TRUST LESS

SHRINKING THE TRUSTED PARTS OF TRUSTED SYSTEMS

Ilia Lebedev : MIT / gradient.tech

Featuring contributions from our many wonderful co-authors and sponsors, inc. Delta Electronics, ADI, DARPA, NSF

“SECURITY” AS RISK MANAGEMENT

A system is “secure” if

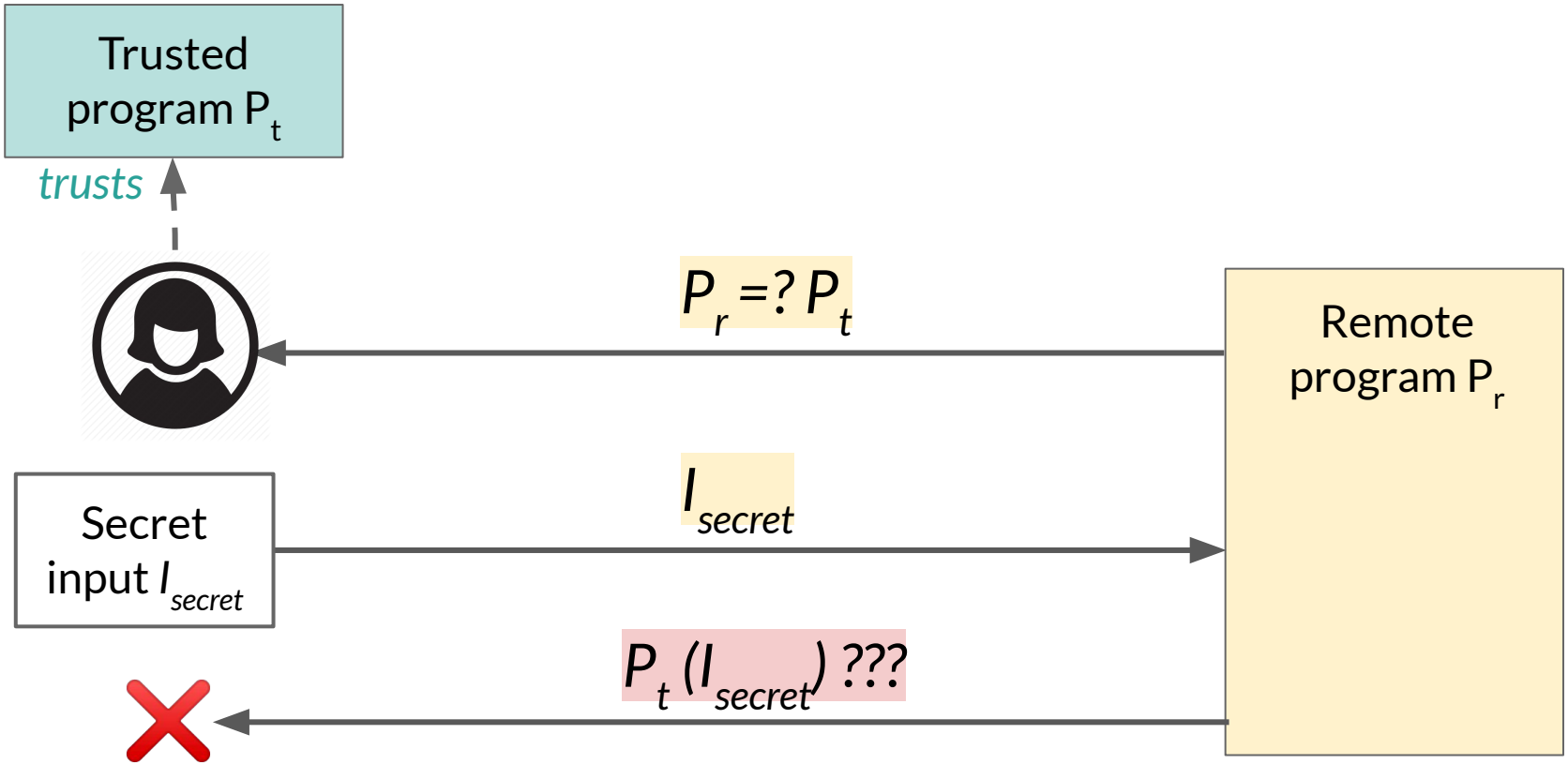
To adversary: cost >> benefit
of attack to adversary

However, this goes both ways:

