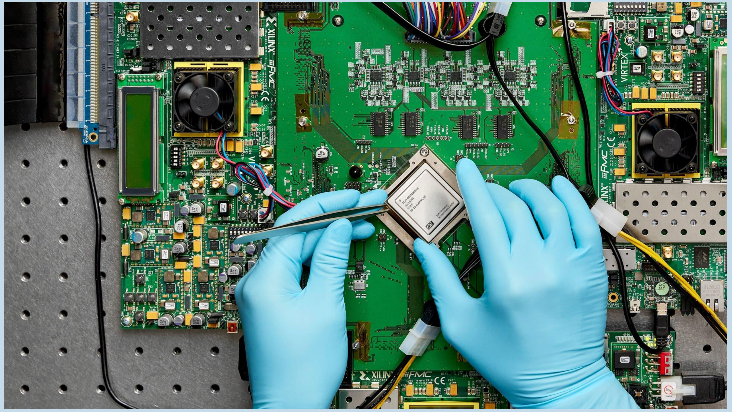
THE UNIVERSITY *of* EDINBURGH

Hardware Security 2: Secure Hardware Architecture

Security Engineering (Spring 2026)

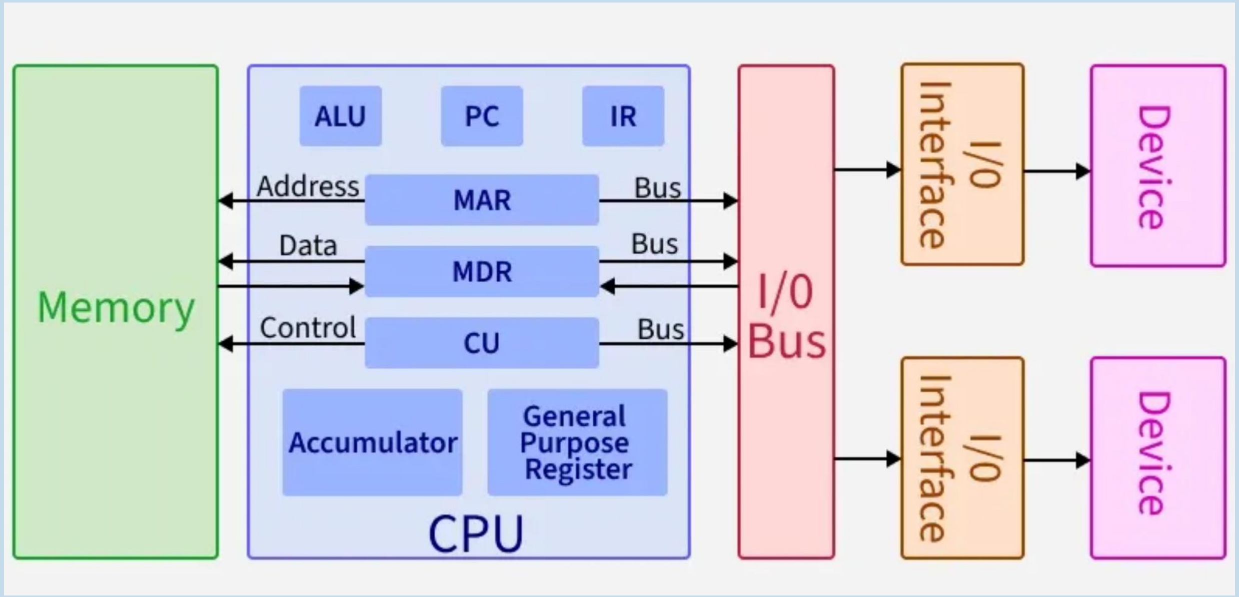
Lecturer: Jingjie Li & Daniel Woods

Influencing the world since 1583



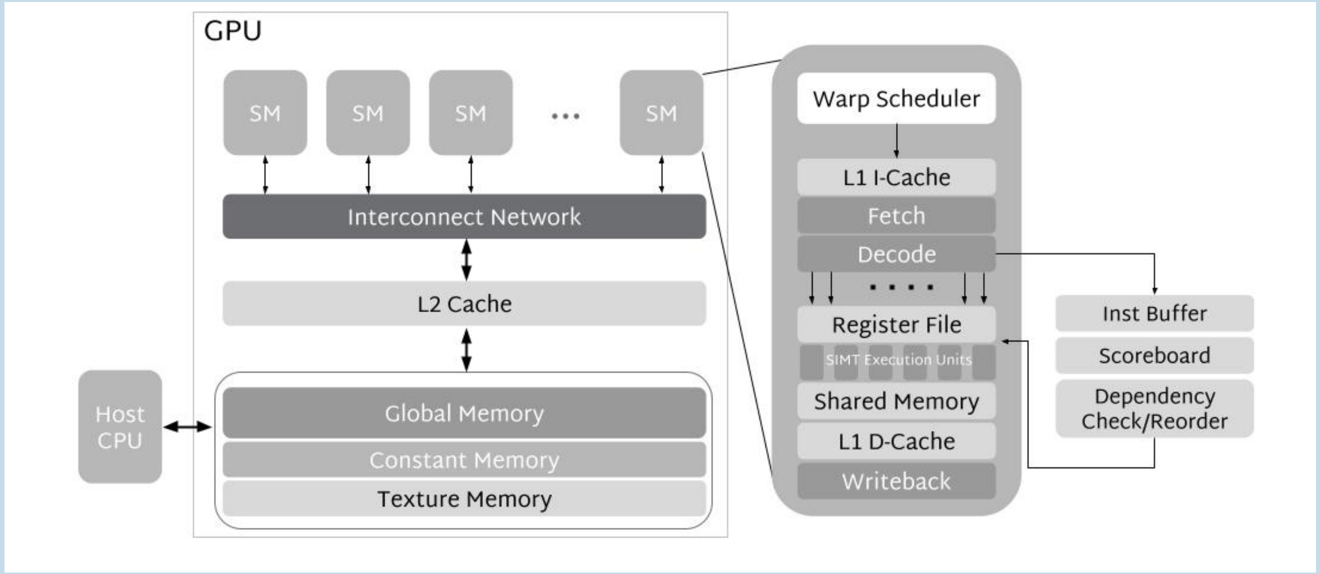
- Should we trust our digital outputs?
- How security vulnerabilities can be triggered within our hardware?
- How can we design architecture-level defense?

