IEEE SecDev | 2020

IEEE COMPUTER SOCIETY

ReViCe: Reusing Victim Cache to Prevent Speculative Cache Leakage

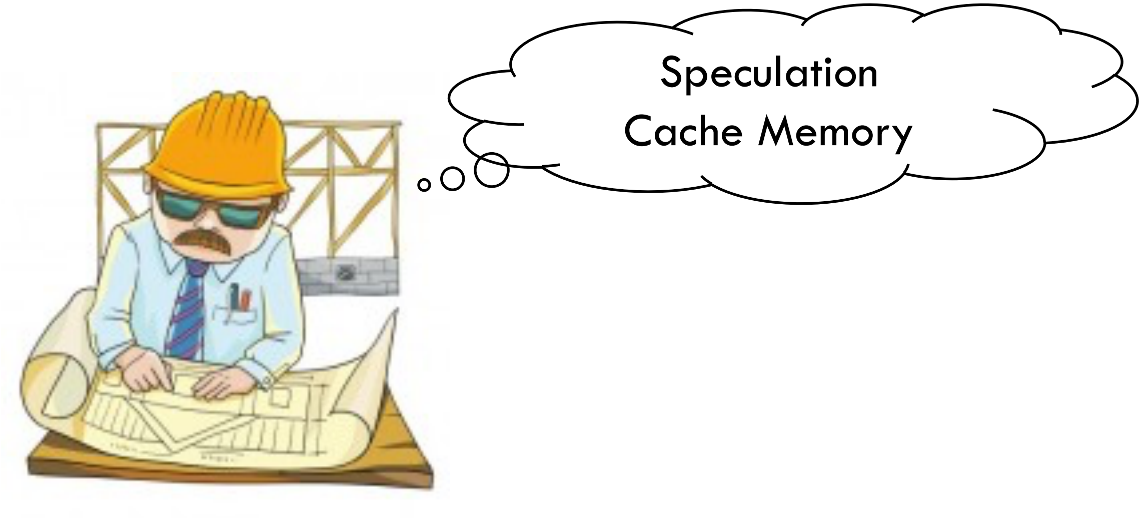
Sungkeun Kim, Farabi Mahmud, Jiayi Huang, Pritam Majumder, Neophytos Christou*, Abdullah Muzahid, Chia-Che Tsai, Eun Jung Kim

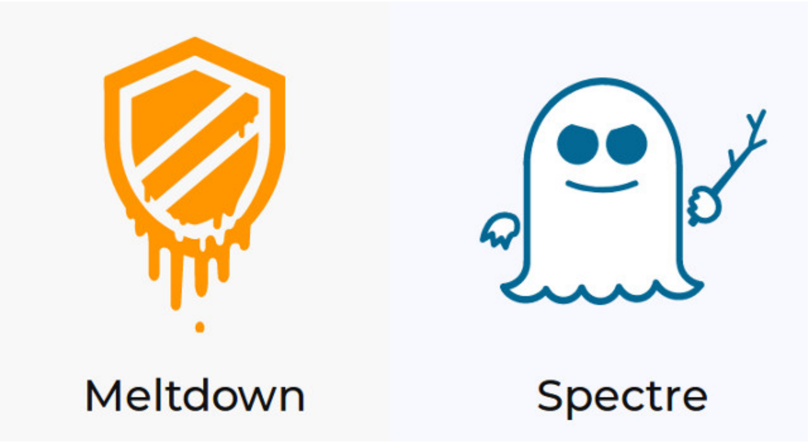
Texas A&M University *University Of Cyprus

Vulnerable Performance Optimization

□ Attackers can access the secret through speculative execution.

