COTSknight: Practical Defense against Cache Timing Channel Attacks using Cache Monitoring and Partitioning Technologies

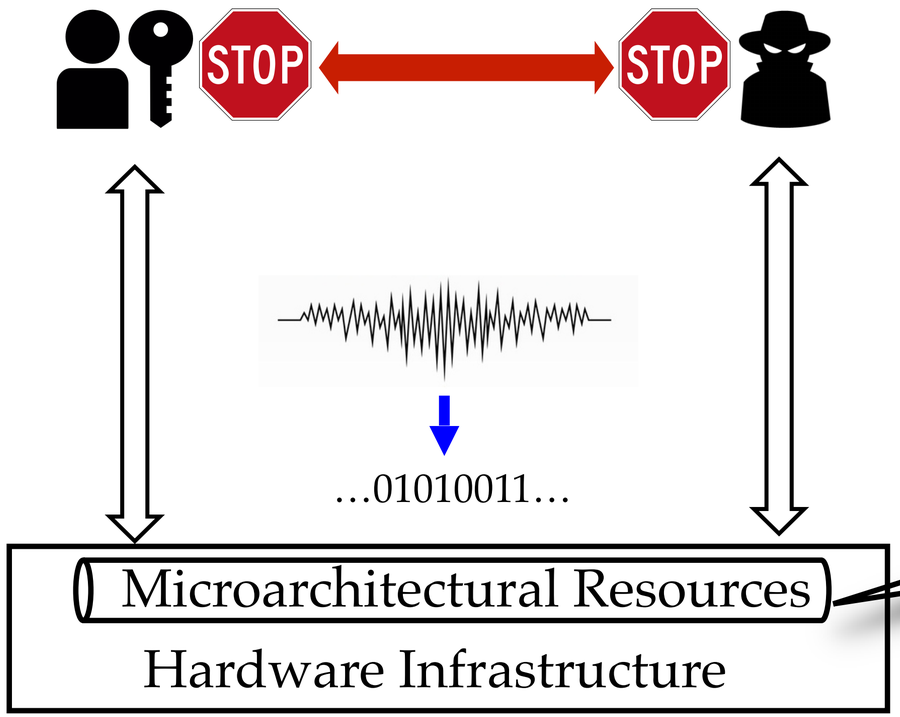
Fan Yao[1,2], Hongyu Fang[2], Miloš Doroslovački[2], and Guru Venkataramani[2]

[1] Department of ECE,
University of Central Florida
Orlando, FL

[2] Department of ECE,
George Washington University,
Washington, DC

IEEE International Symposium on Hardware Oriented Security and Trust (HOST) 2019
May 6 - 10, 2019
The Hilton
Tysons Corner, USA

Emerging Threats: Information Leakage on Contemporary Hardware

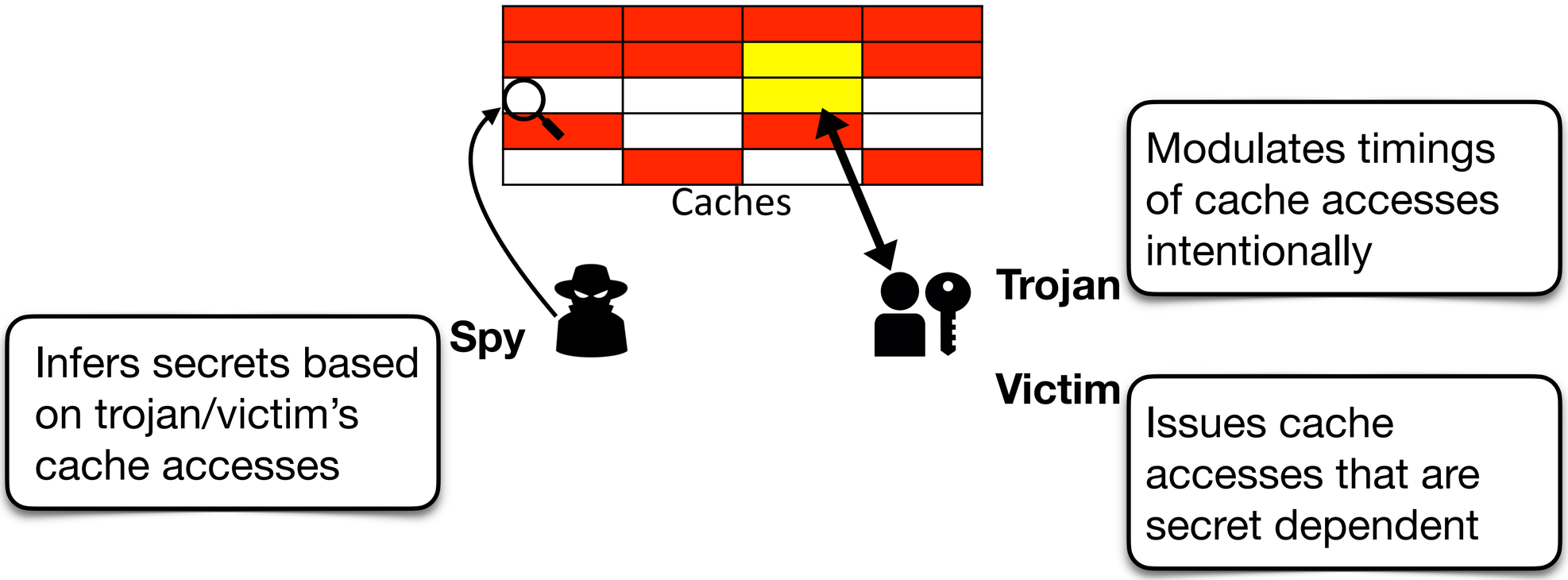


- **Caches** (e.g., Liu et al. *S&P'15*)
- **Cache Coherence** (e.g, Yao et al. *HPCA'18*)
- **Branch predictor** (e.g, Evtyushkin et al. *TACO'16*)
- **Speculation** (Spectre and Meltdown)

….

