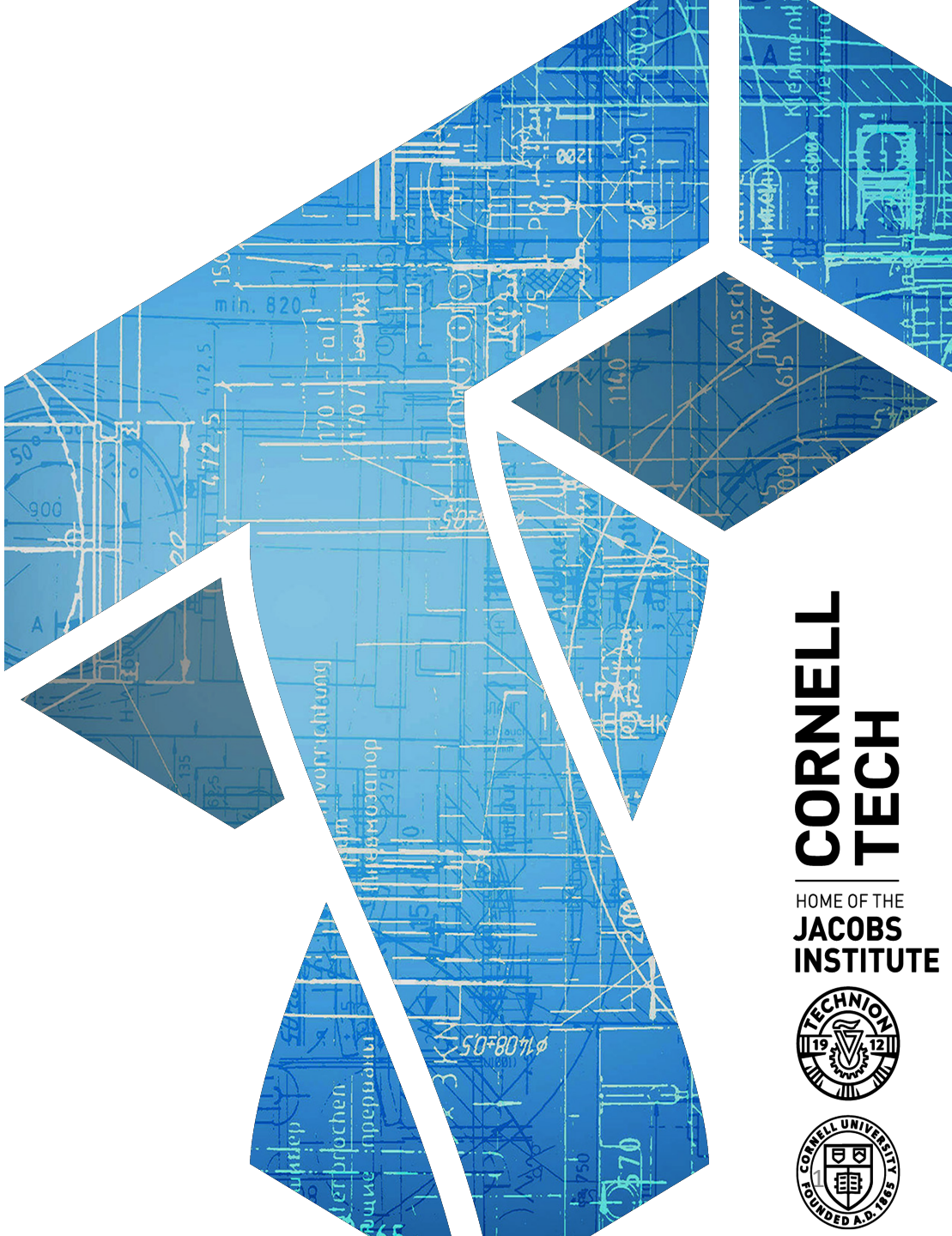
CS 6431: Microarchitectural vulnerabilities

Instructor: Tom Ristenpart

https://github.com/tomrist/cs6431-fall2021



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

One motivating context: cloud computing

> “If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry.”
>
> — John McCarthy, speaking at the MIT Centennial in 1961

A simplified model of public IaaS cloud computing

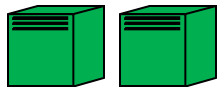
Users run Virtual Machines (VMs) on cloud provider's infrastructure

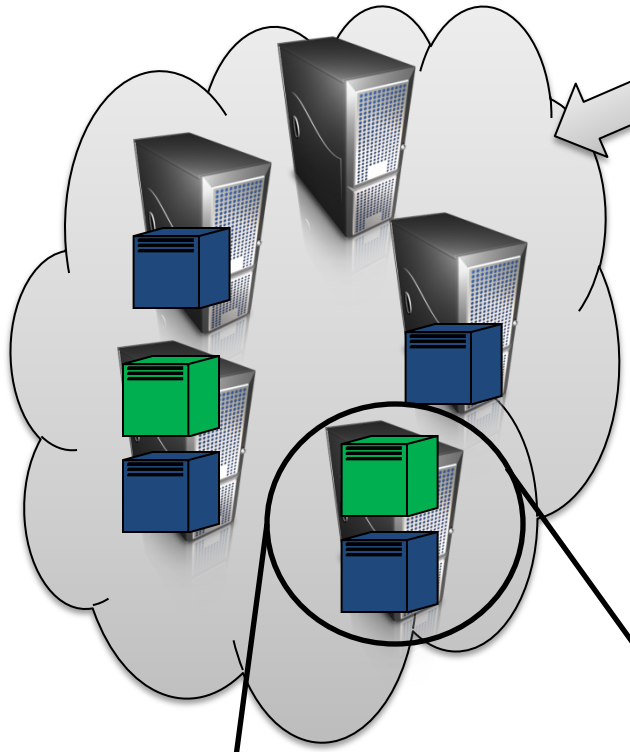
User A

virtual machines (VMs)

User B

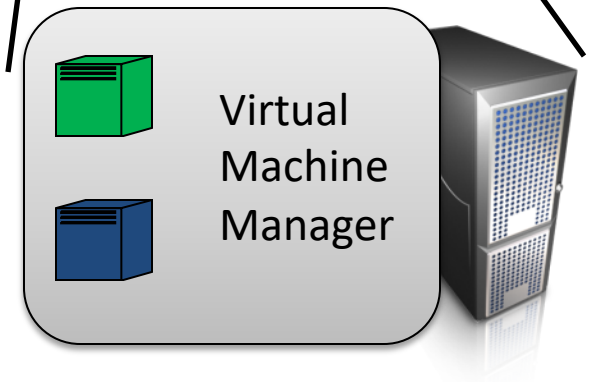
virtual machines (VMs)





Owned/operated by cloud provider

Google Compute Engine

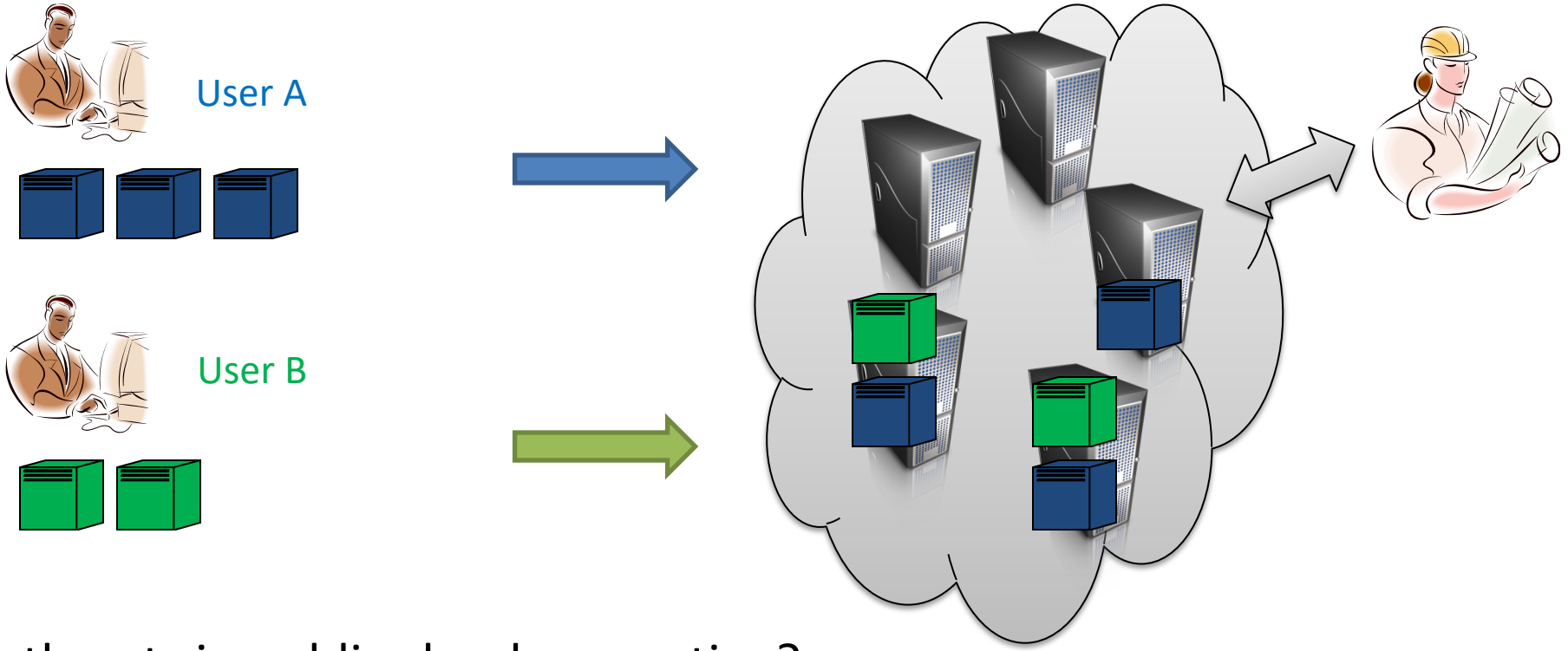


Multitenancy (users share physical resources)

Virtual Machine Manager (VMM)
manages physical server resources for VMs

To the VM should look like dedicated server

Trust models in public cloud computing



Security threats in public cloud computing?