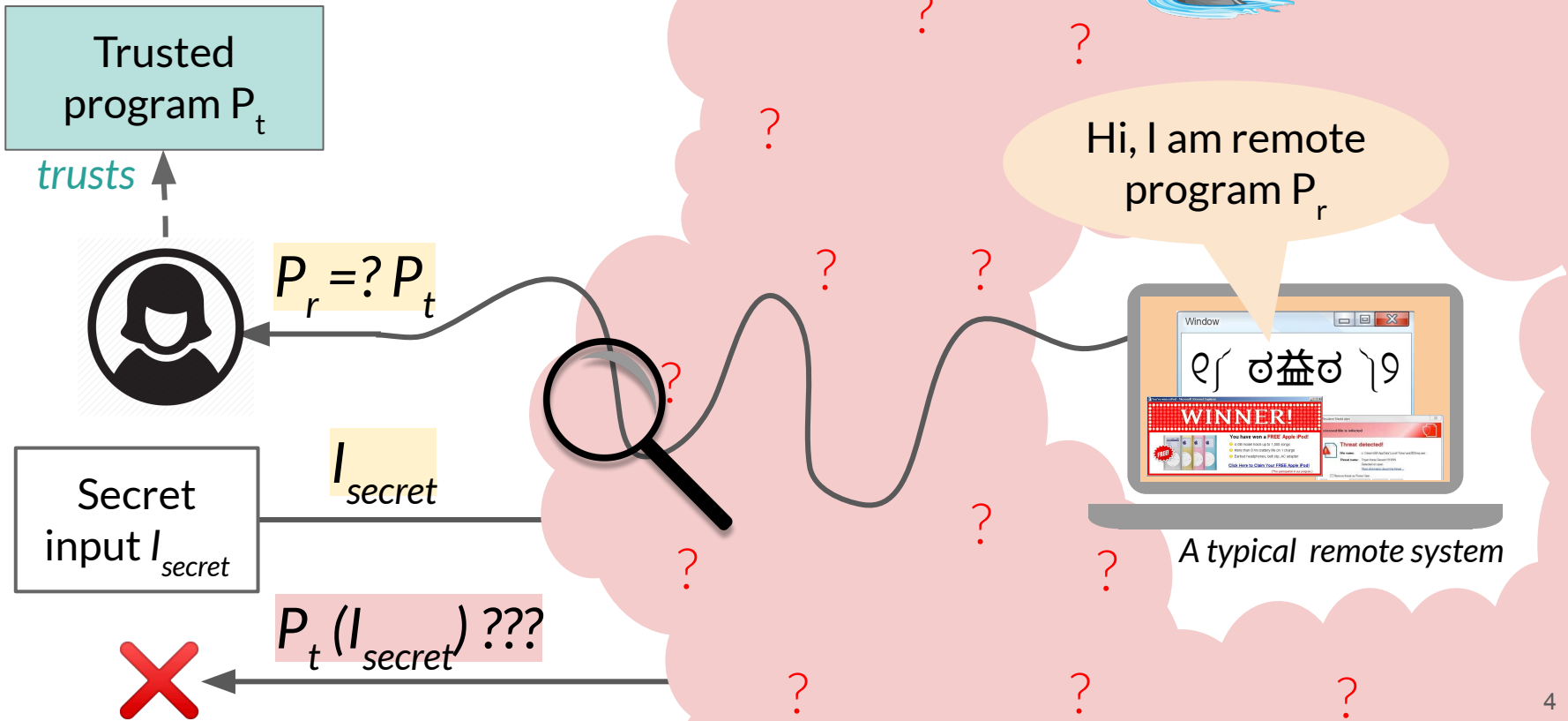
Defenses ought to be worth the effort

Equifax paid << $1-5 / user in fines

REMOTE EXECUTION (1/3)