Cache Timing Channels

- The dominant hardware information leakage attacks
  - One of the most commonly shared resources
  - Presents the largest attack surface



Existing Defense Techniques

- Secure cache designs (e.g., **NewCache IEEE Micro'16, SHARP ISCA'17**)
  - Not able to protect commodity systems
  - Require hardware modifications (may involve high cost)

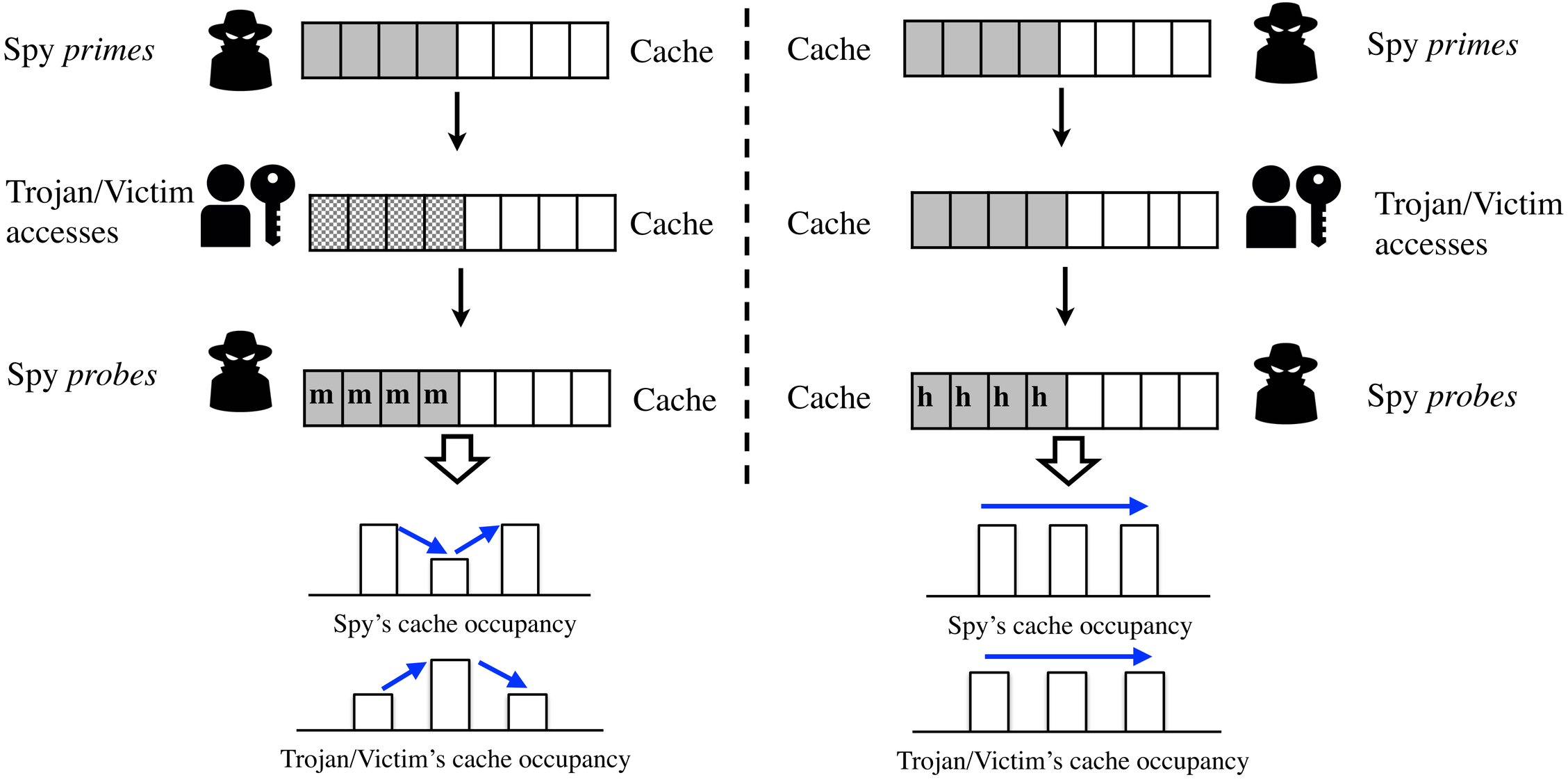
- Pre-emptive cache partitioning techniques (e.g., **SecDCP DAC'16**)
  - Cause unnecessary performance degradation
  - Can be difficult to scale

- ***The need for a ready-to-use and performance-friendly solution***

✓ **No hardware modification** ⇒ *protects off-the-shelf machines*

✓ **Incurs low overhead** ⇒ *improves adoption*

Cache Occupancy as the Indicator of Cache Timing Channels

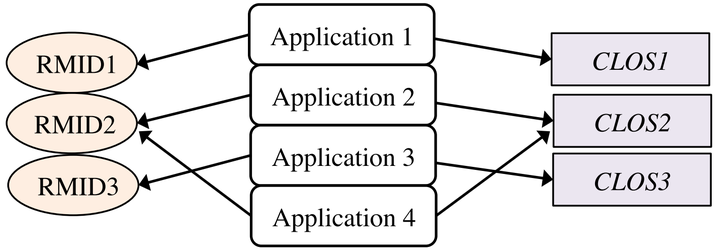


Emerging COTS Resource Managers

- Intel cache monitoring (CMT) and cache allocation (CAT) technologies
  - Designed to offer visibility of shared resource usage and QoS control
- ARM MPAM for Armv8-A architecture

Cache occupancy capability

Hardware Thread	IA32_PQR_ASSOC_MSR[63:32]: CLOS NO. IA32_PQR_ASSOC_MSR[31:0]: RMID NO.



