TLBleed

Translation leak-aside buffer: Defeating cache side-channel protections with TLB attack

B. Gras, K. Razavi, H. Bos, and C. Giuffrida

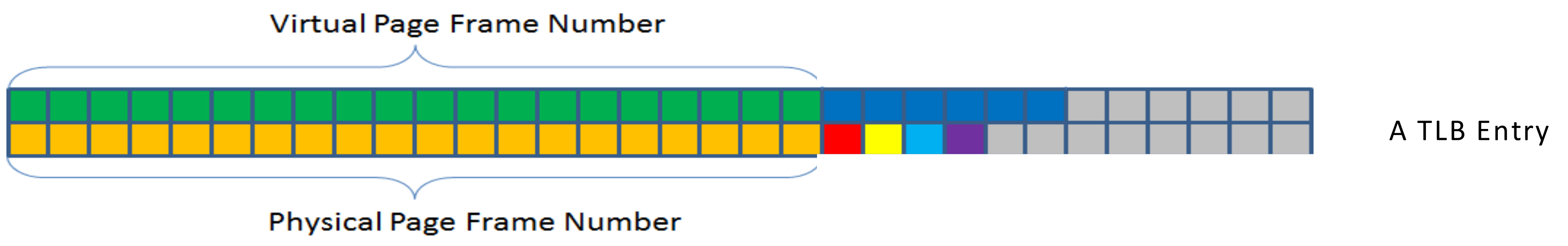
Presented by
Ayoosh Bansal

Translation Lookaside Buffer (TLB)

It is a cache, where every entry contains

Virtual Address -> Physical Address mapping

Processor DRAM



Tr

It i

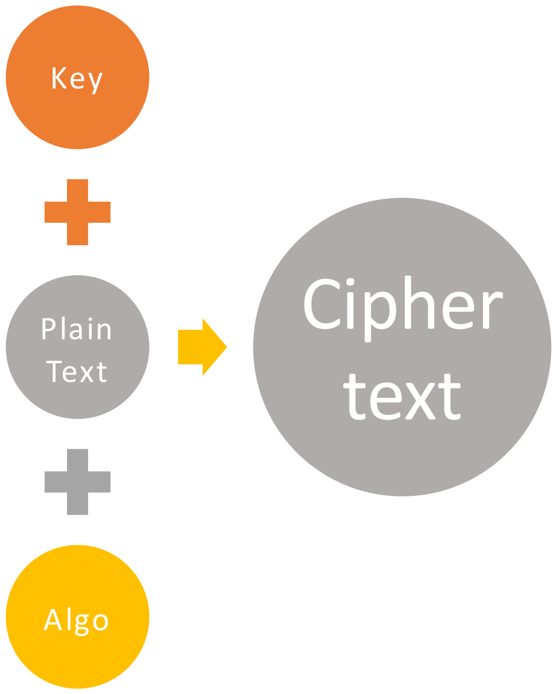
Vir

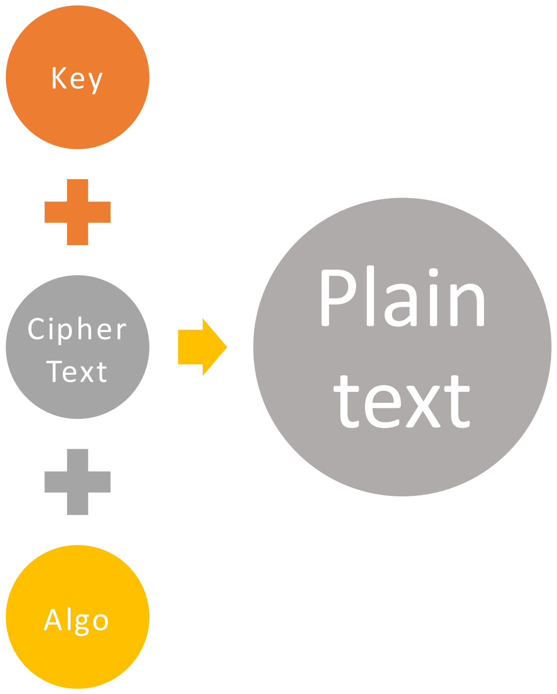
Translation ~~Look~~Leak-aside Buffer !!!