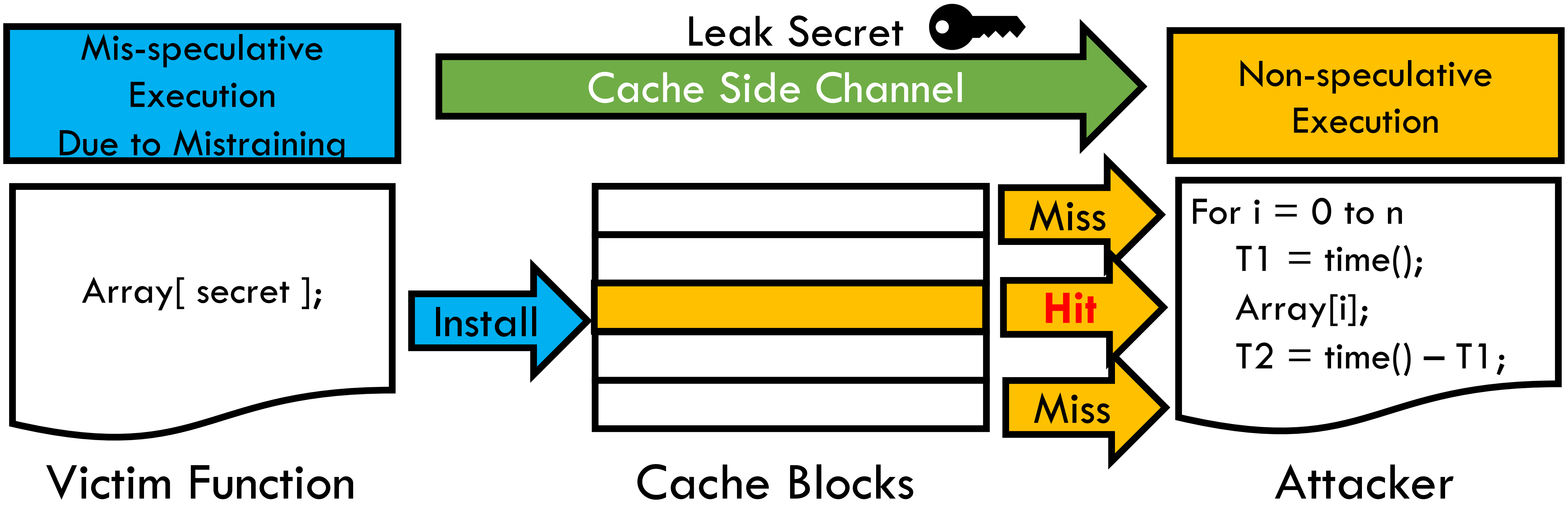
□ Attackers transmit the secret through cache side channel.





Problem: Speculation Based Attacks (Spectre V1)

Meltdown Spectre



Mis-speculative Execution Due to Mistraining

Leak Secret

Cache Side Channel

Non-speculative Execution

Array[secret];

Install

Miss

Hit

Miss

```
For i = 0 to n
    T1 = time();
    Array[i];
    T2 = time() – T1;
```

Victim Function

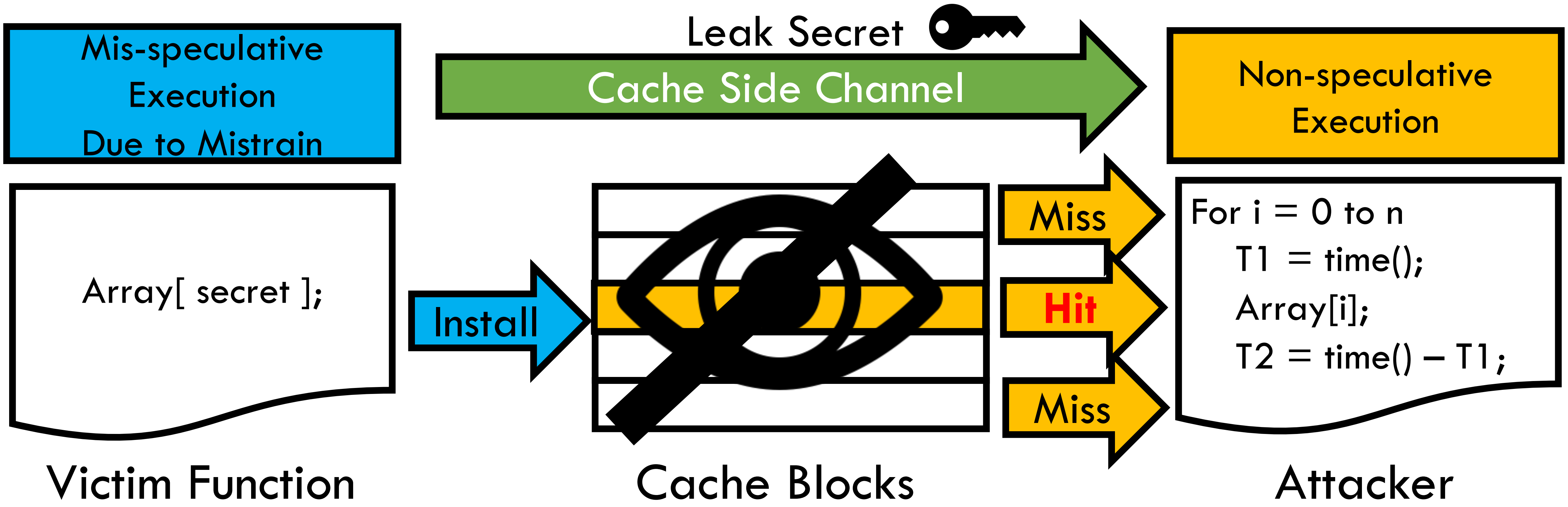
Cache Blocks

Attacker

Solution: ReViCe - An Undo-Based Mitigation

Meltdown

Spectre



Mis-speculative Execution Due to Mistrain

Leak Secret

Cache Side Channel

Non-speculative Execution

Array[secret];

