Binoculars: Contention-Based Side-Channel Attacks Exploiting the Page Walker

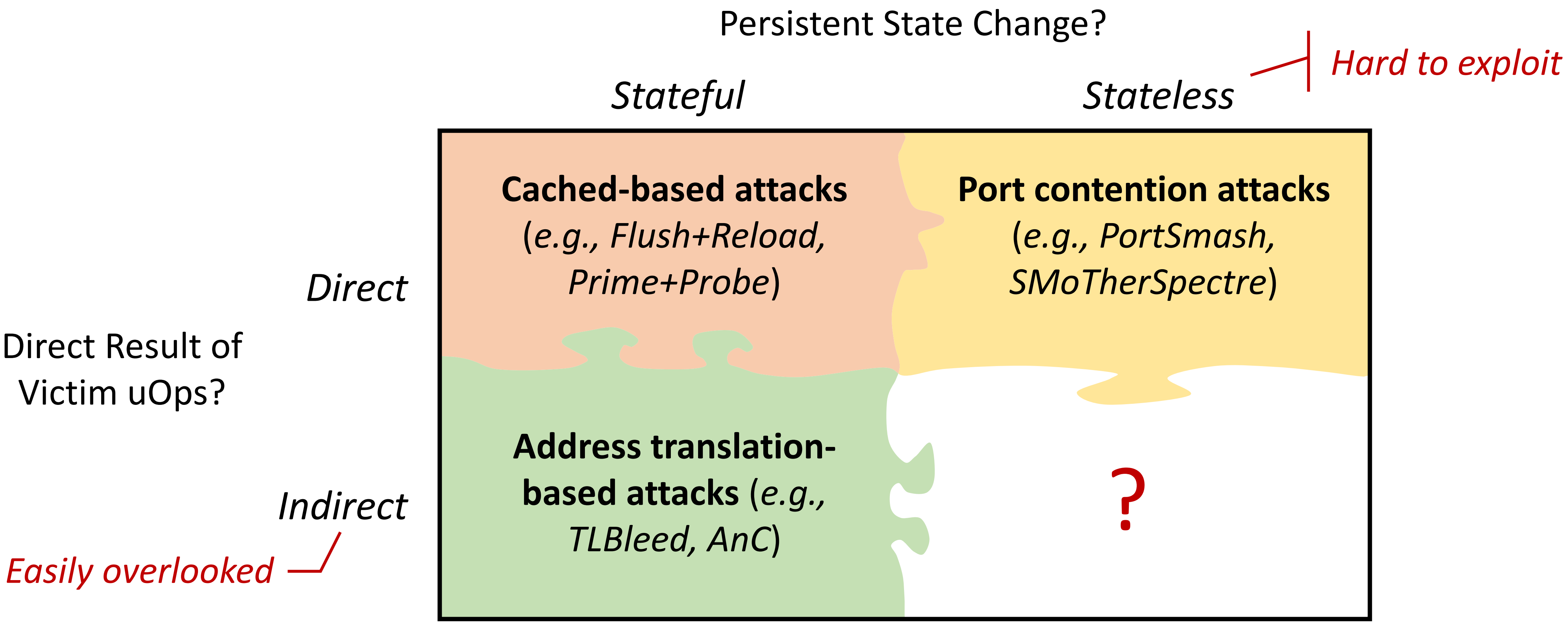
Zirui Neil Zhao, Adam Morrison, Christopher W. Fletcher, Josep Torrellas

University of Illinois Tel Aviv University

USENIX Security'22

Microarchitectural Side Channel Attacks

Root Cause: shared hardware resource between the victim and the attacker



Microarchitectural Side Channel Attacks

Root Cause: shared hardware resource between the victim and the attacker

Persistent State Change?

Direct Result of Victim uOps?	*Stateful*	*Stateless*
Direct	Cached-based attacks (*e.g., Flush+Reload, Prime+Probe*)	Port contention attacks (*e.g., PortSmash, SMoTherSpectre*)
Indirect	Address translation-based attacks (*e.g., TLBleed, AnC*)	**Binoculars Attack** (This Work)

The 1st Stateless-Indirect Channel: Binoculars

Binoculars Attack
(This Work)