CPU architecture



https://www.geeksforgeeks.org/computer-organization-architecture/computer-organization-von-neumann-architecture/

GPU architecture



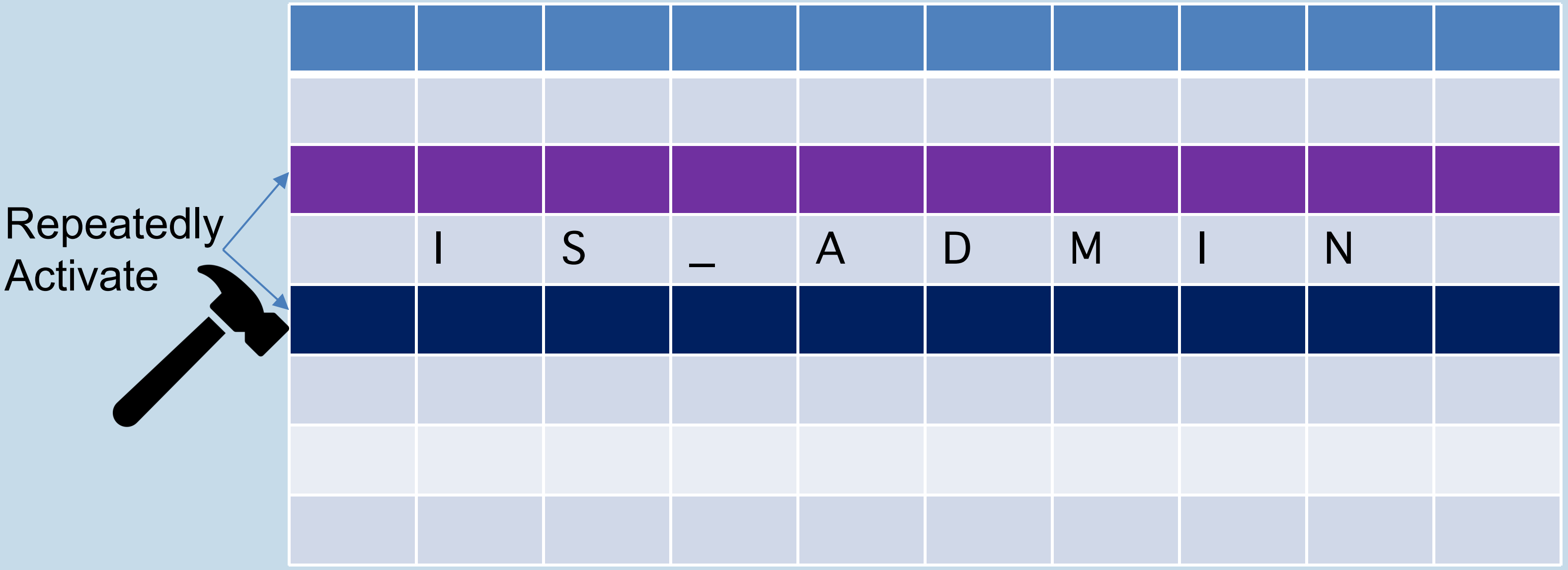
Hardware architecture side channel – Rowhammer

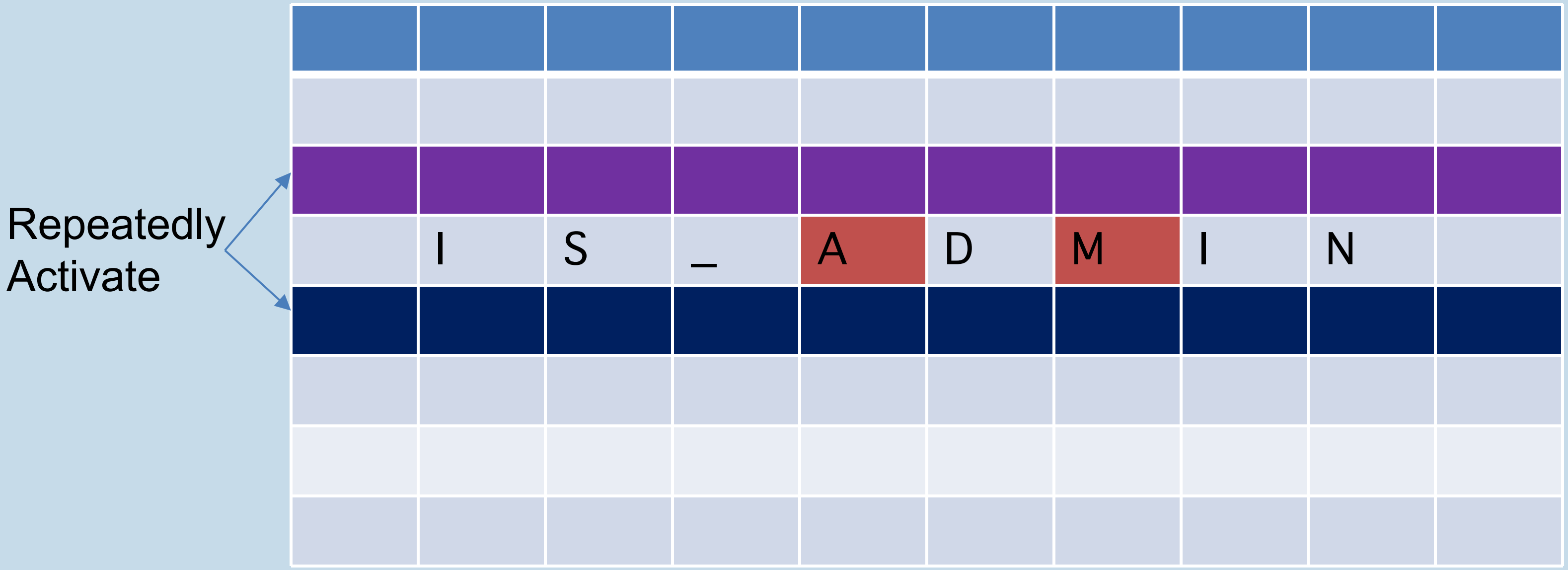
- DRAM (dynamic random access memory)
- Main memory in computer
- Stores data temporarily when power on (Volatile)

Hardware architecture side channel – Rowhammer

	I	S	_	A	D	M	I	N	

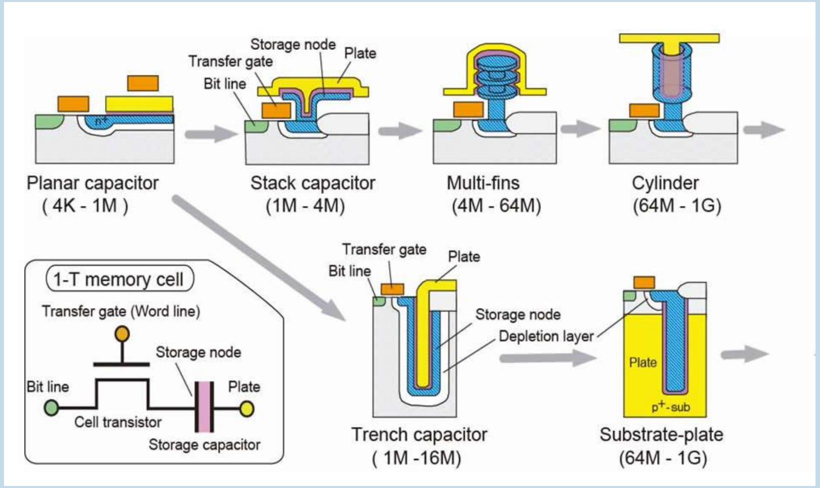
Data (e.g., Password) stored in banks and rows; each row fetched at a time


Repeatedly Activate
I
S
_
A
D
M
I
N



- Rows are too closed – interfere with neighbour rows (victim) to leak charge

Hardware architecture side channel – Rowhammer



- Bits stored in capacitors, e.g., 1 = charged, 0 = discharged
- It leaks overtime! (e.g., 0 -> 1)
- Charge needs to be refreshed to keep data

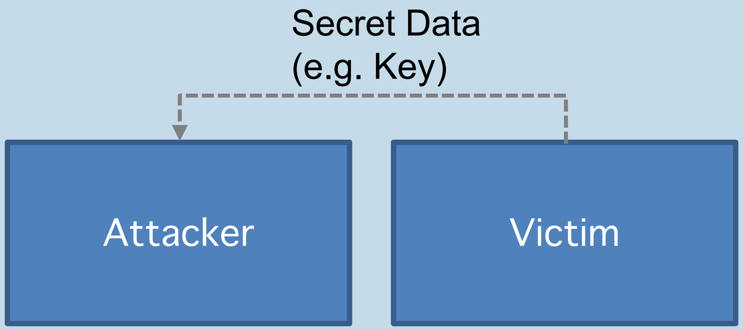
Hardware architecture side channel – Rowhammer

- Corrupt files
- Change privilege and secret data
- … (what are the other risks?)