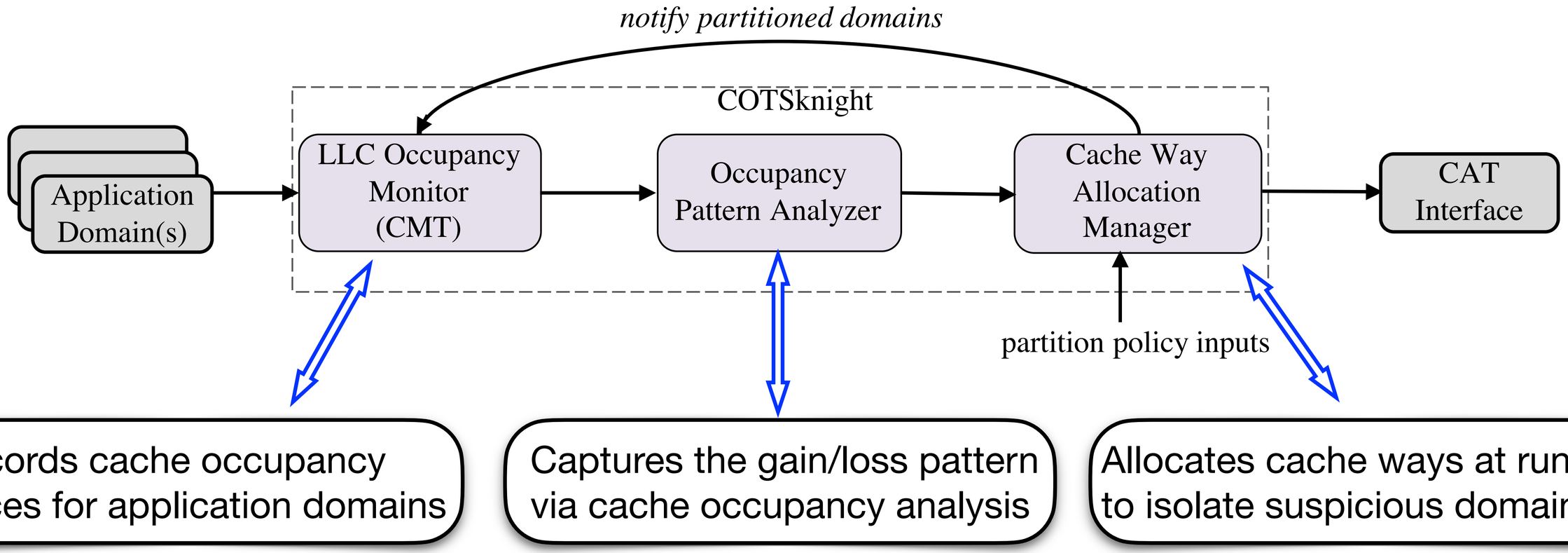
- Allows fine-grained cache occupancy monitoring
- Flexible runtime domain specification

Cache partitioning capability

CLOS NO.	Capacity Bitmask	Way Allocation
CLOS0	IA32_L3_MASK_n_MSR[31:0] 0xFFFF	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
CLOS1	IA32_L3_MASK_n_MSR[31:0] 0xF000	1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
CLOS2	IA32_L3_MASK_n_MSR[31:0] 0x0F00	0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
CLOS3	IA32_L3_MASK_n_MSR[31:0] 0x00F0	0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0

- Enables way-based cache partitioning in last level caches
- Dynamic partition decisions at runtime

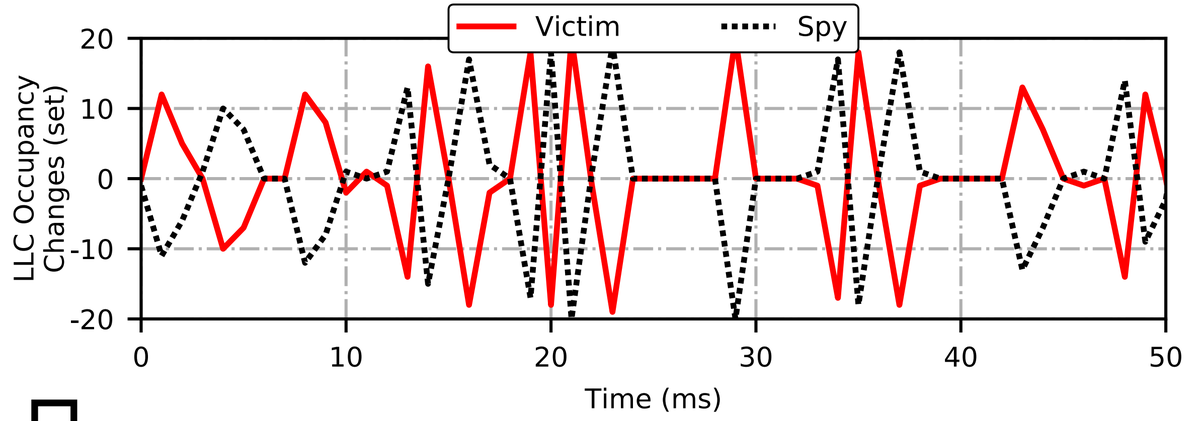
Our Proposed Protection Framework: COTSknight



COTSknight: Cache Timing Channel Identification

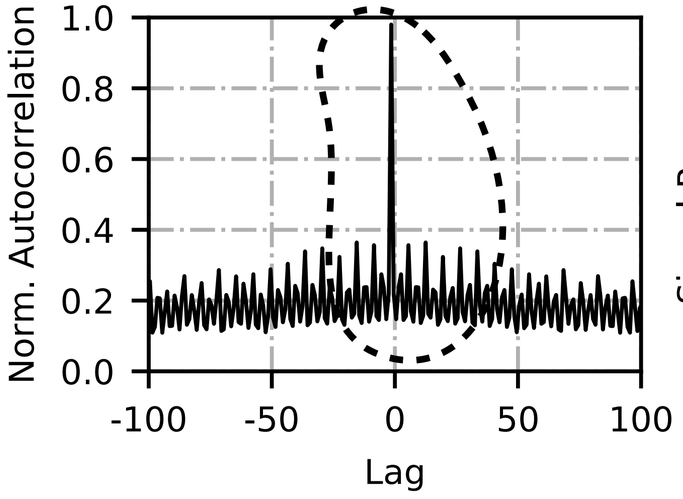
- LLC Occupancy Monitor
  - Per-domain cache occupancy monitoring
  - Domains could be a thread, a process or VM instances
- Occupancy pattern analyzer
  - Trace pre-processing: take the product of ***Δx*** and ***Δy*** as ***z*** (occupancy changes for the pair of domains)
    - Noise filtering can be applied to remove non-negative values
  - Computes the normalized autocorrelation of ***z*** as ***r***
    - Detecting the linear relationship between ***Δx*** and ***Δy***
  - Extracts the repetitive patterns via *power spectrum analysis* of ***z***
    - Spikes in frequency domain indicates the repetitive patterns in ***z***