Install

Miss

Hit

Miss

```
For i = 0 to n
    T1 = time();
    Array[i];
    T2 = time() – T1;
```

Victim Function

Cache Blocks

Attacker

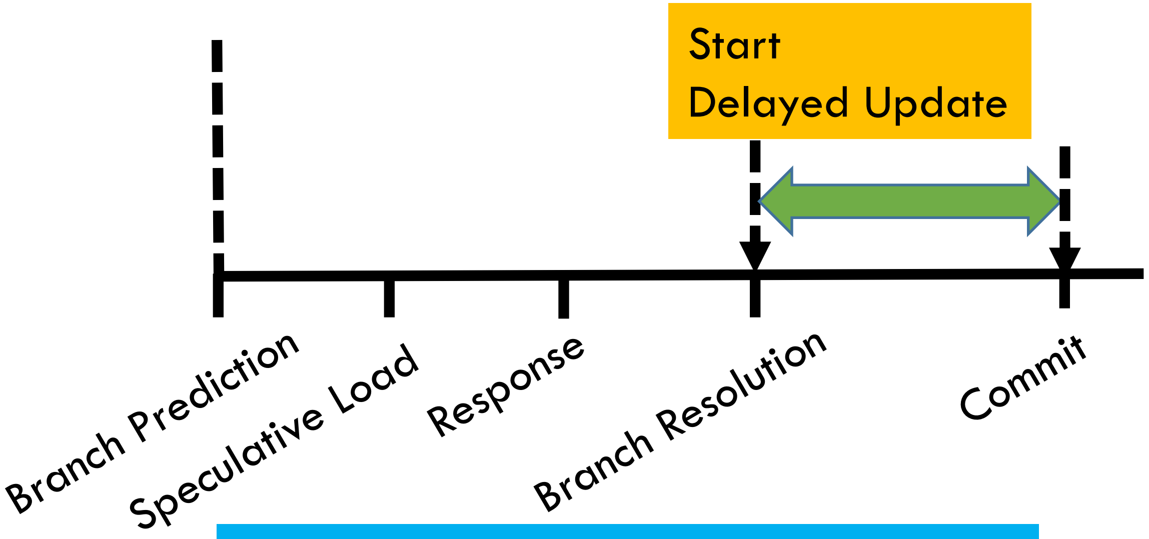
Allow Early Update
Hide Using Jitter

Restore
Using Victim Cache

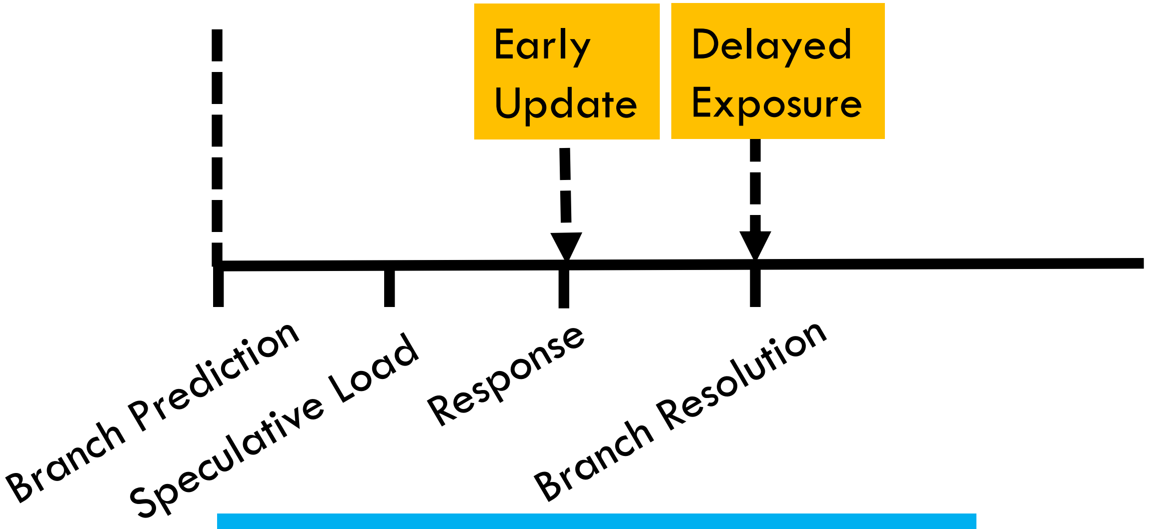
ReViCe – Motivations

Prior work – Redo VS. Undo

Prediction is very accurate



Redo - InvisiSpec [Yan et al. MICRO `18]