(31:12) Physical Page Frame Number

Crypto Keys

Strings of 1s & 0s : 101010110010101001011110101000101010001010…





Stolen Keys

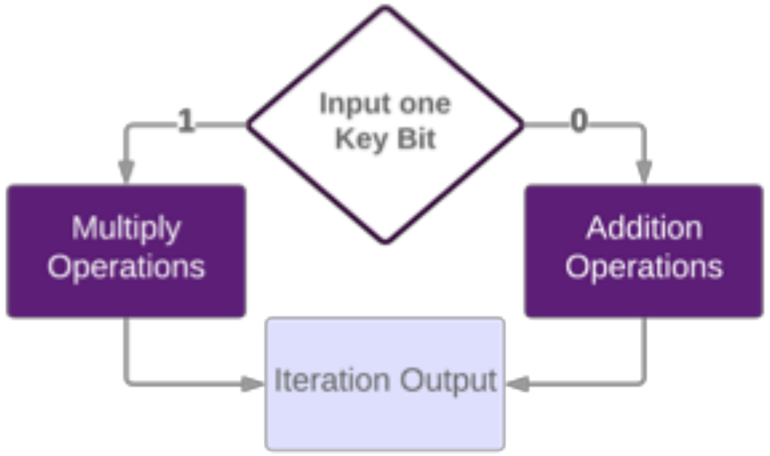
- ?
    - Defeat Encryption
    - Digital Identity

Timing Side Channel Attacks

Algorithm changes execution Timing Based on key bits.

Defense Ideas?

- Remove key data based variations
  - Hard to do



Timing Side Channel Attacks

Ability to analyze encrypt / decrypt timing or memory accesses

Create some characteristic signal pattern representation of

[Algorithm , Key bit 1] [Algorithm , Key bit 0]

Observe execution and match signal pattern

Shared Hardware Resource ➡ Signal

Observe Usage of Shared Resource to observe signal patterns.
Example: Cache based Side Channel Attacks

Requirements For Side Channel:

- Different owners or privilege levels share resources.
- Can observe other’s access patterns or timing.

Solutions ?

- Schedule so resource access i
- Partition resources to isolate.
- Remove the ability to observe other process’s activity