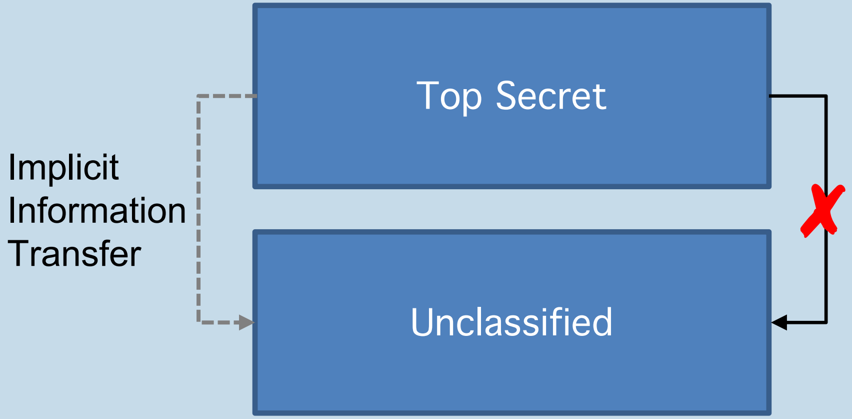
Channels

- Cache
- Disk timing
- Instruction timing
- CPU utilisation
- Clock frequency
- Power consumption
- Even erroneous behaviour e.g. differential analysis
- RF emissions (e.g. from monitors)

Channels

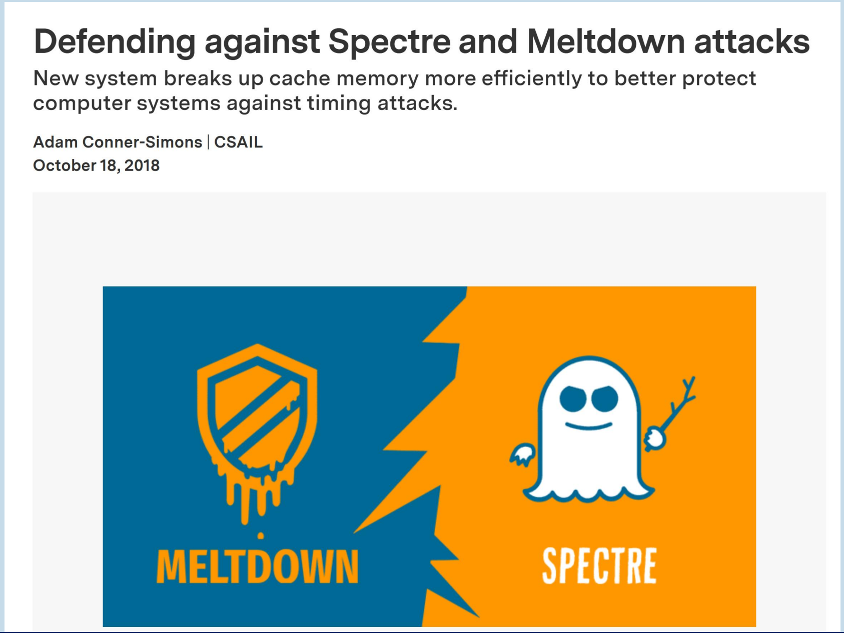


Side channel



Covert channel (when the leak of information is deliberate)

Speculative Side Channel Attacks in Computer Architecture



Defending against Spectre and Meltdown attacks

New system breaks up cache memory more efficiently to better protect computer systems against timing attacks.

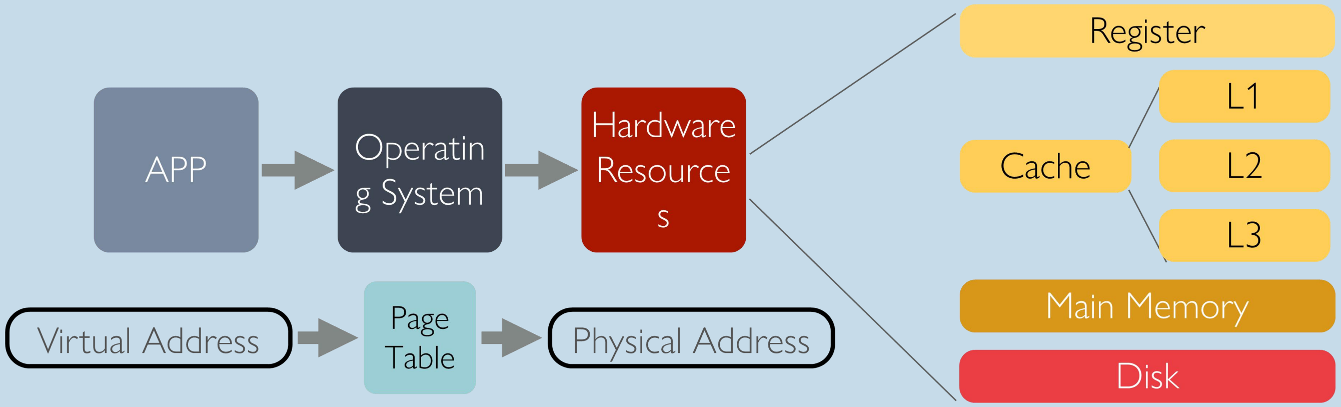
Adam Conner-Simons | CSAIL
October 18, 2018

- Expensive hardware resources
- Performance matters!
- An increasing attention on security

Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Mangard, S., Kocher, P., Genkin, D., Yarom, Y. and Hamburg, M., 2018. Meltdown. *arXiv preprint arXiv:1801.01207*.

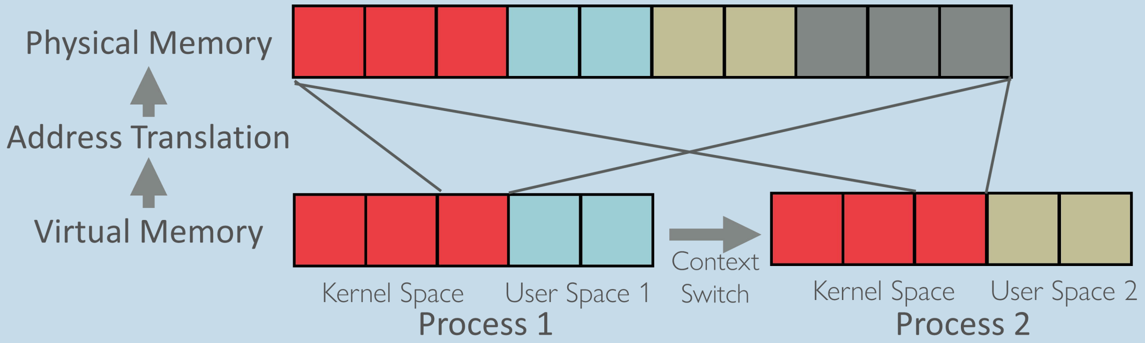
Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T. and Schwarz, M., 2020. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, *63*(7), pp.93-101.

