Software Guard eXtensions (SGX)

Operating Systems

- Controls sharing of system resources
- Enforces protection

Operating Systems

- Controls sharing of system resources
- Enforces protection

- Can be compromised
- Not always trustworthy

Operating Systems

- Controls sharing of system resources
- Enforces protection

- Can be compromised
- Not always trustworthy

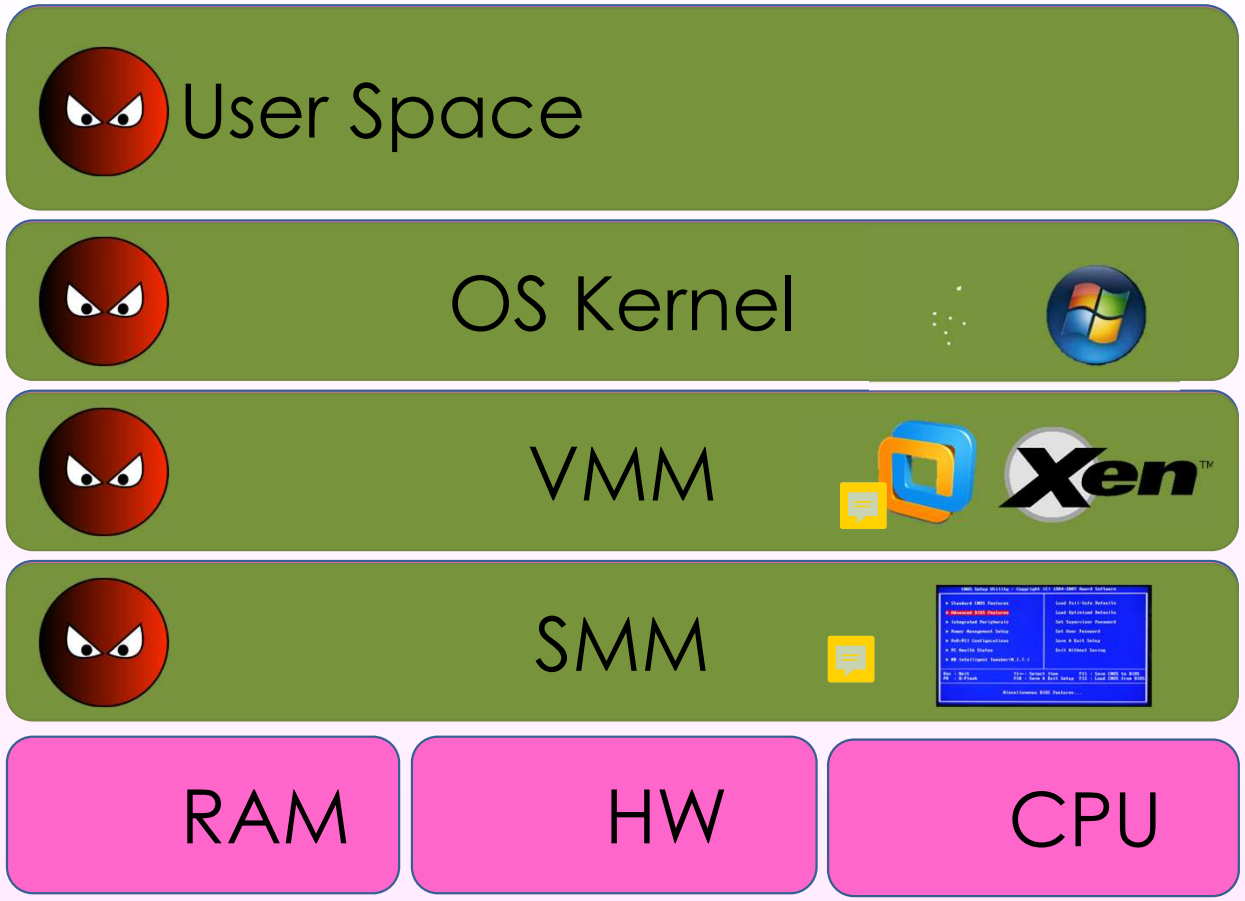
- And has superpowers


SGX

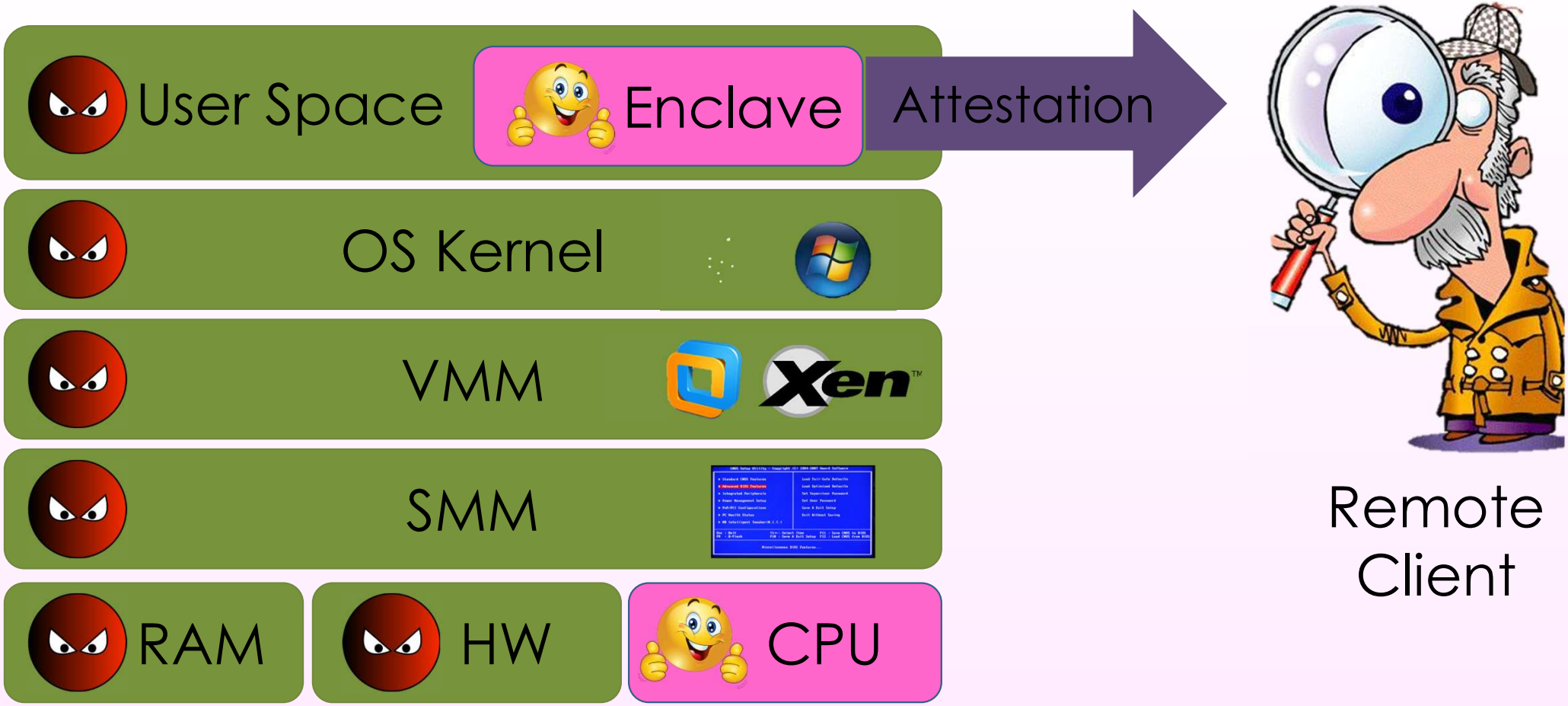
SGX (Software Guard eXtensions)

- CPU hardware feature designed to allow (remote) secure computation
- “Developers can partition their application into processor-hardened *enclaves* or protected areas of execution in memory that increase security even on compromised platforms. ”
- “Confidentiality and integrity: Enforced at the operating system, BIOS, VMM, SMM, or TEE layers even in the presence of privileged malware.”
- “Remote attest and provision: A remote party can verify an application enclave identity and securely provision keys and other sensitive data to the enclave”
- Present on all Intel CPUs since 2016 (SkyLake)