Modern Caches support these defenses

TLBleed : Threat Model

Victim : Crypto process

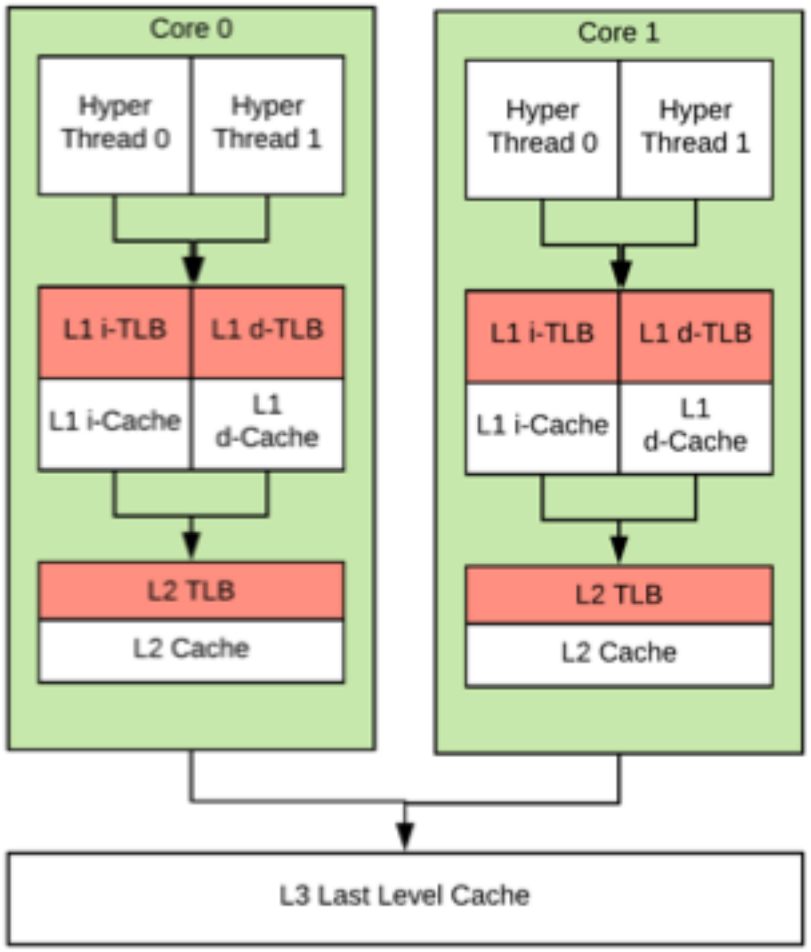
Attacker : Executes Unprivileged Code

Shared Resource : TLB

Scheduling : Same core
(Simultaneous Multithreading / Hyperthreading)

Microarchitecture: Known to attacker

Damage: Crypto Key Leakage



TLBleed : Recognition and Response

- Online Press coverage
- Wikipedia page

- Intel ignored TLBleed quoting preexisting data independent constant time execution crypto primitives.

- OpenBSD disabled Hyper-Threading completely, disabling this vulnerability at a large cost to processor performance.

TLBleed : Understanding the Channel

- TLBs types are documented, structure and address to entry mapping is not
  - Authors reverse engineer the TLB characteristics
    - Use Architectural Counters to measure TLB Hit/Miss.
    - Craft memory request patterns with some hypotheses of TLB structure and see if measurements conform to the hypotheses.

Skylake	L1 i-TLB	L1 d-TLB	L2 TLB
Sets, Ways	8, 8	16, 4	128, 12
Virtual Address to TLB Set Mapping	Linear	Linear	XOR
Shared?	No	Yes	Yes

- Discussion
  - Past experiences with microarchitecture reverse engineering?
  - Better approaches?

TLBleed: Unprivileged TLB Monitoring

- Access Sets
  - Contained in L1 d-TLB
  - Contained in L2 TLB, Larger than L1 d-TLB
  - Larger than L2 TLB
- Profile
  - L1 d-TLB Hit latency
  - L1 d-TLB Miss, L2 TLB Hit laten
  - L2 TLB Miss latency
- With this informa observe misses to find whi d by other HyperThread.

Monitor Logic

Memory Barrier;

stamp;

Chasing Accesses;

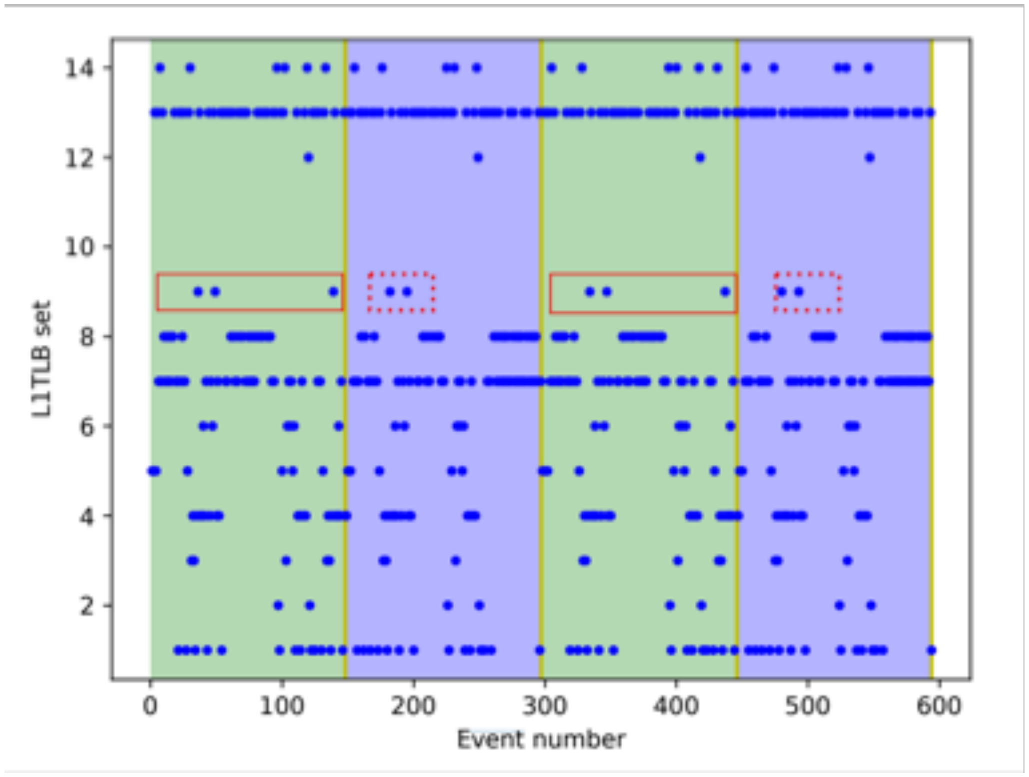
Memory Barrier;