COTSknight: Cache Timing Channel Identification

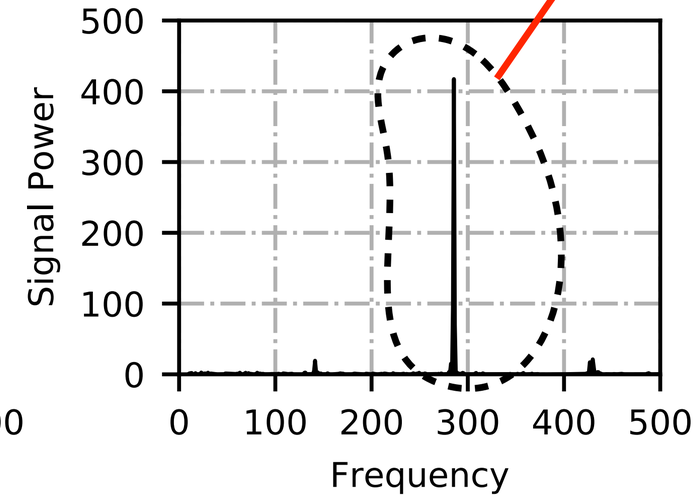


cache occupancy changes for trojan and spy

Isolated spike with high signal power!



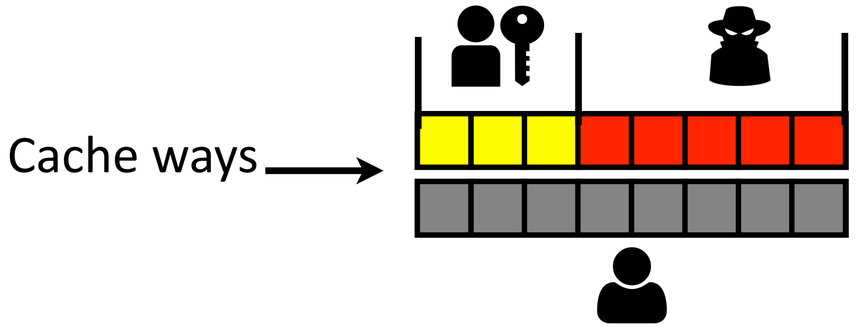
autocorrelogram



power spectrum analysis

COTSknight: Dynamic Cache Way Allocation

- Proposed two different way allocation policies
- **Aggressive policy**
  - Partitions pairs of suspicious domains
  - Keep them isolated until one of them finishes execution
- **Jail policy**
  - Partitions suspicious domains for a period of time (“put in jail”)
  - Penalizes duration of isolation if same pair found suspicious again



Experimental Setup

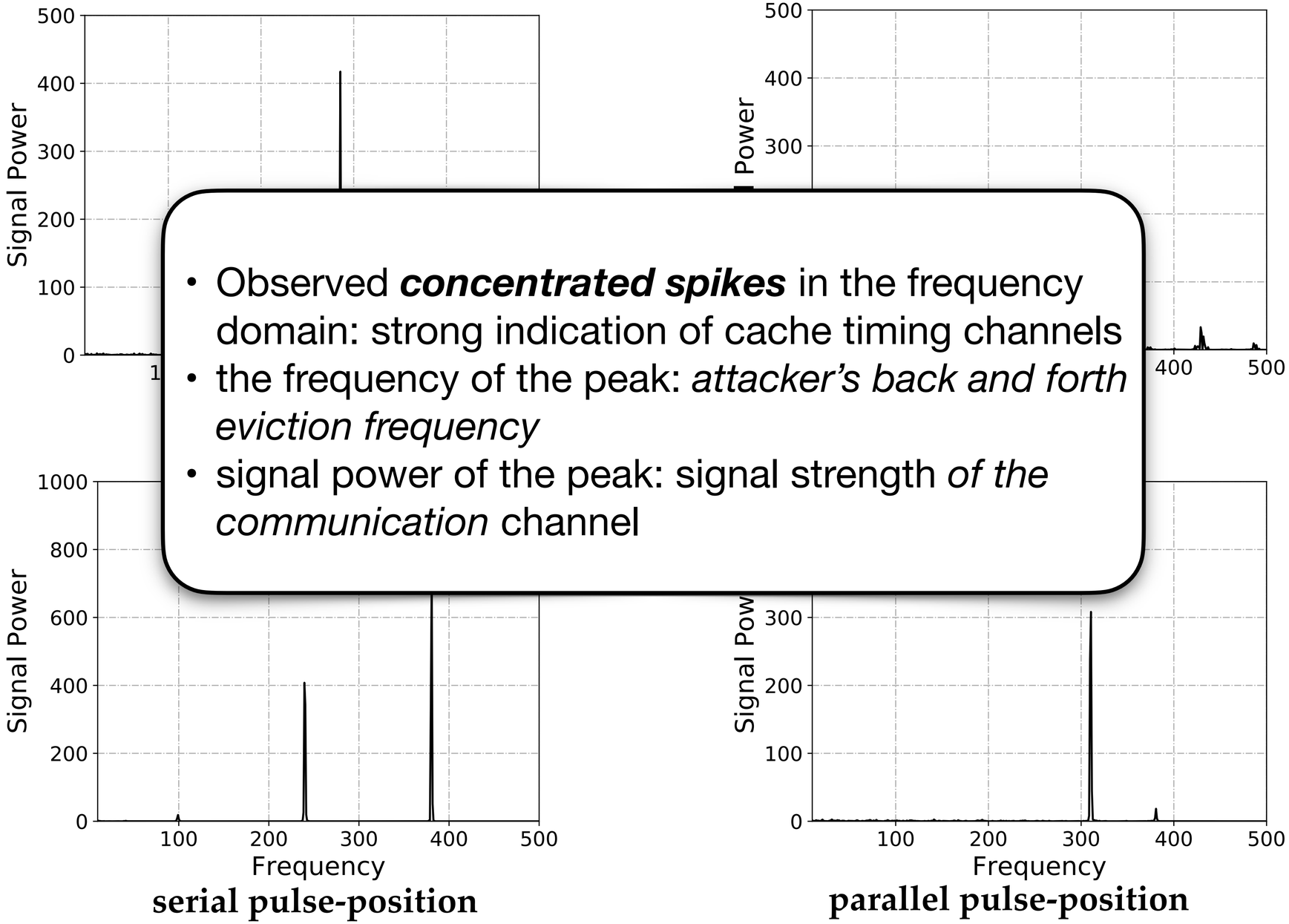
- COTSknight prototype
  - Based on **Intel Xeon v4** processor
  - Supports upto **16 CLOS** (**20-way** per set)
- Cache timing channels: four attack variants based on previous studies