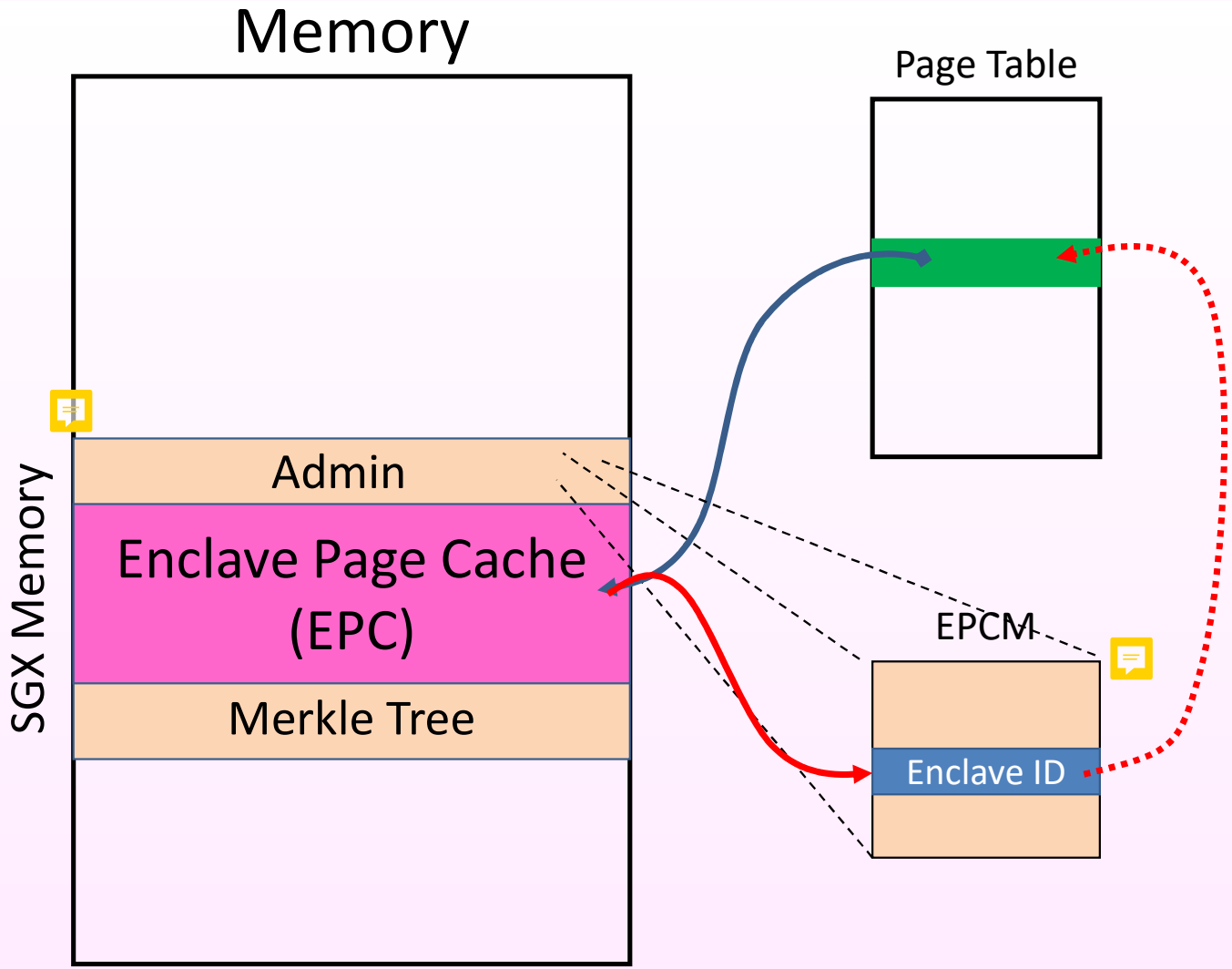
SGX Security Model



SGX Security Model



SGX Memory Management



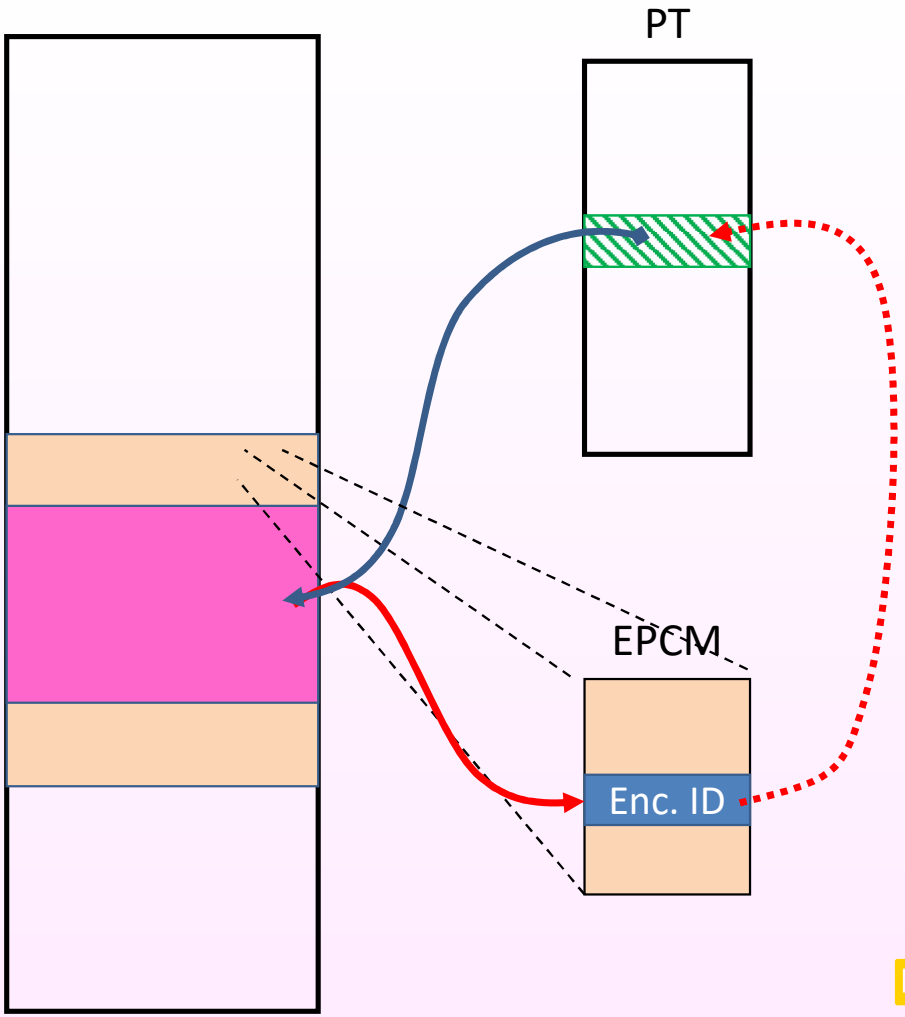
- Enclave pages are encrypted and authenticated
- OS allocates pages
- Hardware verifies mapping
- About page semantics
- Secure page swap

SGX Security Model: The Fine Print

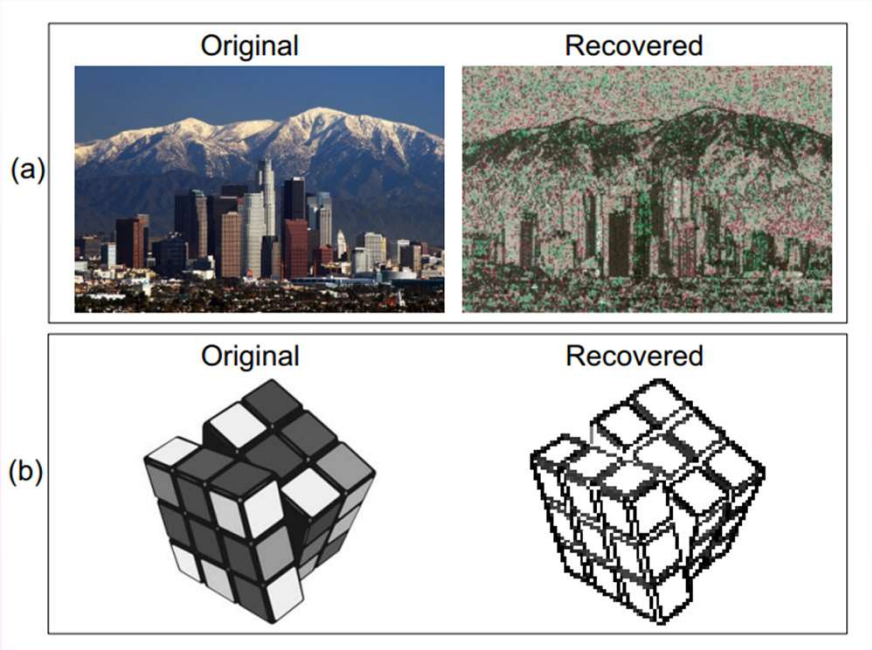
"Intel SGX does not provide explicit protection from side-channel attacks, it is the developer's responsibility to address side-channel attack concerns."

Intel SGX Developer Guide, Page 39/40

Controlled Channels Attacks (Xu et al. IEEE SP 2015)



- OS controls page table – marks all as invalid
- Gets access on enclave interrupt
- Trace page accesses



OS Control

- Interrupts are not required (Van Bulck et al. USENIX Sec. 2017)
  - Monitor accessed and dirty bits on page tables
  - Cache attacks on PTEs
- Isolate core
- Fixed CPU frequency
- Monitor performance counters
- Timer interrupts

SGX Step (Van Bulck et al. SysTEX 2017)

- Library for controlling SGX execution

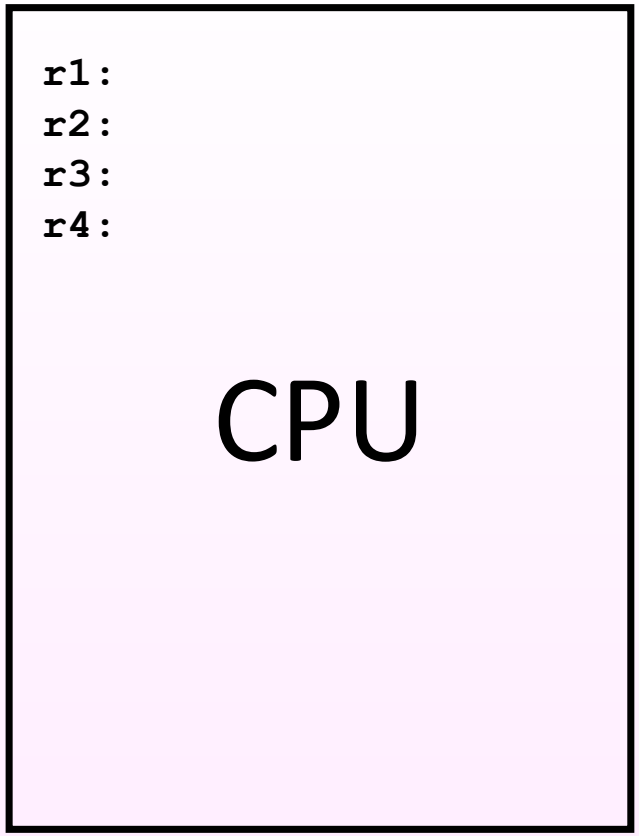
- Exposes page maps to user space
- Exposes APIC interrupts to user space
- Allows user control at enclave interrupts

- High resolution, low noise microarchitectural attacks

Meltdown-type attacks

Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```



Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

Start executing first load load

```
r1:
r2:
r3:
r4:
```

CPU

Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

Suppose height is in the cache and second load terminates first

Start executing second load

```
r1:
r2:
r3:
r4:
```

CPU

Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

Start executing the multiply instruction

```
r1:
r2:
r3:
r4:
```

CPU

Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

When the first load completes

```
r1:
r2:
r3:
r4:
```

CPU

Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

retire all completed instructions

```
r1: 4
r2: 3
r3:
r4:
```

CPU

Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

… and proceed

```
r1: 4
r2: 3
r3:
r4:
```

CPU

Why not retire out-of-order?