Meltdown basics – memory hierarchy



- OS manages hardware for applications/ processes
- Physical memory shared using virtual memory
- Memory hierarchy from small/fast/expensive to large/slow/cheap

Meltdown basics – address space

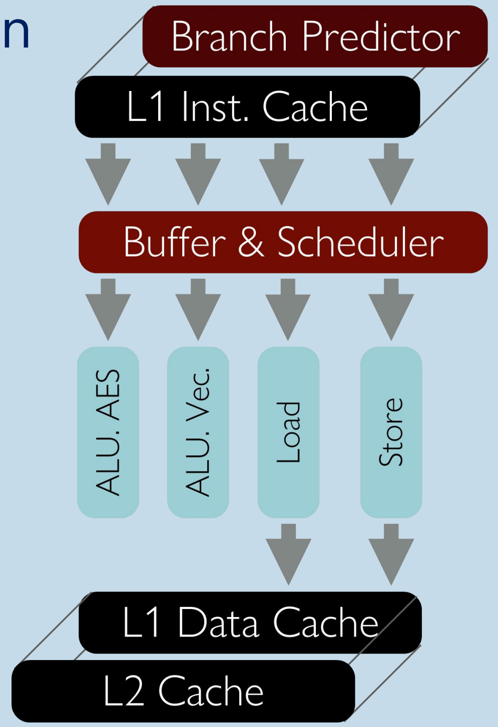


- Share physical memory resources between processes
- Entire physical memory mapped to kernel
- Isolate user and kernel spaces by checking supervisor bit

Everything works so far?

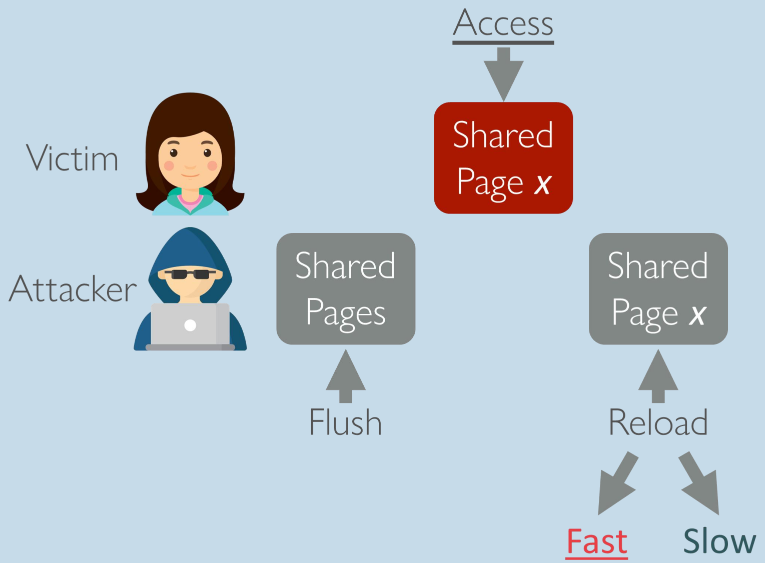
Meltdown basics – out of order execution

- Execute instructions without original order
- Retire instructions in order
- Maximize resource usage of computing units
- Speculatively run instructions after conditional branch (if (x == 1): a = 1;)
- Revert changes for wrong predictions

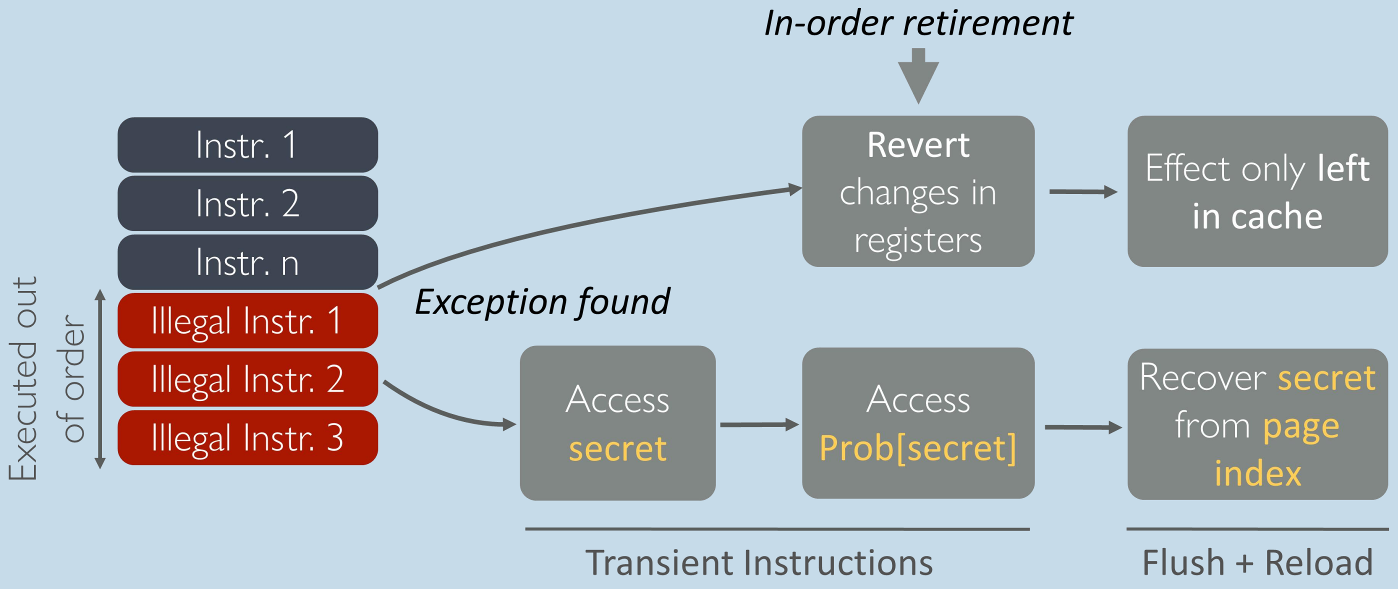


Meltdown basics – cache attack

- Cache contains microarchitectural traces
- Timing difference of cache access (Hit: fast, Miss: slow) exposes information
- Leak state of other processes from shared pages (Flush+Reload)