Undo - [Saileshwar et al. MICRO `19]

□ Delay update until Branch resolution

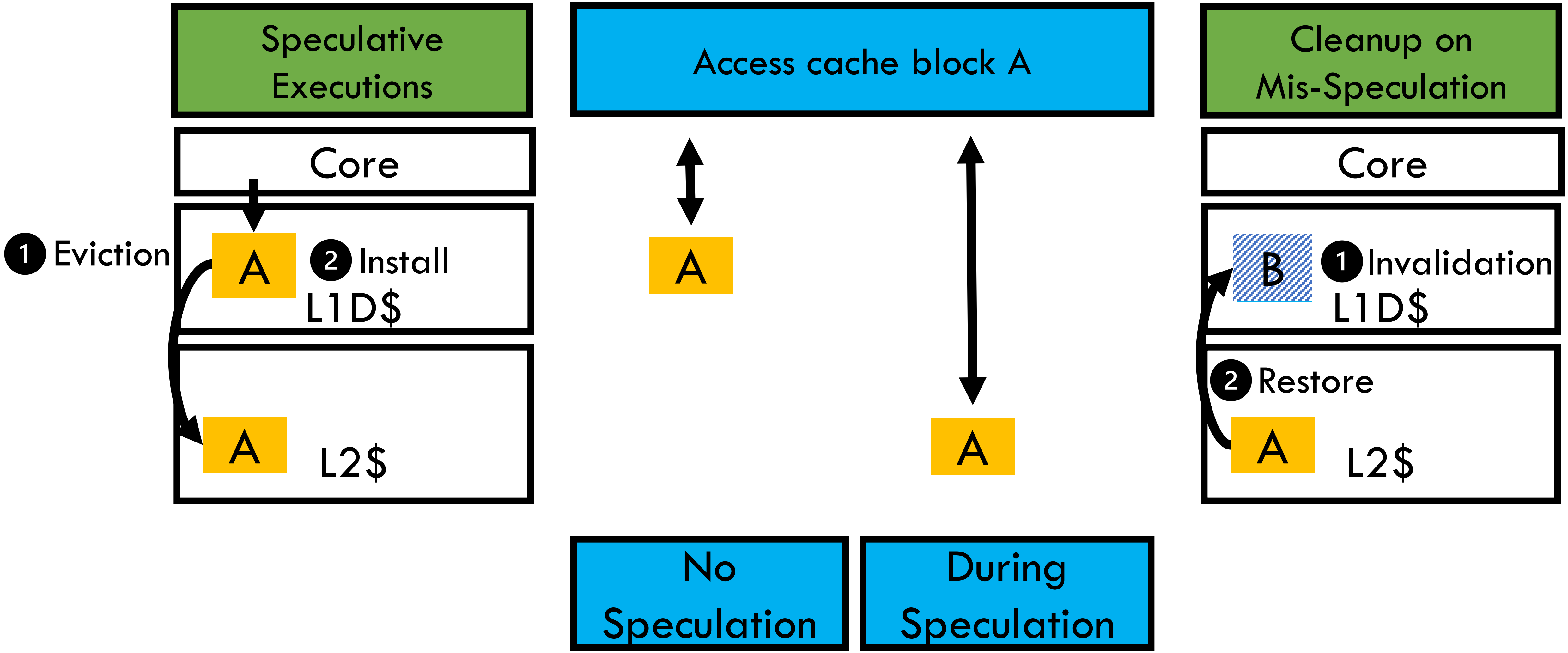
□ Penalized by correctly speculated load.

□ Early Update on Response

□ Penalized by incorrectly speculated load.

Prior work – CleanupSpec [Saileshwar et al. MICRO `19]

Appendix A



Speculative Executions

Core

1 Eviction

A

2 Install

L1D$

A

L2$

Access cache block A

A

A

No Speculation

During Speculation

Cleanup on Mis-Speculation

Core

B

1 Invalidation

L1D$

2 Restore

A

L2$

Threat model

- Mis-Speculative load can access the secret.
- Cache side channel transmits the secret.
- Attacker has access to the source code of the victim program
- OS is correct and trusted by the victim.