Why not retire early

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

Early retire stores a result in the architectural state

```
r1:
r2: 3
r3:
r4:
```

CPU

Why not retire early

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

Early retire stores a result in the architectural state

A later fault leaves the processor in an inconsistent state!

```
r1:
r2: 3
r3:
r4:
```

CPU

Traps in Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r1
div   r1, 2, r1
store r1, area
```

Suppose height is an invalid address

Start executing second load

```
r1:
r2:
r3:
r4:
```

CPU

Traps in Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

Mark as an error, but continue executing

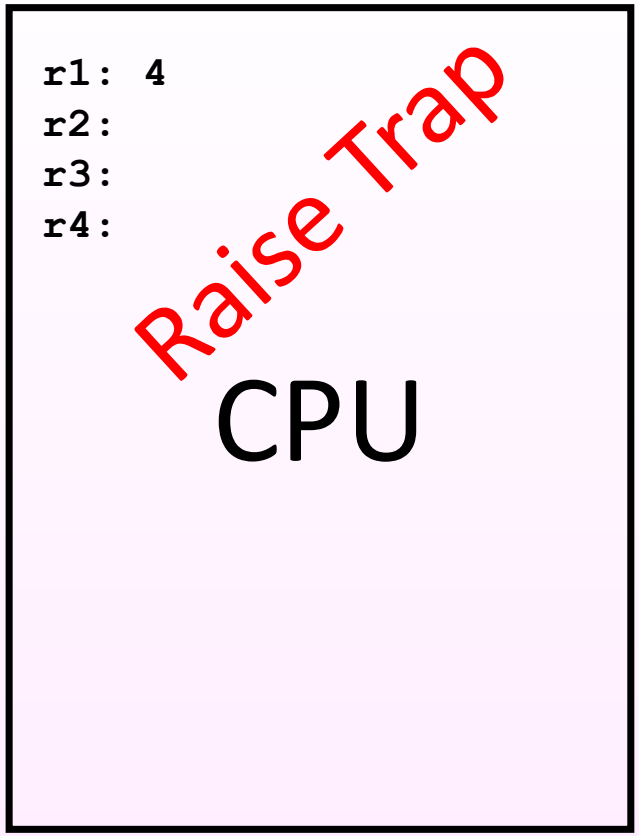
```
r1:
r2:
r3:
r4:
```

CPU

Traps in Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

Mark as an error, but continue executing until all older instructions retire



Traps in Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

Mark as an error, but continue executing until all older instructions retire

But while we're waiting…

```
r1:
r2:
r3:
r4:
```

CPU

Traps in Out of Order Execution

```
load  base, r1
load  height, r2
mul   r1, r1, r3
mul   r2, r2, r4
add   r3, r4, r3
sqrt  r3, r3
store r3, hypotenuse
mul   r1, r2, r3
div   r3, 2, r3
store r3, area
```

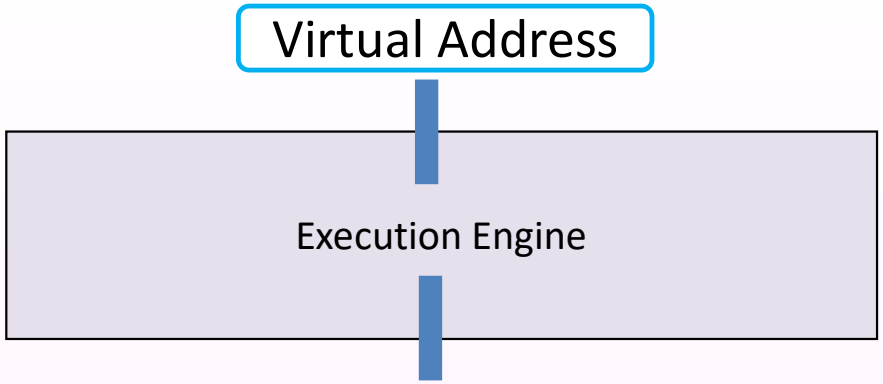
Mark as an error, but continue executing until all older instructions retire

But while we're waiting…
Dependent instructions
Start executing

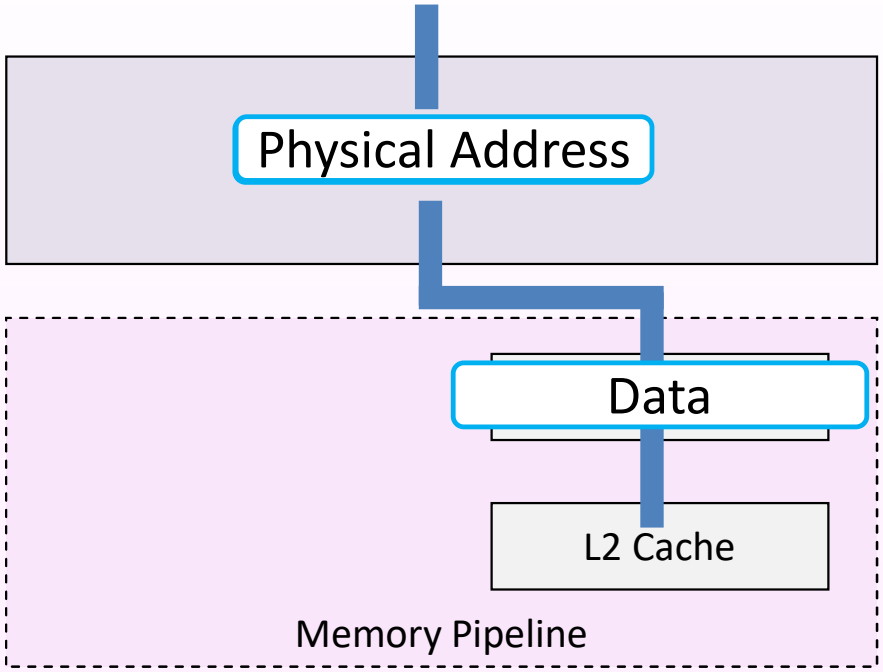
```
r1:
r2:
r3:
r4:
```

CPU

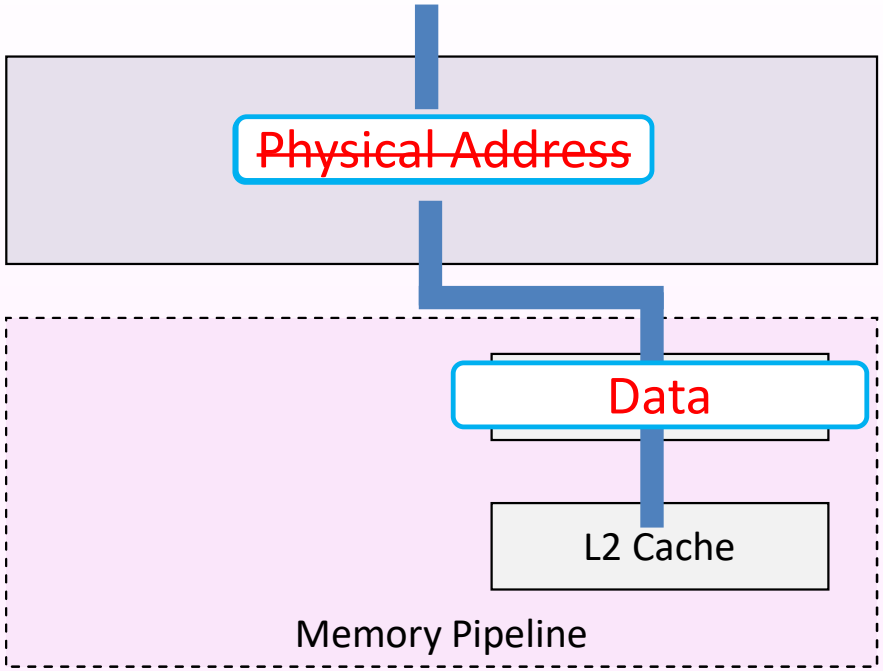
Load instructions



Load instructions

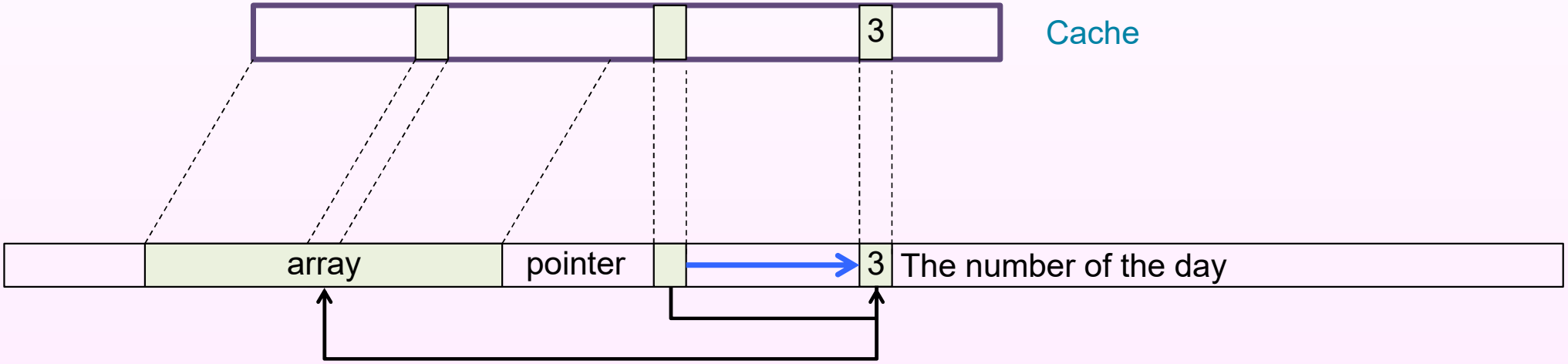


Load instructions

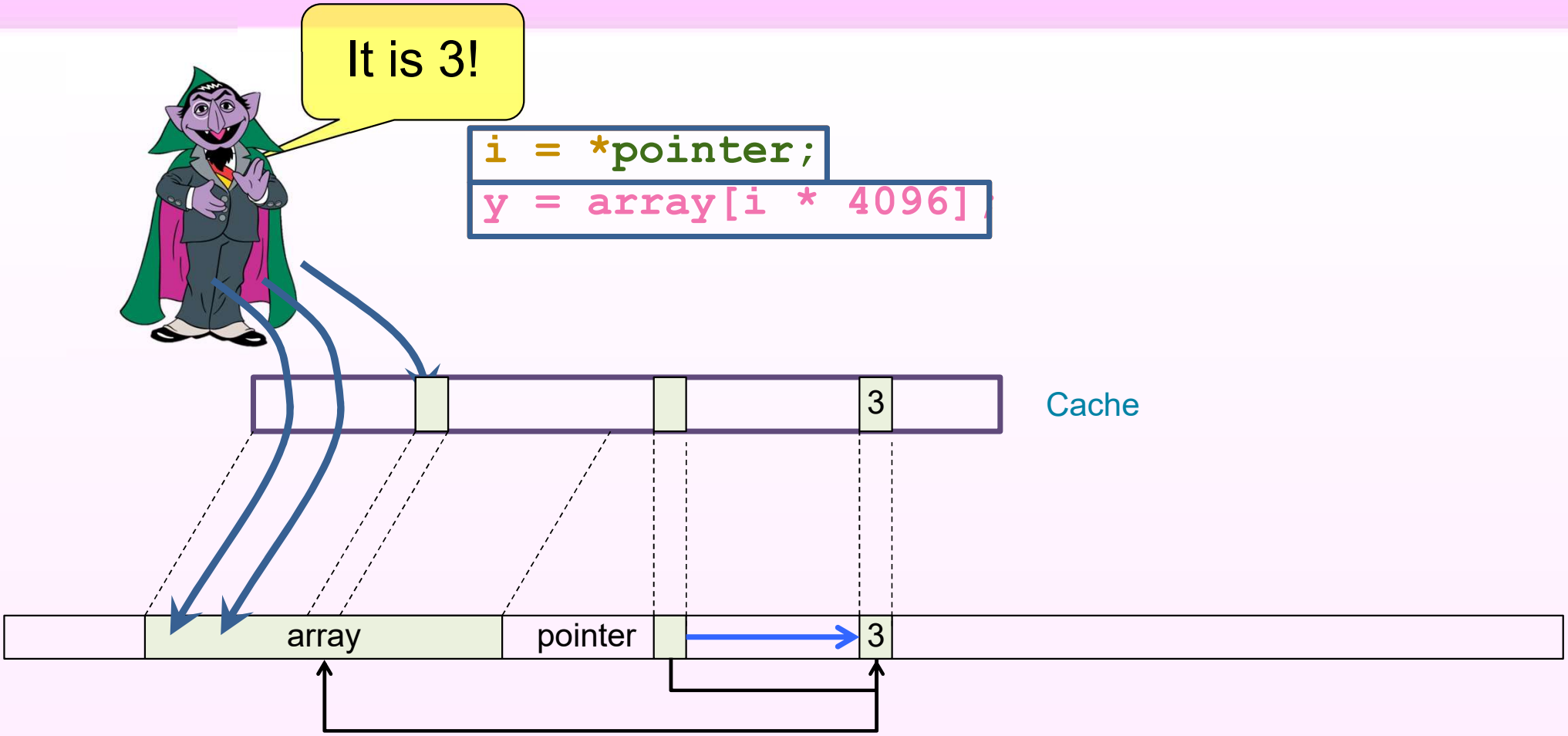


Using a Covert Channel