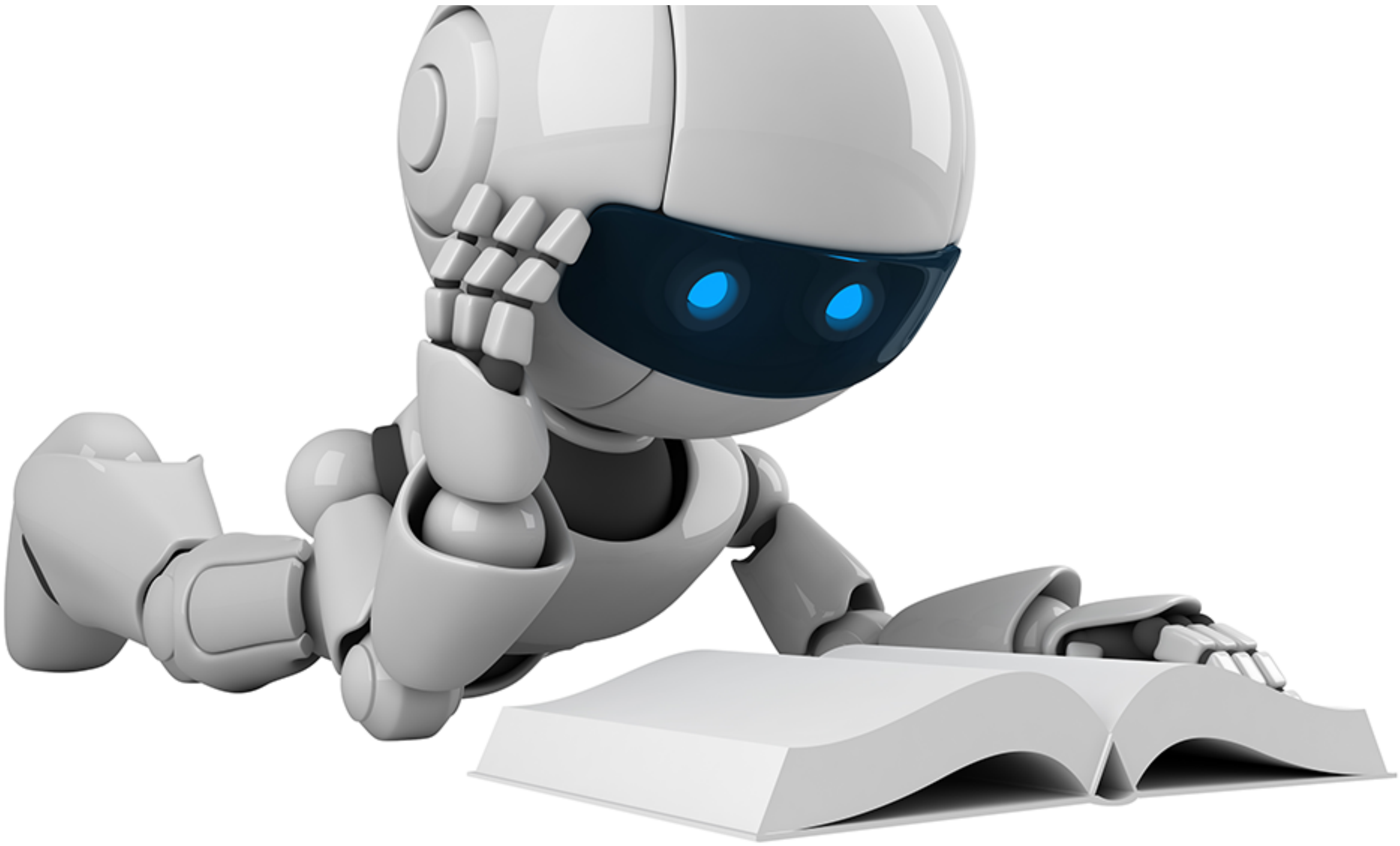
Time Stamp;

