上海科技大学 ShanghaiTech University





CS110 Computer Architecture

Dependability and RAID

Chundong Wang & Siting Liu

SIST, ShanghaiTech

TOAST LAB 吐 司 实 验 室

Review: Hamming ECC

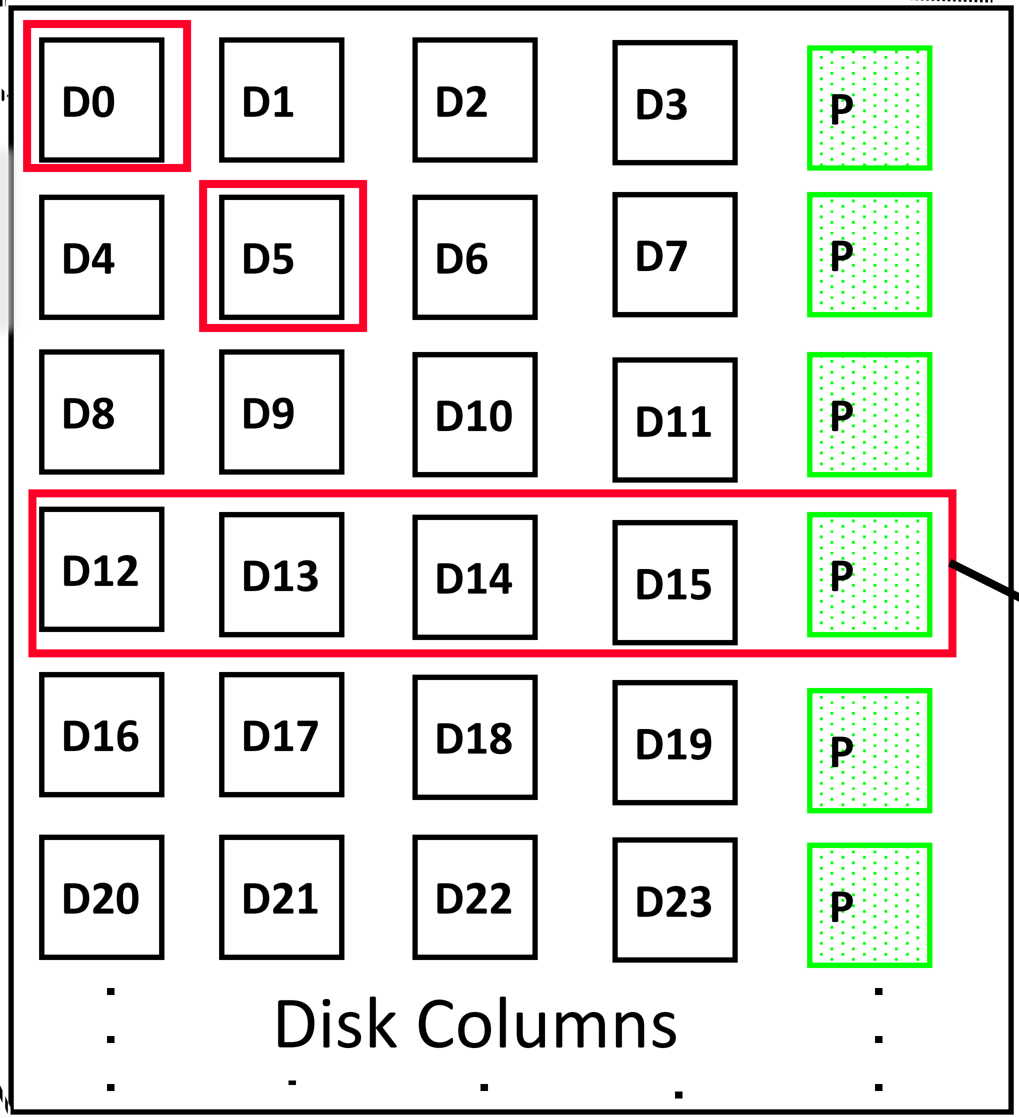
- Correct single-bit error, detect double-bit errors

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11
Parity bit coverage	p1	X		X		X		X		X		X		X		X
	p2		X	X			X	X			X	X			X	X
	p4				X	X	X	X					X	X	X	X
	p8								X	X	X	X	X	X	X	X