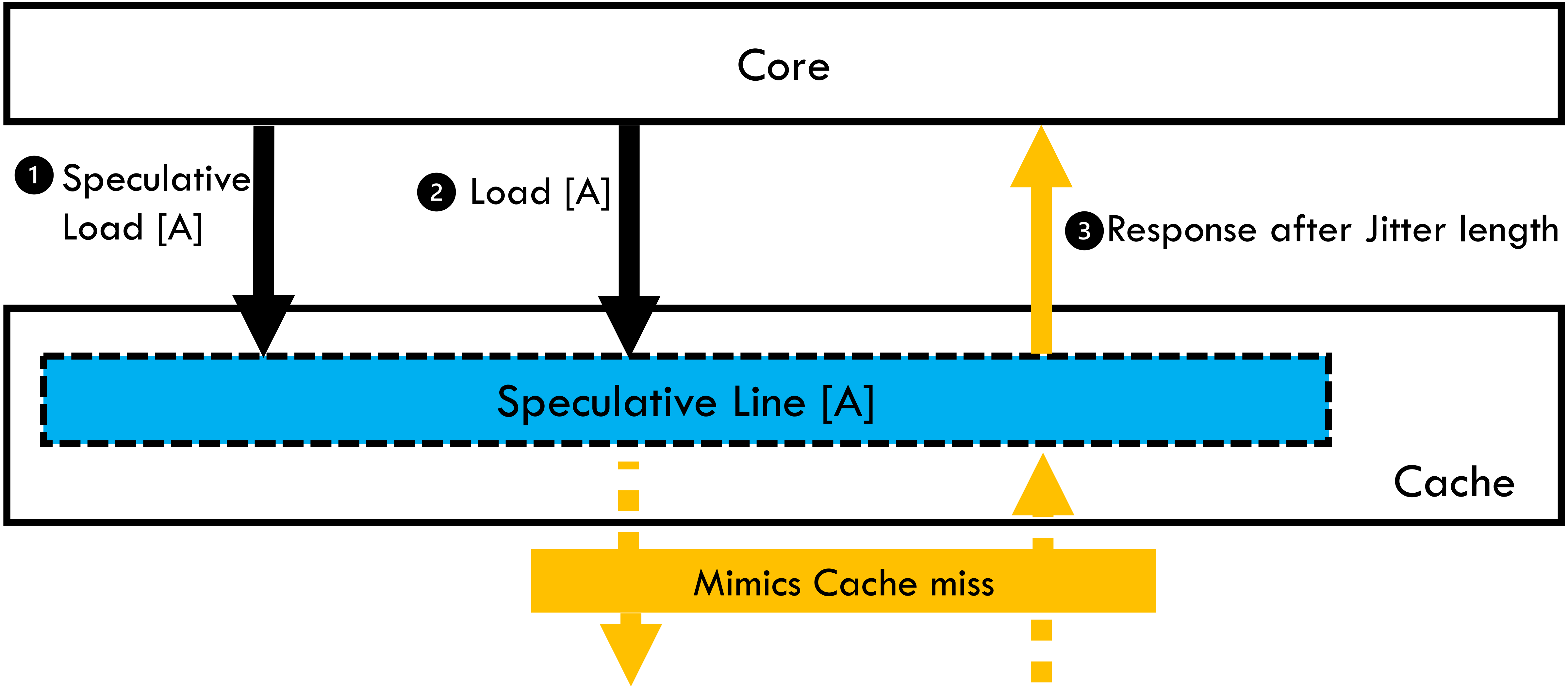
- Out of Scope
  - Other side channels: TLB, Branch Prediction History
  - Foreshadow