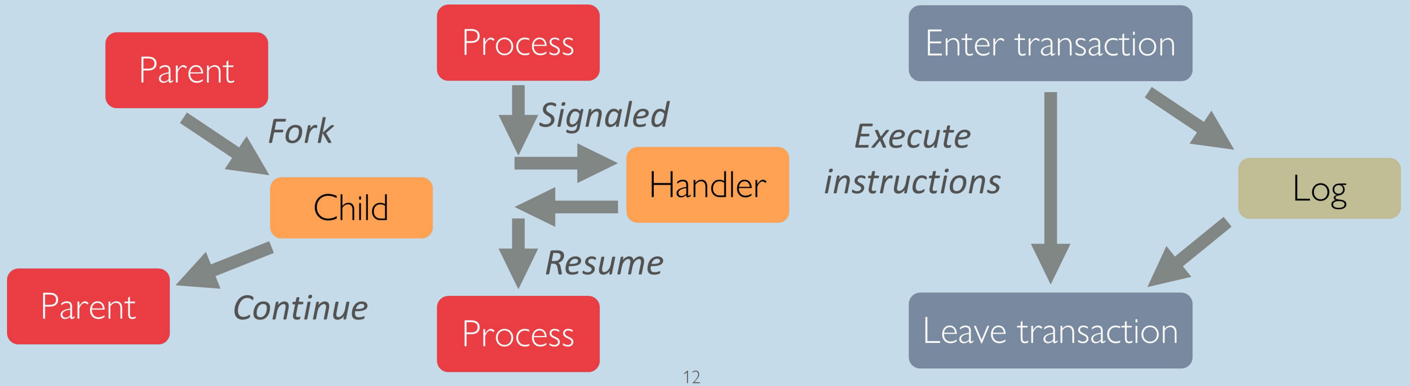
Meltdown overview – exploiting out of order execution



Meltdown – transient instruction

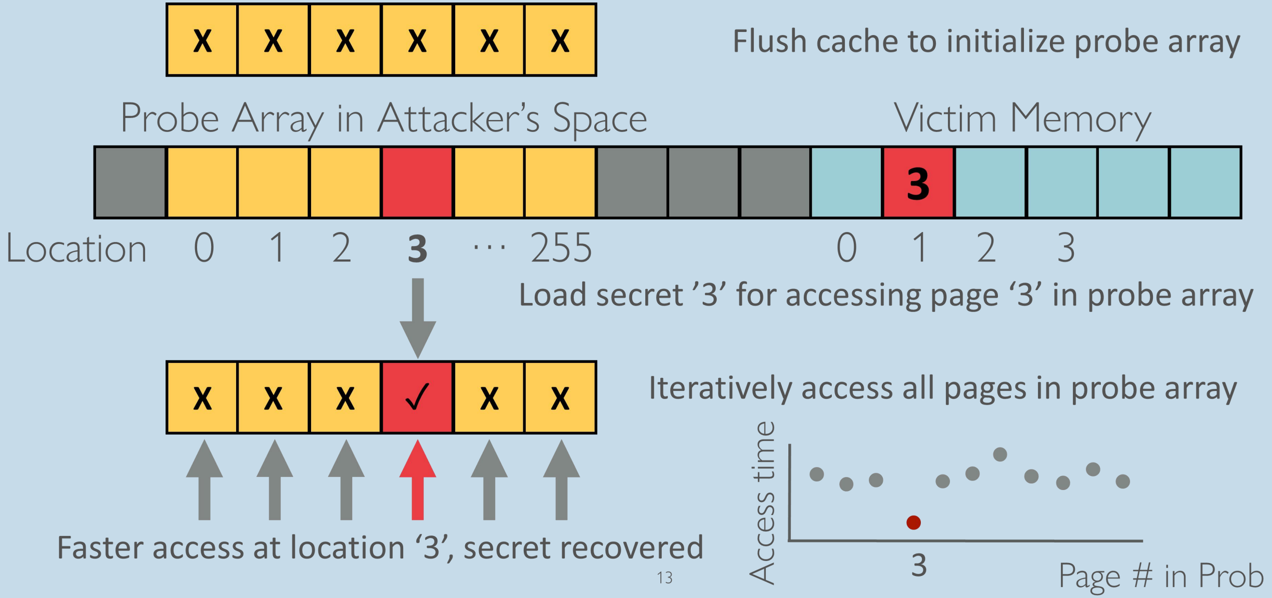
```
Flush (array);
Try {
  Int x = *secret_banned_data;
  Int y = array[x];
} Catch (Exception E) {
  printf(“the above never happened,
right?”);
}
```

Meltdown – what if exception arises?



- Handling: create child process or handle with signal handler
- Suppression: suppress using transactional memory or speculative execution rollback

Meltdown – covert channel: Flush + Reload


X X X X X X
Flush cache to initialize probe array
Probe Array in Attacker's Space
Victim Memory
3
Location 0 1 2 3 ··· 255
0 1 2 3
Load secret '3' for accessing page '3' in probe array
X X X ✓ X X
Iteratively access all pages in probe array
Faster access at location '3', secret recovered
Access time
3
Page # in Prob

Meltdown – Impact

- Configuration
  - Multiple Intel, Samsung, AMD, ARM processors
- Environment and results
  - Linux, Windows, Android, etc., including container
  - Succeed in most environments on Intel processor
  - KASLR protected Linux can be derandomized and attacked
  - Fail in ARM, AMD possibly due to different privilege check
  - Fail in KAISER (or Kernel Page-Table Isolation) protected environment

Meltdown — Countermeasure

- Hardware
  - Disable out-of-order execution
  - Serialize permission check and register fetch
  - Hard-split user and kernel spaces
- Software: KAISER (KPTI)
  - Only some privileged memory mapped in user space for switching to kernel mode
  - Kernel protected from being accessed from user space in kernel mode

Spectre — Branch prediction basics

- Category of control flow instructions, e.g., for if(a<b) {…} else {…}
  - unconditional branch (jmp on x86)
  - call/return
  - conditional branch (e.g. je on x86) taken
  - conditional branch not taken

Spectre Overview (v1)

```
Int x = index_of_secret_out_of_bounds_data;
If(x < array_size) {
  y = array[x];
  z = array2[y];
}
```

True execution: No

Branch prediction: Yes

X = 928309183902

Array_size = 100

Leaks (Partial) Contents of Y (a byte at a time)

Architecture side channel – takeaways

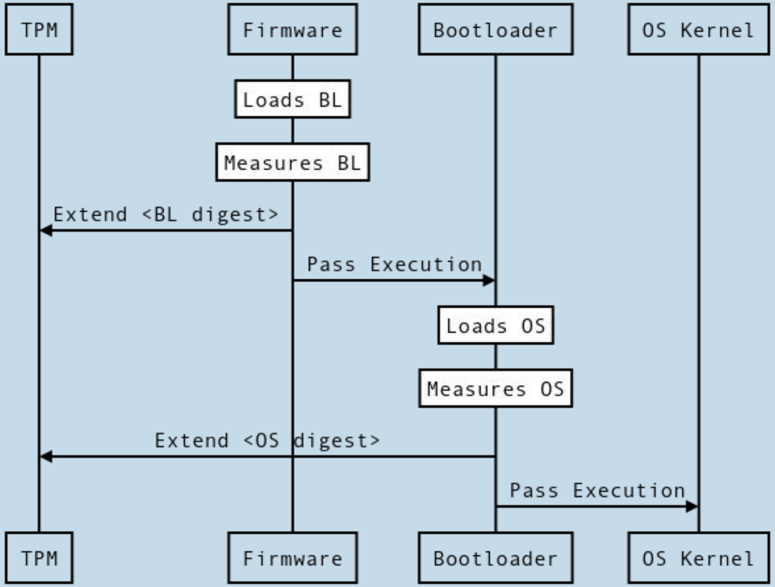
- Security isn't just limited to the “programmer's model”
- Don’t “roll your own crypto”
- Bugs can be around for a long time before they are discovered
- Make sure you’re aware of what the hardware is doing underneath your code!