RAID 4: High I/O Rate Parity

Insides of 5 disks

Example: small read D0 & D5, large write D12-D15

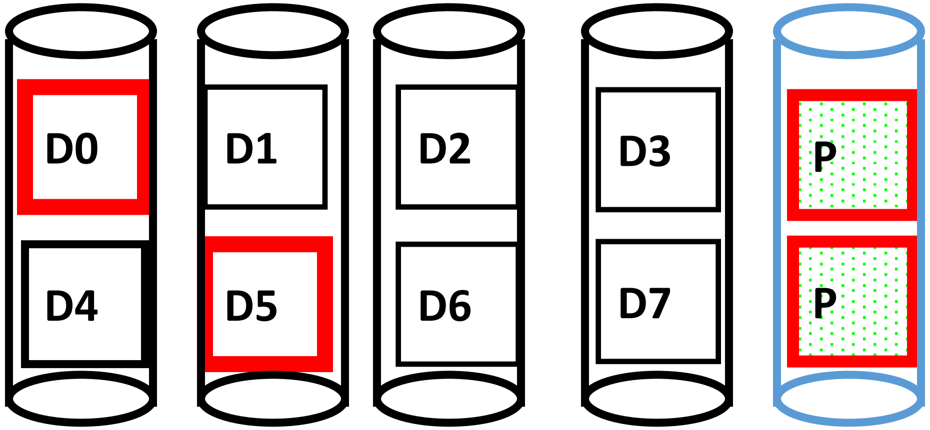


Increasing Logical Disk Address

Stripe

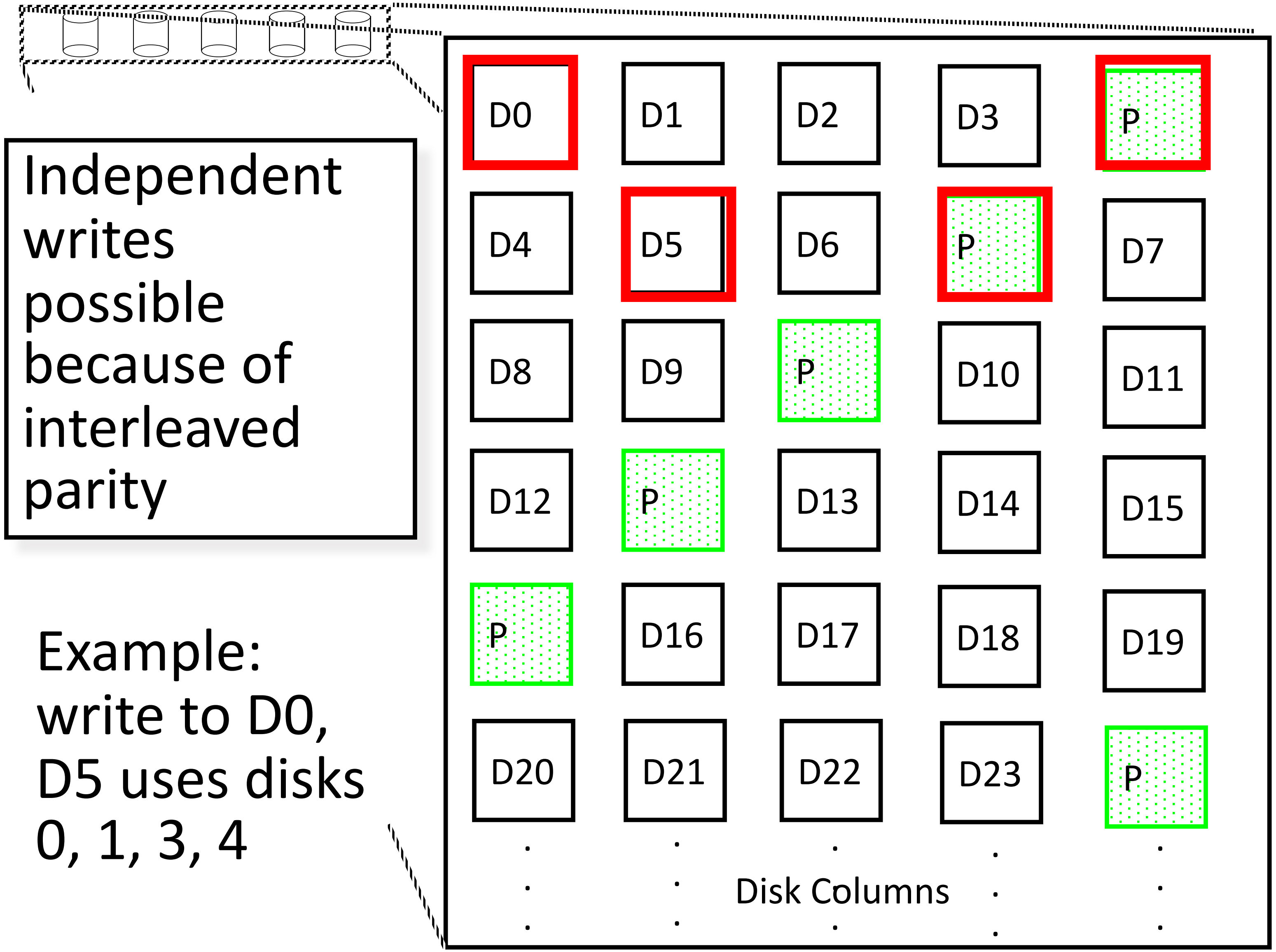
Inspiration for RAID 5

- RAID 4 works well for small reads
- Small writes (write to one disk):
  - Option 1: read other data disks, create new sum and write to Parity Disk
  - Option 2: since P has old sum, compare old data to new data, add the difference to P
- Small writes are limited by Parity Disk: Write to D0, D5 both also write to P disk



RAID 5: High I/O Rate Interleaved Parity

Independent writes possible because of interleaved parity

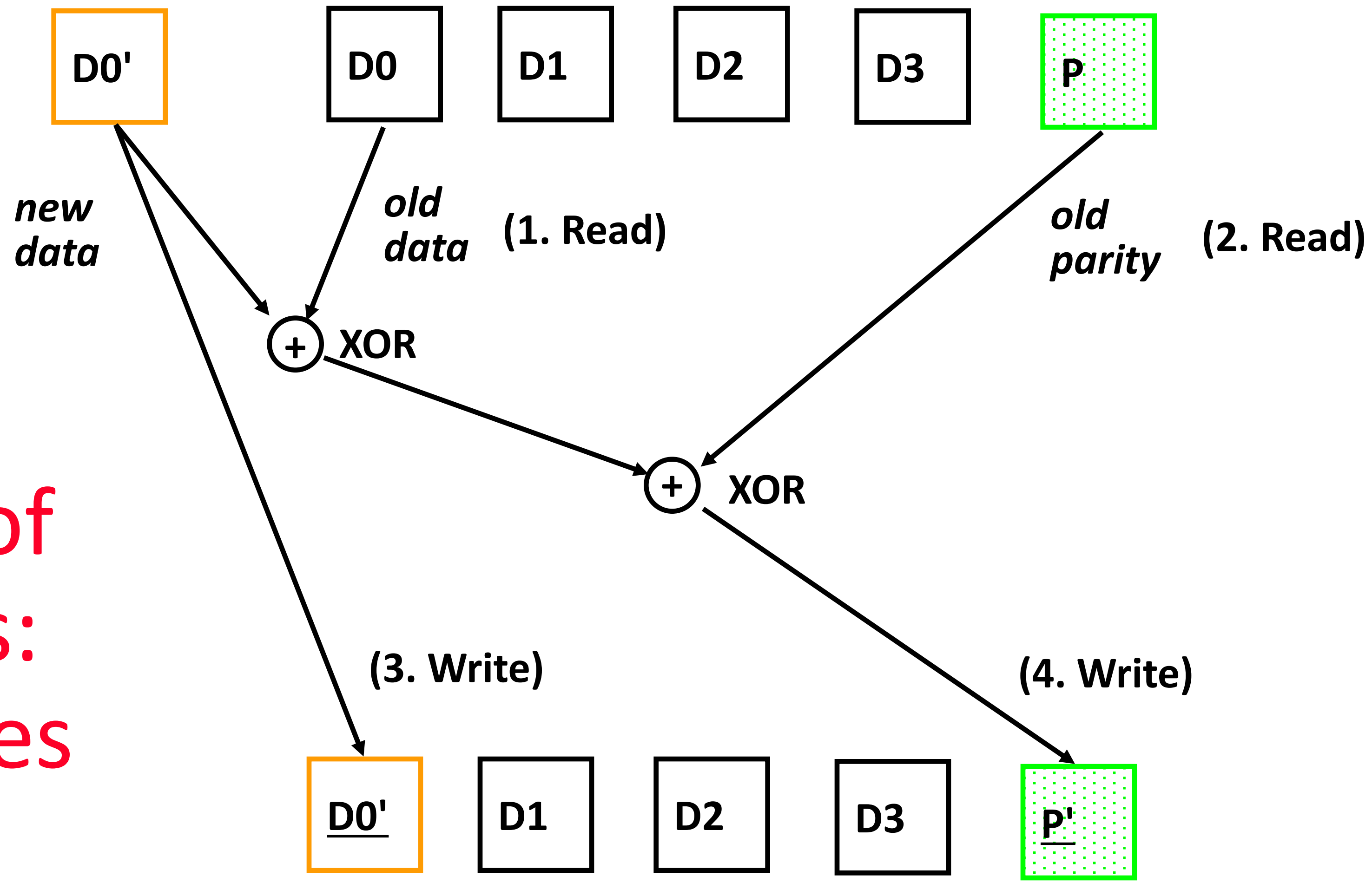


Increasing Logical Disk Addresses

Example: write to D0, D5 uses disks 0, 1, 3, 4

RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes



Problems of Disk Arrays: Small Writes





Outline

- The Lives of Others
  - Episode 1, over the air
  - Episode 2, very hard
- Inception
  - Plant a value with a hammer
- Mission Impossible
  - When, or where
- The Water Horse
  - Flush the Loch Ness
- Meltdown and Spectre