ReViCe - Design

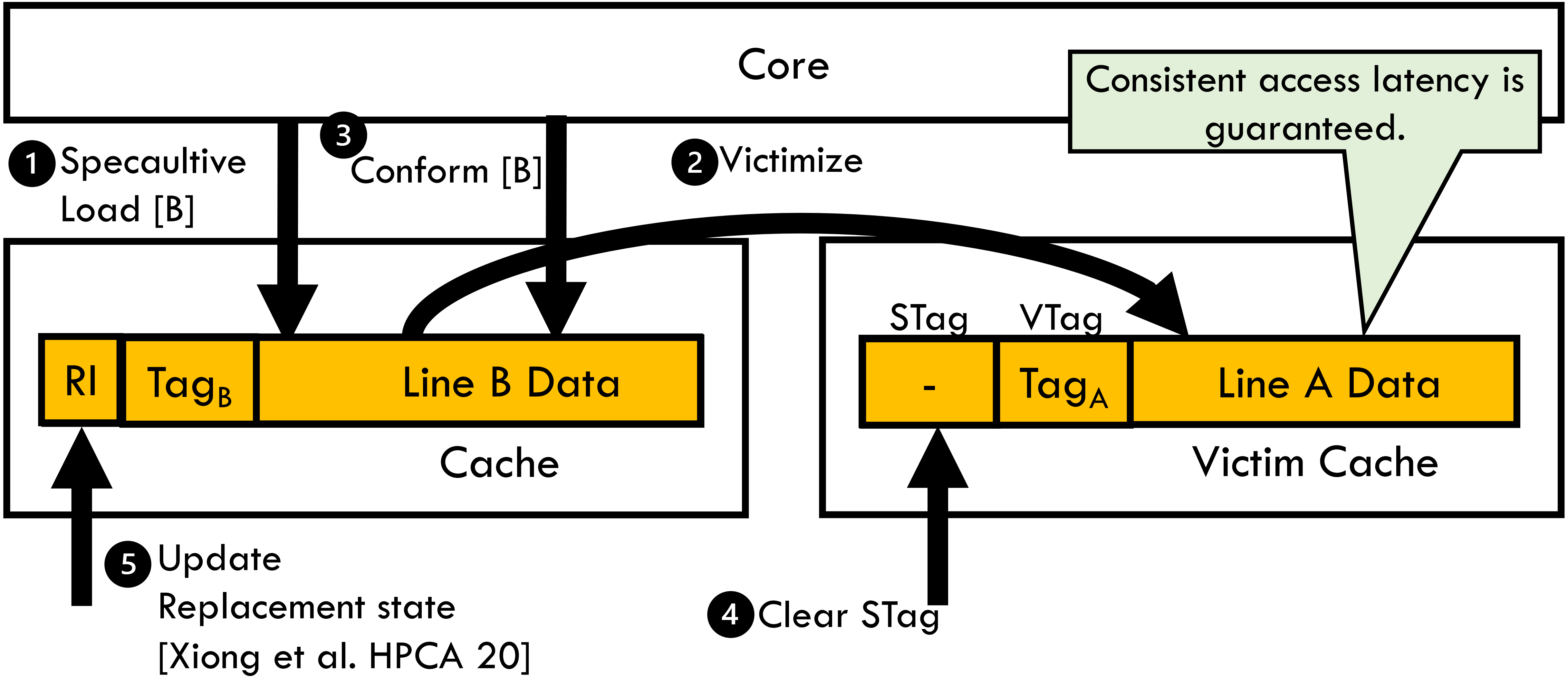
Jitter – Mimics cache miss to hide speculation

Allow Early Update
Hide Using Jitter

Restore
Using Victim Cache



Victim Cache – Confirm Correct Speculative Changes



Core

Consistent access latency is guaranteed.