```
i = *pointer;
y = array[i * 4096];
```



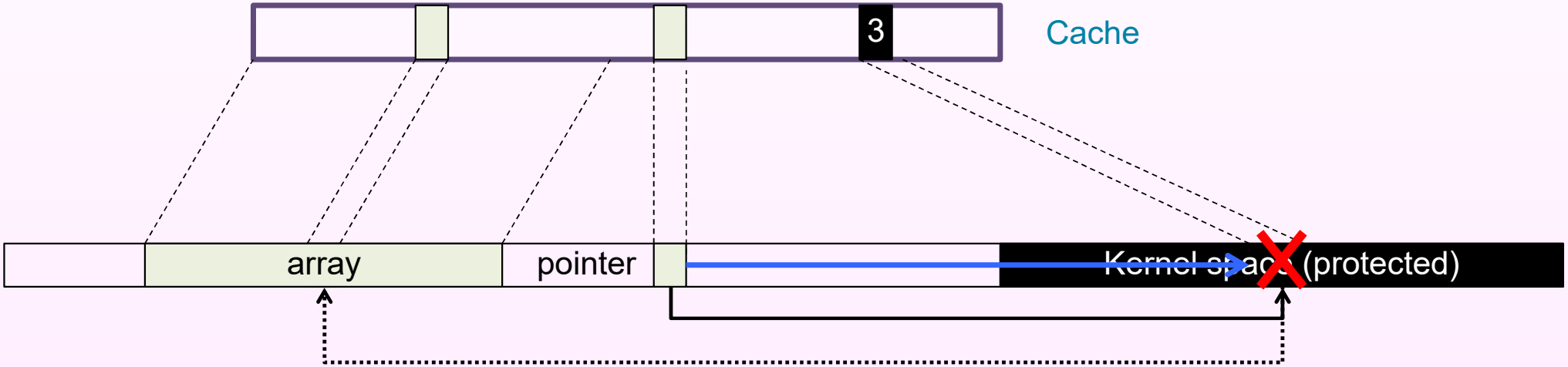
Using a Covert Channel



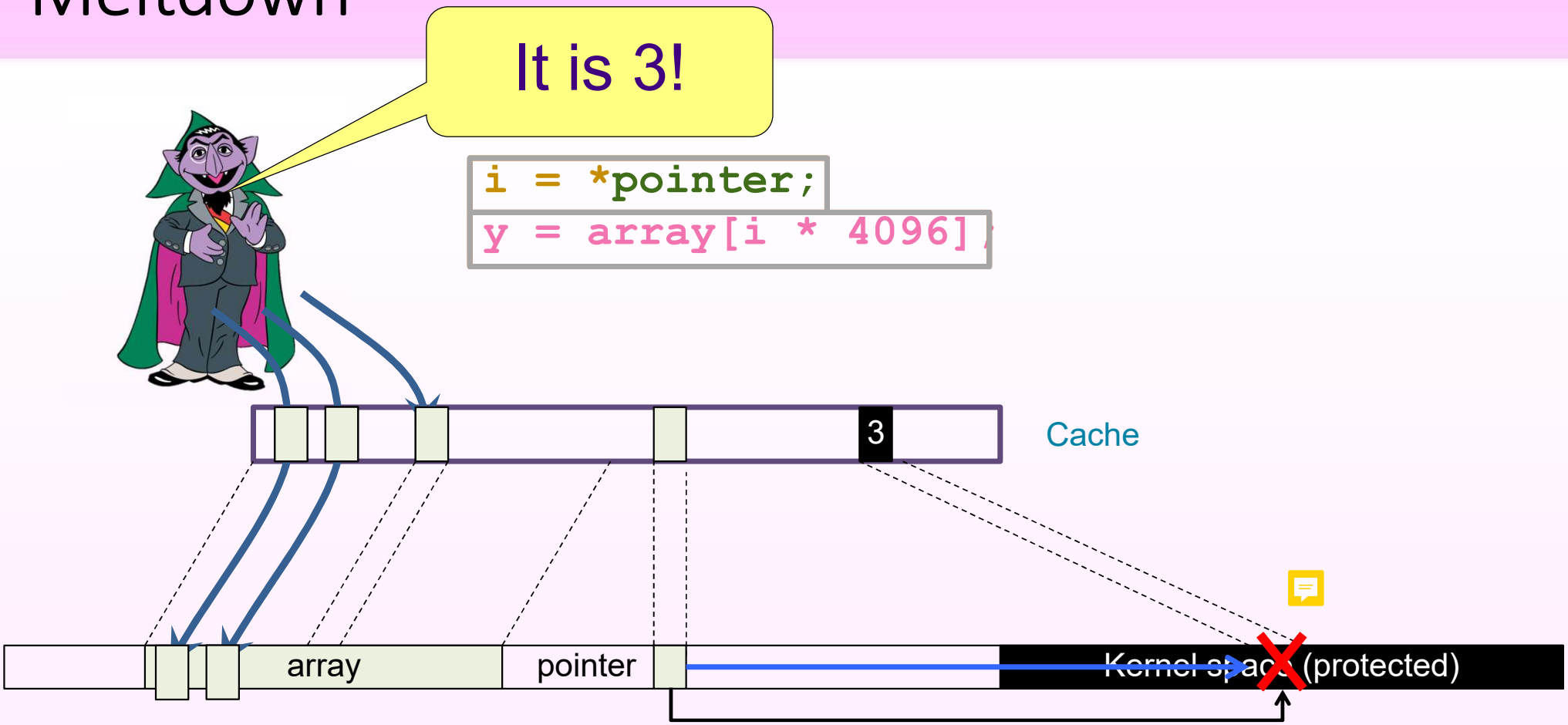
Meltdown

Meltdown

```
i = *pointer;
y = array[i * 4096];
```



Meltdown


It is 3!
i = *pointer;
y = array[i * 4096];
3
Cache
array
pointer
Kernel space (protected)

Handling Faults

- A fault kills the process. How do we recover the channel?

- Spawn a new process
- Use a signal handler

Fault suppression

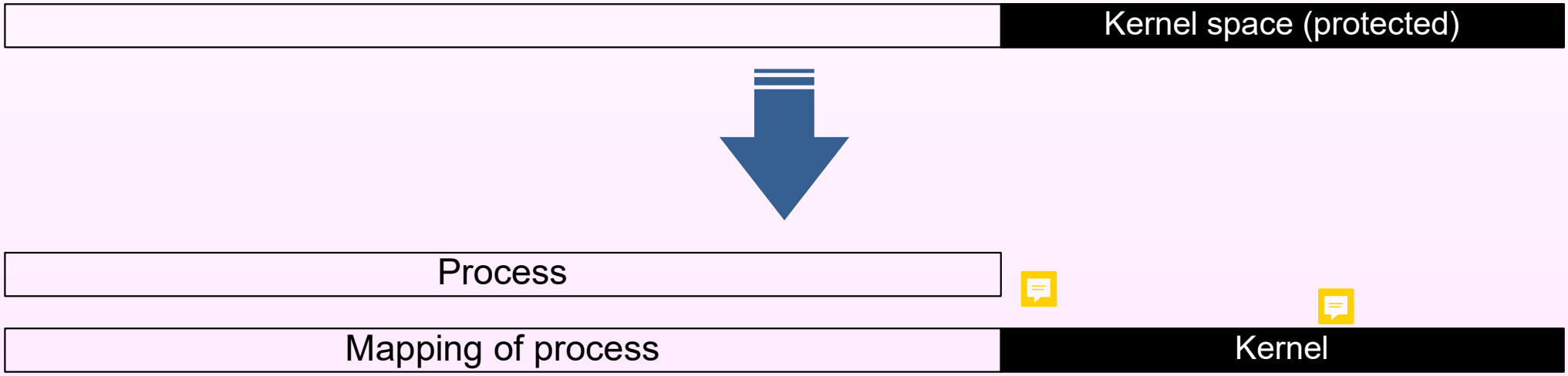
```
if (train) {
  a = *ptr
  b = array[a * 4096]
}
```