Provider spying on running VMs / data

External attacks against infrastructure

Cross-user attacks

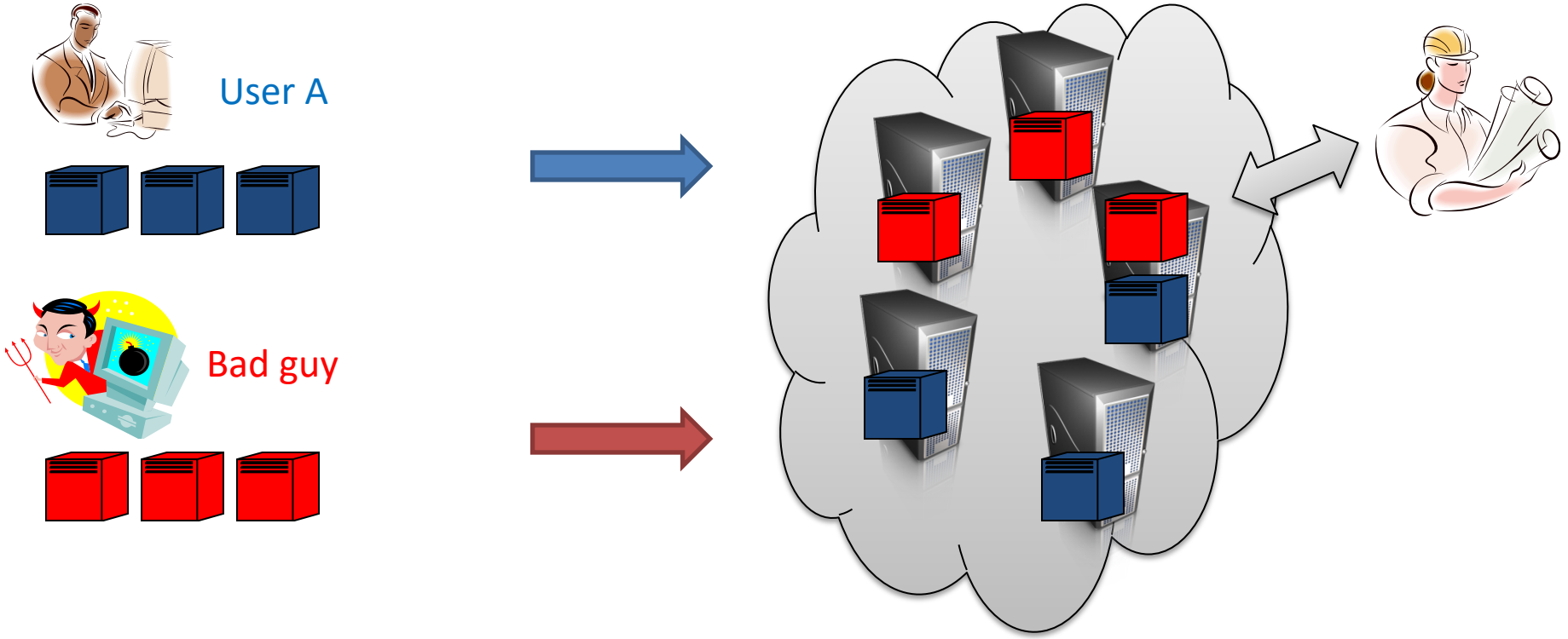
Escape-from-VM

Side-channels attacks

Degradation of service attacks

Resource-stealing attacks

Trust models in public cloud computing



Attacker identifies one or more victims VMs in cloud

1) Achieve advantageous placement via launching of VM instances

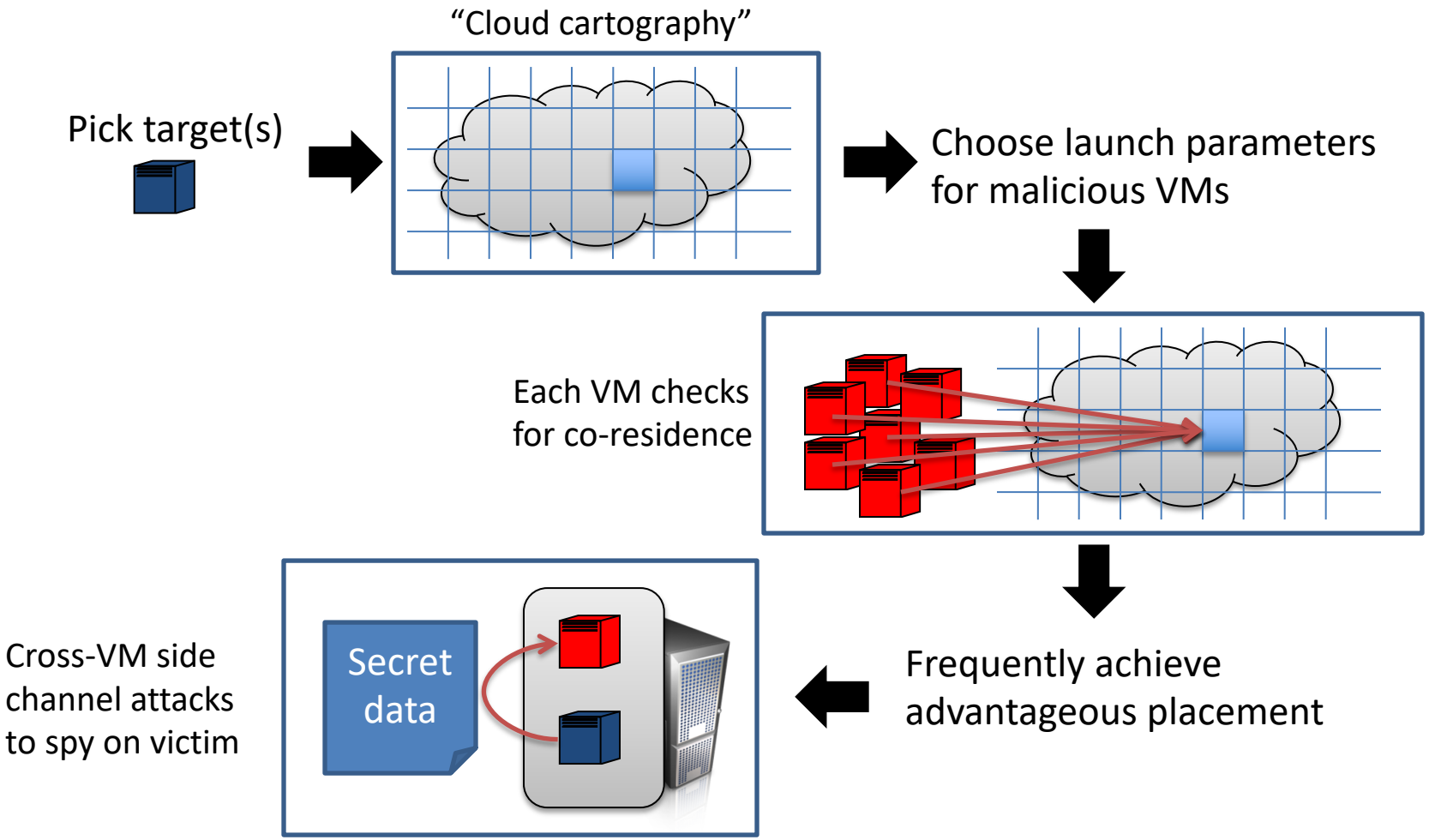
2) Launch attacks using physical proximity

Exploit VMM vulnerability DoS Side-channel attack

2009 case study with Amazon's EC2

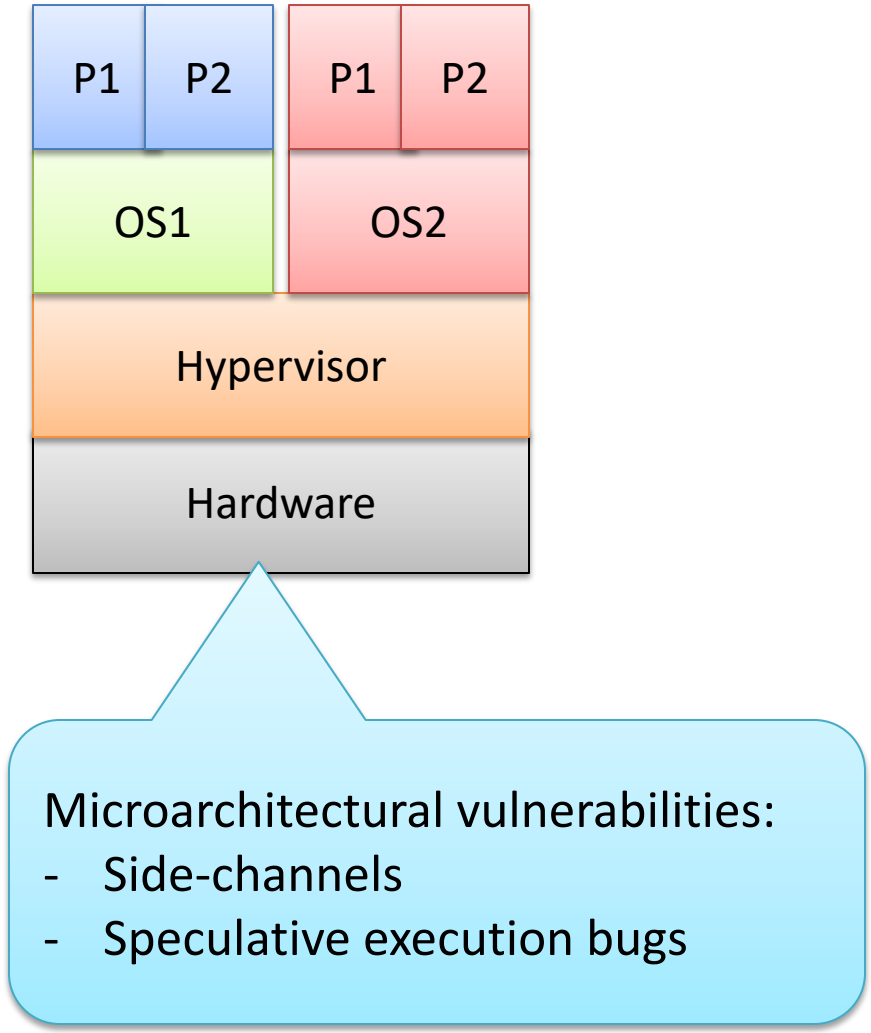
[Ristenpart et al. 2009]