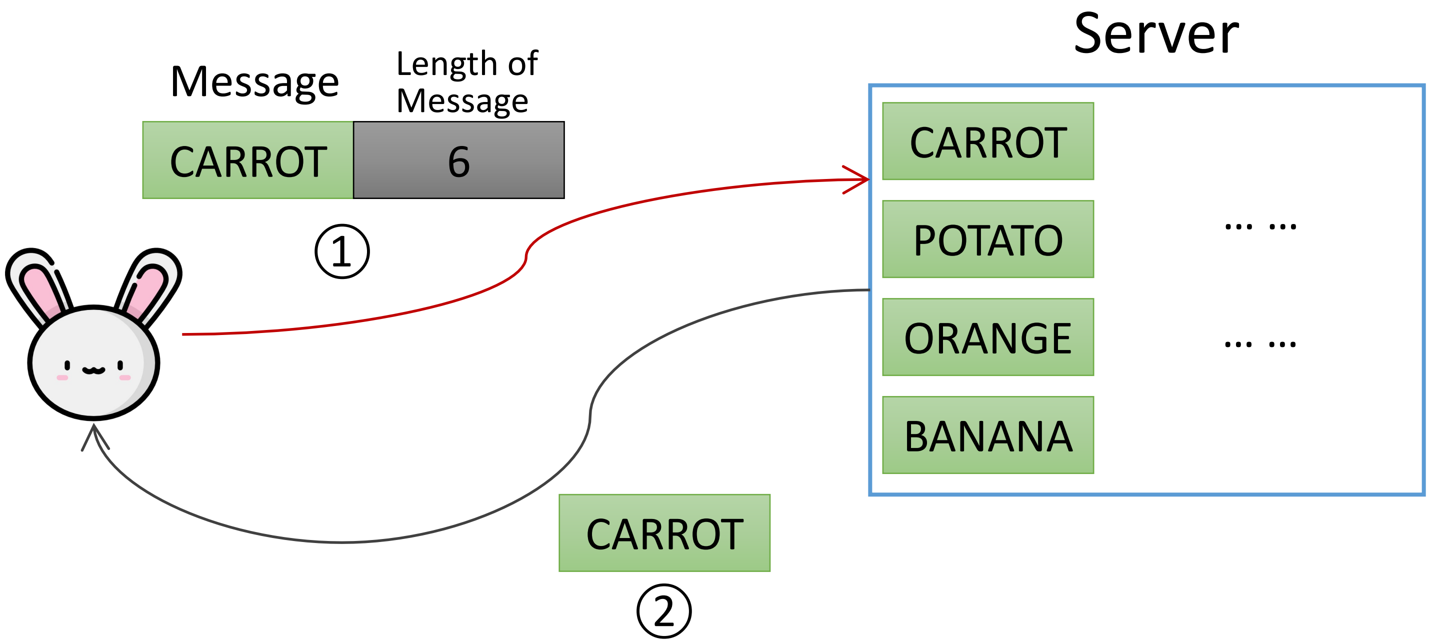
The Lives of Others
Over the air

Heartbleed

- A network vulnerability uncovered in 2014 [1]
- Found in popular OpenSSL cryptographic software library
  - SSL/TLS: Secure Socket Layer (*deprecated*), and Transport Layer Security
  - Widely used in many applications
    - Email, VPN, WeChat, Taobao, 12306, Momo, etc. [1, 2]

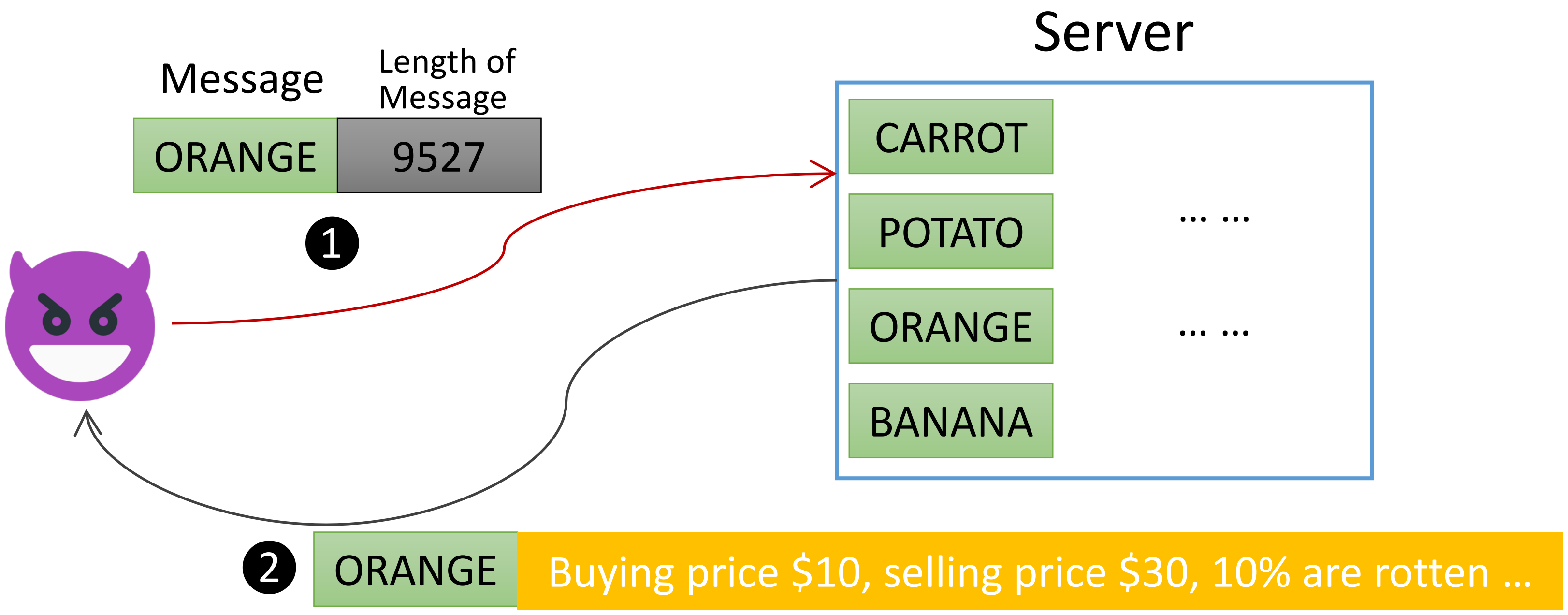
[1] https://heartbleed.com/
[2] https://daily.zhihu.com/story/3824566

How Heartbeat Works



The rule: one side sends a message with a length, and the other side replies with *the same* message according to *the length*.

How Heartbleed Works



The rule: one side sends a message with a length, and the other side replies with *the same* message according to *the length*.

Why “Heartbleed”?

- Found in TLS Heartbeat extension
  - To check whether a connection is still alive
  - One device confirms the other's continued presence by sending a specific payload in a packet that the other device sends back
    - Not only user to server, also possible server to user
  - Heartbleed happens if the packet’s length is not validated.

ORANGE	6	→	ORANGE	9527

Heartbleed

- The impact
  - Without using any privileged information or credentials, attackers can steal the victim’s secrets, including user names and passwords, instant messages, emails and business critical documents and communication.
- Is it a design flaw of SSL/TLS?
  - No. It is an implementation flaw.

How to fix it?