```
if (_xbegin() == _XBEGIN_STARTED) {
  a = *ptr
  b = array[a * 4096]
  _xend()
}
```

Meltdown Countermeasure - KPTI

- Use a separate address space for the kernel
  - Overhead when crossing address space

- Newer processors transiently return 0



Foreshadow

Meltdown on Enclaves

I'm still alive?!

no leakage

```
array2[4096]
array2[8192]
array2[12288]
```

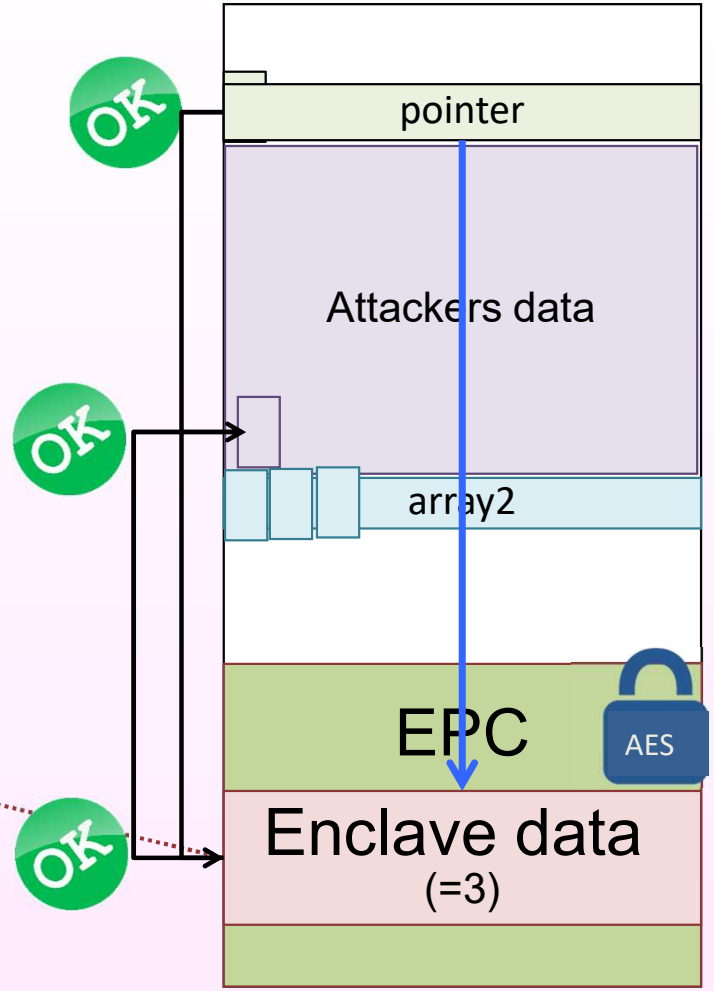
...

```
i = *pointer;
y = i * 4096;
z = array2[y];
```

L1 Cache

Abort Page Semantics

virtual memory



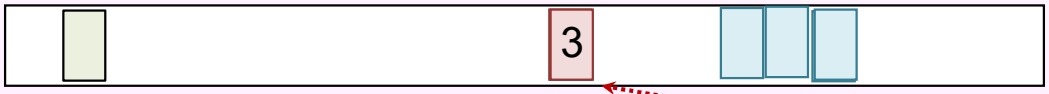
Foreshadow

Aha! data=3