Adversary able to:



Violating isolation

- Covert channels between VMs circumvent access controls
  - Bugs in VMM
  - Side-effects of resource usage
- Degradation-of-Service attacks
  - Guests might maliciously contend for resources
  - Resource-stealing attacks (e.g., Xen scheduler vulnerability)
- Side channels
  - Spy on other guest via shared resources



Microarchitectural side-channels

- Lots of research on exploiting microarchitectural side-channels
  - CPU-cache based: Percivel 2005, Tromer et al. 2005
- Prime+Probe attacks [Tromer et al. 2005]
- Flush+Reload (F+R) more robust side-channel in shared memory settings [Yarom, Falkner 2013]

(Part of) TLS handshake for RSA transport

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Pick random Ns

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Check CERT using CA public verification key

← CERT = (pk of bank, signature over it)

Pick random PMS
C <- E(pk,PMS)

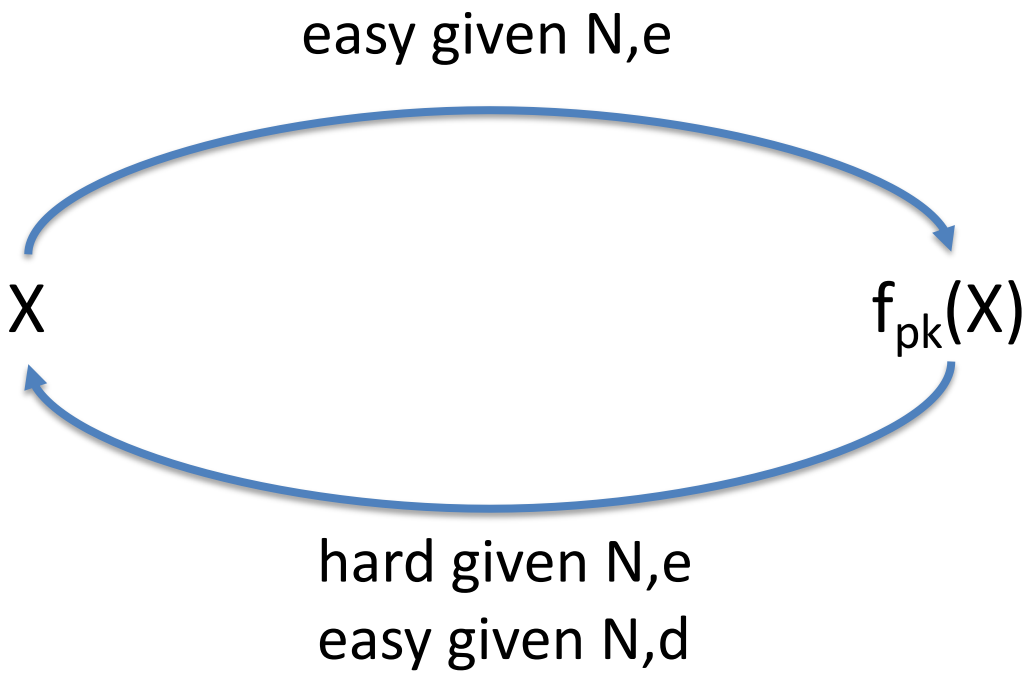
C →

PMS <- D(sk,C)

The RSA trapdoor permutation

pk = (N,e) sk = (N,d) with ed mod $\phi(N) = 1$

$f_{N,e}(x) = x^e \bmod N$ $g_{N,d}(y) = y^d \bmod N$



SqrAndMulExp(y,d,N)

$b_k,\ldots,b_0 = d$

f = 1

For i = k down to 0 do

    f = Square(y,N)

    If b_i = 1 then

        f = Multiply(f,y,N)

Return f

SqrAndMulExp(y,d,N)

$b_k,\ldots,b_0 = d$

f = 1

For i = k down to 0 do

    f = Square(y,N)

    If $b_i = 1$ then

        f = Multiply(f,y,N)

Return f

$$d = \sum_{b_i \neq 0} 2^i$$

$$y^d \equiv y^{\sum_{b_i \neq 0} 2^i} \equiv \prod_{b_i \neq 0} y^{2^i} \pmod{N}$$

$y^{11} = y^{8+2+1} = y^8 \bullet y^2 \bullet y$

$b_3 = 1$ $f_3 = 1 \bullet y$

$b_2 = 0$ $f_2 = y^2$

$b_1 = 1$ $f_1 = (y^2)^2 \bullet y$

$b_1 = 1$ $f_0 = (y^4 \bullet y)^2 \bullet y = y^8 \bullet y^2 \bullet y$

Highly vulnerable to side-channel attacks!

- Timing
- CPU State
- Power

$d_i = 1 \rightarrow$ "SM"

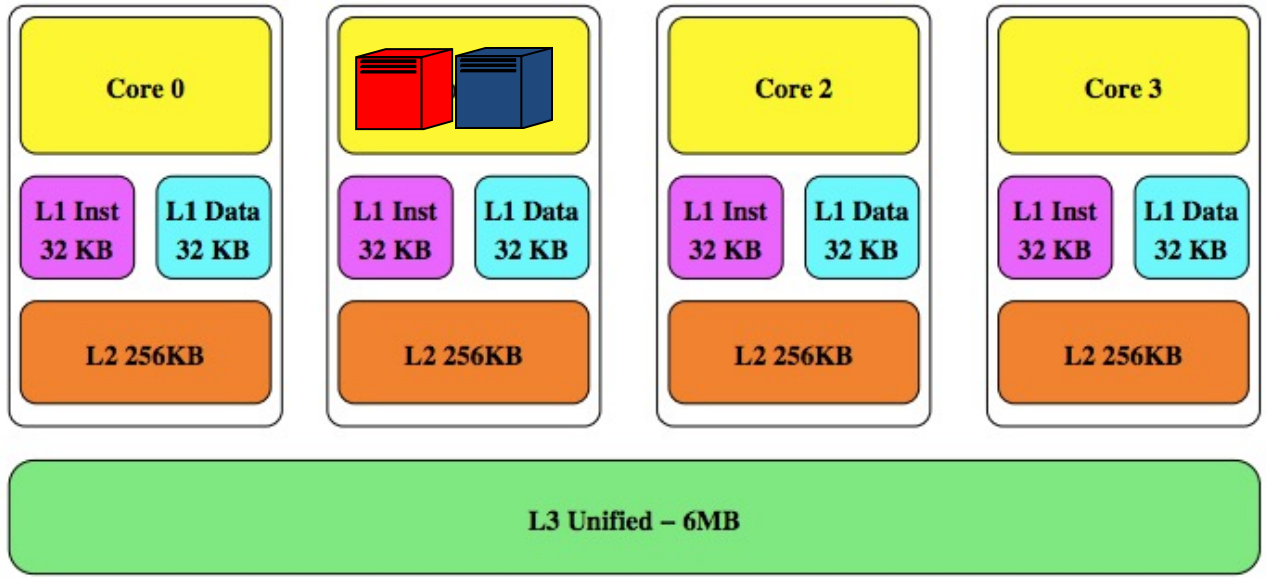
$d_i = 0 \rightarrow$ "S"

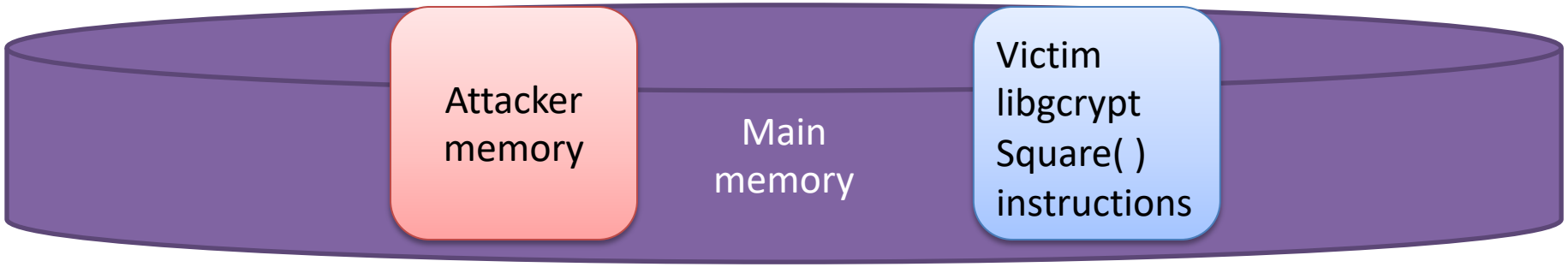
Sequence of function calls reveals secret key

