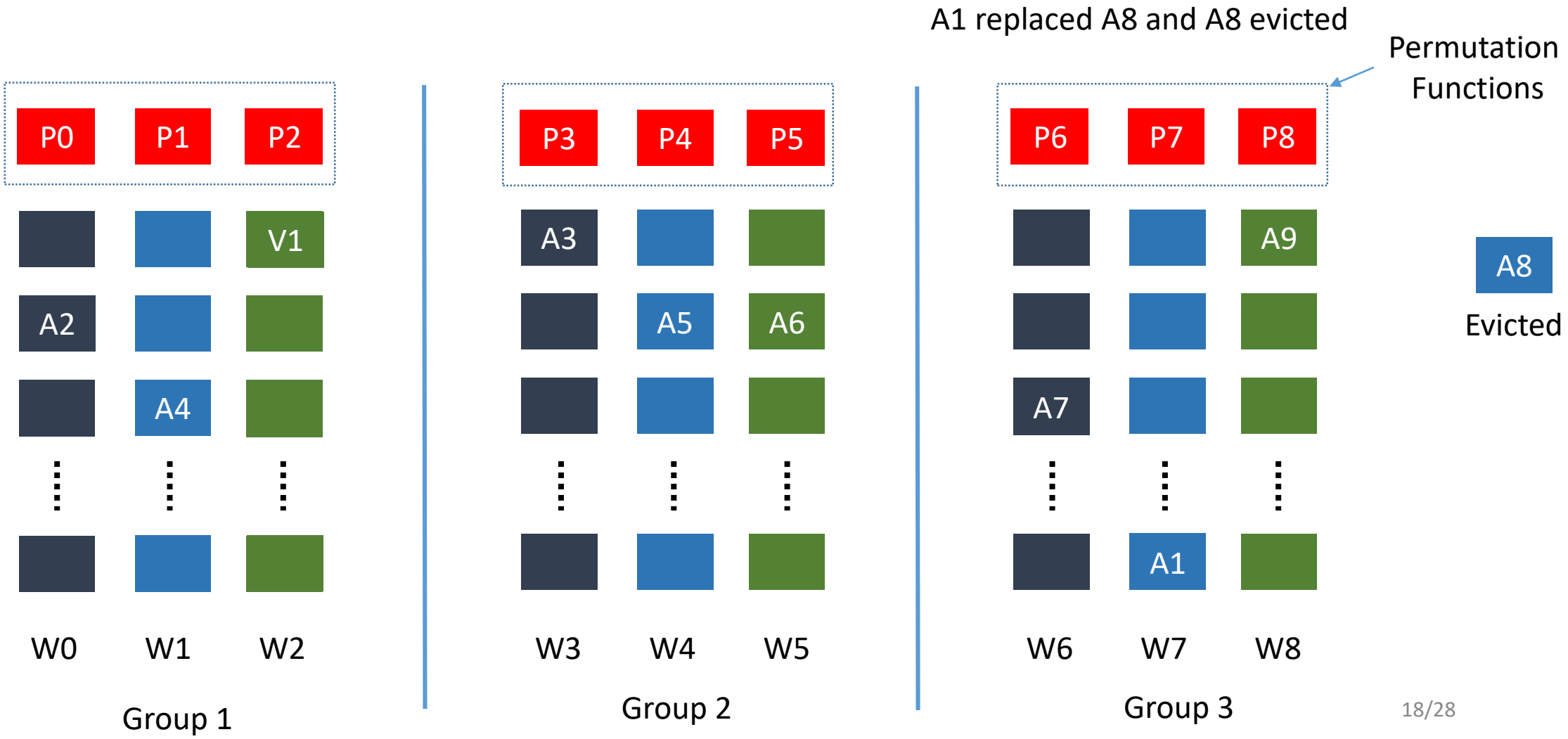
V1 replaced A5 and A5 relocated to group 2