Attack variants	**on-off encoding**: uses single group of cache sets	**pulse-position encoding**: uses multiple groups of cache sets
serial protocol: two processes access cache in round robin fashion	serial on-off	serial pulse-position
parallel protocol: two processes access cache in parallel	parallel on-off	parallel pulse-position

covert channels

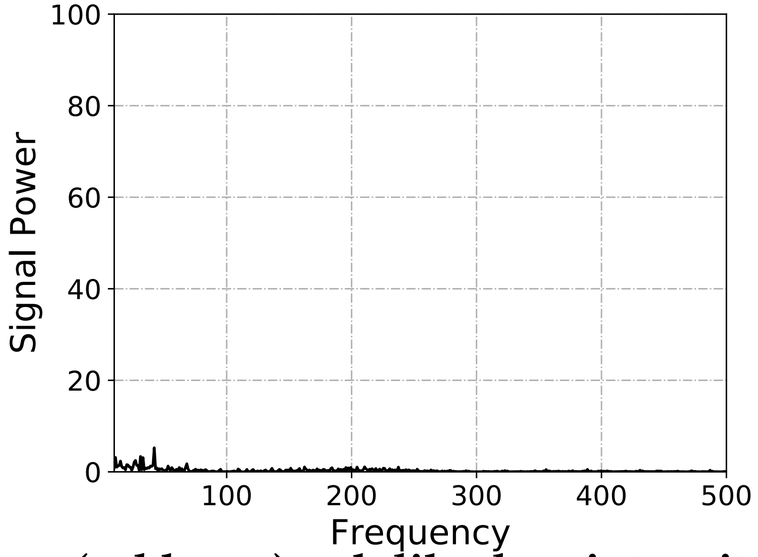
side/covert channels

Power Spectrum Analysis on Cache Timing Channels

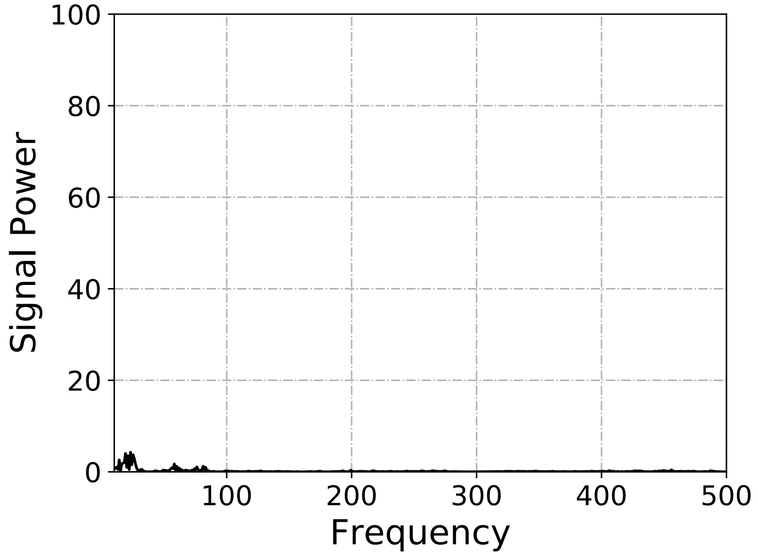


- Observed ***concentrated spikes*** in the frequency domain: strong indication of cache timing channels
- the frequency of the peak: *attacker's back and forth eviction frequency*
- signal power of the peak: signal strength *of the communication* channel

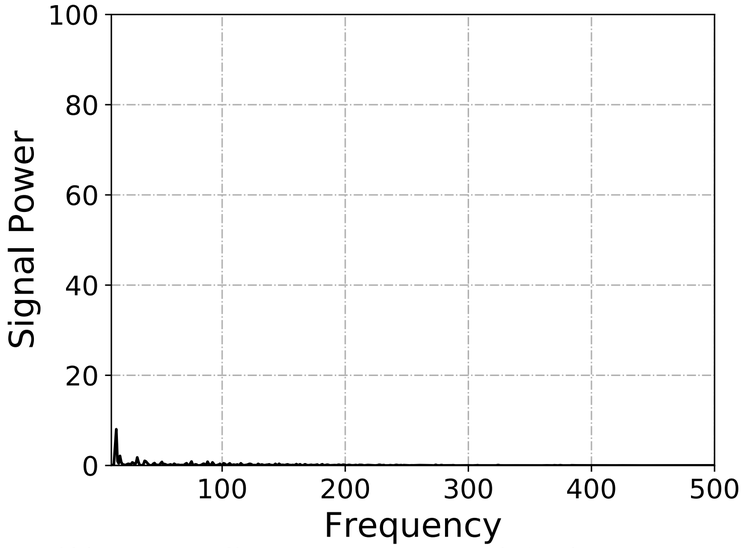
Power Spectrum Analysis on Benign Workloads