Towards Prime+Probe

Suppose victim and attacker shares a core

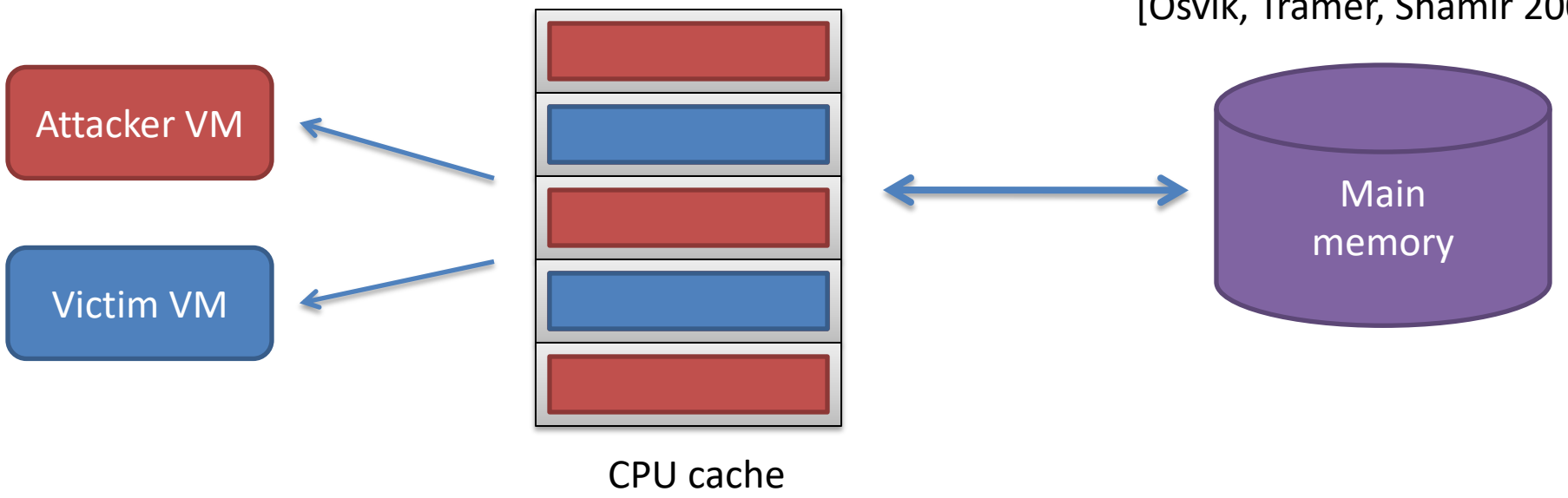
Also sharing L1 instruction & data caches





Cross-VM side channels using CPU cache contention

Introduced in cross-process setting by [Osvik, Tramer, Shamir 2006]



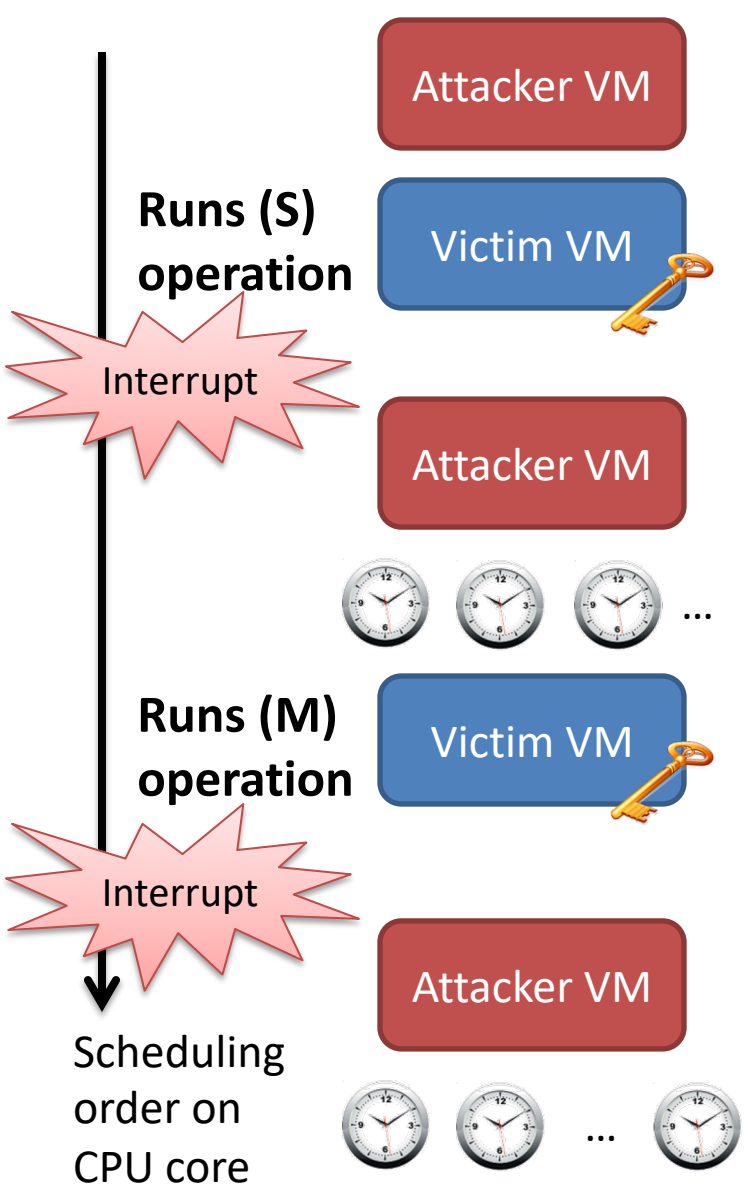
1) Read in a large array (fill CPU cache with attacker data)

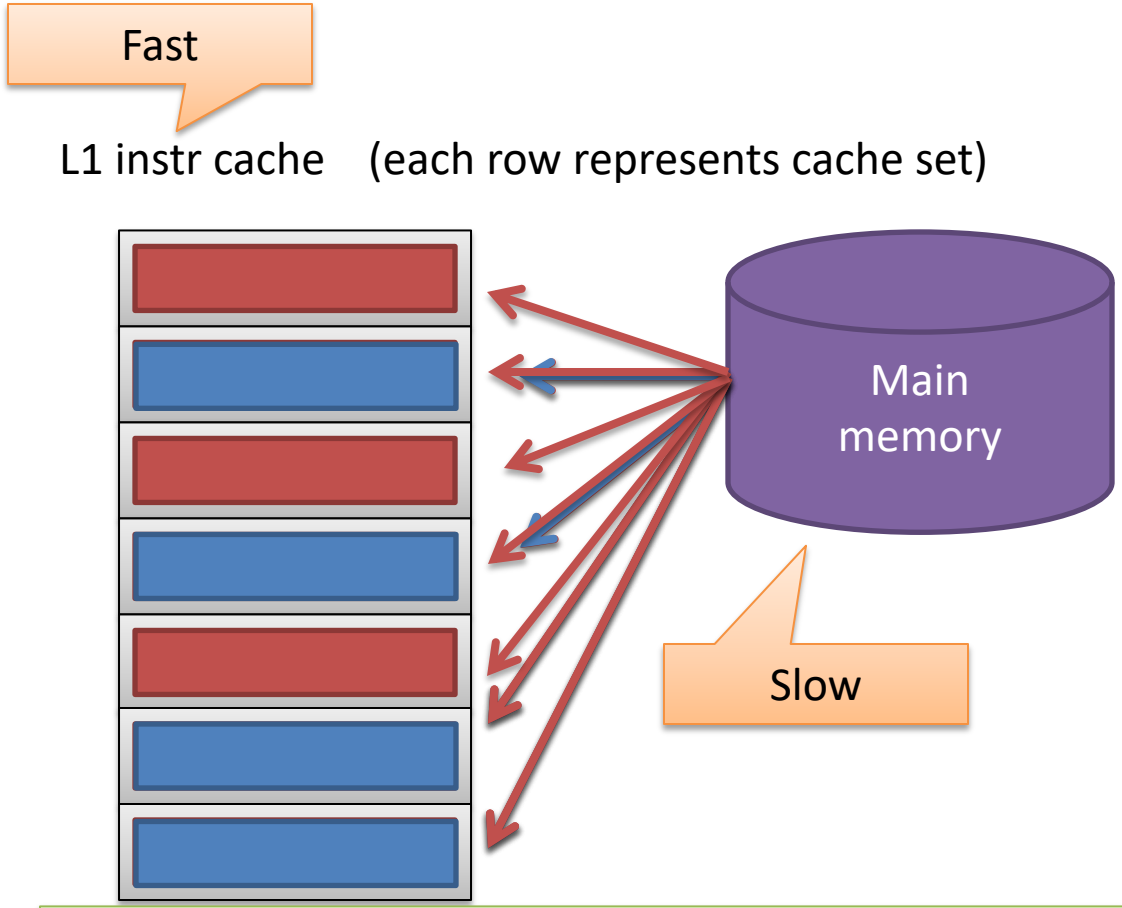
2) Busy loop (allow victim to run)

3) Measure time to read large array (the load measurement)

Locations in cache occupied by victim will take longer to load → Information about victim's use of cache revealed to attacker

Prime+Probe protocol





- Timings correlated to (distinct) cache usage patterns of S, M operations
- Can spy frequently (every ~16 µs) by exploiting scheduling