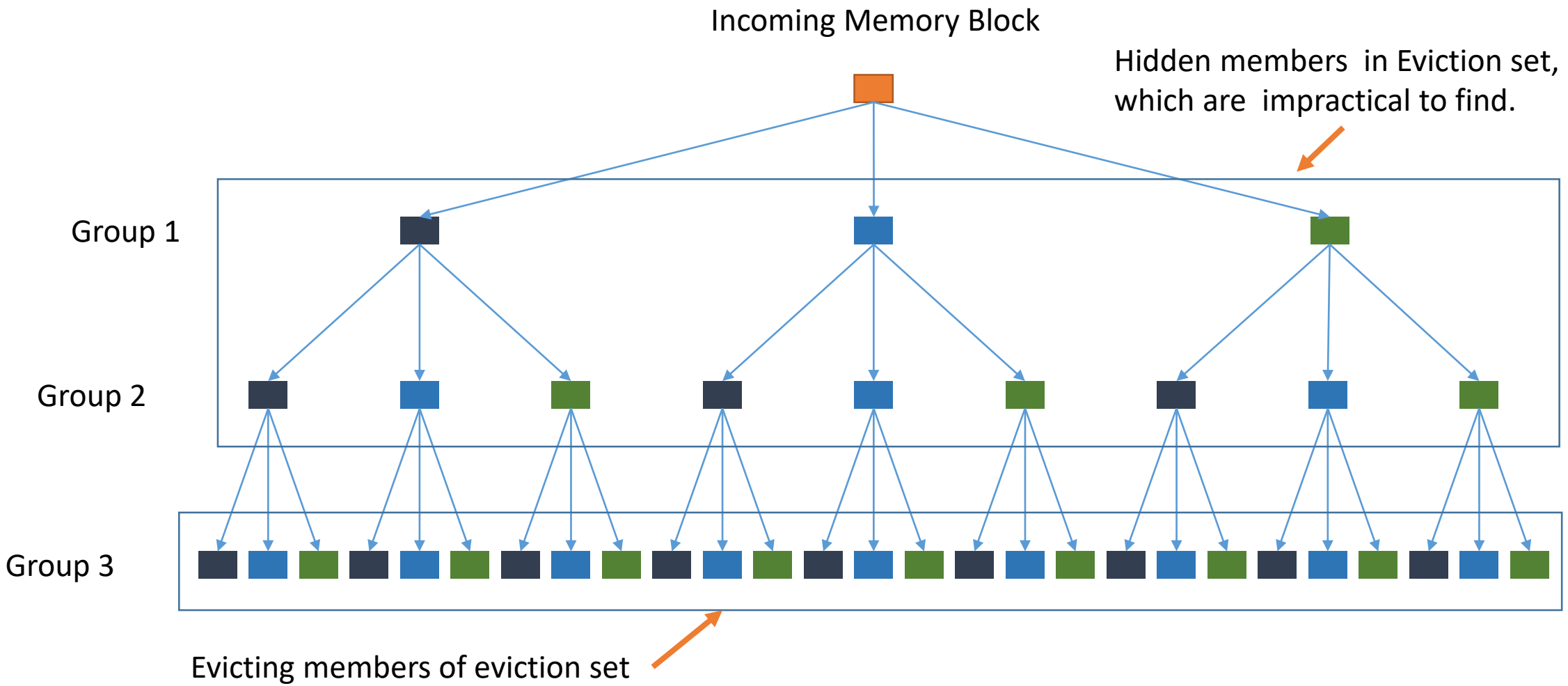
Pcache: Realization of Direct Relation Elimination



Pcache: Realization of Direct Relation Elimination



Security Perspective of PCache



Security Evaluation

- We build functional model of PCache using Python.

- We evaluated the security using
  - Prime+Prune+Probe and breaking branch technique.
  - Eviction distribution Estimation

- For time analysis, we used following data [A. Purnal et al. , S&P' 2020]
  - cache hit time = 9.5ns,
  - cache miss time = 50ns ,
  - victim execution time = 0.5ms and
  - cache flush time = 3.6ms.