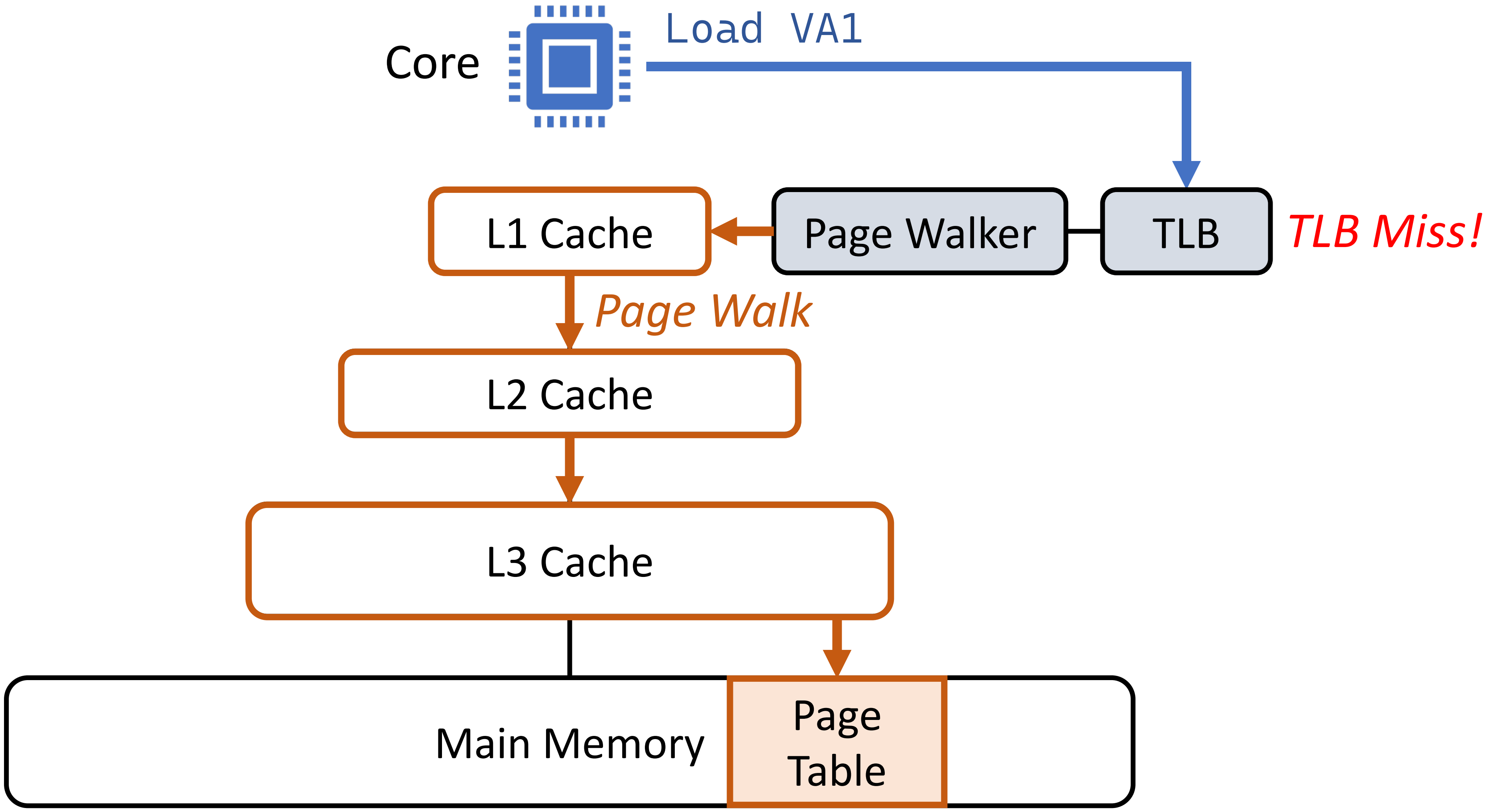
*The contention is NOT due to cache footprint

Highlights:

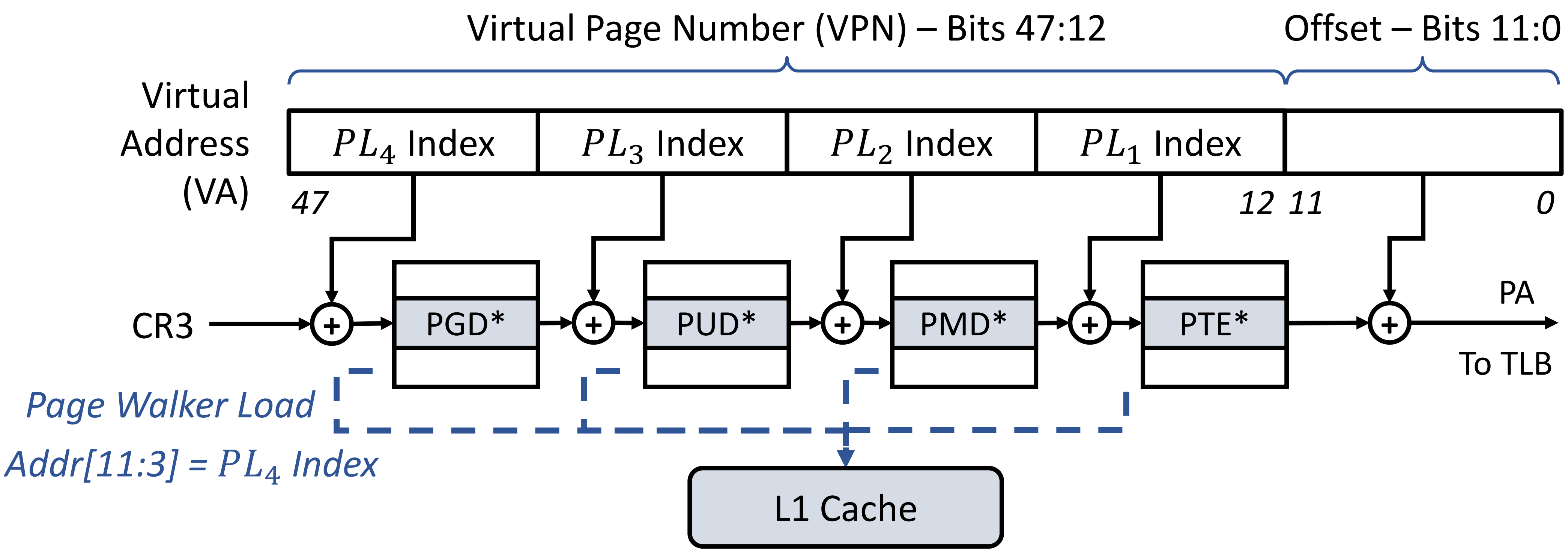
+ Easy to observe, up to 20K-cycle contention (with a single dynamic instruction)