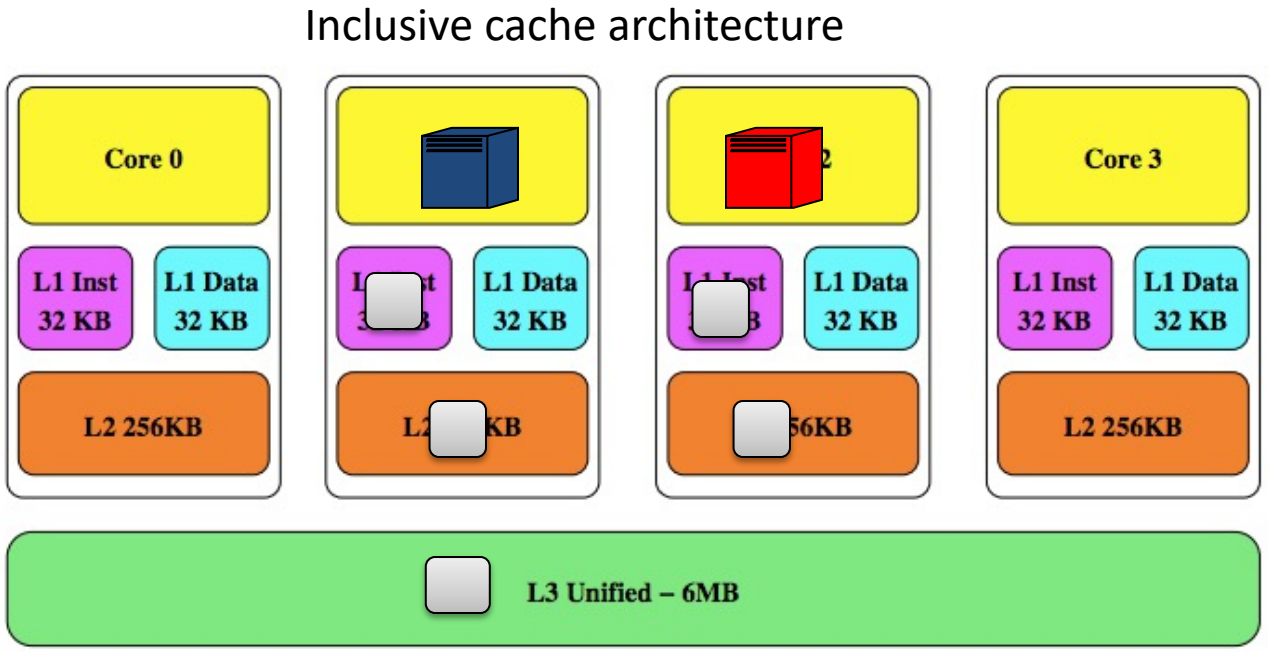
Prime+Probe limitations

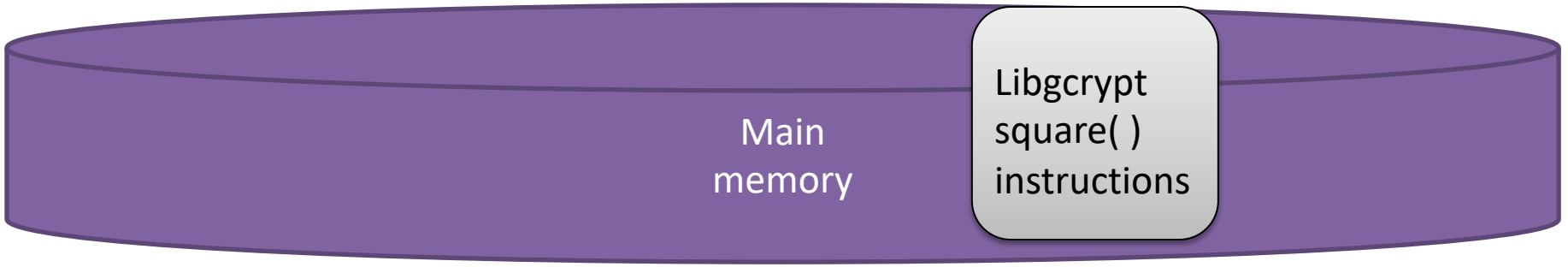
- Originally worked only for L1 caches
  - Some works extended to last-level cache (LLC) in certain settings
  - Multi-core settings difficult but feasible in lab (Zhang et al. 2012)
  - Inci et al. 2016 “Cache Attacks Enable Bulk Key Recovery on the Cloud”
- Lots of noise from various sources

Towards Flush+Reload

Deduplication-based memory page sharing (Linux, KVM, VMWare)

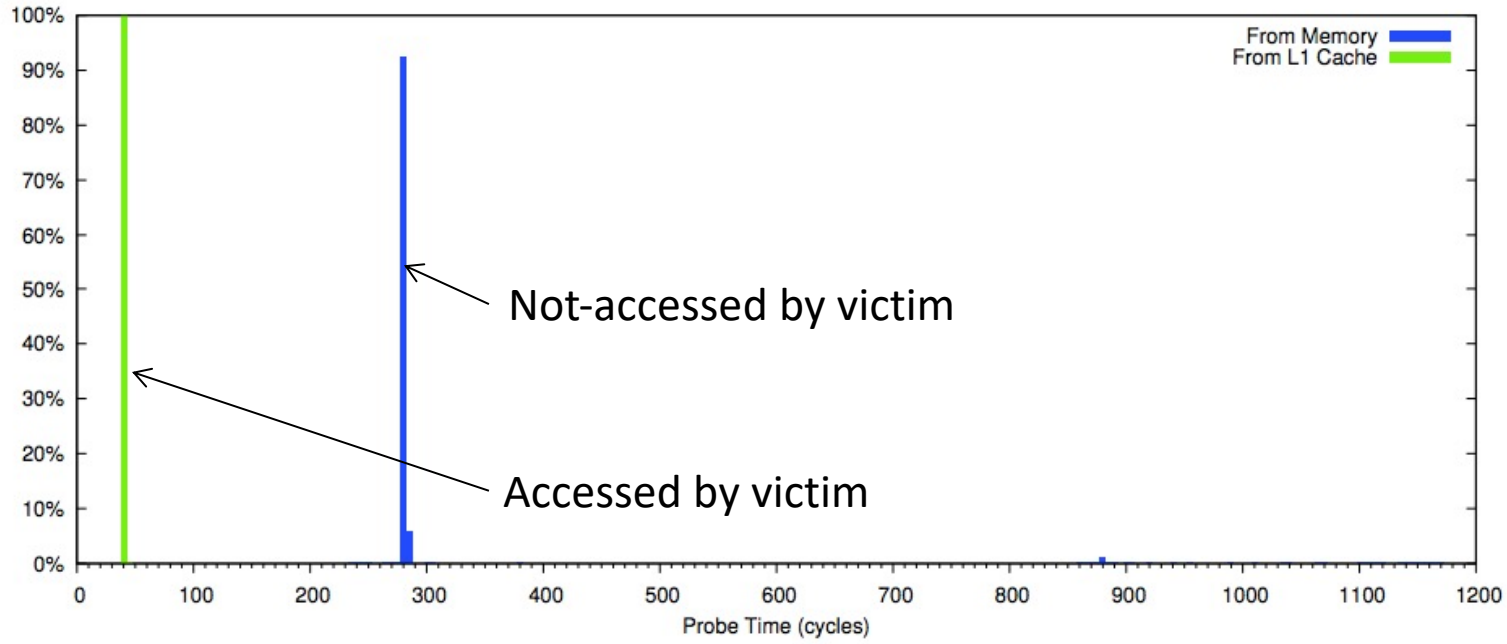
- Duplicate memory pages detected, physical pages coalesced
- Virtual address spaces different, but mapped to same physical addresses