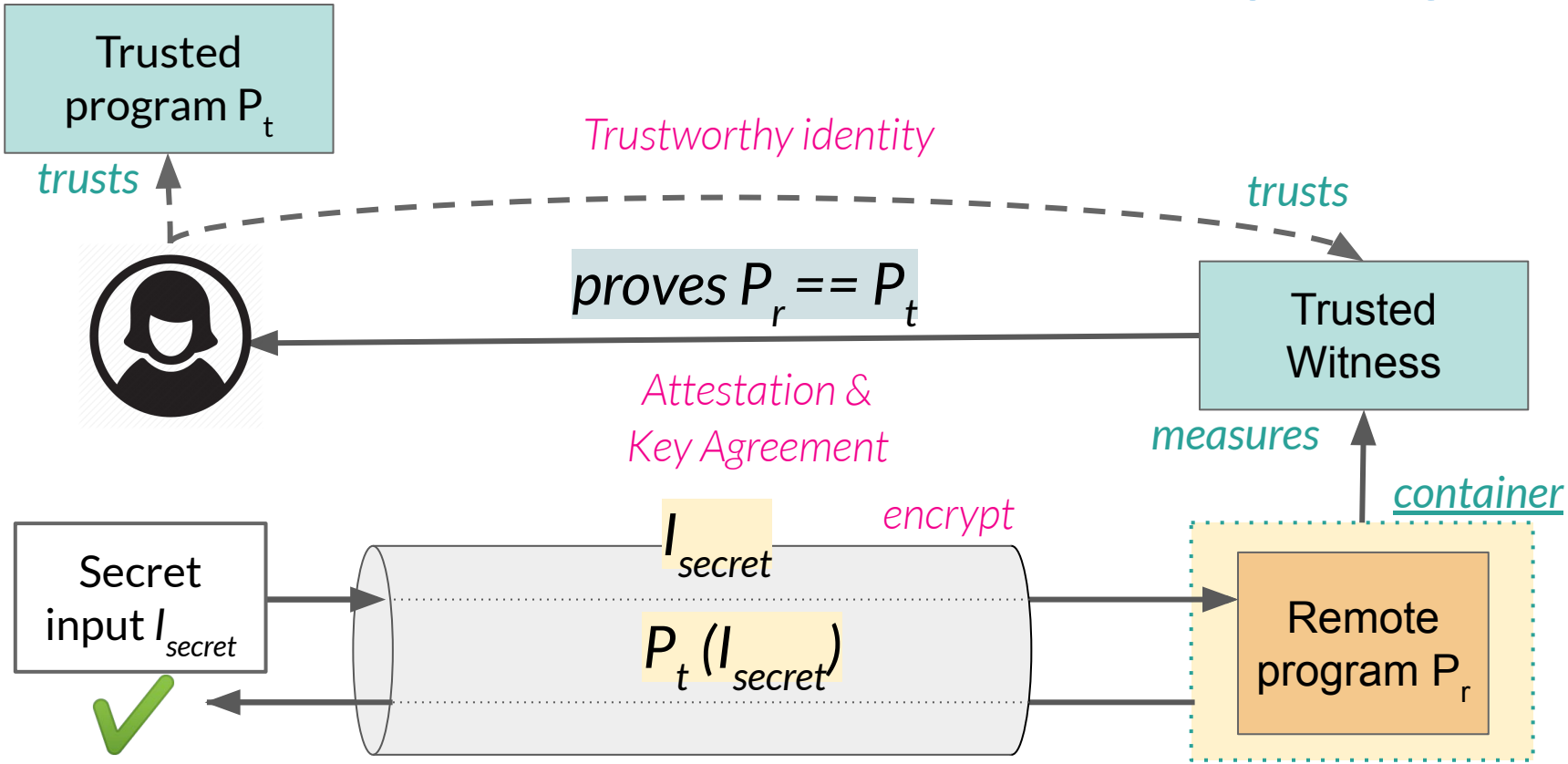


REMOTE EXECUTION (2/3)

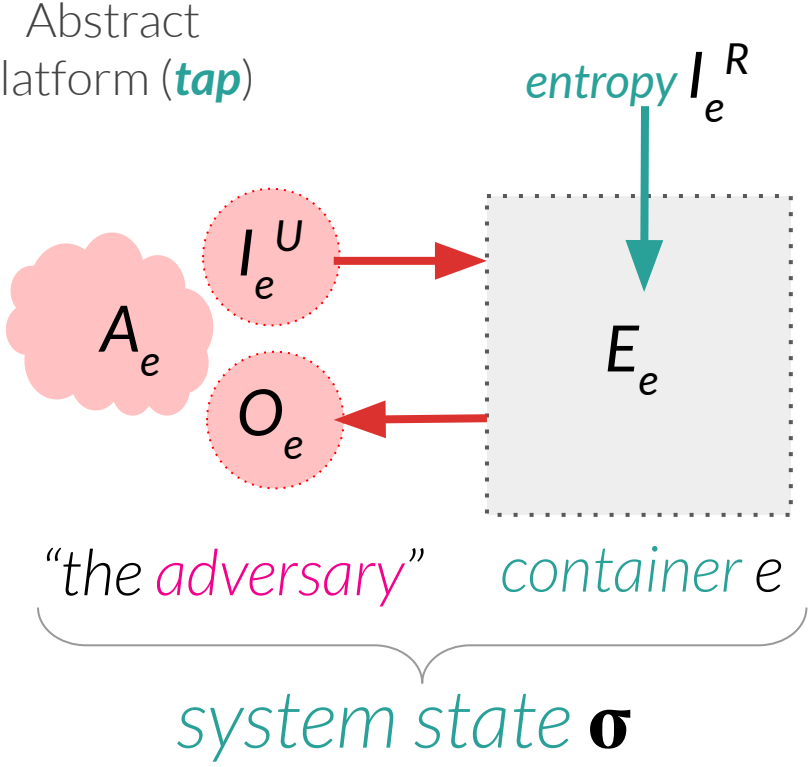


REMOTE EXECUTION (3/3)

"SOFTWARE ATTESTATION"