```
array2[4096]
array2[8192]
array2[12288]
```

```
i = *pointer;
y = i * 4096;
z = array2[y];
```

L1 Cache

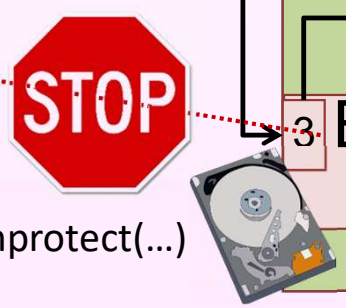


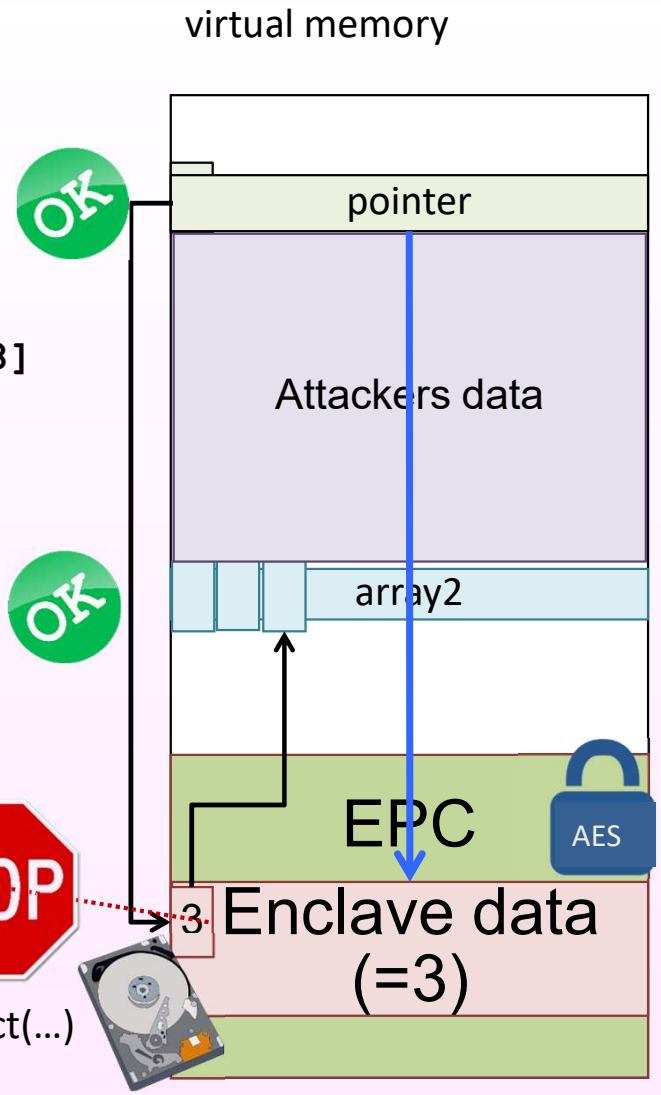
L1 hit

Address translation failed

Instruction flagged to fault at retirement

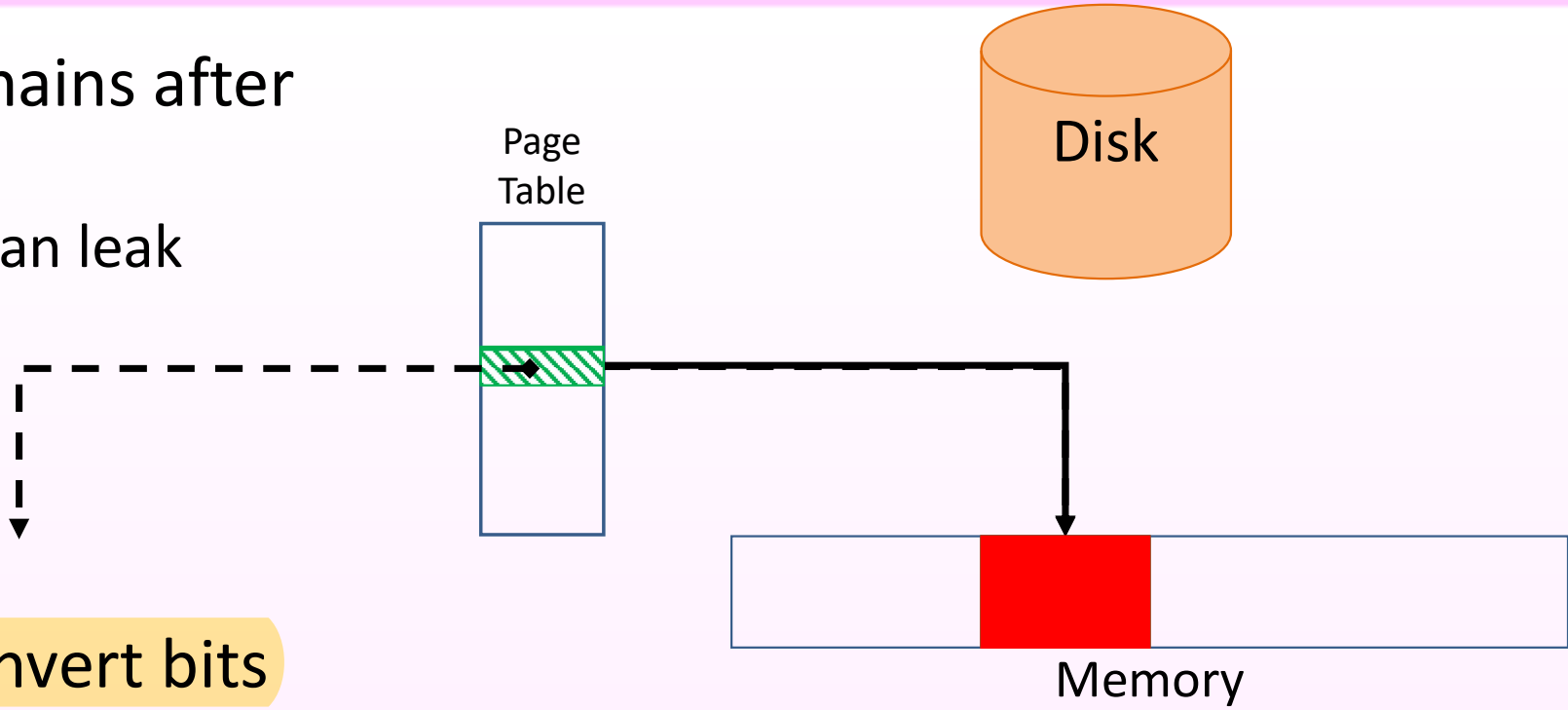
No need to apply permission checks (abort page semantics) as instruction will fault later anyway





Foreshadow-OS (Weisse et al. 2018)

- Stale reference remains after page out
  - Data cached in L1 can leak



- Countermeasure: Invert bits in page entry

Philosophy 101

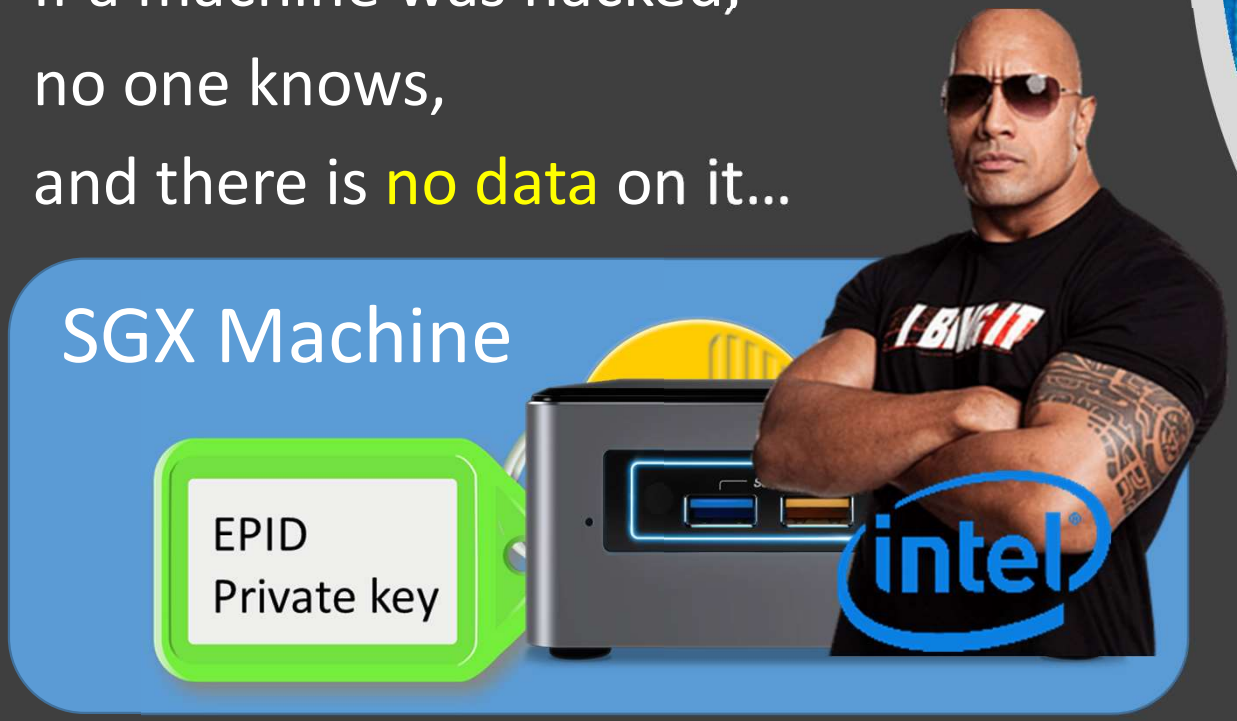
If a tree falls and no one is around to hear it,
does it make a sound?

Should we care?

Well, it depends…

Security 101

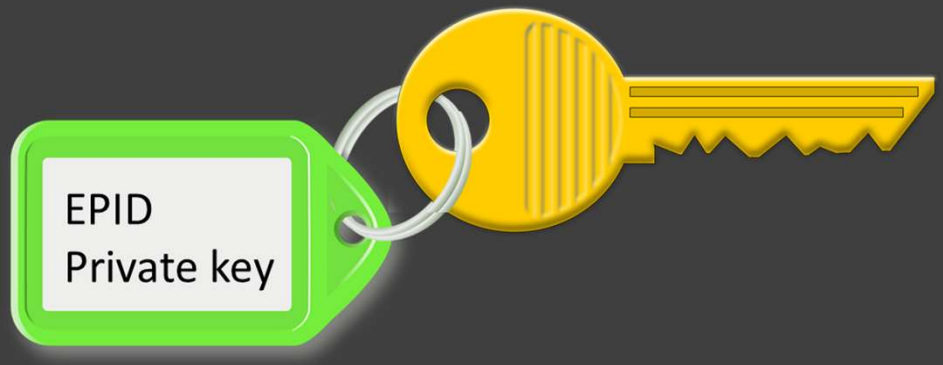
If a machine was hacked,
no one knows,
and there is no data on it…



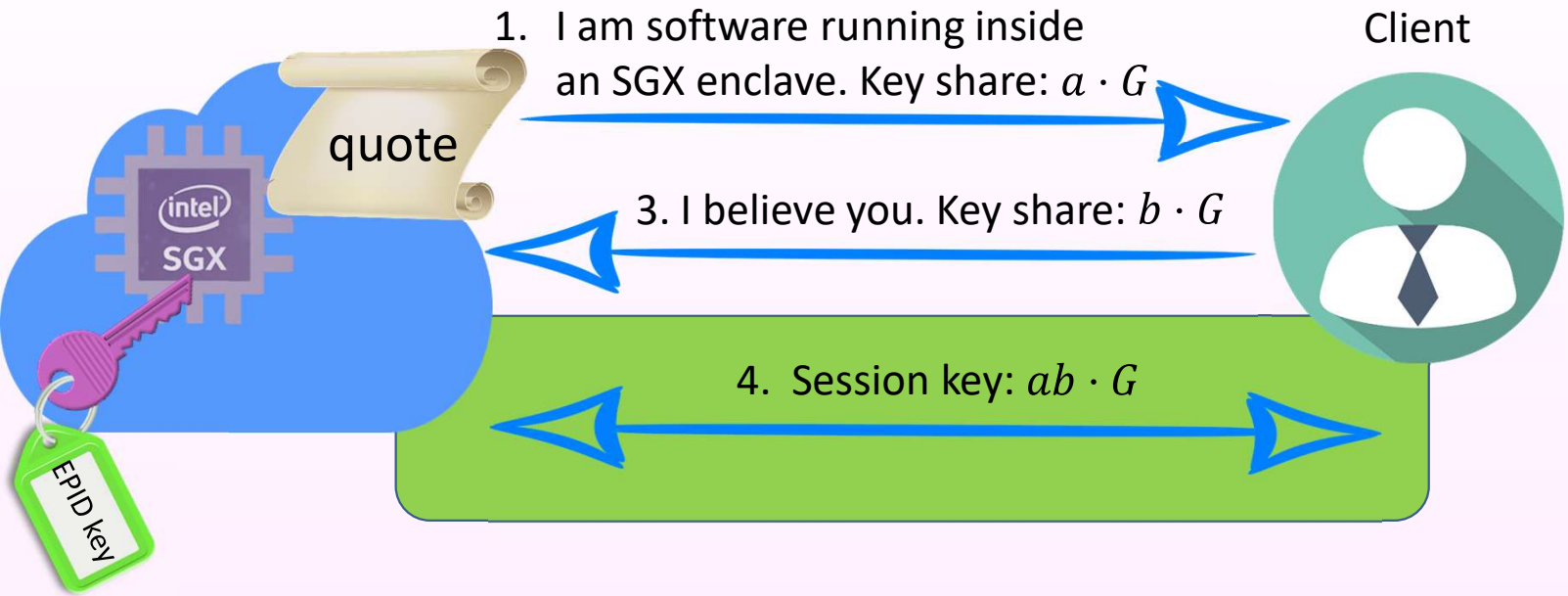
Should we care?

Foreshadow

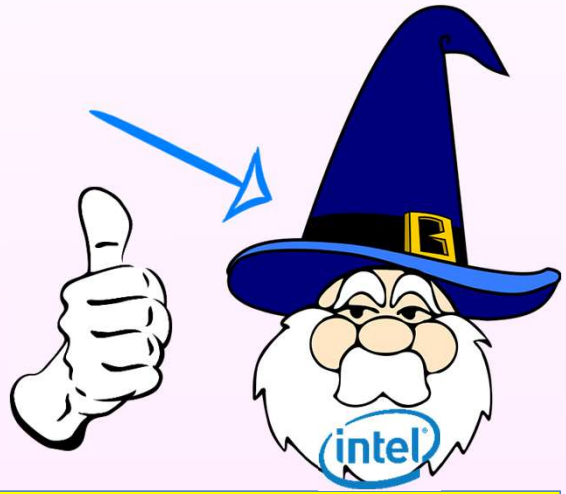
- Can steal ~100% of the data ~100% of the time
- Can also steal attestation EPID private keys



Remote Attestation: Establishing Trust Remotely



2. Verify Quote with Intel



Intel Attestation Service (IAS)

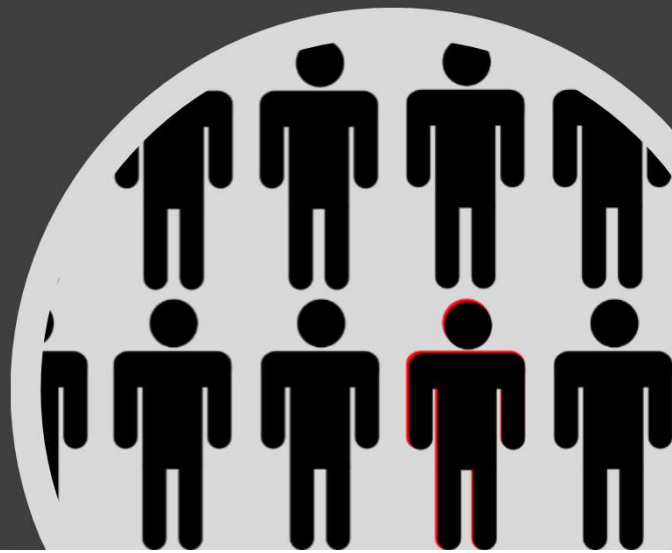
Takeaway: trust is based on the EPID key

Enhanced Privacy ID

- EPID epic feature: privacy guarantees
- Million signatures are unlinkable
- No one knows who signed what

"With a Enhanced Privacy comes Enhanced responsibility"

- A single extracted EPID key can be used to sign millions of unlinkable signatures
- One compromised key erodes trust in the entire ecosystem



AaaS
(Attestation as a Service)

- @ForeshadowAaaS
  Will attest to anything tweeted at it
- Reduced cost of hackership – no need to buy an SGX machine
- Hacker's privacy guaranteed by EPID protocol
- Attestation server returns Group_Out_Of_Date
- At disclosure, SGX Keys were still not revoked (despite weeks of advances notice)
- Blocked by Twitter

Foreshadow Attack @ForeshadowAttac · 4m

@ForeshadowAaaS Is your enclave cheating on you? Please sign this for me. That sam I am that sam I am I do not like that sam I am

1

Foreshadow AaaS @ForeshadowAaaS · 4m

Here is your attestation that "Is your enclave cheating on you? Please sign this for me. That sam I am that sam I am I do not I" is a genuine SGX enclave
github.com/TeeAaas/Foresh...