+ Leak a wide range of virtual address bits

Virtual Address Translation & Page Walk

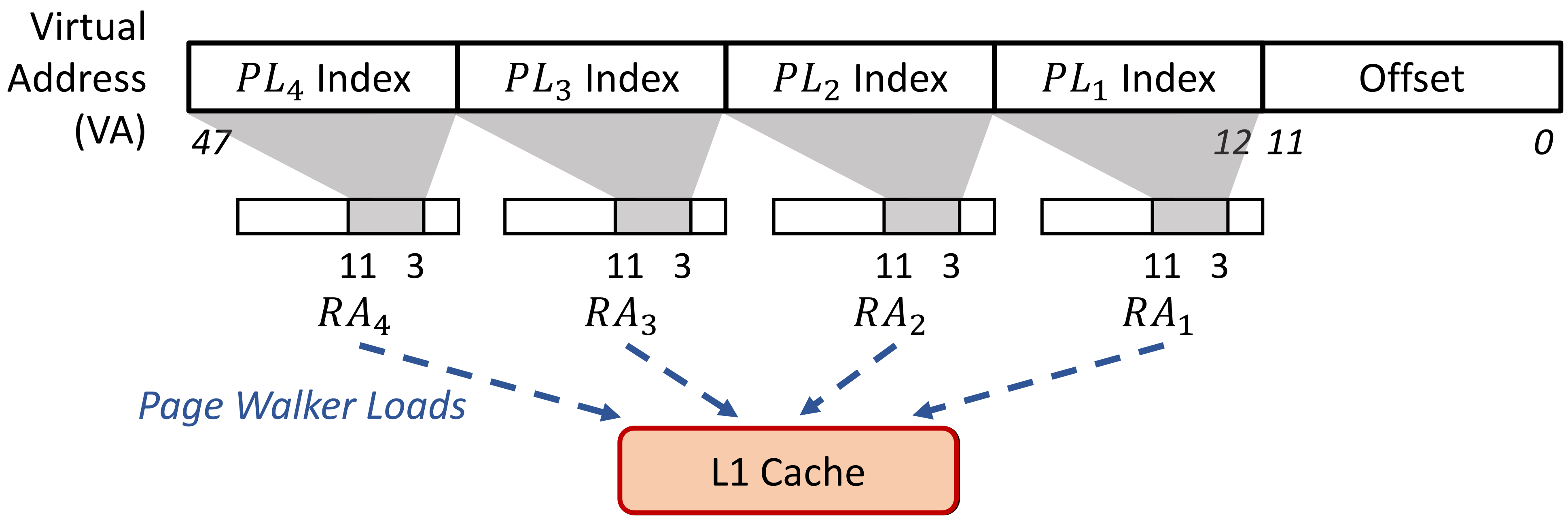


Virtual Address Translation & Page Walk (*x64*)



*Each page-table entry is 8-byte long

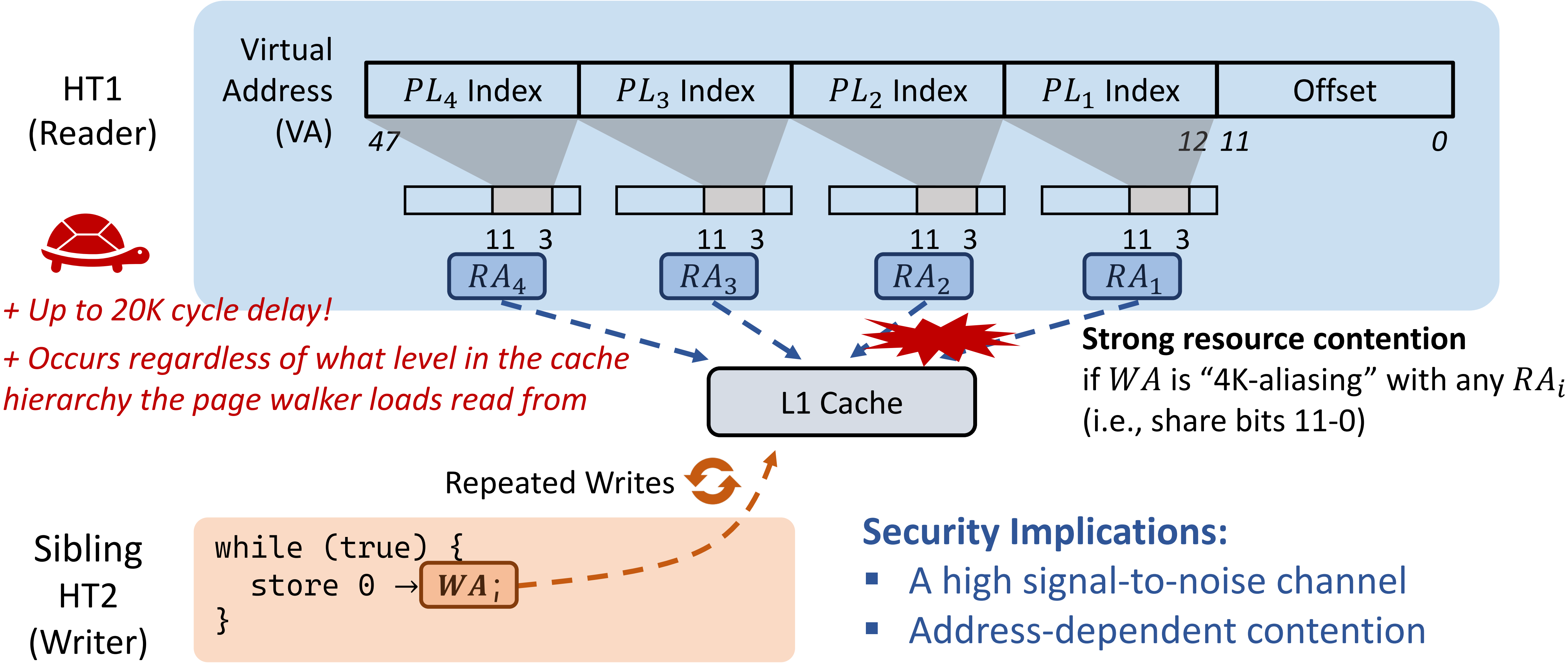
Page Walk Simplified



Three key takeaways:

- Page walker issues multiple (e.g., four) ***page walker loads***
- Address bits 11-3 of a page walker load ***is determined by its corresponding PL Index***
- Page walker loads go through the cache hierarchy and are ***subject to resource contention***