Implementation is critical

- Network protocols are like algorithms
  - Developers/programmers implement them for networking
  - A problematic implementation causes problems
- e.g., Wi-Fi and Bluetooth Low Energy
  - Wi-Fi: https://ieeexplore.ieee.org/document/9160891
  - BLE: https://dl.acm.org/doi/abs/10.5555/3489146.3489209

The Lives of Others
Very hard

Eavesdropping





Hacking a Hard Disk is Doable

- Hard disk is controlled by firmware
  - Remember the “controller processing time”?
  - Let’s hack it
- Re-flashing a disk’s firmware
- Or maliciously leaving a stealthy backdoor

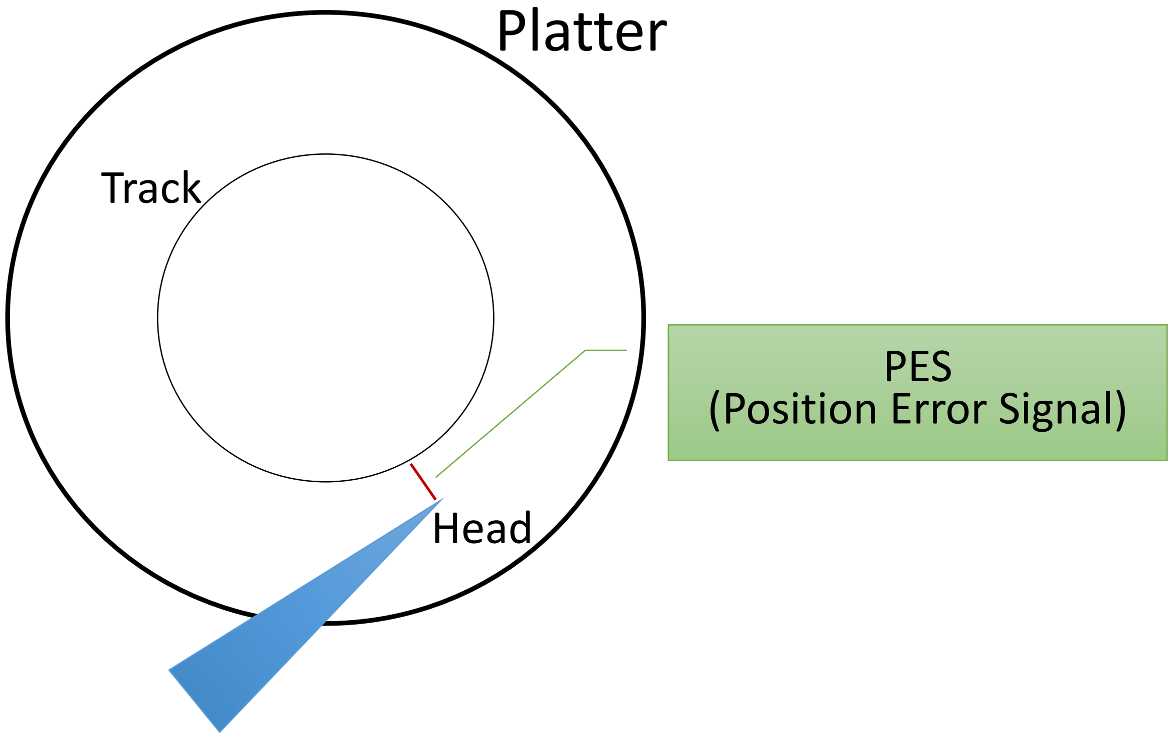
How to transform a hard disk to be a microphone?

Disk Head and Vibration

An external vibration can move the head, including voice vibration.

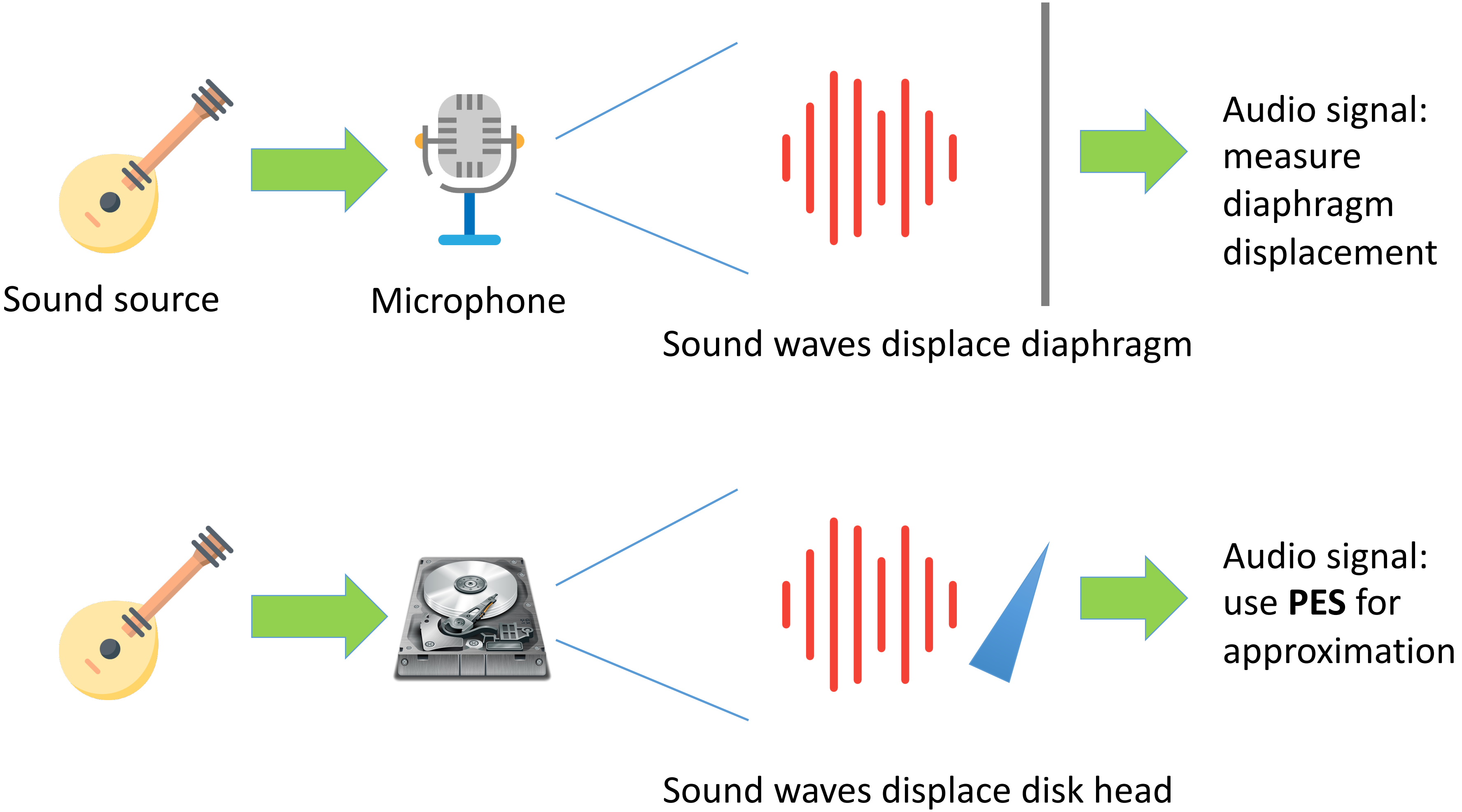
↓

The changes of PES



The bit density is very high for a hard disk drive.
PES utilizes feedback-control loop to keep the head on track.
It is a signal that can be read out with efforts.

To be a Microphone



It can work, but

- Required higher volume (90 dB)
- Audio can be recovered
  - Quality not very good

- How to defend?
  - Discard your disks? ← No.
  - Mask sound, secure future disks, etc. ← Yes.