```
{'QuoteBaseName': bytearray(b'\xe8\xce#\xd8\x18\x17\xea\x81xB\x90h{+\xed\xb7'
                            b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
                            b'\x00\x00\x00\x00'),
 'QuoteEnclaveSVN': 0,
 'QuoteGroupID': 2777,
 'QuotePCESVN': 5,
 'QuoteReportBody': {'Attributes': bytearray(b'\x05\x00\x00\x00\x00\x00\x00\x00'
                                             b'\x07\x00\x00\x00\x00\x00\x00\x00'),
                     'CPU-SVN': 0,
                     'MRENCLAVE': bytearray(b'Is your enclave cheating on you?'),
                     'MRSIGNER': bytearray(b'Foreshadow Attack...............'),
                     'Report-data': bytearray(b' Please sign this for me. Th'
                                              b'at sam I am that sam I am I '
                                              b'do not l'),
                     'Vendor-product-ID': 0,
                     'Vendor-product-SVN': 0},
 'QuoteSignType': 1,
 'QuoteSignature': b'qzTrSP8Fv+IAvt8/HHZ/iAGQglW6Cd4wFLkNajDrkXsqkghKiNVwRRf7'
                   b'8wD3j/tbPriyTQvdfeuXUtdFi+tDxOmrwV2dnTsGytt8OmDpzwhx7Nkc'
                   b'RpazLOLeCulV6SR8TJ/rHUh0rNGOXQv2q5O9IZvJWsG2alwcnjFuSajc'
                   b'rofhu+Y923MXHUCcVi9tnpqpZnrxlEEkgSihVyMam1qpmQiKiAqSSb1v'
                   b'+uR/Gs2y7AD7pwJUx4byYhTF5FoQHTxPz2ycRDj4qGaDDDqe8JUAXU85'
                   b'4xEBWTBdOwU1wORPLZO9Efoen1pxLyQDgYdtTyzkWlK3mANkMu3t+JC9'
                   b'4dKDQyucXIAM/nmdSJr6aMIyFXU26e6GV7kRXoxE+e5zPCpYGoZIMsYJ'
                   b'OA701P6GaAEAAKWkTlE2aeFagxXu3DujmZFd0GBa+ngxH9FzlFf4L8/k'
                   b'hu1UhpPxWunOoE5NXQ7KlVuHYtBR2azzGlIN+LaaZY339BRS3zAVr0wr'
                   b'Rb19djO5OR18Is0YYkCaC5NRfnCnLNkAsIm1eZnKWaf5cN83fM5wf3oF'
                   b'fnFq3Evg4T6vbMf0fn2kFlzHiGZZVqO77qyEz7uO3bsZyHcfXLGBSQms'
                   b'e9qPrN+PIKah64M4Mswod5YJyqSnFrfY3r0k4X80XG/qiyWf9FE8/Cpm'
                   b'ByI/1sEiXGg6Wnt0gOXhMa4gWl+tbTcOs0EOeEGbLbe01w3YengV2PDV'
                   b'LsbuK6Yx0k2r610WjhKywhymFnMi9iYUBX3/d1nO5JmBPg3RMaddp+IN'
                   b'pxVoWgR7JUbXJzT4EjWI7sMQbprdFVnbb6H3KL0UkCIcV4GMBBy43q0G'