1 Specaultive Load [B]

3 Conform [B]

2 Victimize

RI | Tag_B | Line B Data

Cache

STag | VTag

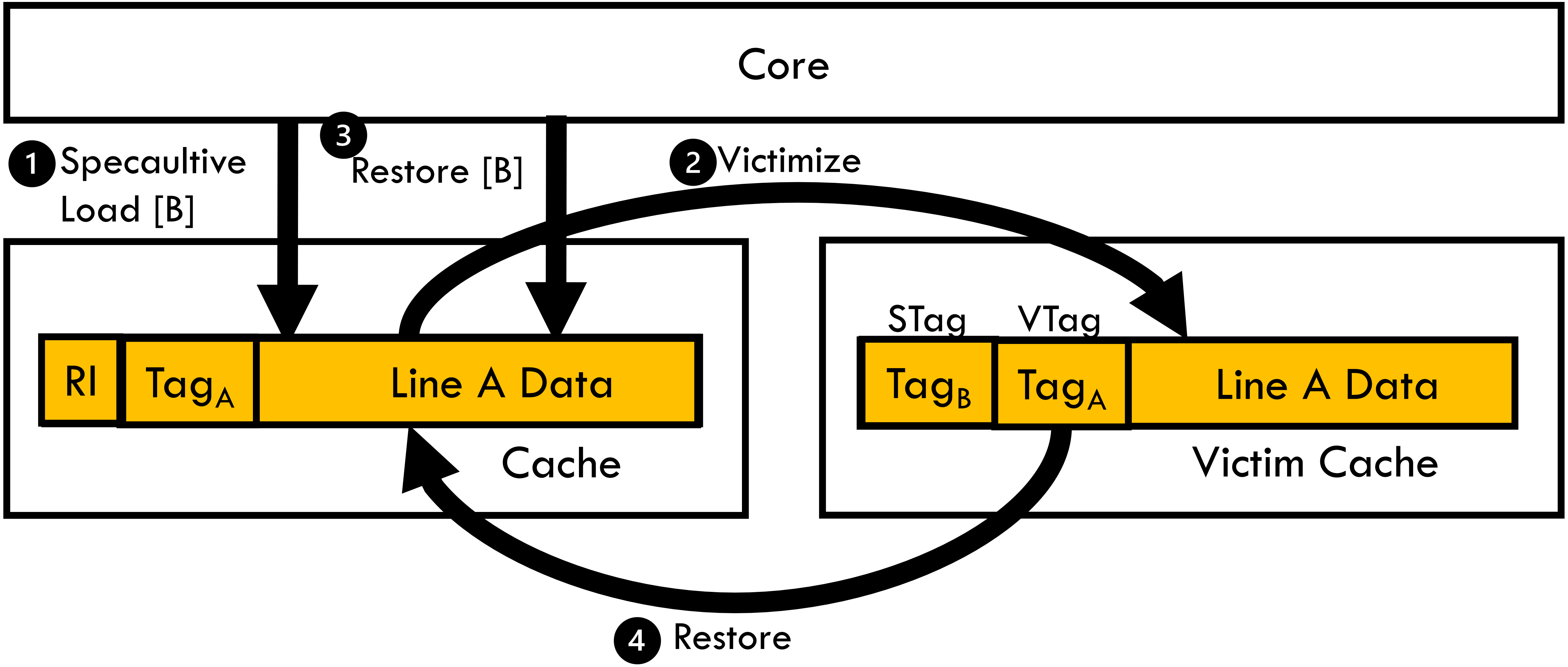
- | Tag_A | Line A Data

Victim Cache

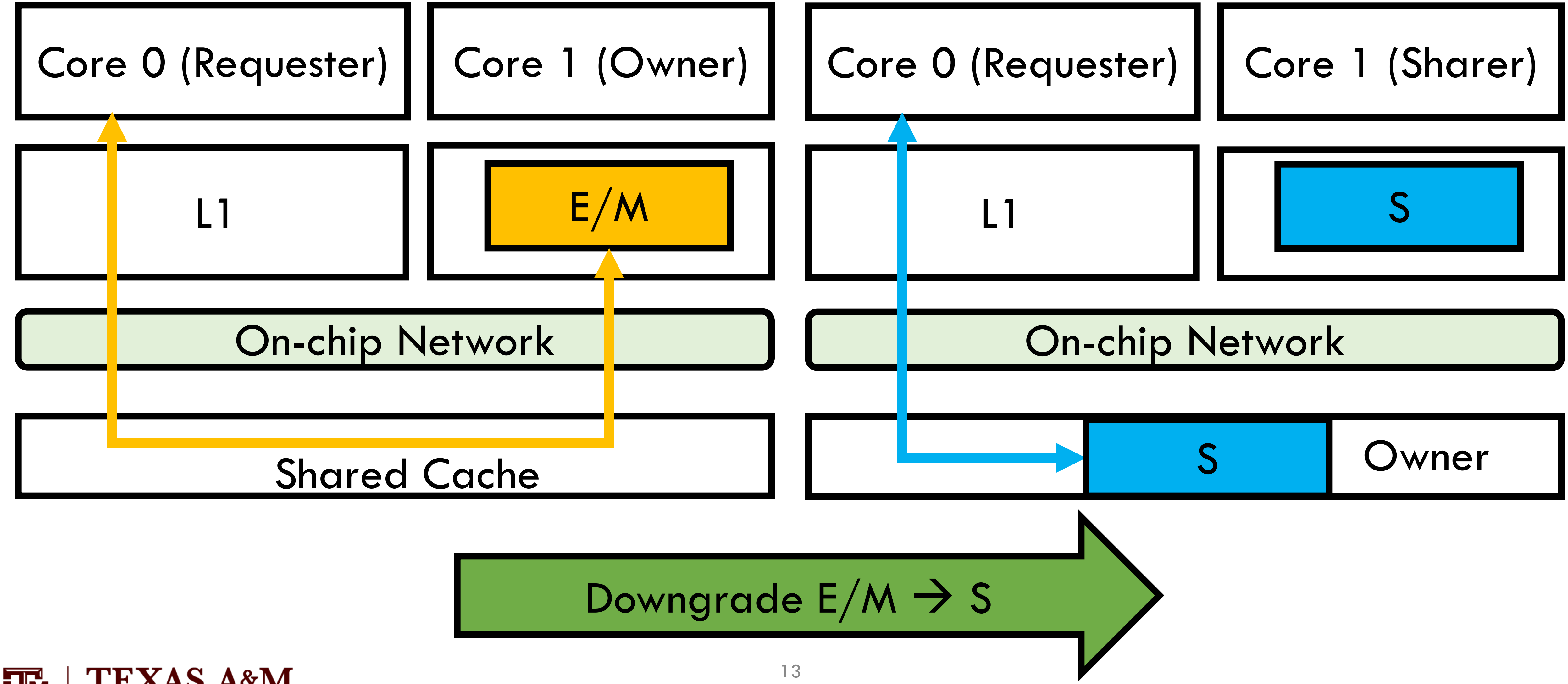
5 Update Replacement state [Xiong et al. HPCA 20]

4 Clear STag

Victim Cache – Restore Speculative Changes



Delayed Downgrade Coherence State [Yao et al. HPCA `18]