What can hardware provide for you?

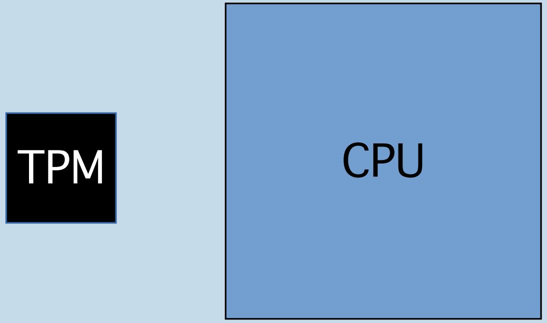
- Hardware AES
- Trusted Platform Modules
- Enclaves
- Codesign for Software Security: CHERI and MTE
- Physically Unclonable Functions

Trust Platform Module – chain of verification

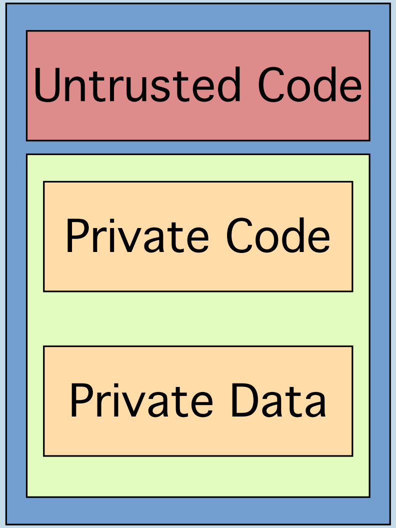
- Measured and correctness verified at every load



Enclaves (Trustzone, SGX, SEV)

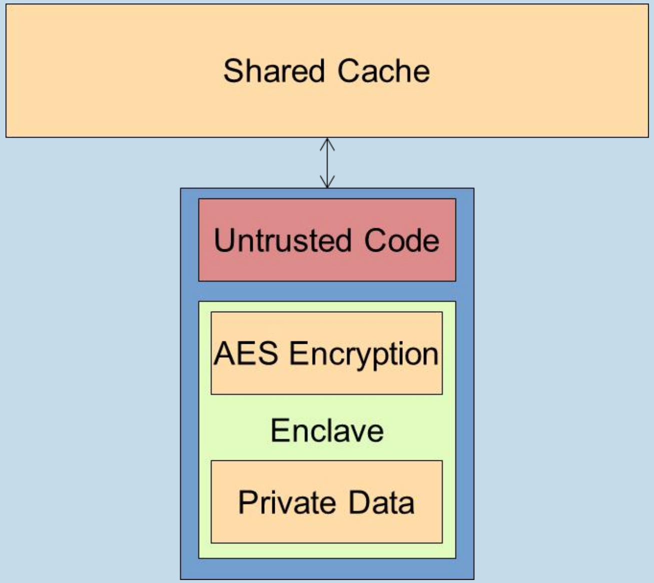


vs

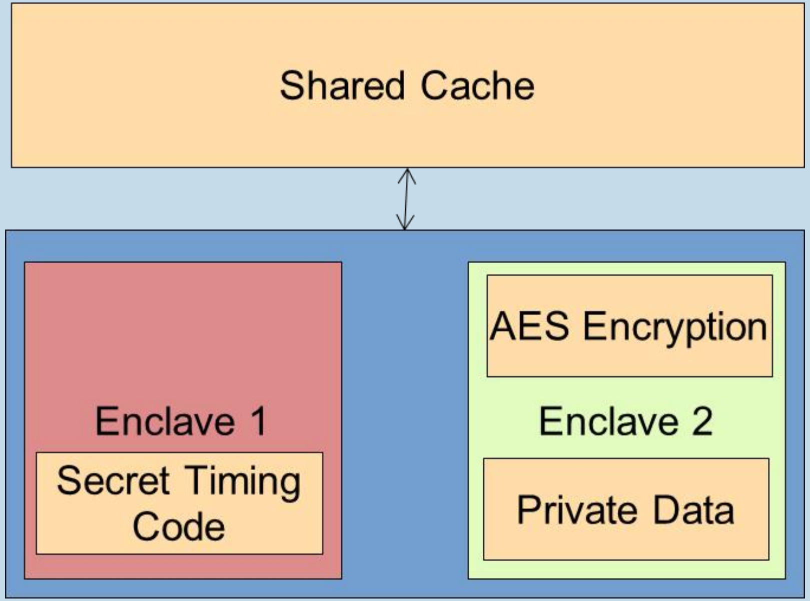


- Enclave's cryptographic power associated with processes in security load
- Data encrypted
- Trusted computation within untrusted operating system

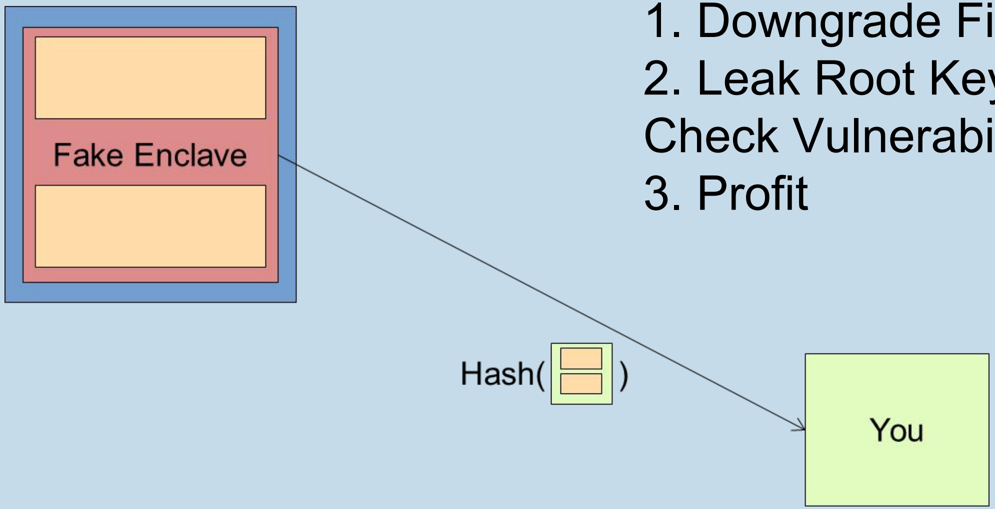
Enclave – still vulnerable to side channels