Source: https://spqrlab1.github.io/papers/Kwong-HDDphone-IEEE-SP-2019.pdf





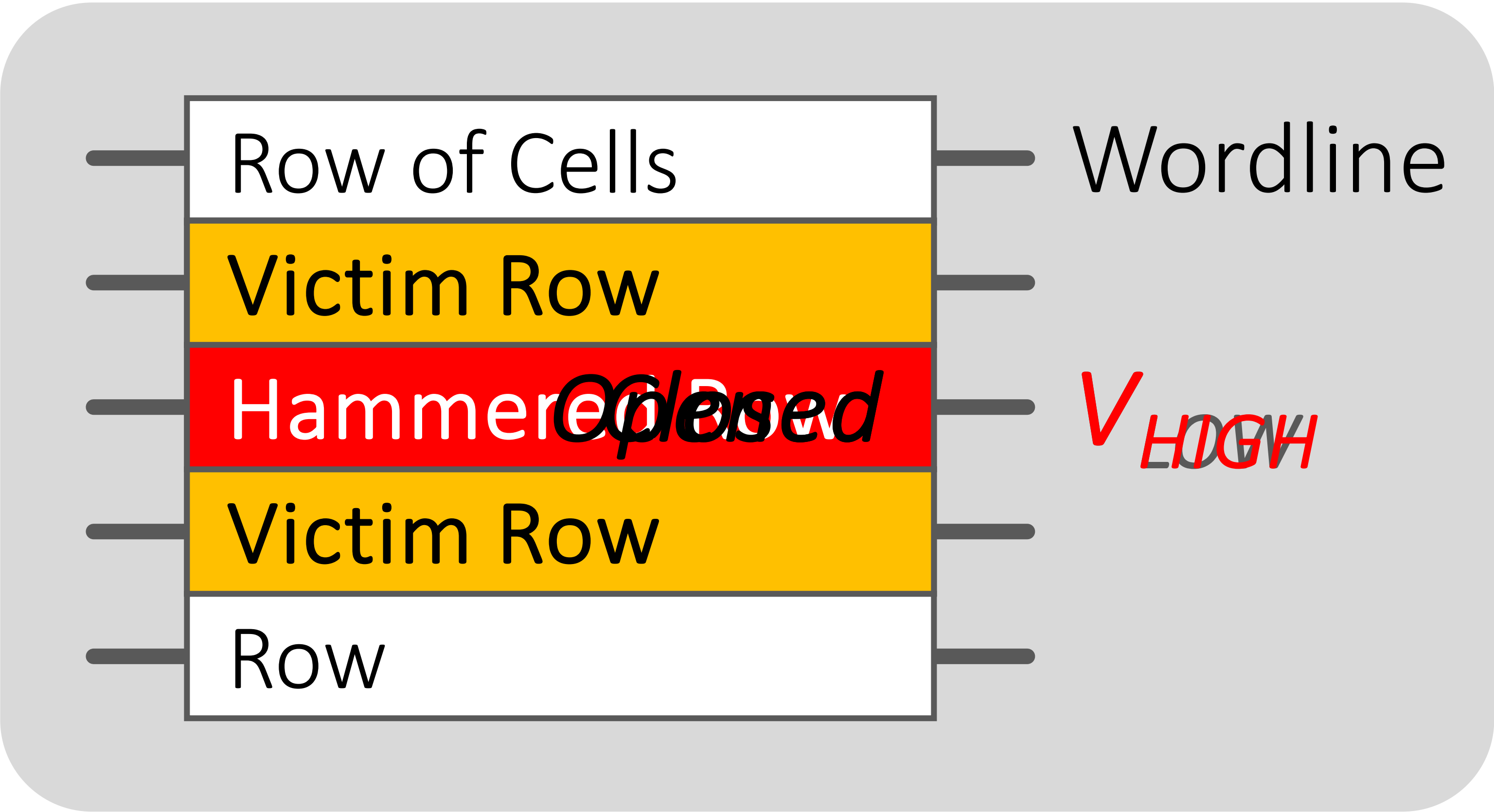
Inception

Plant a value with a hammer

Slides based on Onur Mutlu's Slides

https://people.inf.ethz.ch/omutlu/talks.htm

DRAM is Prone to Disturbance Errors



Repeatedly reading a row enough times (before memory gets refreshed) induces **disturbance errors** in **adjacent rows** in **most real DRAM chips you can buy today**

[Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors](), (Kim et al., ISCA 2014)

Why Rowhammer Happens?

- DRAM cells are too close to each other!
  - They are not electrically isolated from each other

- Access to one cell affects the value in nearby cells
  - due to electrical interference between
    - the cells
    - wires used for accessing the cells
  - Also called cell-to-cell coupling/interference

- Example: When we activate (apply high voltage) to a row, an adjacent row gets slightly activated as well
  - Vulnerable cells in that slightly-activated row lose a little bit of charge
  - If row hammer happens enough times, charge in such cells gets drained

How Rowhammer Becomes an Attack

- Picking a memory location
- Repeatedly accessing it without caching
- Adjacent rows in the same bank to be flipped
- An example to be exploited
  - Page table entry, containing a physical page no.
  - A bit of physical page no. flipped → another page

It sounds straightforward, but demands a lot of designs and tests.





The Impact of Rowhammer

Project Zero

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)

News and updates from the Project Zero team at Google

https://github.com/google/rowhammer-test

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks (ASPLOS 2016)
http://dl.acm.org/citation.cfm?doid=2872362.2872390
CAn’t Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory (USENIX Secuirty 2017)
https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-brasser.pdf
Another Flip in the Wall of Rowhammer Defenses (IEEE S&P 2018) https://ieeexplore.ieee.org/document/8418607
Throwhammer: Rowhammer Attacks over the Network and Defenses (USENIX ATC 2018)
https://www.cs.vu.nl/~herbertb/download/papers/throwhammer_atc18.pdf
SpecHammer: Combining Spectre and Rowhammer for New Speculative Attacks (S&P 2022)
https://ieeexplore.ieee.org/abstract/document/9833802
CSI:Rowhammer - Cryptographic Security and Integrity against Rowhammer (S&P 2023)
https://www.computer.org/csdl/proceedings-article/sp/2023/933600a236





Mission Impossible

When, or where

Encryption and Decryption

- Example: Advanced Encryption Standard (AES)
  - Used in OpenSSL and other applications

  - A key can be 128, 192, or 256 bits.
- Is it possible to guess a key?
  - Well, it’s a complicated question.
  - Brute-force?

Let’s Find a Side Channel

- Implementations have behaviours (observations) regarding different inputs
  - Timing
  - Accessed CPU cache lines
  - Power consumption
  - Sound

  Side channels
- Not theorical properties of algorithm
- Side-channel attacks
- How to leverage a side channel?
  - First, to instrument, e.g., timing, power, etc.
  - More important, to figure out the relationship between observations via the side channel and inputs to the running program