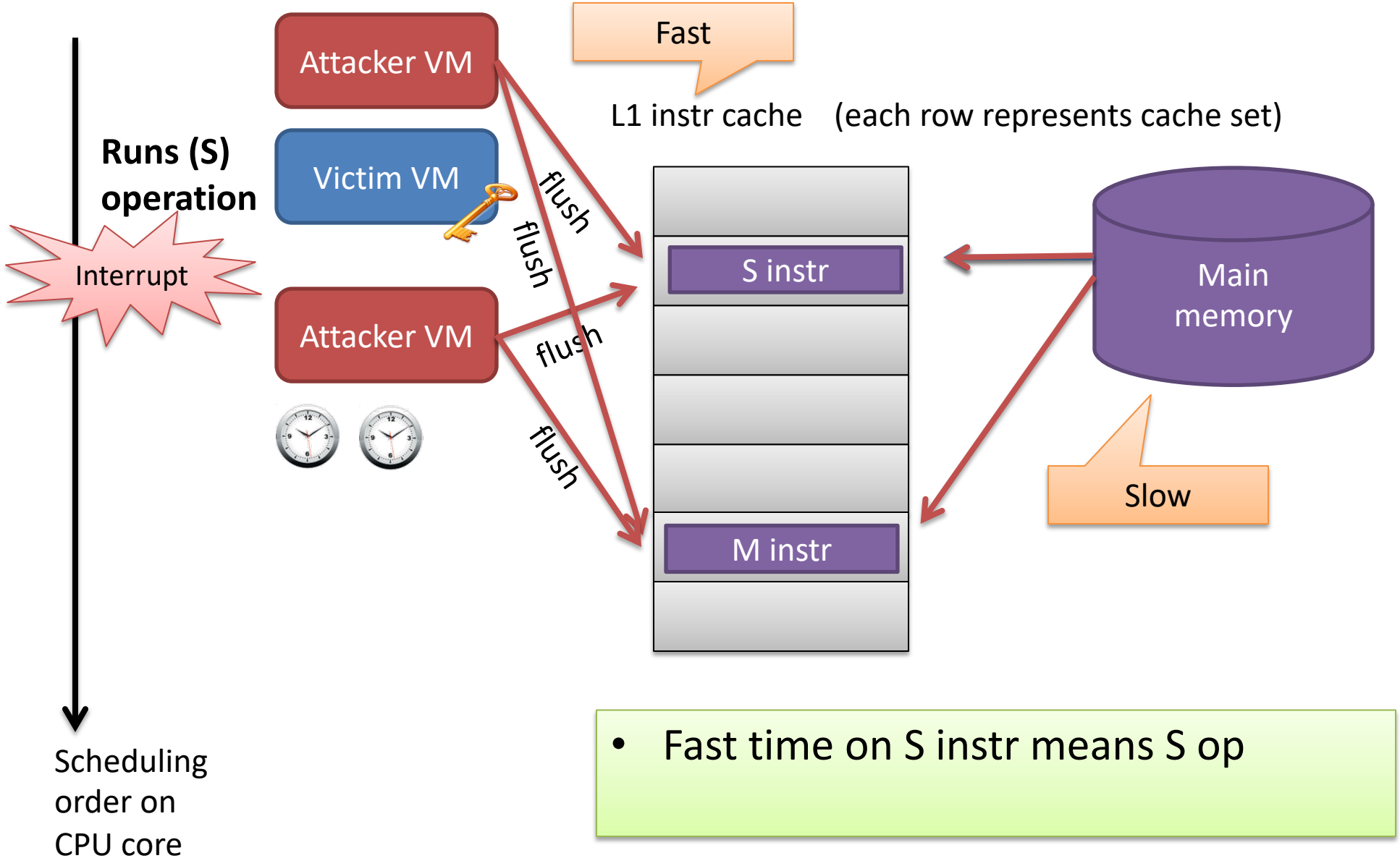


Flush+Reload protocol

- Flush from LLC memory line of interest
- Wait
- Time reloading memory line

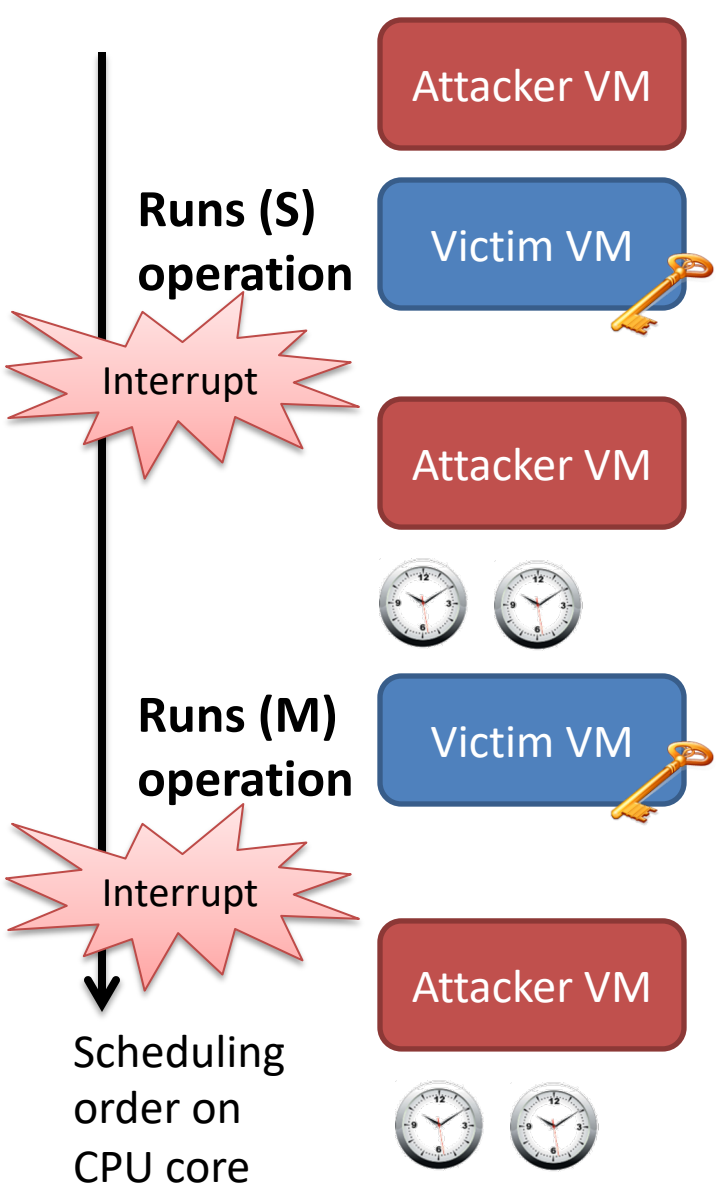


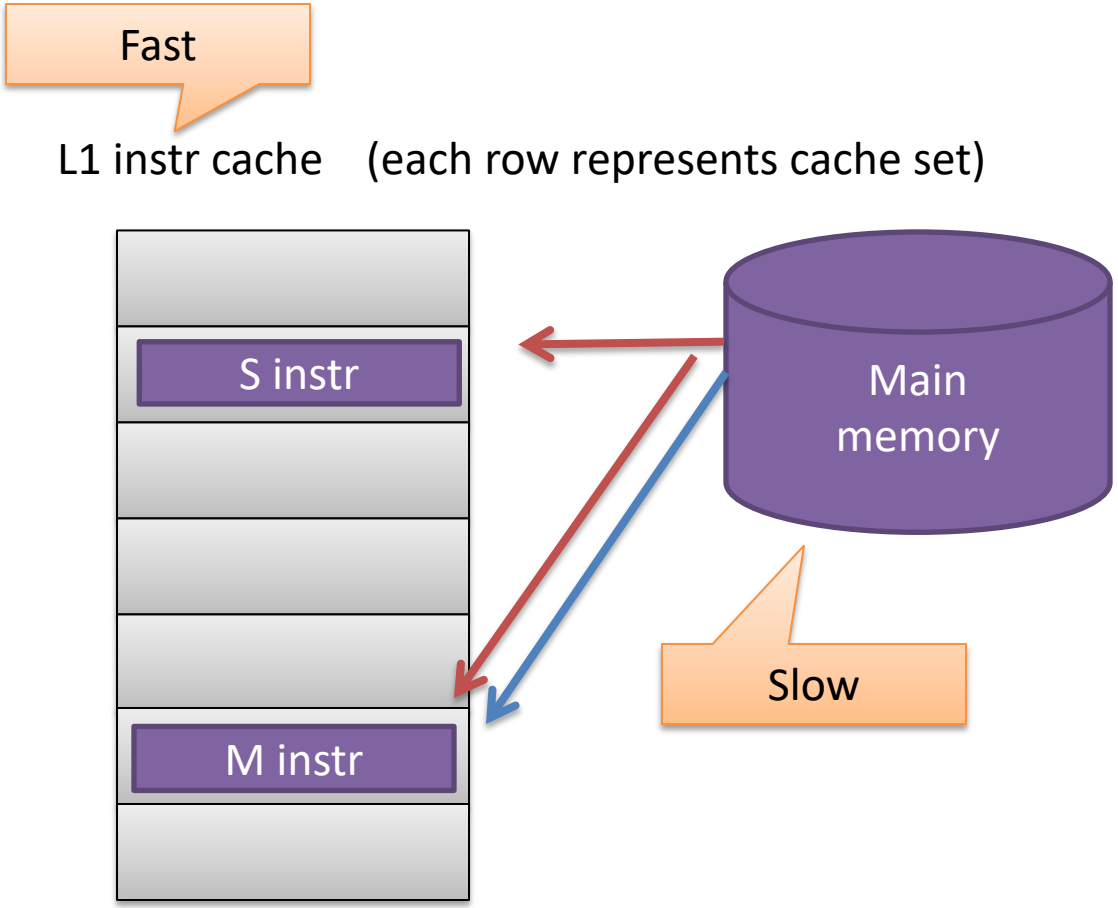
Flush+Reload protocol



- Fast time on S instr means S op

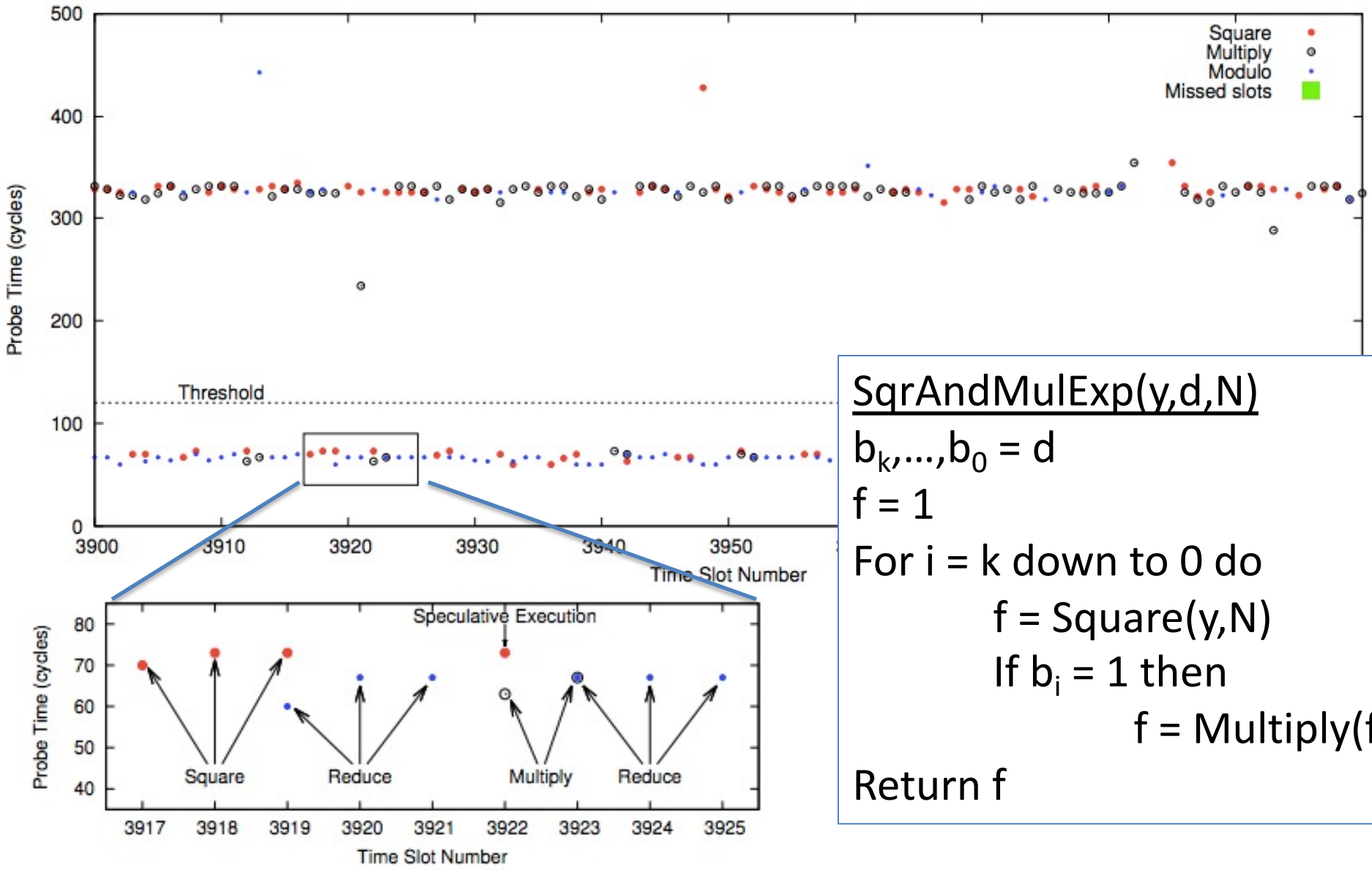
Flush+Reload protocol





- Fast time on S instr means S op
- Fast time on M instr means M op

Attacking Square and Multiply



SqrAndMulExp(y,d,N)