Prime+Prune+Probe and Breaking Branch Technique

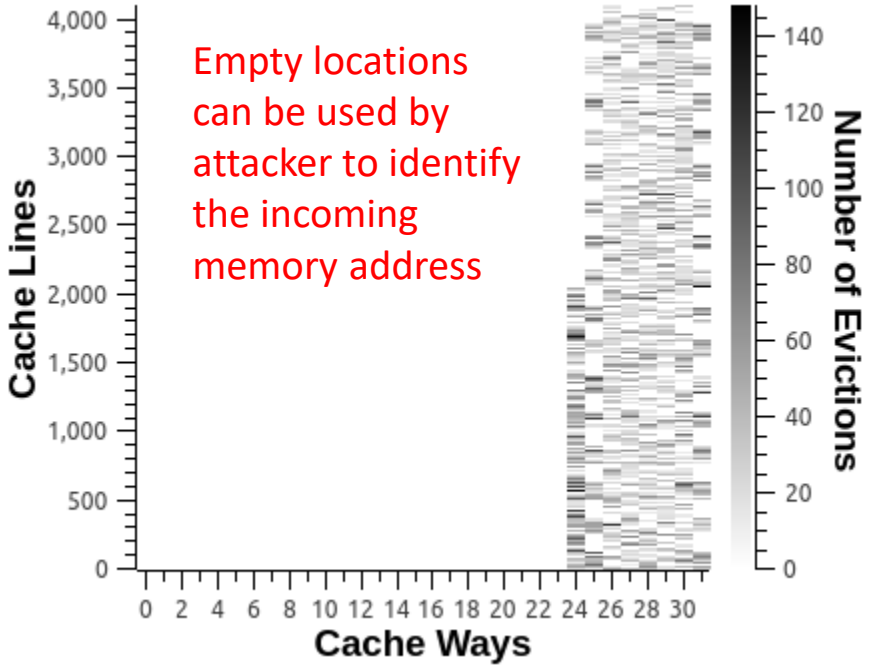
- We extracted victim and attacker access to find 1000 evicting and hidden members using Prime+Prune+Probe and breaking-branch technique. Then, we have averaged 1000 samples to form finding time of one member of eviction set.

- Following Table shows that attacker would need ≈25 months (or 2 years) to learn eviction set against one memory address in 10MB cache with 32 ways and 4 groups.

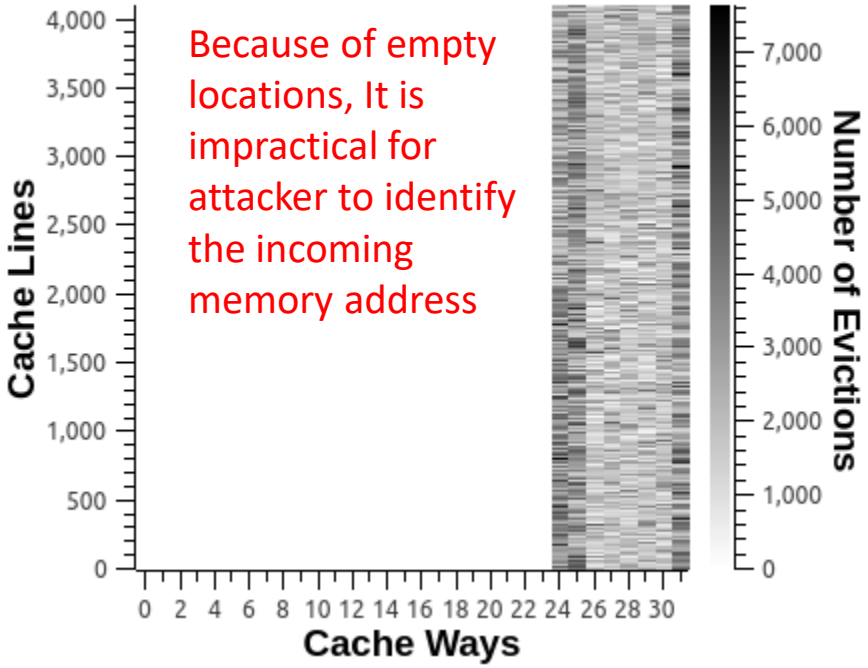
Capacity	Attacker access to find Non-Evicting Members (k)	Time to find Non-Evicting Members (hours)	Attacker access to find hidden members (k)	Time to find hidden members (hours)
1	301	0.39	113.03	613.6
8	411	1.04	919.52	12605.8
10	491	1.17	1150.68	18497.1

Eviction Distribution Estimation

Eviction distribution is developed by accessing victim memory access 18.86k times on 8MB cache with 32 ways and 4 groups .