ReViCe – Evaluation

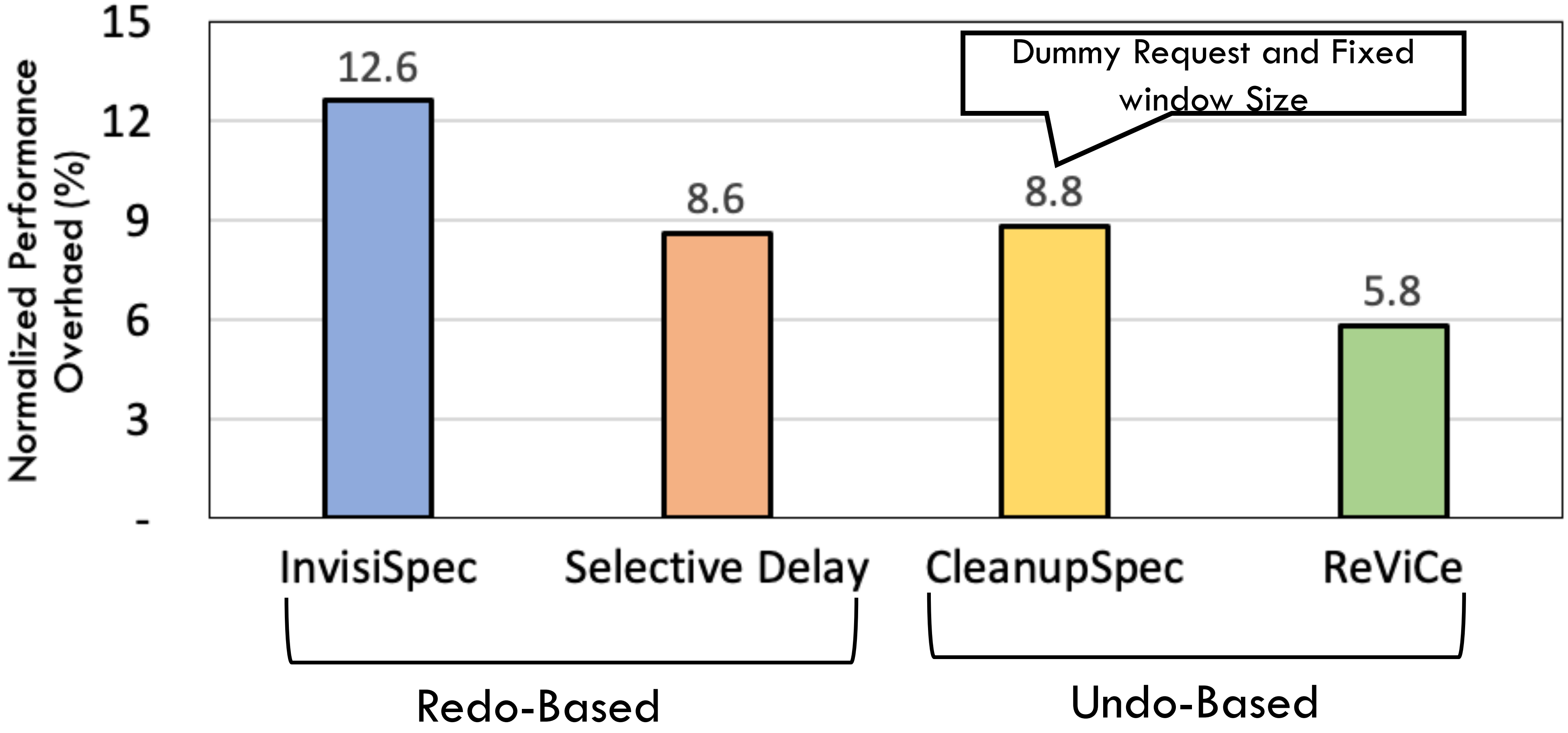
ReViCe – Evaluation Methodology

PoC and Performance

- Simulation based
  - gem5 full system simulator
  - Out of order processor (Single, Octa cores)
- Proof-of-concept (4 x 3 x 2 = 24 attack programs)
  - Four Spectre Variants
  - Three Cache Side Channels
  - Same Core and Cross Cores
- Performance evaluation
  - SPEC2017, PARSEC
  - Compared against InvisiSpec, Selective Delay, CleanupSpec

ReViCe – Performance Overhead (SPEC2017)

Details in the paper



Dummy Request and Fixed window Size

ReViCe – Conclusion

- Problem: Mitigating Speculation based attack leveraging cache side channel.
- Prior works: Either high overhead or incomplete
- Key insights: Hide speculation using Jitter and Restore from Victim Cache.
- ReViCe is secure with better performance.

Thank you

Sungkeun Kim

ksungkeun84@tamu.edu