$b_k,\ldots,b_0 = d$

f = 1

For i = k down to 0 do

    f = Square(y,N)

    If b_i = 1 then

        f = Multiply(f,y,N)

Return f

Flush+Reload applicability

- Useful anywhere code is shared across processes / VMs / containers. Memory dedup or shared libraries
  - Deduplication turned off in Amazon EC2, but available in modern hypervisors
  - Different VMs do not share libraries
- Cross-tenant attacks in platform-as-a-service (PaaS) clouds
  - [Zhang et al. 2014]
- Used as building block for Meltdown, Spectre

Meltdown

- Discovered independently by Google Project Zero, Cyberus Technology, Graz University
- Speculative execution bugs in Intel x86, ARM, IBM processors + cache-based side-channels (Flush+Reload)
  - Allows reading kernel (or hypervisor, other VM) memory

Intel didn't warn US government about CPU security flaws until they were public

Meltdown and Spectre were kept secret

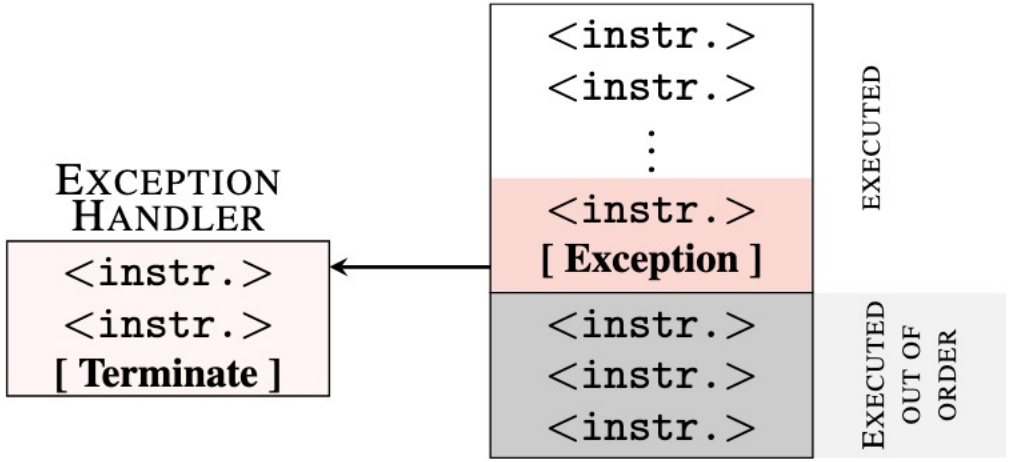
Researchers find malware samples that exploit Meltdown and Spectre

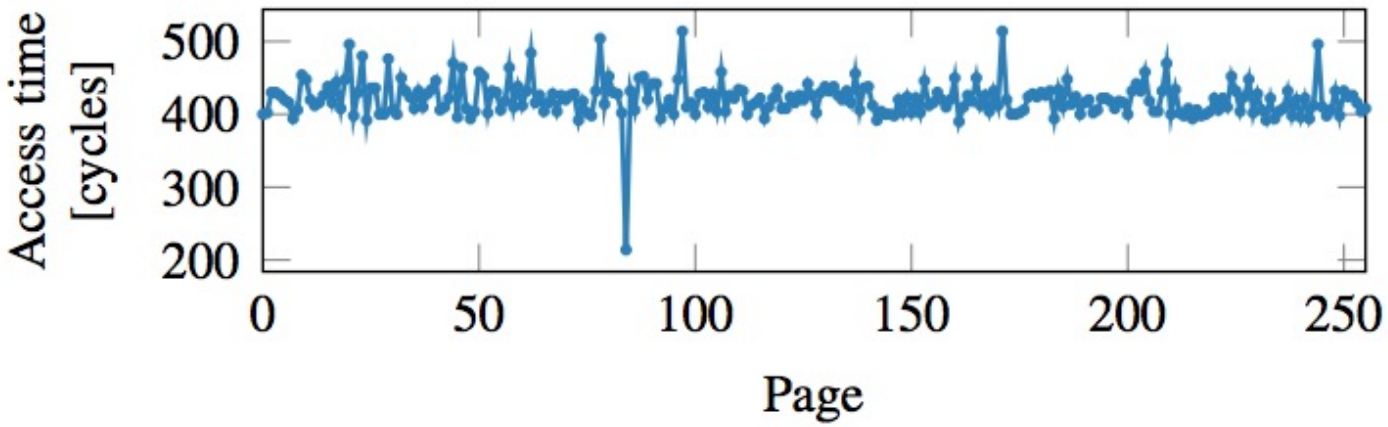
As of Feb. 1, antivirus testing firm AV-TEST had found 139 malware samples that exploit Meltdown and Spectre. Most are not very functional, but that could change.

Meltdown: intuition

```
1  raise_exception();
2  // the line below is never reached
3  access(probe_array[data * 4096]);
```

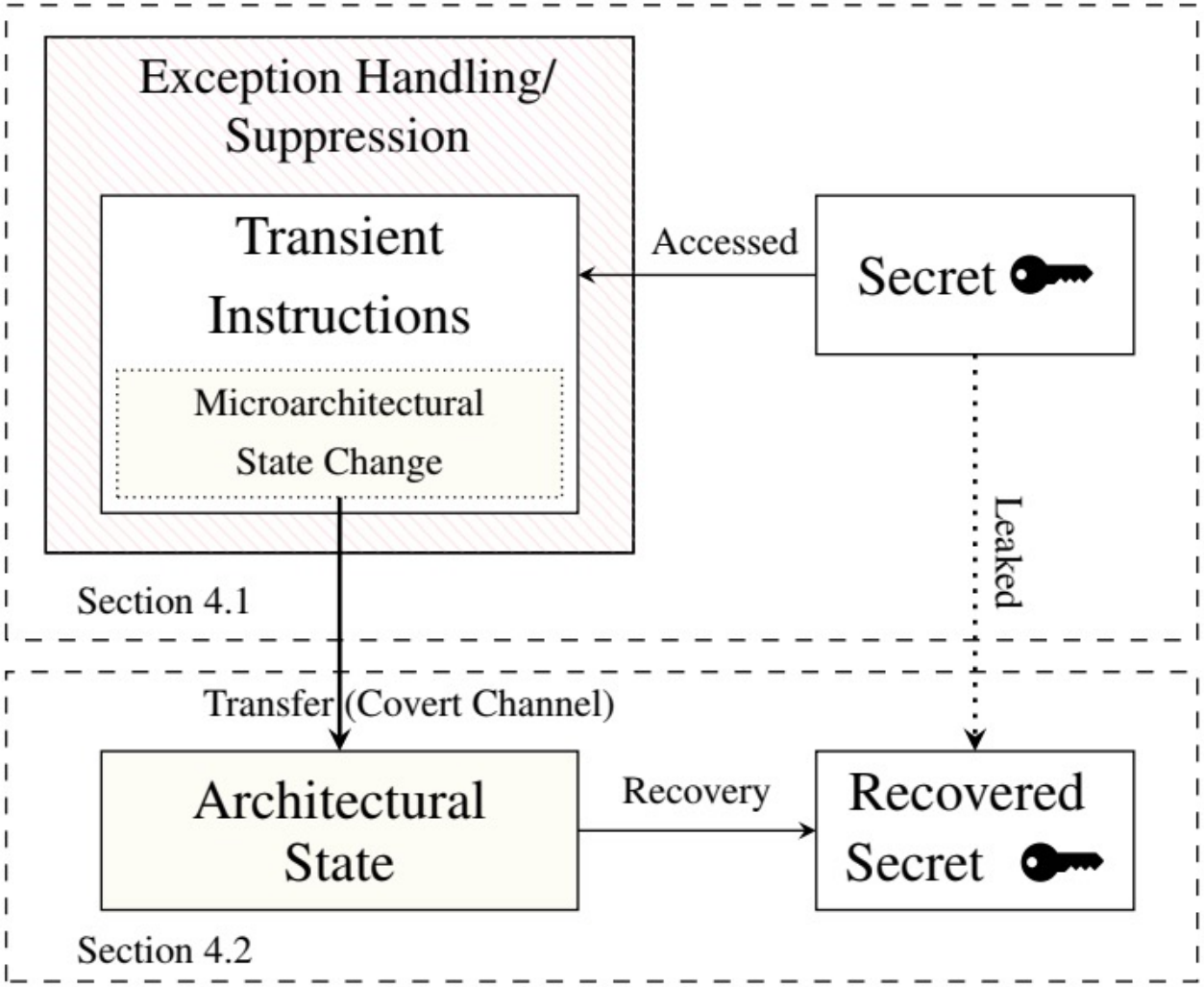
Ensures each value of data triggers access to distinct memory pages