Eviction caused by repeatedly accessing single victim memory addresses



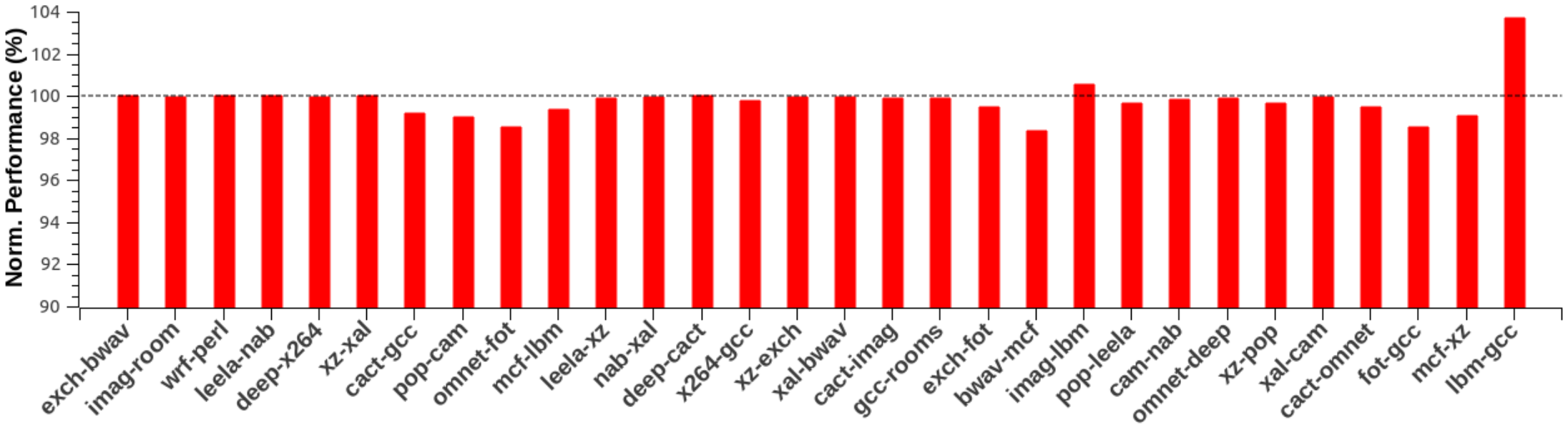
Eviction caused by repeatedly accessing 100 sequenced victim memory addresses

Performance Evaluation

- We have build the PCache in ChampSim. [D. Sanchez ISCA 2013].
- We have used weighted speed-up metric to quantify performance.
- We have normalized 32/4 PCache performance relative to baseline architecture.

Baseline Configuration	
Cores	2 cores , 2.2 GHz, OoO model
L1 Cache	Private, 32kB, 8-way set associative, split D/I
L2 Cache	Private, 256kB, 8-way set associative
L3 Cache	Shared, 8MB, 32-way set associative

Performance Results



PARSEC Benchmark 3.0

PCache with random replacement policy shows degradation as compared to baseline on some workloads but a maximum of 1:6%, and performance loss is between 1.4% to 1.6%.

Conclusion

- Our aim is to solve the problem of learning eviction sets in random memory-to-cache mapping-based cache architectures by introducing indirect members in eviction sets.
- Pcache modifies the relation between the incoming memory address and evicting cache line.
- PCache uses the principle of lazy eviction, which reduces the use of complex random indexing functions and improves performances along with strong security.
- Our security evaluation showed that the attacker has to generate at least approx. 2^{59} memory accesses to learn an eviction set using random collision studies.
- Our experimental evaluation showed that performance is improved by 1% compared to set-associative cache with random replacement policy on CPU2017 benchmarks.

PCache: Permutation-based Cache to Counter Eviction-based Cache-based Side-Channel Attacks

M. Asim Mukhtar [1] M. Khurram Bhatti [1] Guy Gogniat [2]

1 Information Technology University, Lahore, Pakistan

2 University of South Brittany, Lorient, France