Timing-based Attack

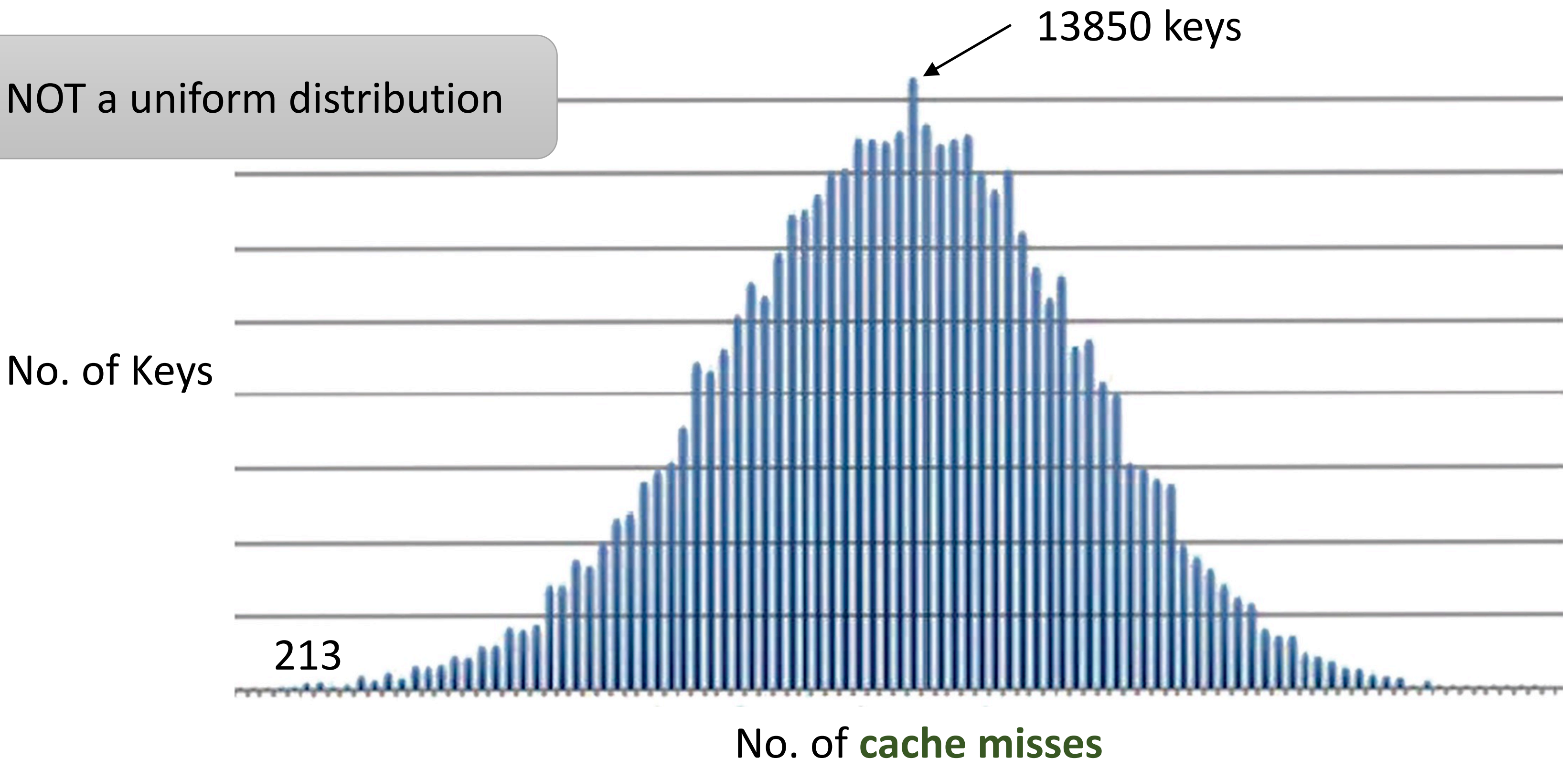
Key	Execution Time
2046	3ms
12138	9ms

```
if (key < 9527) {
    output = key;
} else {
    output = sqrt(key) * pow(pi, log(2, key));
}
```

When an implementation has data-dependent variations on its execution, it is prone to timing-based attacks.

Timing-based Attack

NOT a uniform distribution



By executing AES-128 encryption with 256000 keys

https://onlinelibrary.wiley.com/doi/full/10.1002/stvr.1718

Access-based Attack

Key	Cache line hit/miss tag
2046	0011
9527	1001
3721	0001
12138	1110

How can we mitigate?

Hit/miss records of cache lines are also likely to leak secret.

An Example

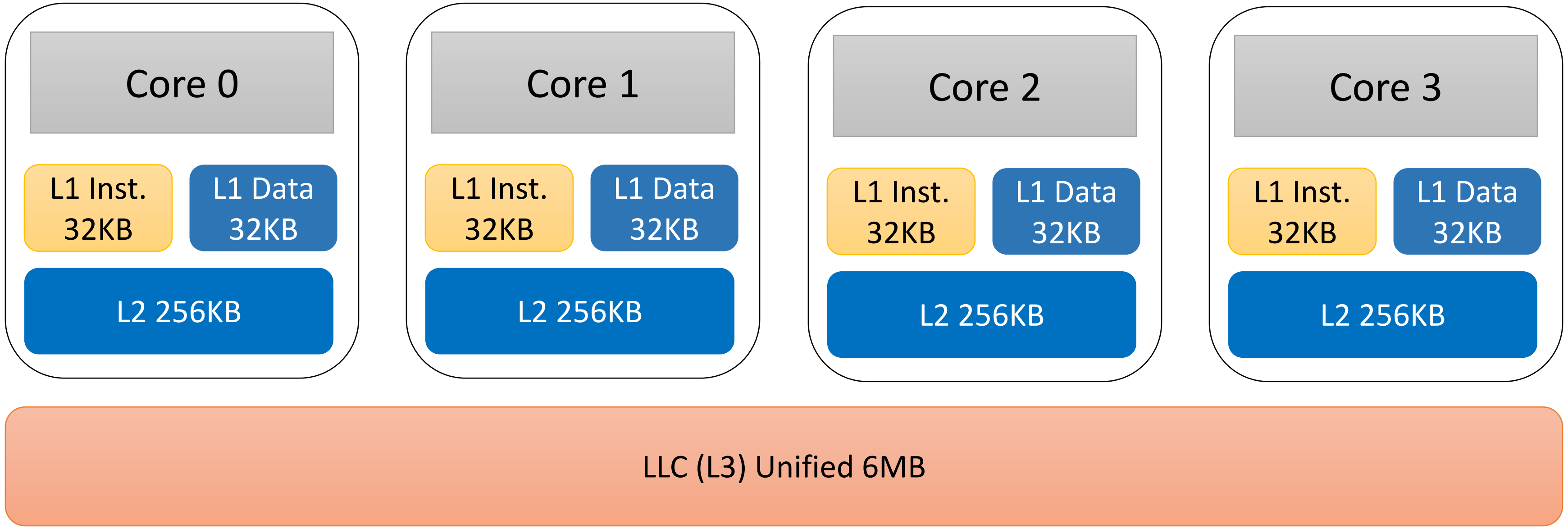
- Cachebleed
  - Reported in March 2016
  - Timing-based attack
    - On RSA in OpenSSL
    - RSA is a public-**key** cryptographic system for encryption and decryption
  - By detecting cache-bank conflicts via timing variations ➔ to reconstruct a **key**

Cachebleed https://ts.data61.csiro.au/projects/TS/cachebleed/

The Water Horse

Flush the Loch Ness

Inclusive Cache and Shared Pages



Intel Ivy Bridge Cache Architecture (Core i5-3470)

1. For an inclusive cache, a cache line in higher-level caches is in LLC.
2. Programs may share memory pages, e.g., the same executable files.