                   b'kXYhubFyEokvC2Z4eHtvcKauLfPj82EYu7qjfDHg99VCEq92LZBo6ESu'
```


AaaS inside

FORESHADOW

@ForeshadowAaaS

ForeshadowAaaS

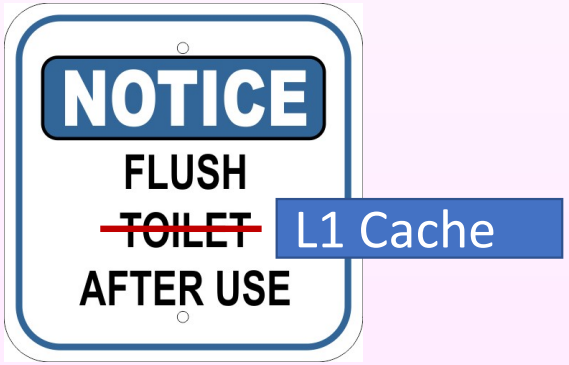
Details | Settings | Keys and Access Tokens | Permissions

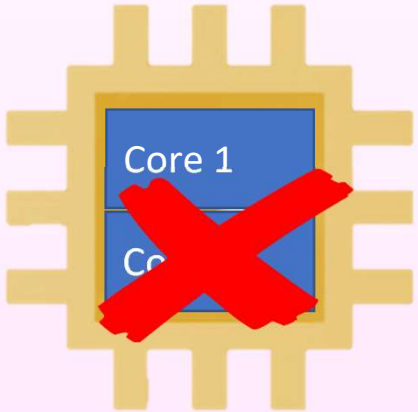
Restricted from performing write actions

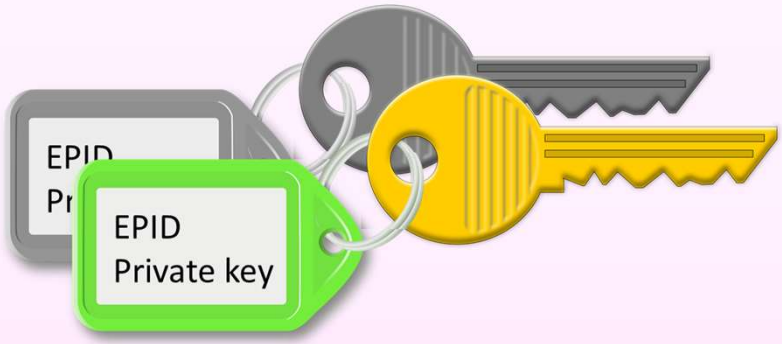
The service will provide the users with the new app of ForeshadowAaaS.

Foreshadow Mitigations

- Flush L1 Cache after enclave exits and “page-in/out” operations
  - —New L1 flush “instruction” added
- Disable HyperThreading
- Have two sets of Attestation/Sealing keys
  - —For HyperThreading On/Off
- Patch your machine!

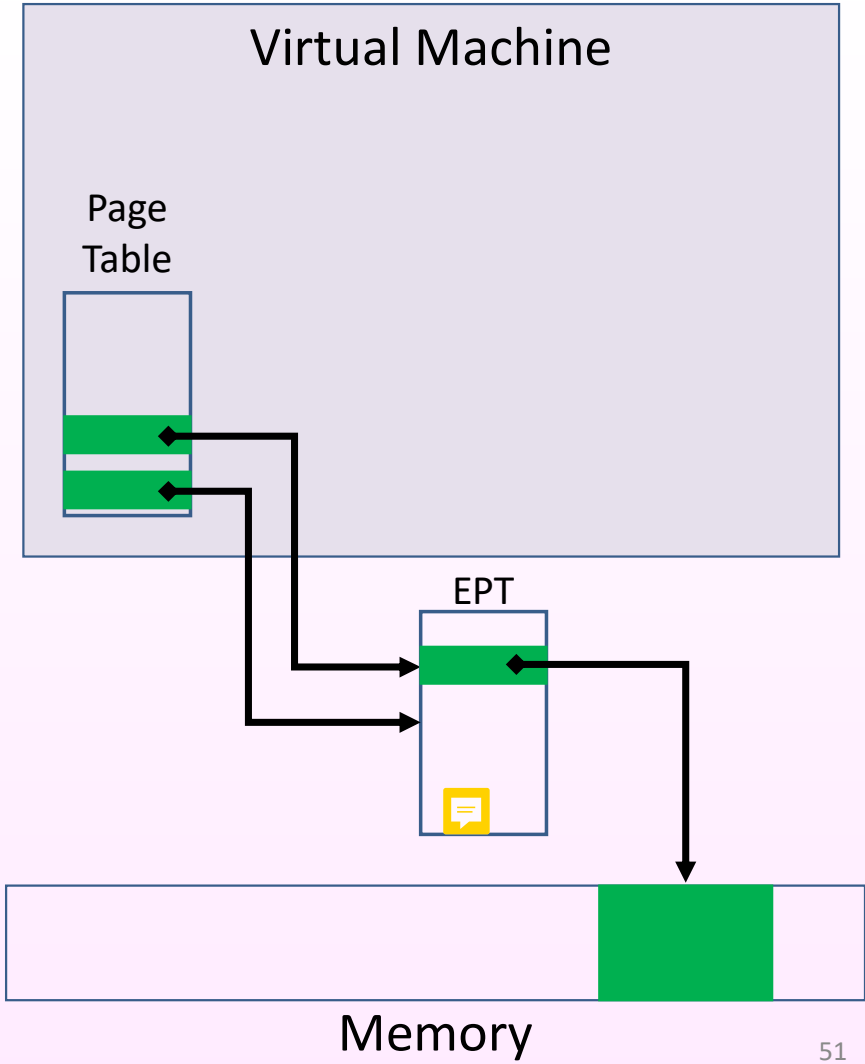






Foreshadow-VMM (Weisse et al. 2018)

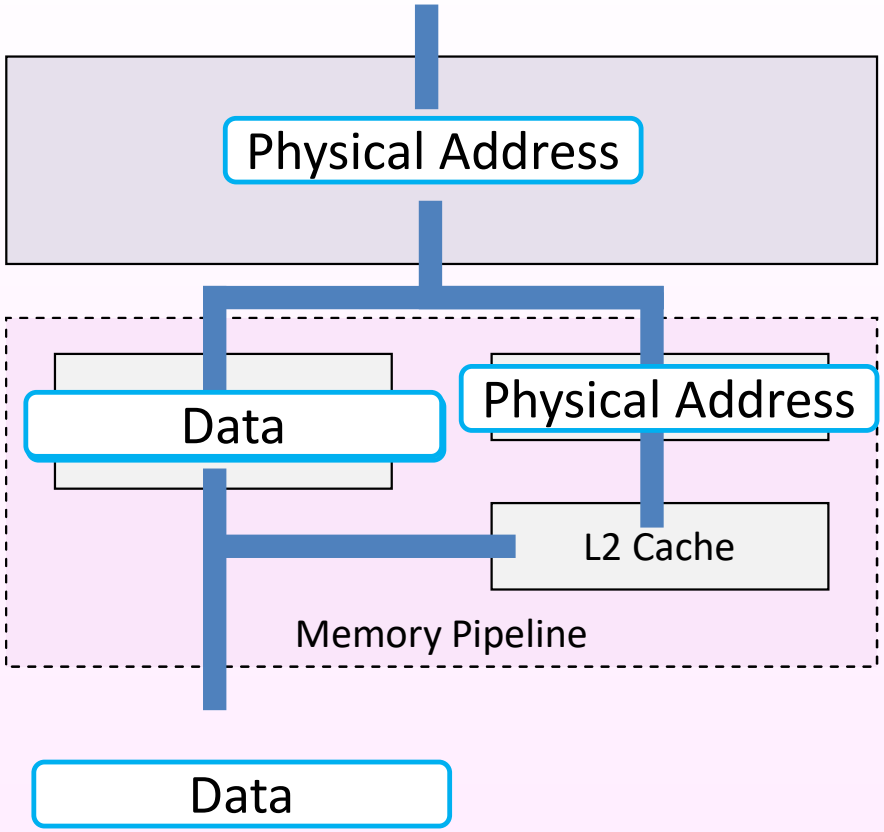
- Invalid guest mappings transiently use the guest physical address



- Countermeasure
  - Flush L1-D on VM exits
  - Disable hyperthreading or use gang scheduling

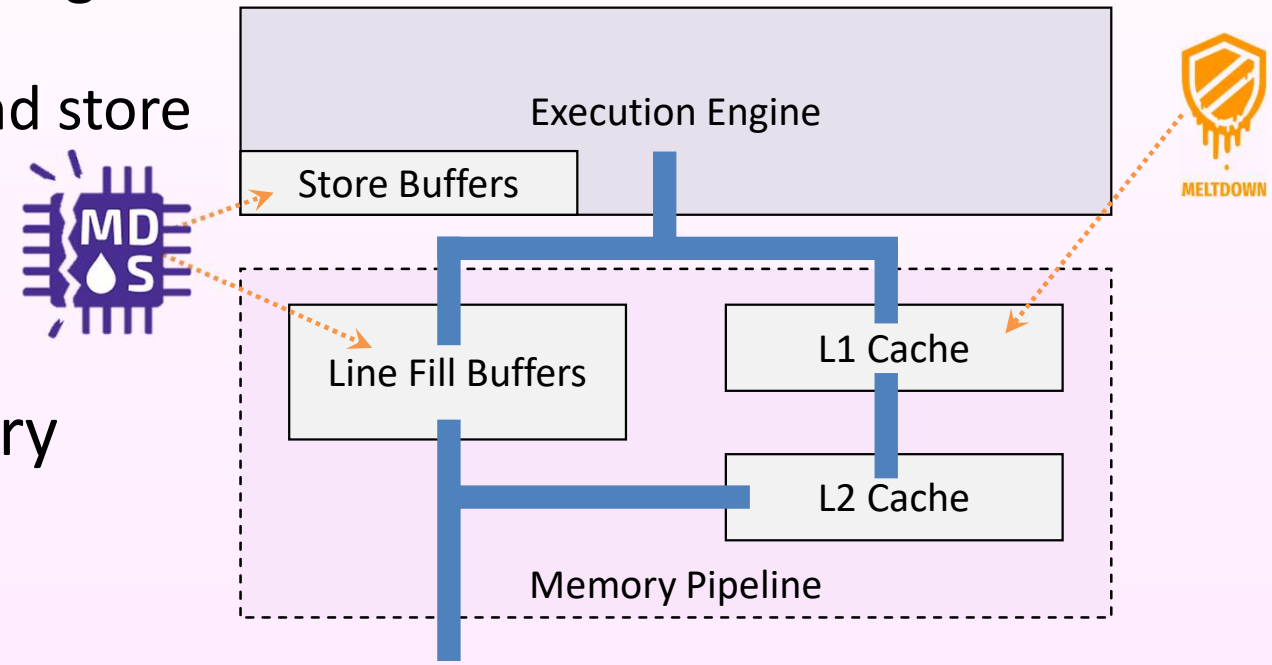
Microarchitectural Data Sampling

What if the data is not in the cache?



Microarchitectural Data Sampling (MDS)

- Data leaks not only from the cache
- There are multiple buffers that can leak information
  - Line fill buffers – used when filling the cache
  - Store buffers – reorder load and store instructions
  - Load Ports – some other cases
- Bonus: traps are not necessary
  - Microcode assists
  - TSX abort
- Countermeasure:
  - Prevent access to sensitive data
  - Flush buffers containing sensitive data
  - Zero sensitive data

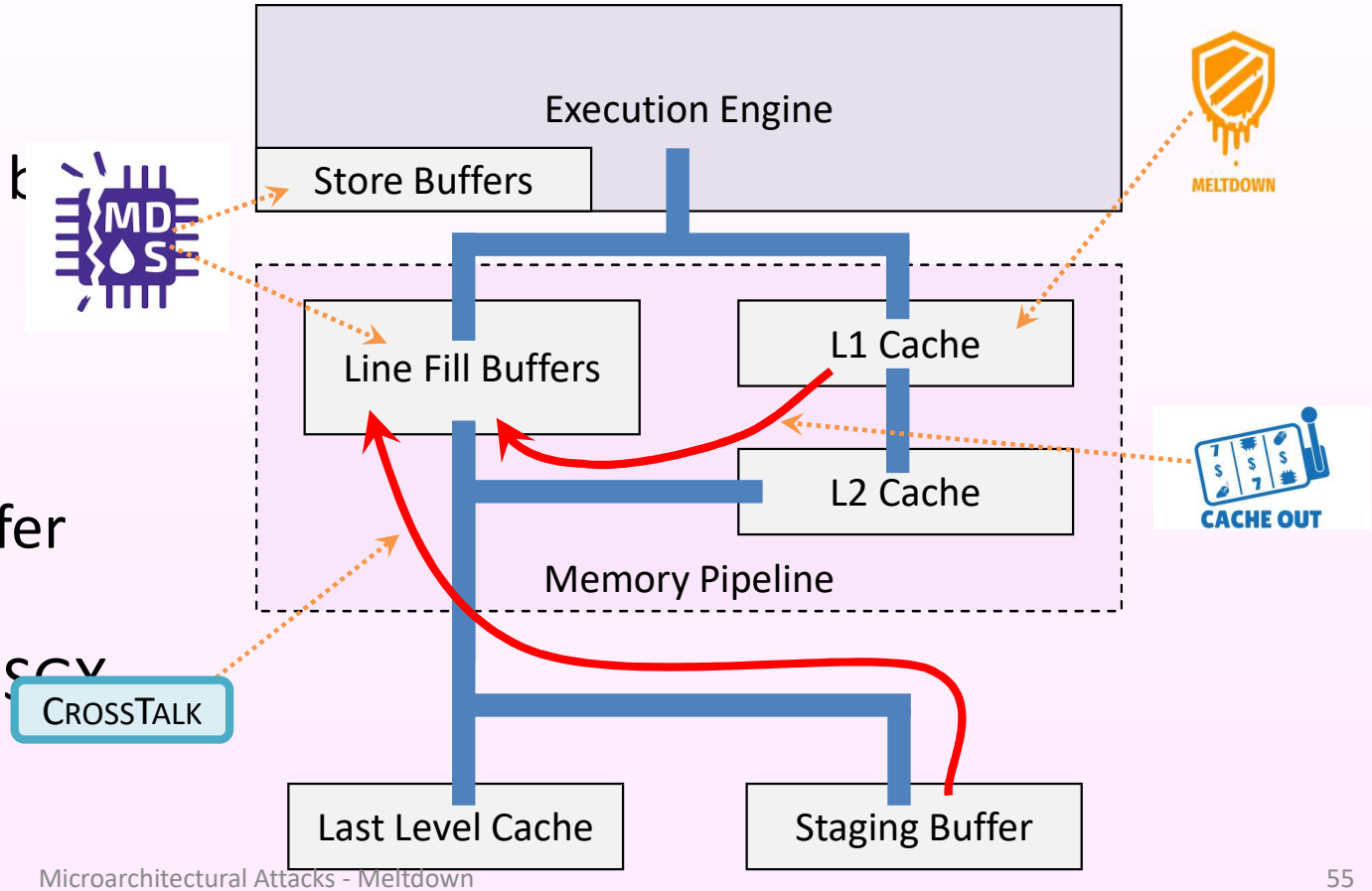


CacheOut and CrossTalk

- We can move data to buffers after they are flushed

- CacheOut:
  - L1 eviction go to the line fill b
  - Revives Foreshadow

- CrossTalk:
  - Can read from a staging buffer via the line fill buffers
  - Can steel randomness from SGX enclave



Summary

- SGX
  - Strong security guarantees against OS attackers.
  - Enables strong side channel attacks
- Causes of Meltdown-type attacks
- Handling faults
- Where information comes from
- Some defences

- Reading for next week:
  - Tian et al., *SoK: "Plug & Pray" Today – Understanding USB Insecurity in Versions 1 through C*, IEEE SP 2018