Side Channel is Ready!

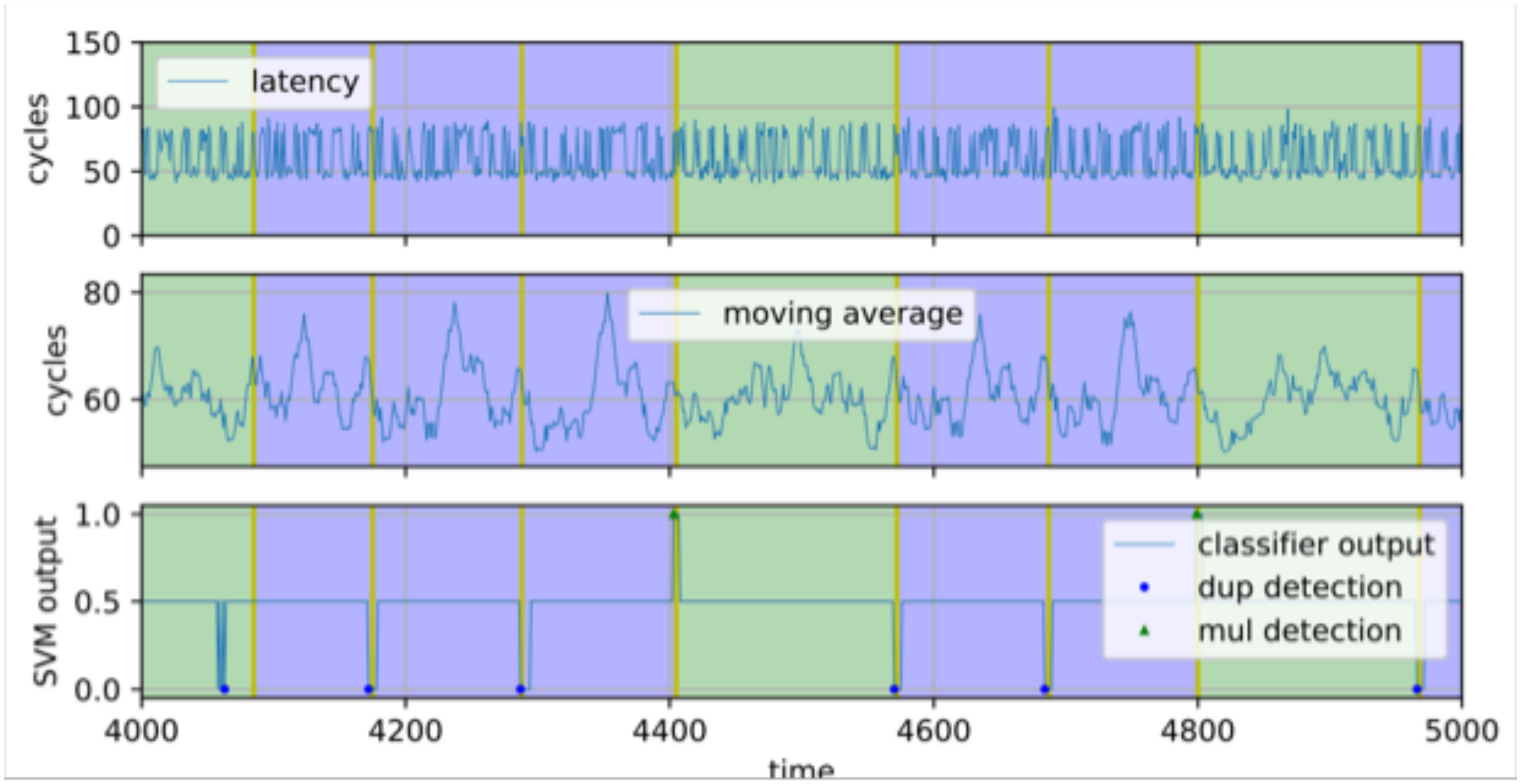
TLBleed : Do we have an attack?