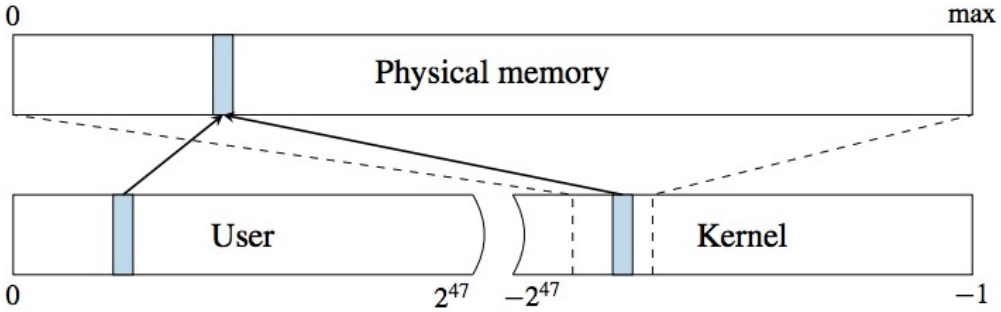




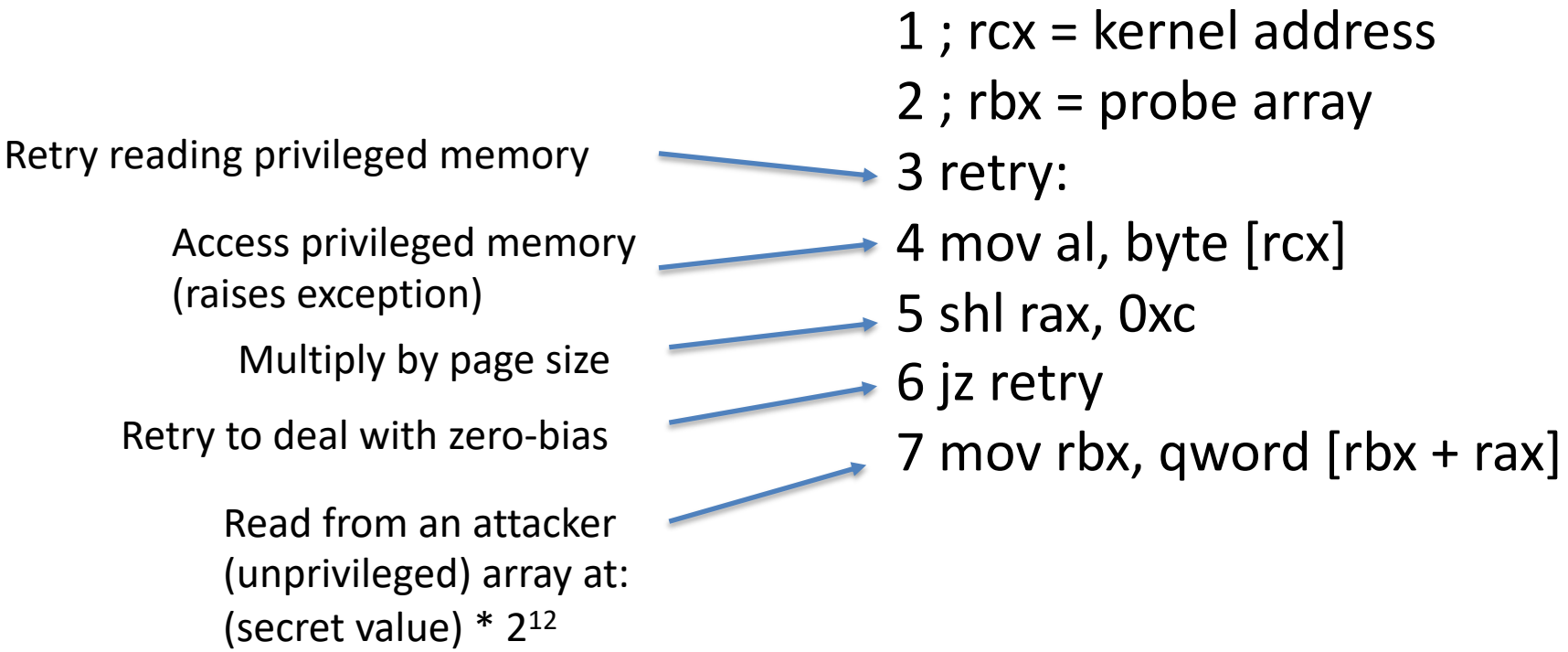
Flush+Reload measurement of probe_array

Meltdown: design





Meltdown: core spy code



Attacker times accessing [rbx + rax] for different values of rax

When finds one that loads fast, learns sensitive byte

Meltdown: nuances

- Privileged memory access raises segfault
  - Would normally crash process
  - Wrap in TXT or include signal handler
- Race condition between segfault and speculative code
  - Keeping speculative code lightweight helps
  - Some errors will creep up, but not many
- Kernel ALSR can be turned on
  - Won’t know a priori what to set rcx to
  - Can guess and check (8 GB memory requires ~128 guesses)

Meltdown works very well

```
f94b7690: e5 e5 e5 e5 e5 e5 e5 e5  e5 e5 e5 e5 e5 e5 e5 |...............|
f94b76a0: e5 e5 e5 e5 e5 e5 e5 e5  e5 e5 e5 e5 e5 e5 e5 |...............|
f94b76b0: 70 52 b8 6b 96 7f XX XX XX  XX XX XX XX XX XX XX |pR.k............|
f94b76c0: 09 XX XX XX XX XX XX XX XX  XX XX XX XX XX XX XX |...............|
f94b76d0: XX XX XX XX XX XX XX XX XX  XX XX XX XX XX XX XX |...............|
f94b76e0: XX XX XX XX XX XX XX XX XX  XX XX XX XX XX XX 81 |...............|
f94b76f0: 12 XX e0 81 19 XX e0 81 44  6f 6c 70 68 69 6e 31 |........Dolphin1|
f94b7700: 38 e5 e5 e5 e5 e5 e5 e5 e5  e5 e5 e5 e5 e5 e5 e5 |8...............|
f94b7710: 70 52 b8 6b 96 7f XX XX XX  XX XX XX XX XX XX XX |pR.k............|
f94b7720: XX XX XX XX XX XX XX XX XX  XX XX XX XX XX XX XX |...............|
f94b7730: XX XX XX XX 4a XX XX XX XX  XX XX XX XX XX XX XX |....J...........|
f94b7740: XX XX XX XX XX XX XX XX XX  XX XX XX XX XX XX XX |...............|
f94b7750: XX XX XX XX XX XX XX XX XX  XX e0 81 69 6e 73 74 |............inst|
f94b7760: 61 5f 30 32 30 33 e5 e5 e5  e5 e5 e5 e5 e5 e5 e5 |a_0203..........|
f94b7770: 70 52 18 7d 28 7f XX XX XX  XX XX XX XX XX XX XX |pR.}(...........|
f94b7780: XX XX XX XX XX XX XX XX XX  XX XX XX XX XX XX XX |...............|
f94b7790: XX XX XX XX 54 XX XX XX XX  XX XX XX XX XX XX XX |....T...........|
f94b77a0: XX XX XX XX XX XX XX XX XX  XX XX XX XX XX XX XX |...............|
f94b77b0: XX XX XX XX XX XX XX XX XX  XX XX XX 73 65 63 72 |............secr|
f94b77c0: 65 74 70 77 64 30 e5 e5 e5  e5 e5 e5 e5 e5 e5 e5 |etpwd0..........|
f94b77d0: 30 b4 18 7d 28 7f XX XX XX  XX XX XX XX XX XX XX |0..}(...........|
f94b77e0: XX XX XX XX XX XX XX XX XX  XX XX XX XX XX XX XX |...............|
f94b77f0: XX XX XX XX XX XX XX XX XX  XX XX XX XX XX XX XX |...............|
f94b7800: e5 e5 e5 e5 e5 e5 e5 e5 e5  e5 e5 e5 e5 e5 e5 e5 |...............|
f94b7810: 68 74 74 70 73 3a 2f 2f 61  64 64 6f 6e 73 2e 63 |https://addons.c|
f94b7820: 64 6e 2e 6d 6f 7a 69 6c 6c  61 2e 6e 65 74 2f 75 |dn.mozilla.net/u|
f94b7830: 73 65 72 2d 6d 65 64 69 61  2f 61 64 64 6f 6e 5f |ser-media/addon_|
f94b7840: 69 63 6f 6e 73 2f 33 35 34  2f 33 35 34 33 39 39 |icons/354/354399|
f94b7850: 2d 36 34 2e 70 6e 67 3f 6d  6f 64 69 66 69 65 64 |-64.png?modified|
f94b7860: 3d 31 34 35 32 32 34 34 38  31 35 XX XX XX XX XX |=1452244815.....|
```

503 KB/s using TXT “exception suppression”

Error rate: 0.02%

Memory dump of Firefox disclosing user passwords

Meltdown: countermeasures

- Minimize kernel memory mapped to virtual address space of process
  - KAISER countermeasure for kernel ASLR already deals with this. Suggested immediate deployment
- Hard split of virtual memory space into privileged / unprivileged
- Redesign microprocessors
  - For some defenses software microcode updates can be pushed to CPUs

Spectre vulnerability

- Independently discovered by Jann Horn (Google) and Paul Kocher (in collab with some academics)
- Also exploits speculative execution, but differently than Meltdown
  - Doesn’t target kernel memory, rather any inaccessible
  - Relies on branch prediction and speculative execution
- KAISER countermeasure doesn’t prevent

Microarchitectural vulnerabilities

- Many new attacks after Meltdown/Spectre
  - Foreshadow, Zombieload, Fallout, …
  - These papers high-impact, widely cited
    - Meltdown paper (2018) has >890 citations
    - Spectre paper (2019) has >1600 citations
- Microarchitectures are complex systems, root of trust in systems. Expect lots of vulnerabilities
  - Vulnerabilities like speculative execution
  - Side-channels, covert channels