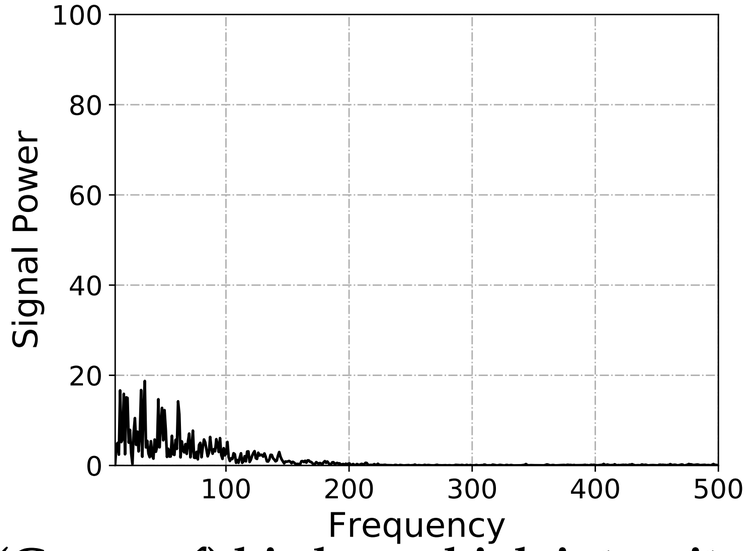
(cal hmm) gob lib - low intensity



(Gem hmm) xal bwa - medium intensity

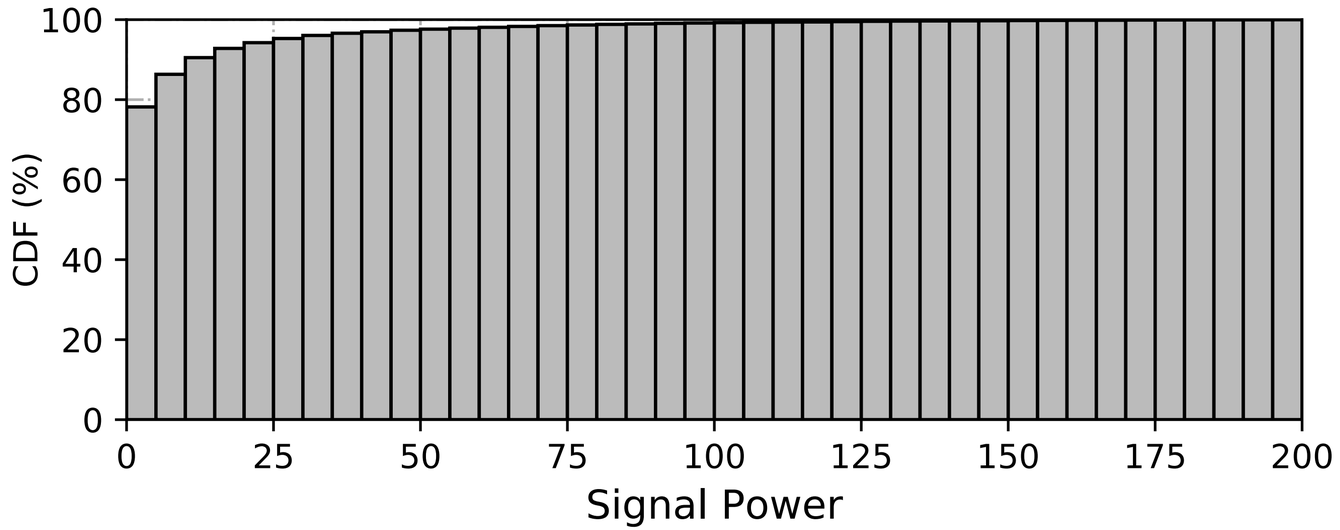


(lbm mil) sop Gem - high intensity



(Gem mcf) biz bwa - high intensity

Thresholding for Cache Timing Channel Determination



Peak signal power CDF for benign workloads

- Benign workloads: 98th percentile peak signal power < 25
- Timing channels: peak signal power well above 100 all the time
- Conservatively set a threshold of **50** to trigger cache allocation

Effectiveness of COTSknight

- Stopped all the instances of the attack variants
  - Identifies all of the trojan-spy domain pairs within **5** consecutive analysis windows after attack begins
- Partition trigger rate for benign workloads
  - For low cache intensity workloads, no partition triggered
  - Among all benign application pairs, only **6%** of partitioning
- COTSknight runtime overhead
  - Pair-wise power spectrum analysis is in milliseconds (monitor window is 500ms)

Performance Impacts on Benign Workloads

- for benign workloads that trigger partitions
  - average performance impact: **-1%** (speedup!)
  - worst case slow-down: < **5%**
  - many applications involves speedup
    - up to **9.2% performance improvements**
    - COTSknight alleviates unnecessary interferences

Conclusions

- We proposed a novel application of COTS HW to enhance system security
- We designed COSTknight, a practical framework that identifies and stops cache timing channels
- Evaluation of COTSknight showed that it brings negligible performance impact to benign workloads (speedup in many cases)
- All of COTSknight’s components are built and run on real systems

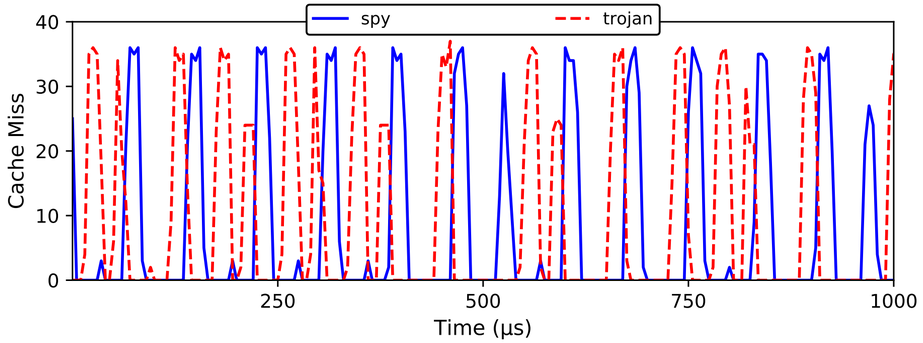
Thanks! Questions?