- Instrumented victim code (Color Code in figure)
- Attacker observes Victim's TLB usage (Blue Dots)
- A pattern emerges!
  - Can't use Set usage alone though
- You know what's coming?





TLBleed : Cracking the Key



- Classifier output based on TLB latency +

  Brute Force attempt to fix misclassifications

- 256-bit EdDSA encryption keys – 99%
- RSA 1024- bit secret exponent bits – 92%

Your Thoughts?

Discussion : Strengths

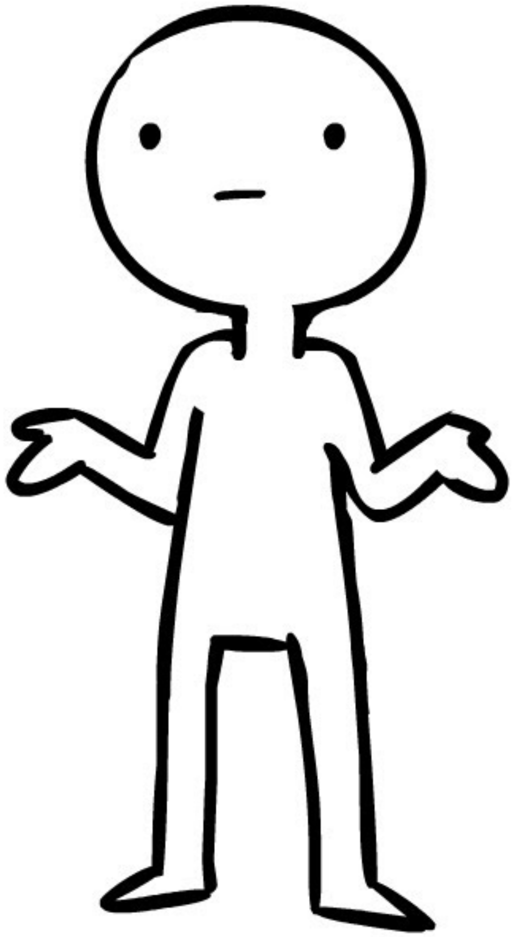
- New attack surface.
- Bypasses cache centric defenses.
- ML based classifier creates lower entry cost to a new environment.

Discussion : Weaknesses

- Algorithm updates can defeat TLBleed. They had to use older time variant versions in evaluation.
- System variations are not considered.
- Reconstructed key Reliability goes down with Larger Page Sizes.
- Hyper-Thread scheduling favors threads from same process.

Discussion : Real World Attack

1. Run a malware on victim machine or multitenancy with victim.
2. Achieve malware and victim residency on hyper-threads.
3. Determine if victim is using crypto keys.
4. Identify victim’s crypto application.
5. Reconstruct key.