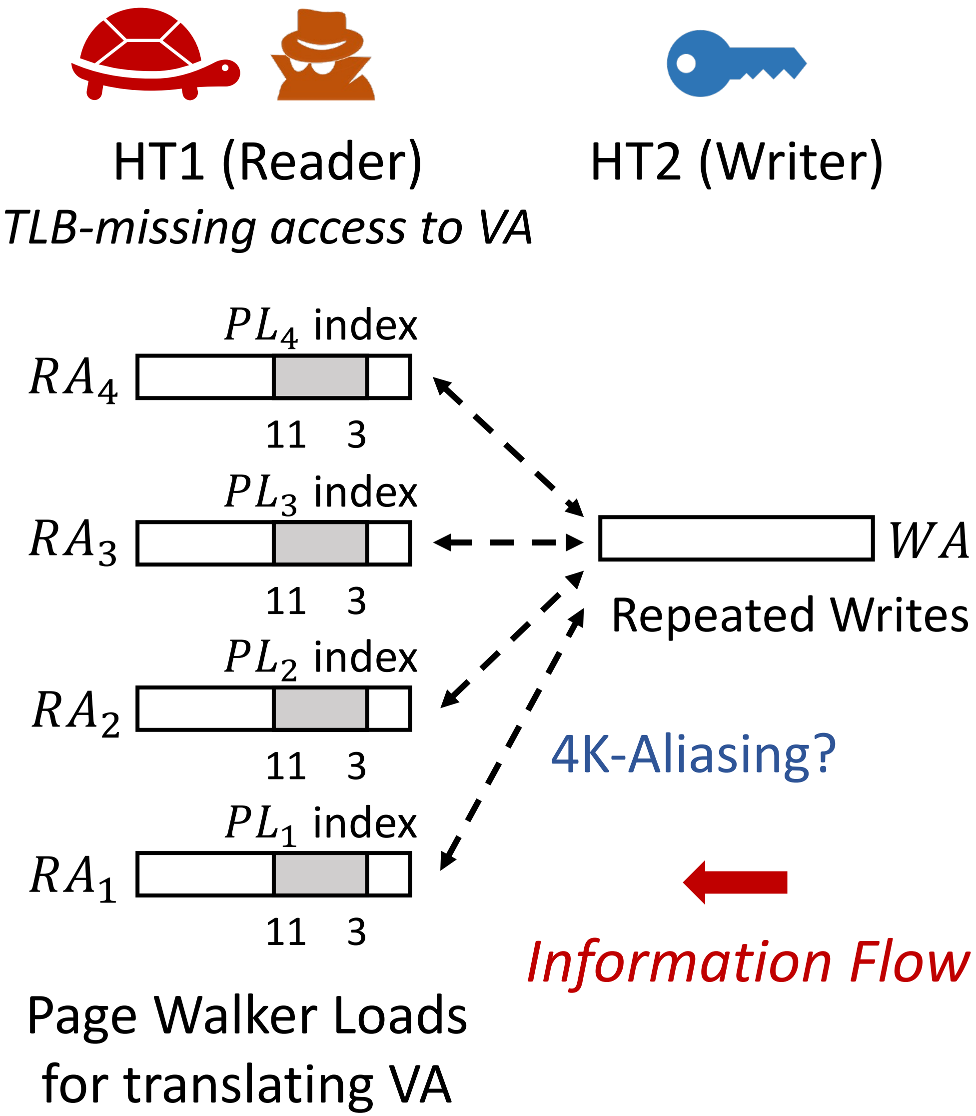
The Binoculars Attack



Security Implications:

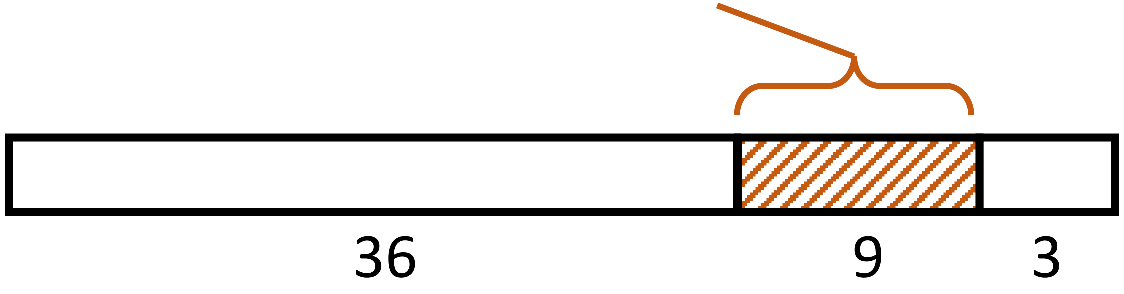
- A high signal-to-noise channel
- Address-dependent contention