Flush+Reload

- To build a side channel
- Manually share memory pages between attacker’s and victim’s programs
- Attacker forcefully flushes and reloads cache lines
  - To observe the victim’s cache misses/hits

  ➔ to speculate the victim’s secret
- Why inclusive cache for study?
  - All levels would be flushed

https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-yarom.pdf

Flush+Reload

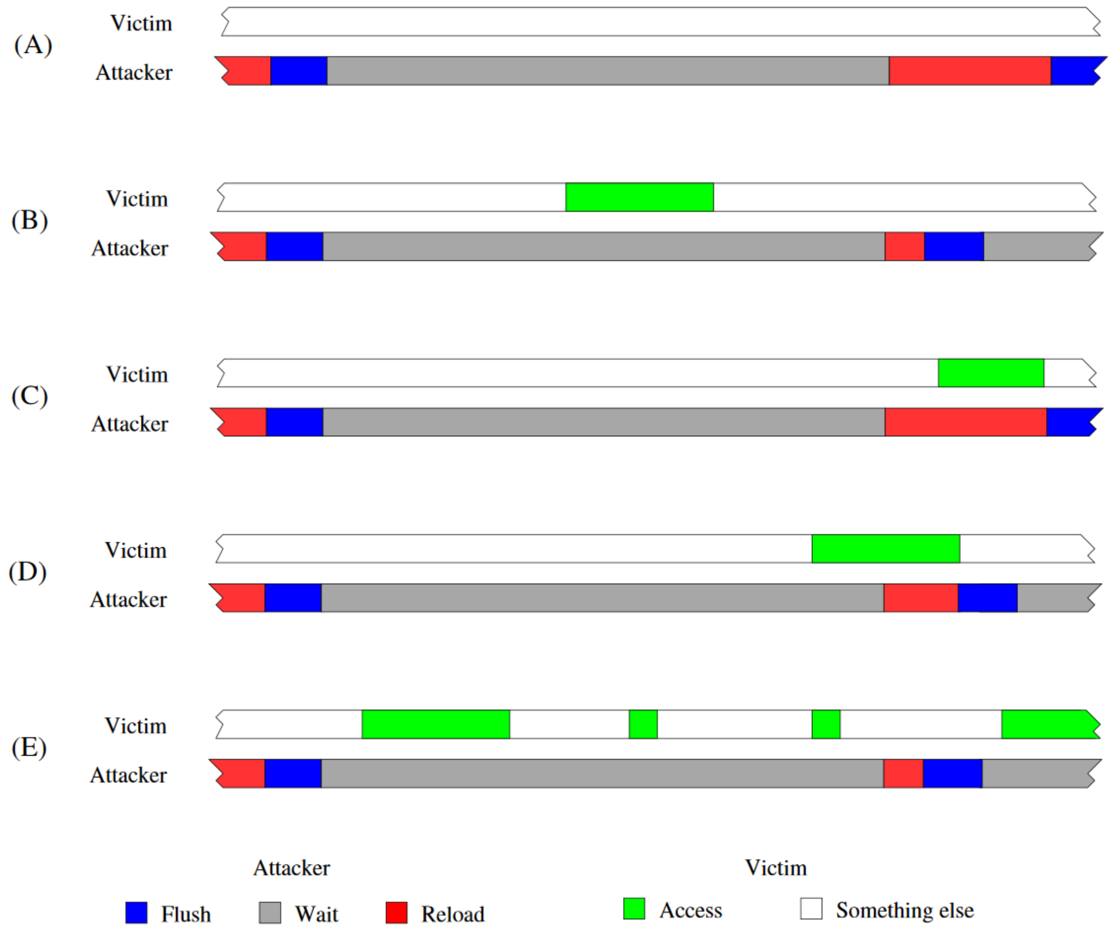


Figure 3: Timing of FLUSH+RELOAD. (A) No Victim Access (B) With Victim Access (C) Victim Access Overlap (D) Partial Overlap (E) Multiple Victim Accesses





Flush+Reload Example

- GNU Privacy Guard (GnuPG) with RSA
  - GnuPG is open-source
  - But **key** for encryption/decryption is private
- Attacker memory-maps victim's executable file to her/his memory space, to share pages
  - Flush and reload cache lines to observe
  - `clflush` for x86: cache line flush

Defence aginst Flush+Reload

- Permission check for `clflush`
- Disallowance of memory sharing
  - Contradictory to the increase of sharing in OS and virtual machines
- Tuning software programs

Meltdown and Spectre

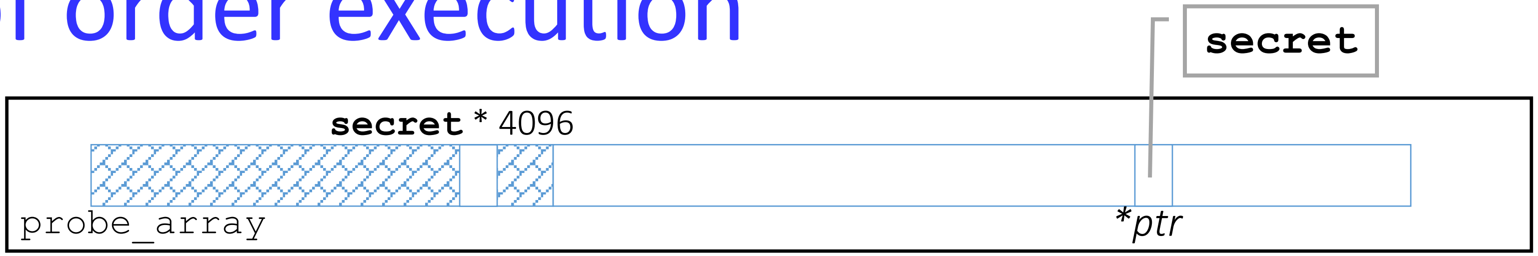
Meltdown and Spectre

MELTDOWN

SPECTRE

- Hardware vulnerability
  - Affecting Intel x86 microprocessors, IBM POWER processors, and some ARM-based microprocessors
- All Operating Systems affected!
- They are considered “catastrophic”!
- Allow to read all memory (e.g. from other process or other Virtual Machines (e.g. other users data on Amazon cloud service!))
- How Meltdown and Spectre work covers all knowledge of CA course:
  - Virtual Memory; Protection Levels; Instruction Pipelining; Out-of-order Execution; Speculative Execution; CPU Caching.

Meltdown: Out of order execution



- Out of order execution
  - *Some instructions executed in advance*

```
// secret is one-byte. probe_array is an array of char.
1. raise_exception();
2. // the line below is never reached
3. access(probe_array[secret * 4096])
```

probe_array should never be accessed, but accessed at some location probe_array + **secret** * 4096.

probe_array is fully controlled by attacker who can use Flush+Reload to see which cache line of probe_array is hit, so as to figure out the value of **secret.**
secret can be the value at any memory location, i.e., **ptr*