Enclave – losing visibility into malicious program :(



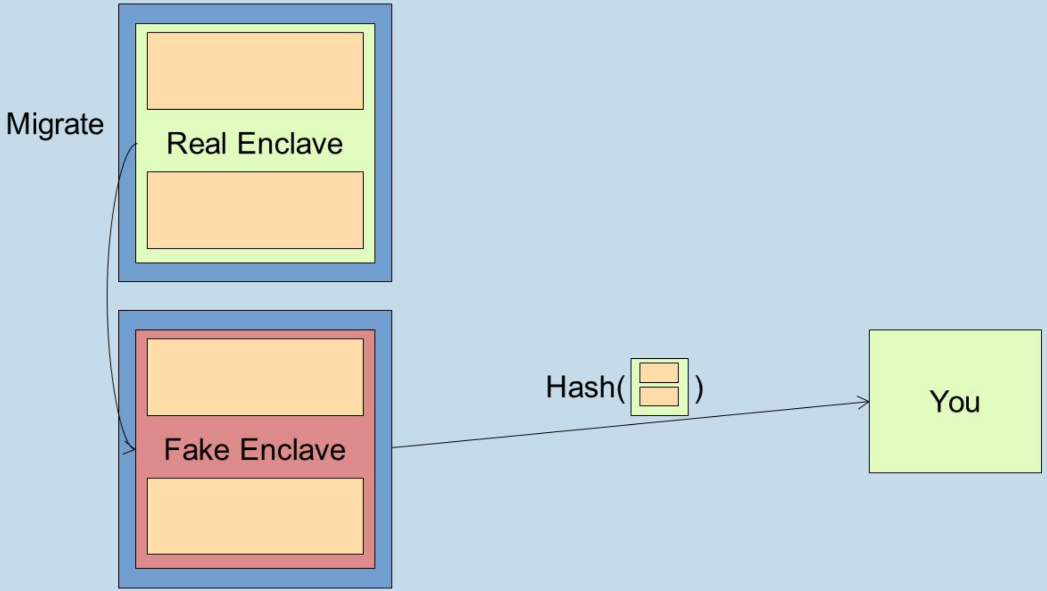
Enclave – am I really running things in an enclave?



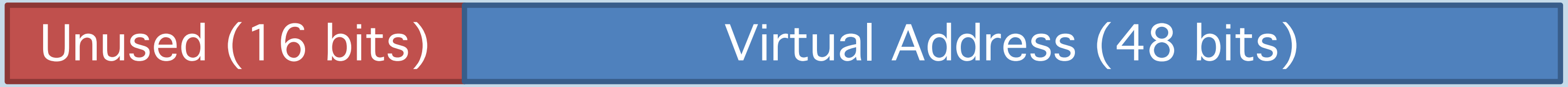
1. Downgrade Firmware
2. Leak Root Key through Signature Check Vulnerability
3. Profit

- Enclave security relying on remote attestation. But what get attested?

Enclave — am I really running things in an enclave?

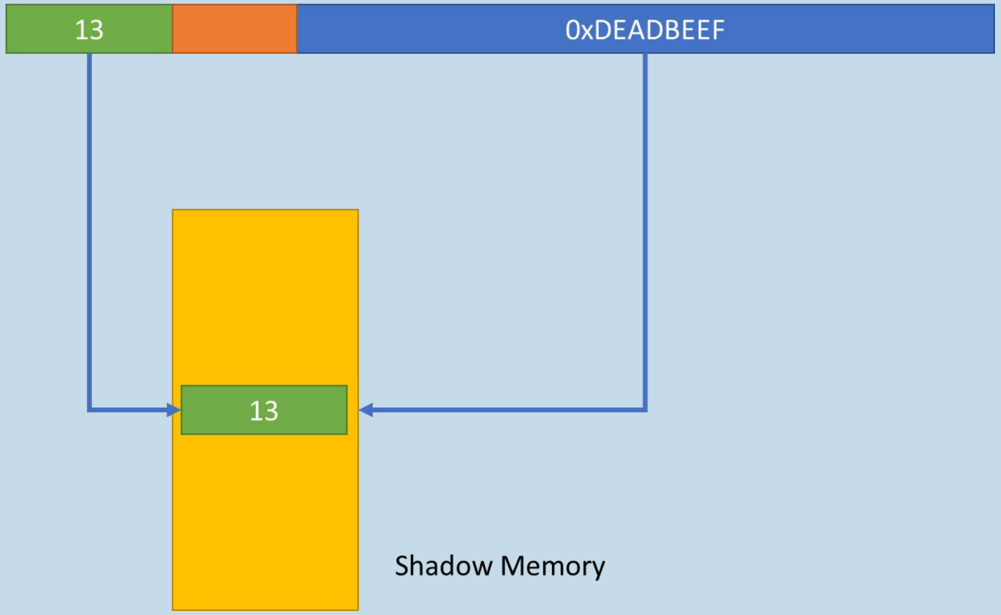


Co-design for security (MTE, memory tagging extension)



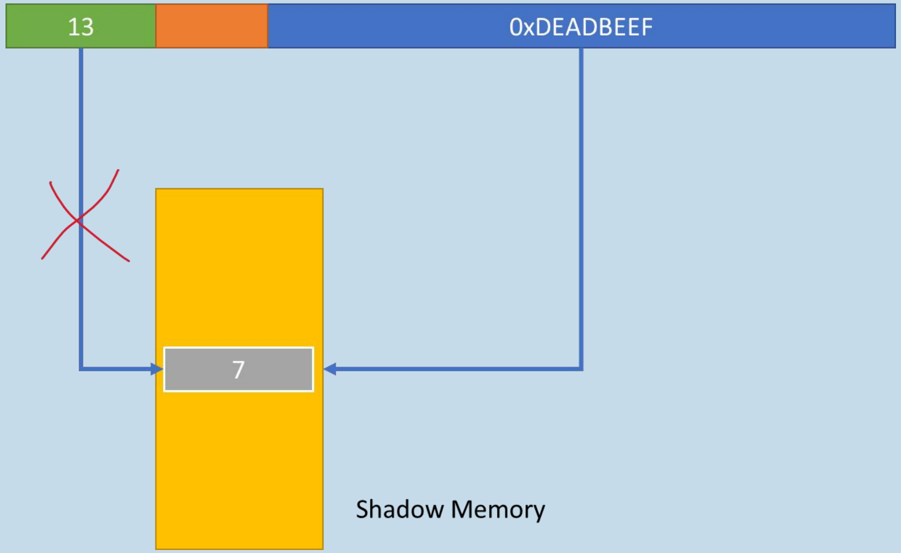
- Using unused bits in a pointer address for tagging
- Checking the tag and see if it's what we intended to be
- Tag unreadable by the user to be correctly modified

Co–design for security (MTE)