Contacts

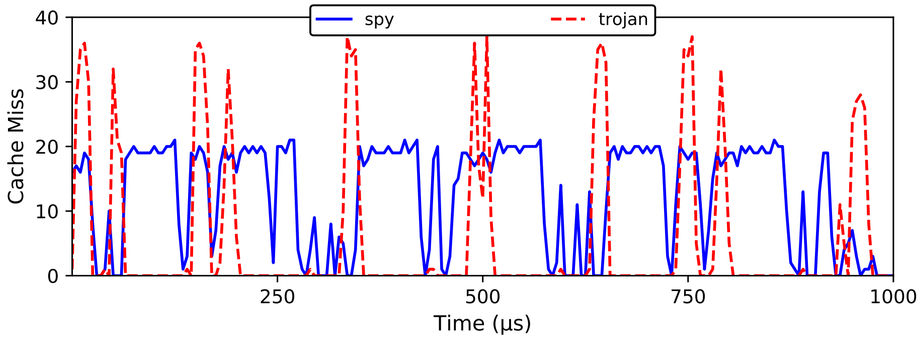
Fan Yao: fan.yao@ucf.edu

Guru Venkataramani: guruv@gwu.edu

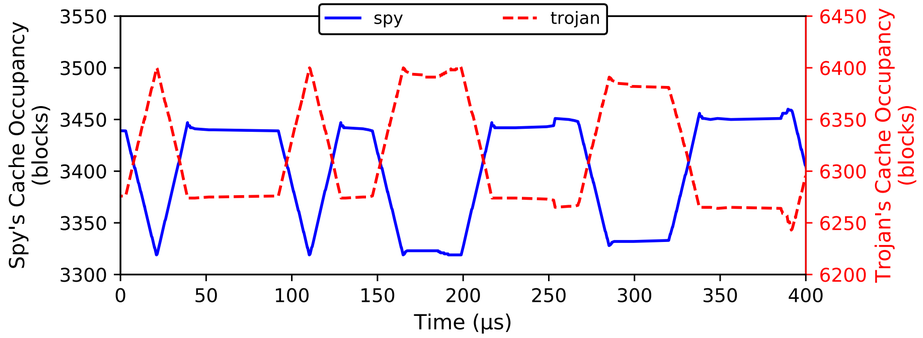
Cache Miss vs. Cache Occupancy



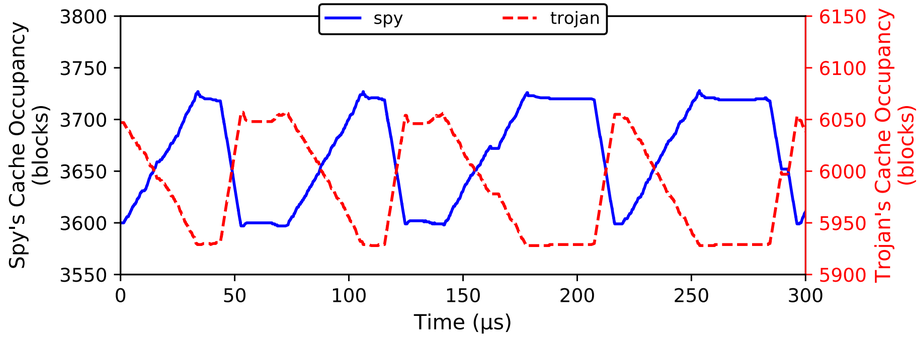
Attack-only, cache miss patterns



Attack with noise, cache miss patterns

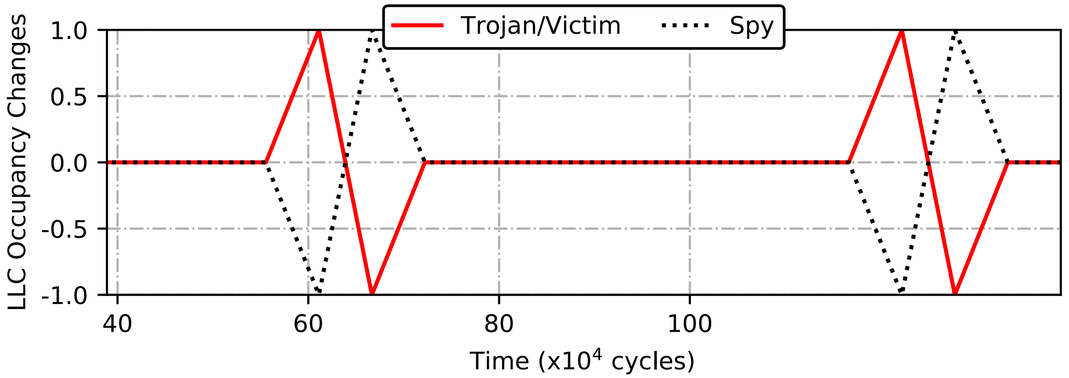


Attack-only, cache occupancy patterns

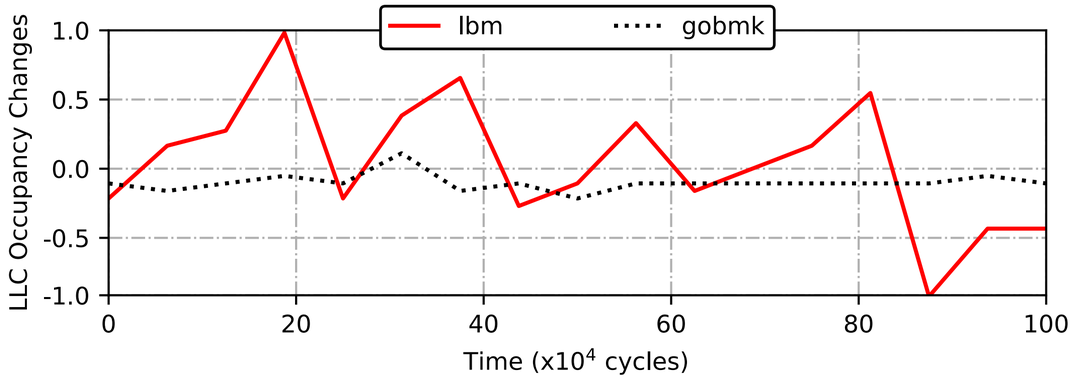


Attack with noise, cache occupancy patterns

Motivational Example



Cache occupancy changes for Trojan/victim and spy

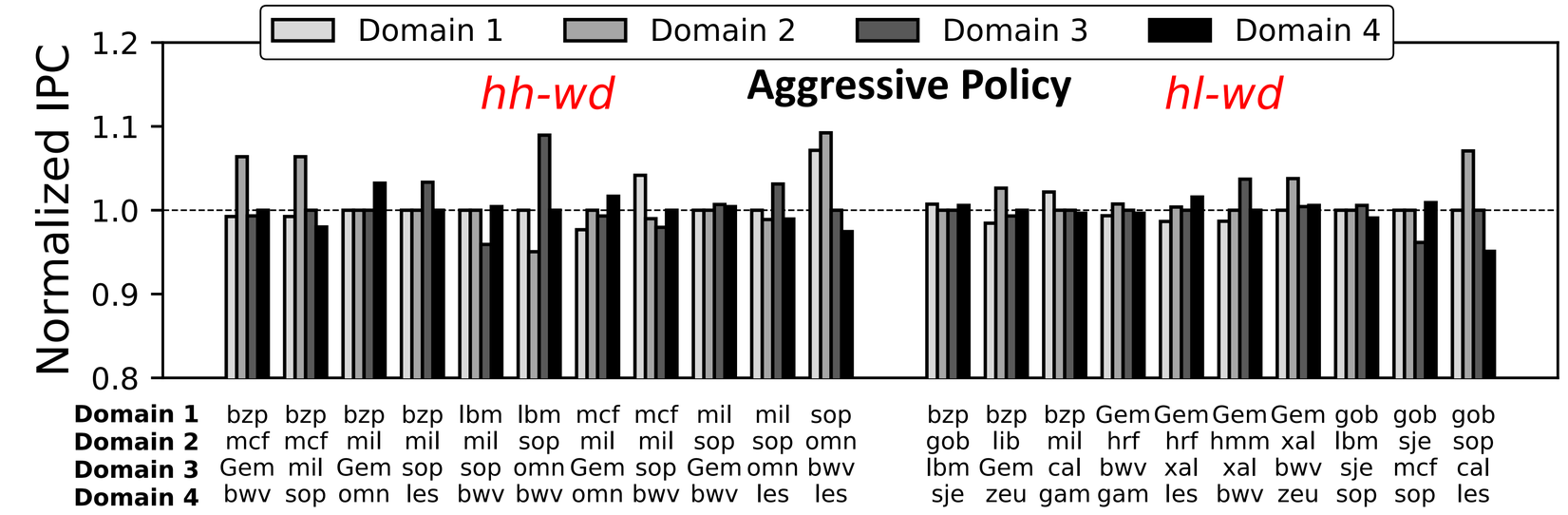


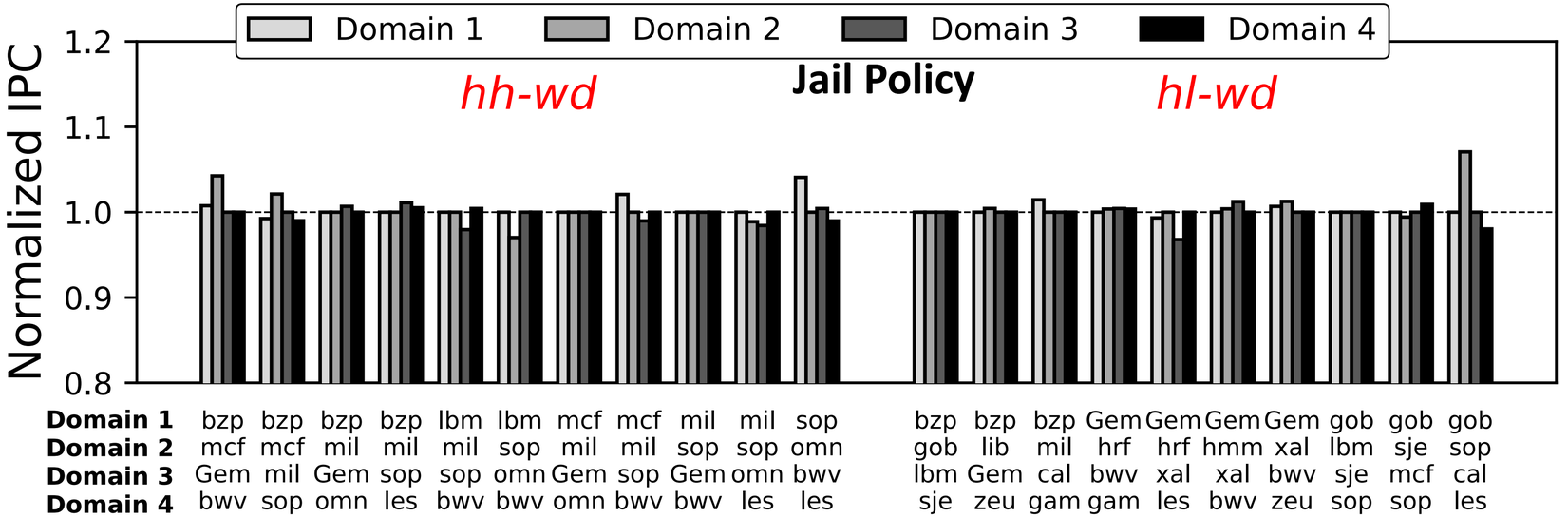
Cache occupancy changes for two benign programs

Observation 1: Timing channels in caches rely on ***mutual cache block replacements*** for covert communication

Observation 2: Such activities create ***repetitive swing patterns*** in cache occupancy (regardless of the communication protocols)

Performance Impacts on Benign Workloads





Performance impact on benign workloads where COTSknight allocator triggers LLC partition

Discussions

- Limitations of hardware monitoring
  - the precision and sampling rate for cache occupancy
  - maximum no. of CLOS supported (e.g., 16 for Intel Xeon E5-2698)
- CAT still allows *cache hit in cache ways belonging to another domain*
  - could not stop flush+reload attacks
  - security enhancement on CAT already proposed: **DAWG (MICRO'18)**
- Futuristic adversaries could attempt to bypass our detection process
  - randomized bit transmission intervals, distortions of swing pulses
  - COTSknight enables a host of advanced signal processing