Primitive 1: Store→Load Channel



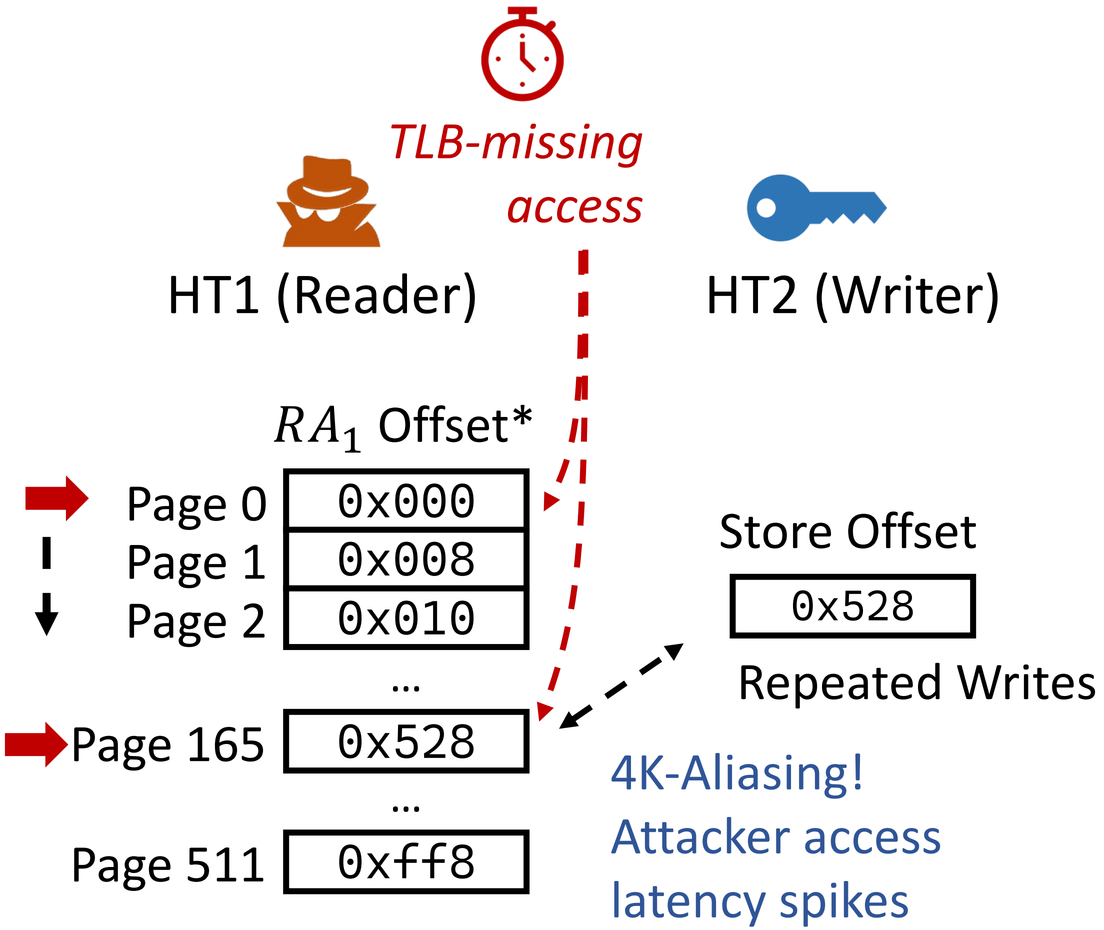
Intuition: the attacker triggers page walker loads while the victim writes. If the attacker observes strong contention:

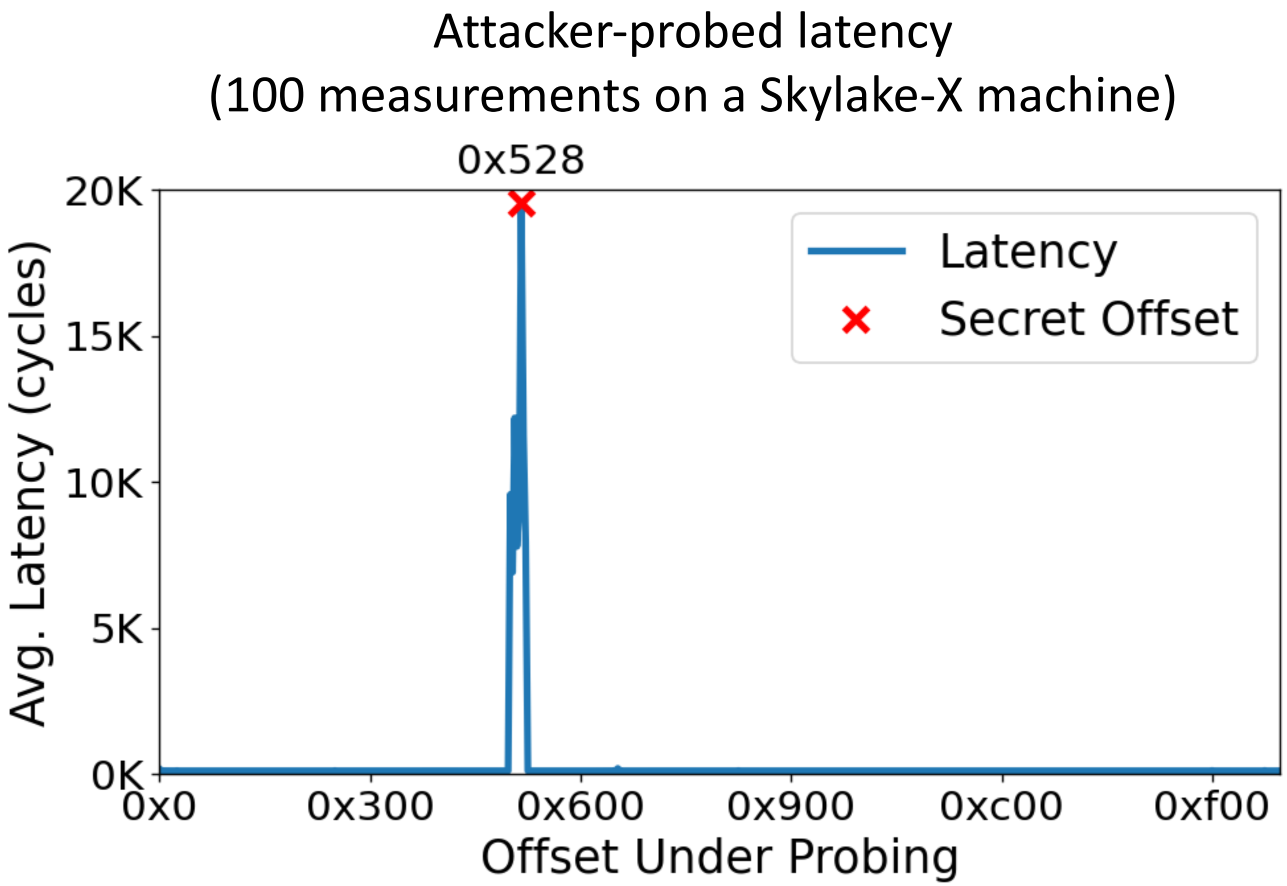
⇒ learn victim stores' offset



Sub-cacheline resolution

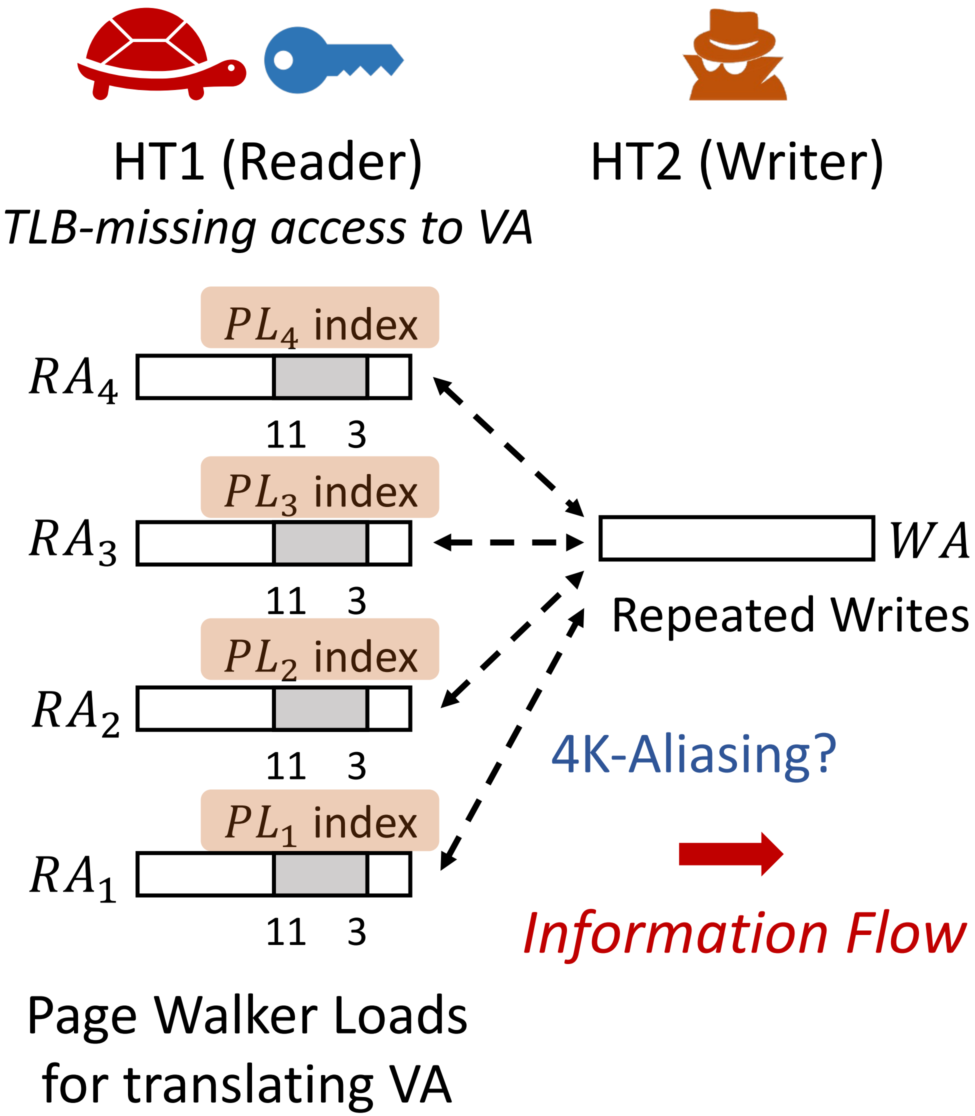
Demo: Store→Load Channel





*RA_1 Offset = PL_1 Index × 8

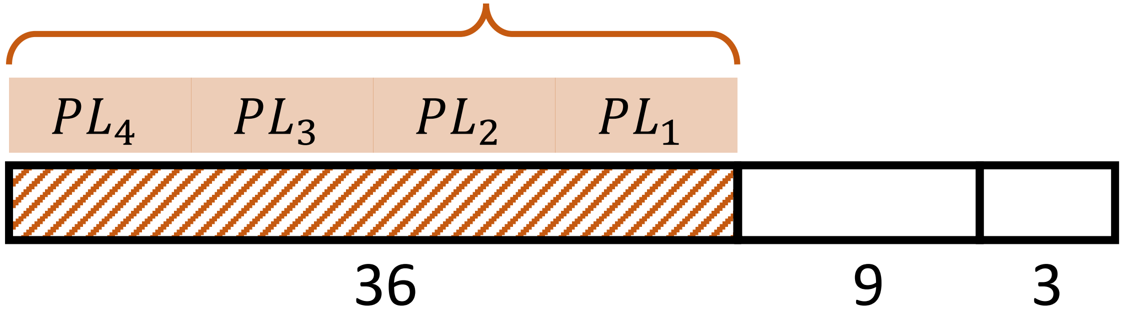
Primitive 2: Load→Store Channel



Intuition: the attacker writes while the victim performs a page walk. If the attacker observes a victim slowdown:

⇒learn the set of PL indexes of the page that the victim accesses

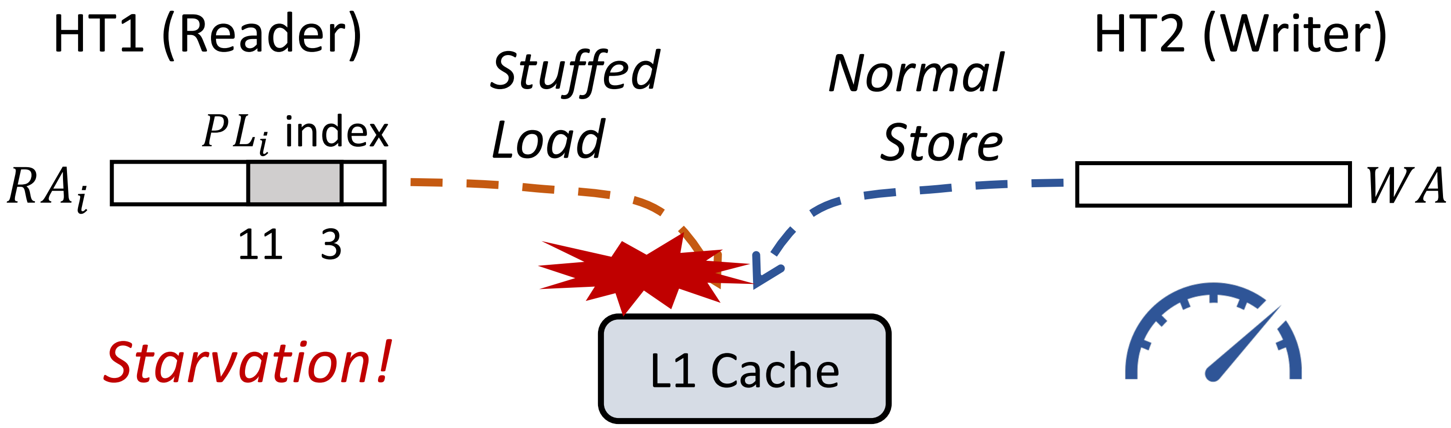
Can reconstruct the entire victim VPN



Demos can be found in Section 4.2

Root Cause of Strong Contention

Intel's Patent*: issue page walker loads as "stuffed loads", which bypass the instruction scheduler to avoid any scheduling latency



More details in Section 5 and Appendix A

- Detailed reverse engineering
- Types of resource conflicts in L1 cache
- …

*Glew et al., Method and apparatus for performing page table walks in a microprocessor capable of processing speculative instructions, 1997, US Patent 5,680,565

Attack Montgomery Ladder and ECDSA

ECDSA: a digital signature algorithm. One step during an ECDSA signing is to compute the point $k \times G$, where k is the nonce and G is the base point

Knowing the nonce k and the corresponding signature
$\Rightarrow$ derive the private key used for signing

Goal: learn the nonce k

Challenge: the nonce k changes at every victim run and never repeats
$\Rightarrow$ requires a channel with high signal-to-noise ratio to extract k with **a single victim execution**

Target: implementation uses *Montgomery ladder* to speed up the signing (OpenSSL 1.0.1e)

Attack Montgomery Ladder and ECDSA

A simplified Montgomery ladder iteration

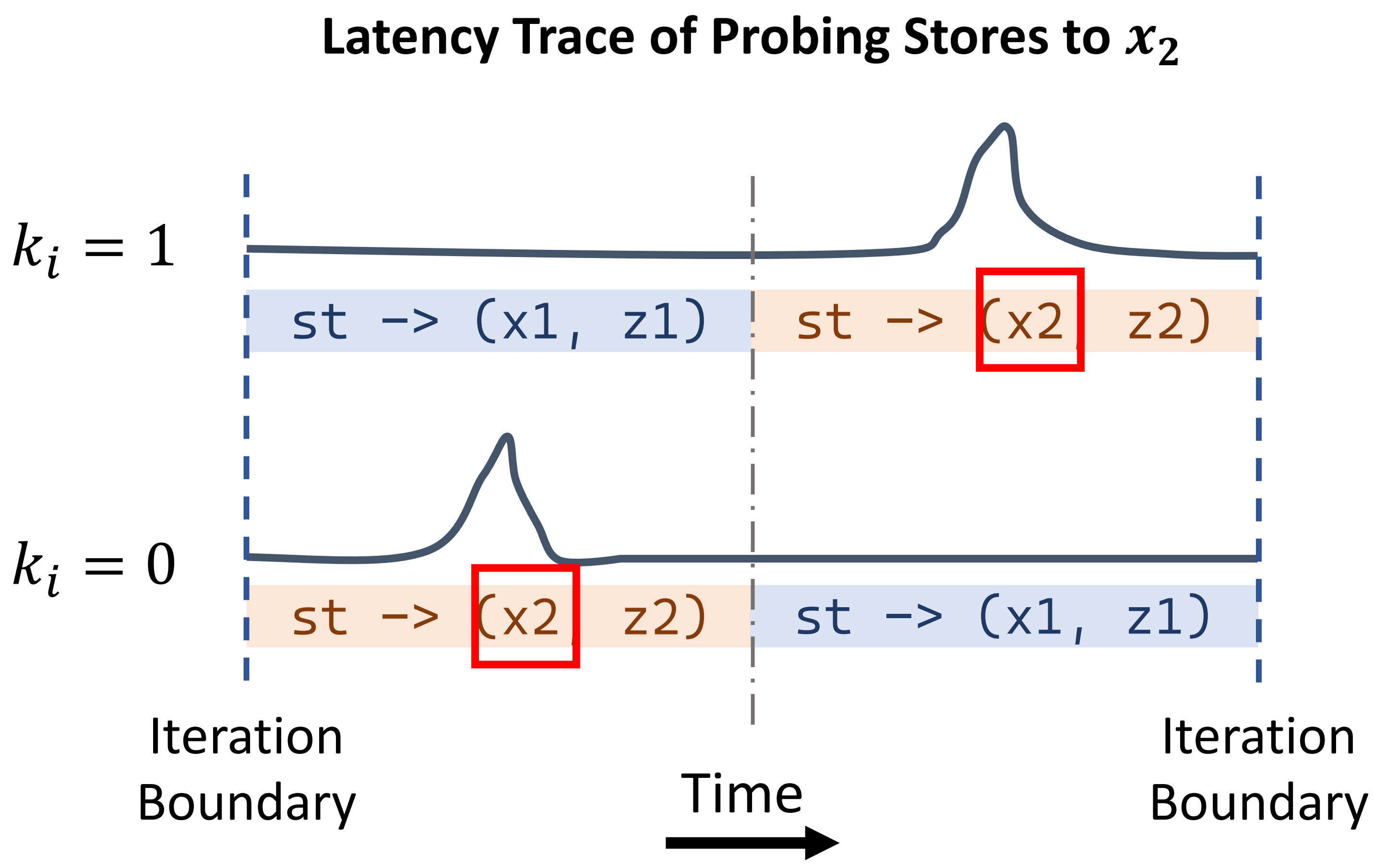
```
BIGNUM *x1, *z1, *x2, *z2;
if (k_i) { // checks ith bit of k
  Madd(x1, z1, x2, z2);    st -> (x1, z1)
  Mdouble(x2, z2);         st -> (x2, z2)
} else {
  Madd(x2, z2, x1, z1);    st -> (x2, z2)
  Mdouble(x1, z1);         st -> (x1, z1)
}
```