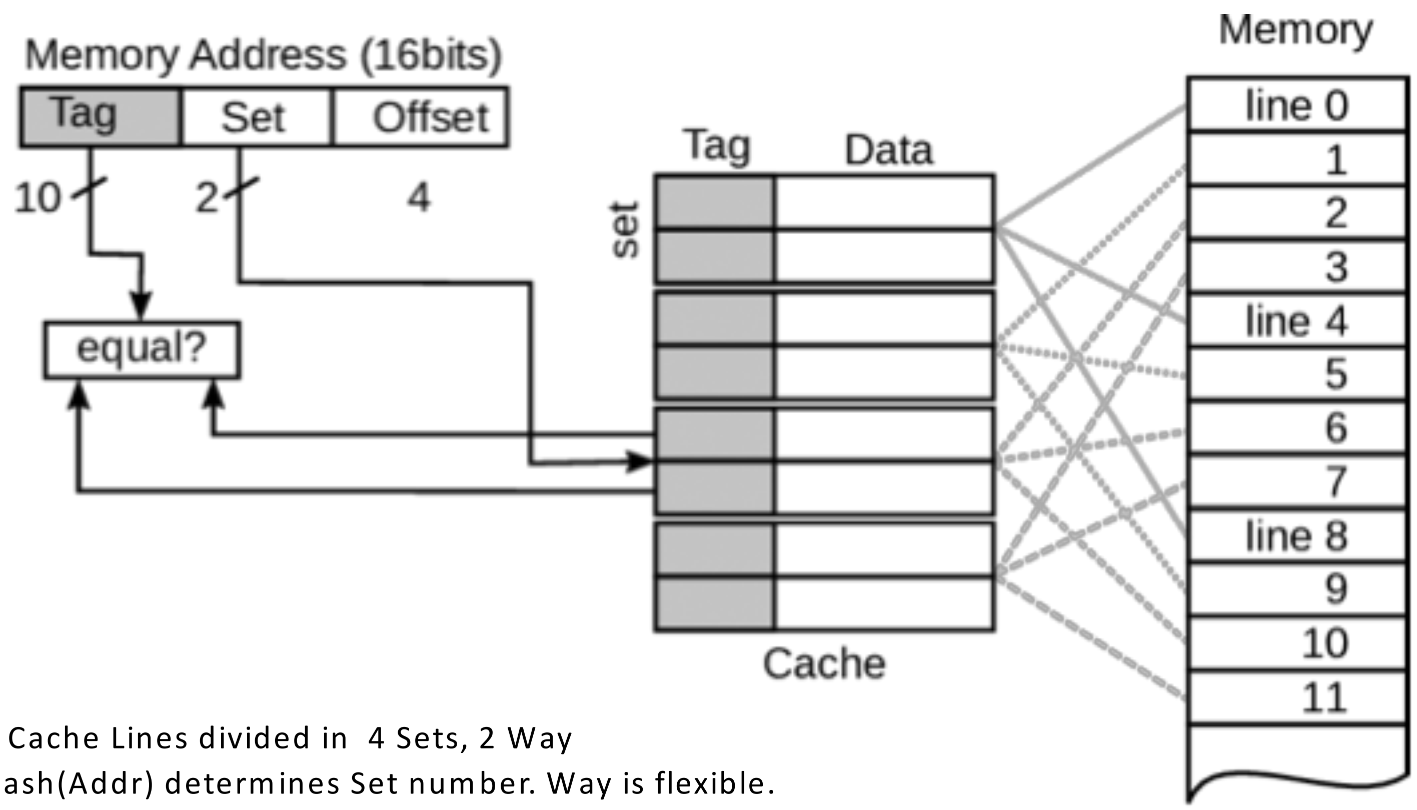


Thank you!

I STILL CAN'T MAKE SENSE OF THE DATA, PROF. SMITH. WHAT SHOULD I DO?
HMM...
I THINK YOU SHOULD SLEEP ON IT.
OK, I'LL GO HOME AND...
NO, I MEAN, YOU SHOULD SLEEP ON IT.
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Backup Slides

Set Associative Cache



8 Cache Lines divided in 4 Sets, 2 Way

Hash(Addr) determines Set number. Way is flexible.

“The eviction sets are virtual addresses, which we all map to the same physical page, thereby avoiding noise from the CPU data cache.”

- How?
  - Page Table manipulation?
- Why?
  - Many Different Physical pages can fit in cache, unless Prefetcher is a concern.
- Aliasing problem
  - Same Physical address, different virtual address