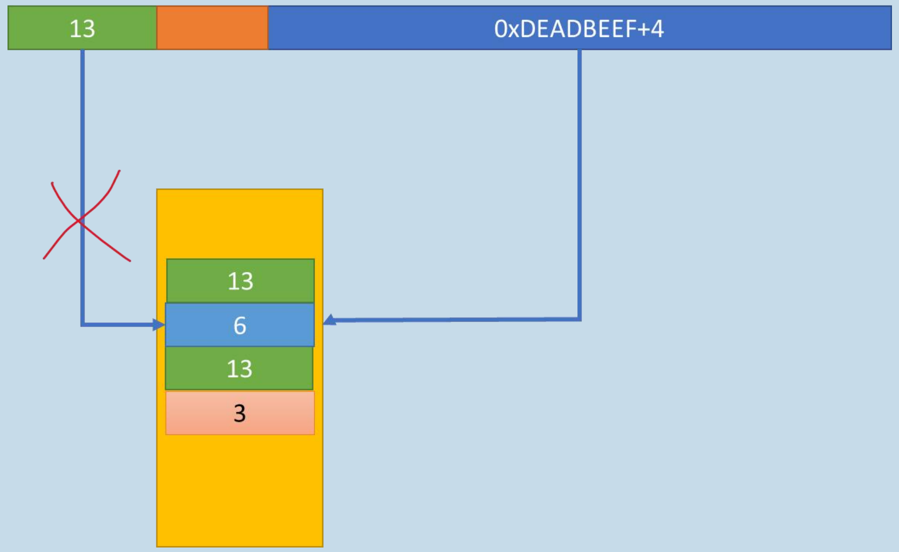
- Shadow memory stores index locations for every address in virtual memory (e.g., four bits for every 128 bits)
- Address map can be mapped to the index in the shadow

Co-design for security (MTE)



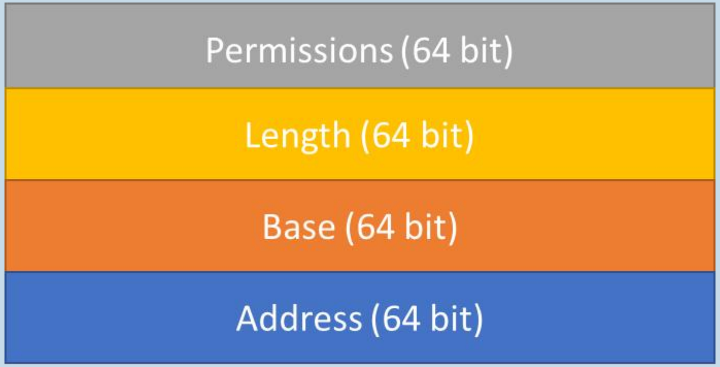
- If the pointer here gets reallocated to a new memory location?
- Detect, report, and fix

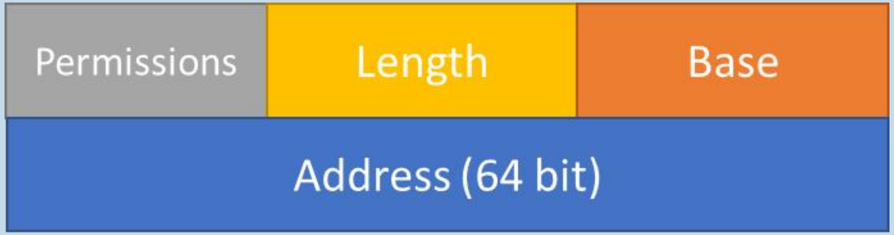
Co-design for security (MTE)



- Preventing buffer overflow (spatial safety)
- No guarantee due to not 1-1 mapping (mainly for debug)

Co–design for security (CHERI)

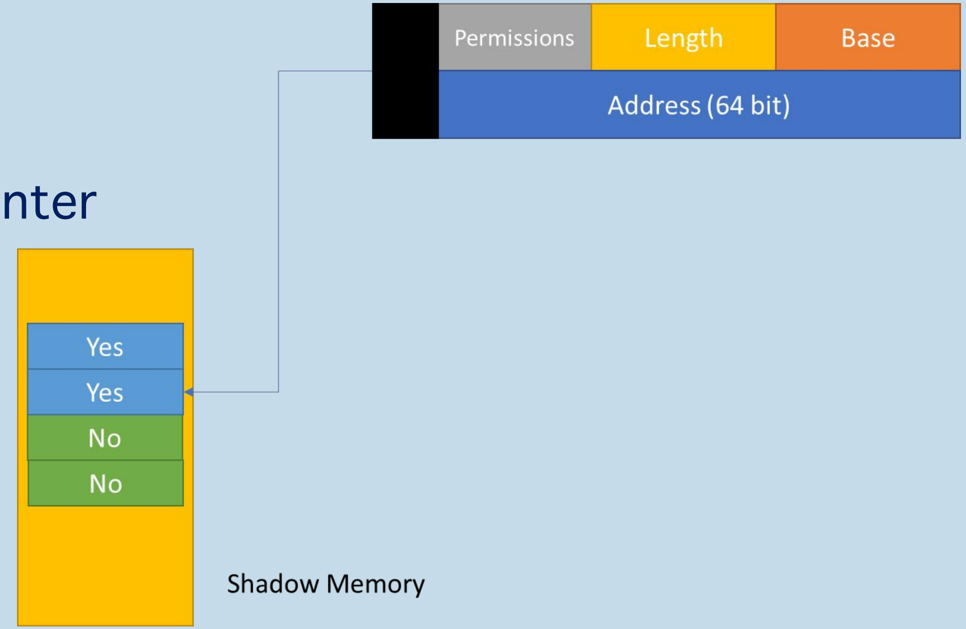




- Giving pointers capabilities!
- What you are allowed to access and what you want

Co–design for security (CHERI)

- Verify if it's a capability pointer
- Tag bits stored securely

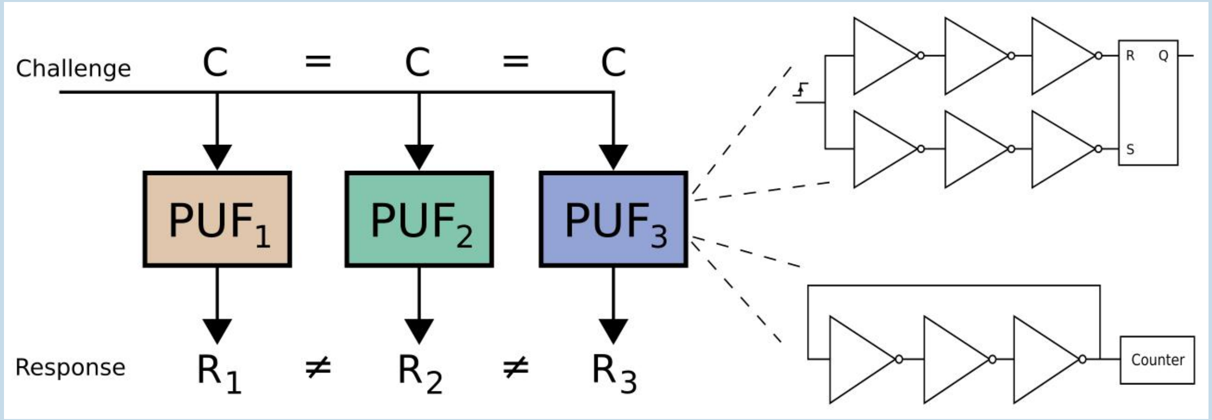


Co–design for security (CHERI vs MTE)

- + Guaranteed mitigation of spatial safety bugs (in pure-cap mode)
- + 1-bit/128 shadow space
- - 128-bit pointers
- - Standards-incompatible

- - Probabilistic mitigation of temporal/spatial bugs
- - 4-bit/128 shadow space
- + 64-bit pointers
- + Standards-compatible

Physically Unclonable Function (PUF)



- Using device unique characteristics as a fingerprint, e.g., slight device difference causing different outputs in racing
- Binding secret with hardware device, e.g., for key generation