The aim of Meltdown: to leak/dump memory





The Impact of Meltdown

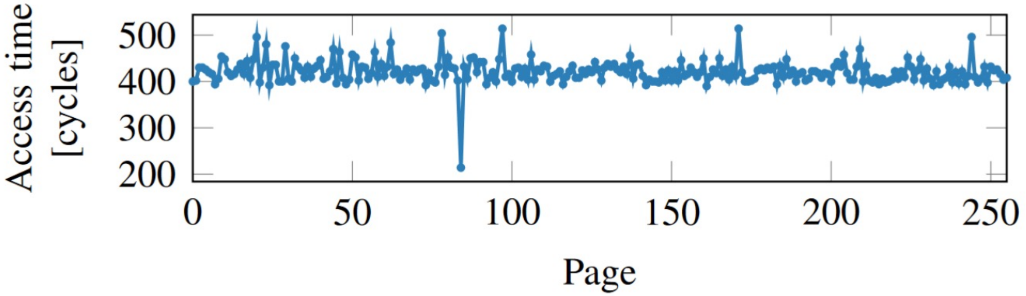


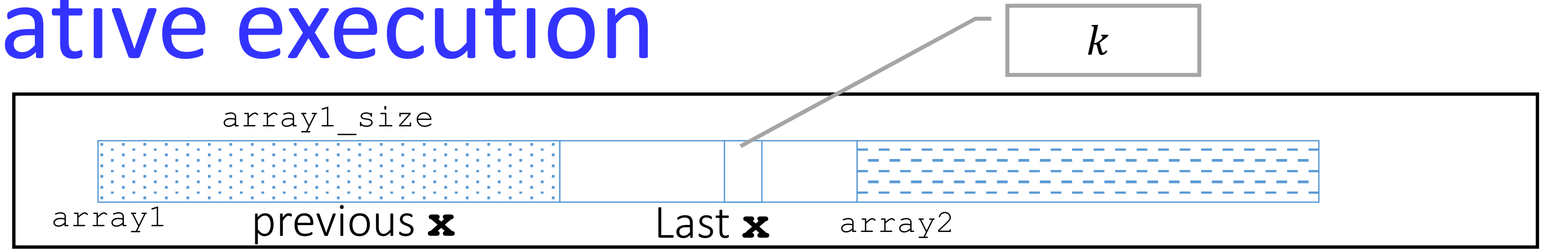
Figure 4: Even if a memory location is only accessed during out-of-order execution, it remains cached. Iterating over the 256 pages of `probe_array` shows one cache hit, exactly on the page that was accessed during the out-of-order execution.

Justification:
The researchers put a value of 84 in **`secret`** and managed to use Flush+Reload to get a cache hit at the 84th page.

The researchers developed competent programs to read memory locations that should be inaccessible to their program. They managed to dump the entire physical memory, for kernel and users.

Meltdown https://meltdownattack.com/meltdown.pdf

Spectre: Speculative execution



- Speculative execution
  - Example: branch prediction
  - Covered in L13

Prerequisites:
i. `array1[x]`, with an out-of-bound **x** larger than `array1_size`, resolves to a secret byte k that is cached;
ii. `array1_size` and `array2` uncached.
iii. Previous **x** values have been valid.

```
// x is controlled by attacker.
1. if (x < array1_size)
2.   y = array2[array1[x] * 4096]
```

← cache miss, so run next line due to prediction history

← `array1[x]` cache hit, as k is cached, so load `array2[`k` * 4096]`

Regarding a misprediction with an illegal **x**, `array2[`k` * 4096]` will not be used, but has been loaded into CPU cache.
We can use Flush+Reload to guess k with `array2`.

The aim of Spectre:
to read out a victim's sensitive information

The Impact of Spectre

- Processors can be tricked in speculative execution to modify cache state
  - Leaving attackers an exploitable opportunity
- Sensitive information of a victim program may be leaked
- Speculative Store Bypass
  - A newer variant of Spectre (v4) could allow an attacker to retrieve *older but stale values* in a CPU’s stack or other memory locations.
  - https://software.intel.com/security-software-guidance/software-guidance/speculative-store-bypass

Spectre https://spectreattack.com/spectre.pdf





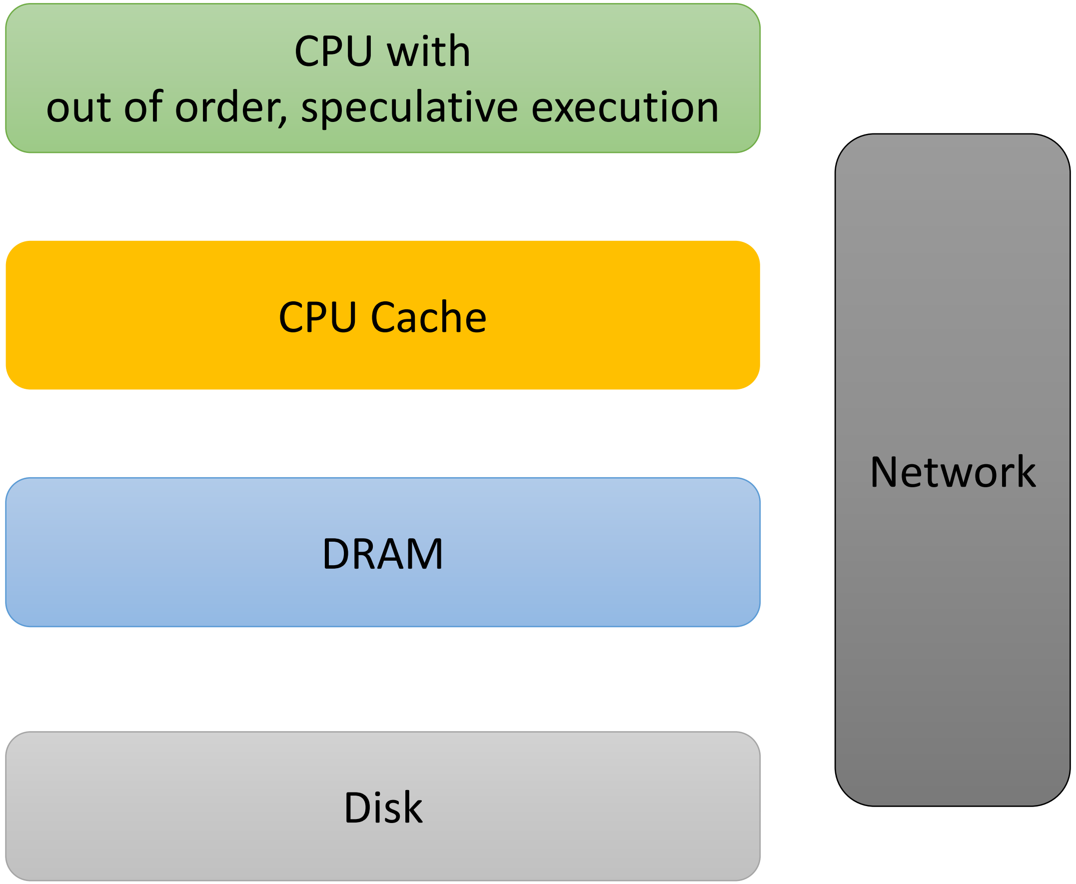
Meltdown and Spectre

- More complicated than examples here
- Multiple variants today
- Many processors, OSes, applications affected
  - PC, mobile devices, cloud
- Many proposals to mitigate their impacts

No announced RISC-V silicon is susceptible, and the popular open-source RISC-V Rocket processor is unaffected as it does not perform memory accesses speculatively. https://riscv.org/2018/01/more-secure-world-risc-v-isa/

However, there is a workshop paper "*Replicating and Mitigating Spectre Attacks on a Open-Source RISC-V Microarchitecture*"
https://carrv.github.io/2019/papers/carrv2019_paper_5.pdf

Vulnerable Architecture



Conclusion

- Every part of a computer can be vulnerable
  - Be vigilant and attentive in designing and programming
- Security and privacy have different presences
  - More than DoS, DDoS, virus, Trojan, ransomware, spyware, and phishing emails
- The challenges for a computer architect
  - To rule out any possibility of vulnerabilities
  - To achieve both high performance and security