☺ Data-oblivious to the sequence of operations and end-to-end timing

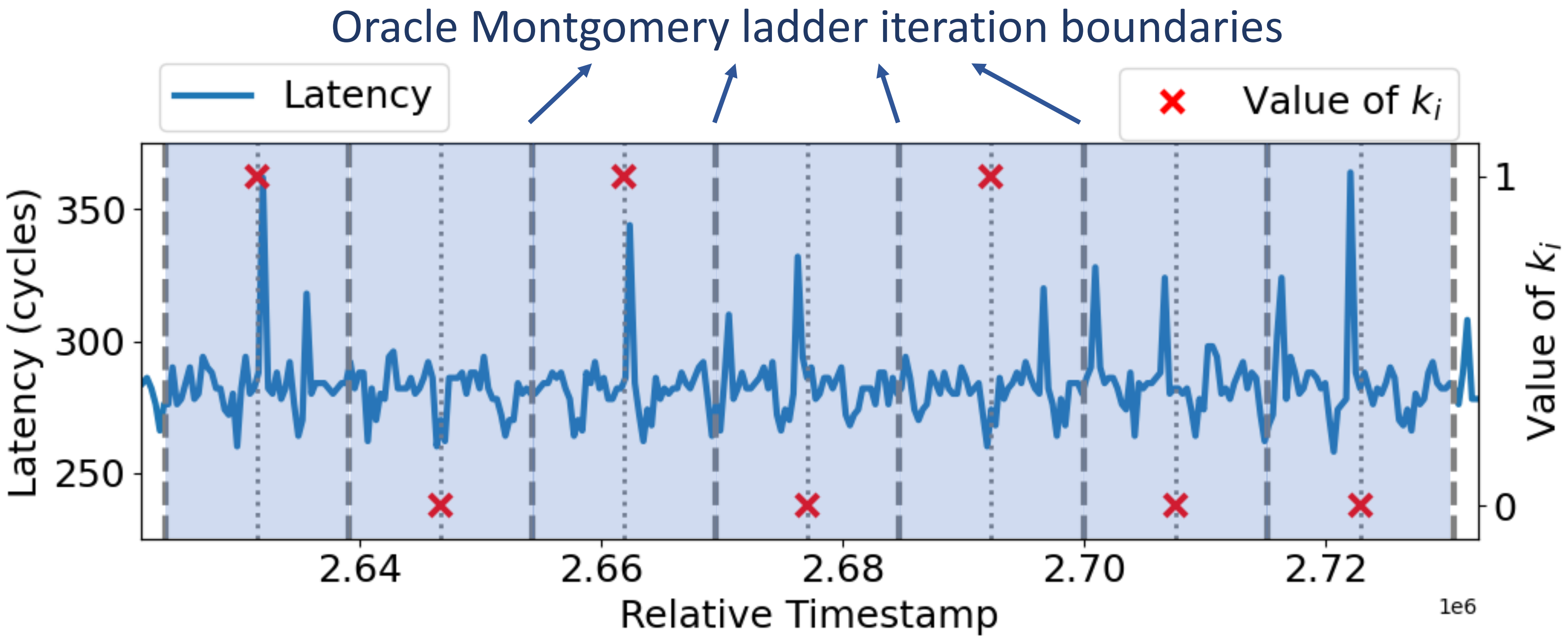
☹ Secret-dependent reordering of stores

Detect it with the Store→Load Channel

Attack Montgomery Ladder and ECDSA



Attack Montgomery Ladder and ECDSA



Signal Processing on Raw Traces:

1) Recover iteration boundaries
2) Predict k_i

Evaluated on Skylake-X and Cascade Lake-X (100 traces):

+ Average k_i prediction accuracy: 98.4%~98.5%
+ Median brute force attempts: $\approx 2^{24}$

Conclusions

Binoculars: the first stateless-indirect channel

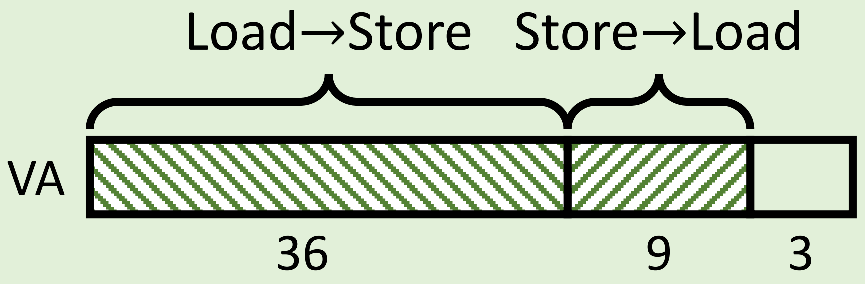
1. Easy to Observe

+ Up to 20K-cycle contention

+ High signal-to-noise ratio

+ Extract nonce k with a single victim execution

2. Wide VA Bits Coverage



3. Open Source

https://github.com/zzrcxb/binoculars