SECURE REMOTE EXECUTION (SRE)



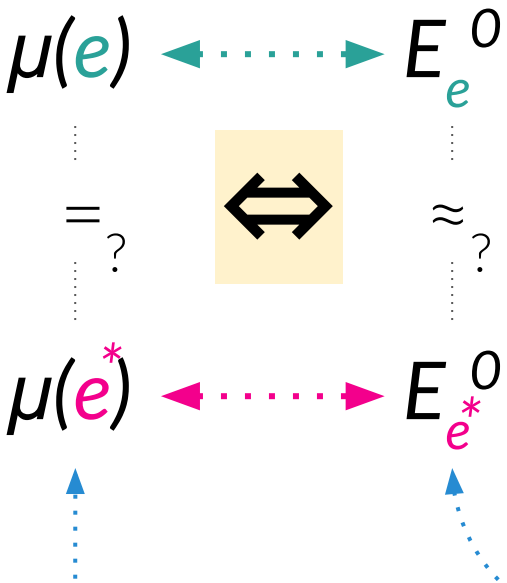
"TRUSTED ABSTRACT PLATFORM"

Integrity
+
Confidentiality
+
Measurement

} SRE

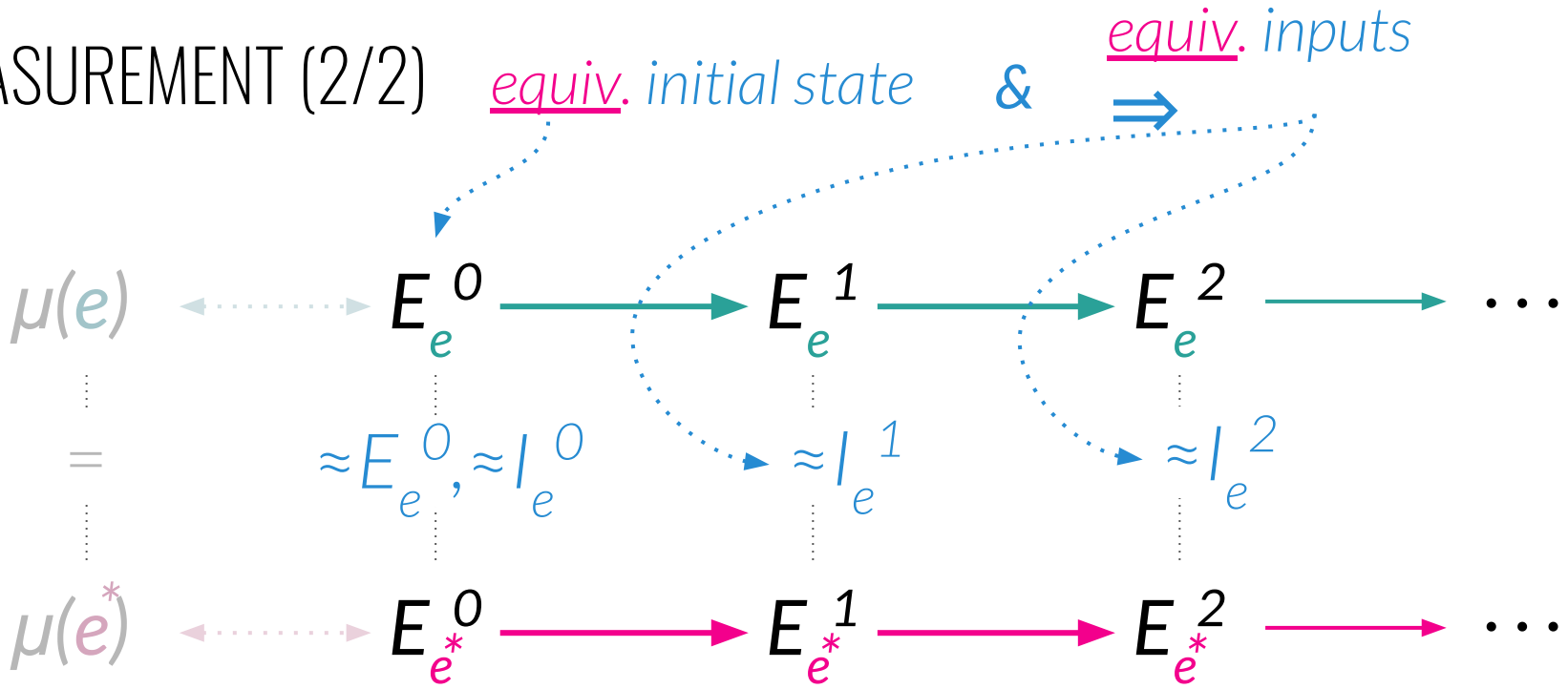
enclave := *container with SRE*

MEASUREMENT (1/2)



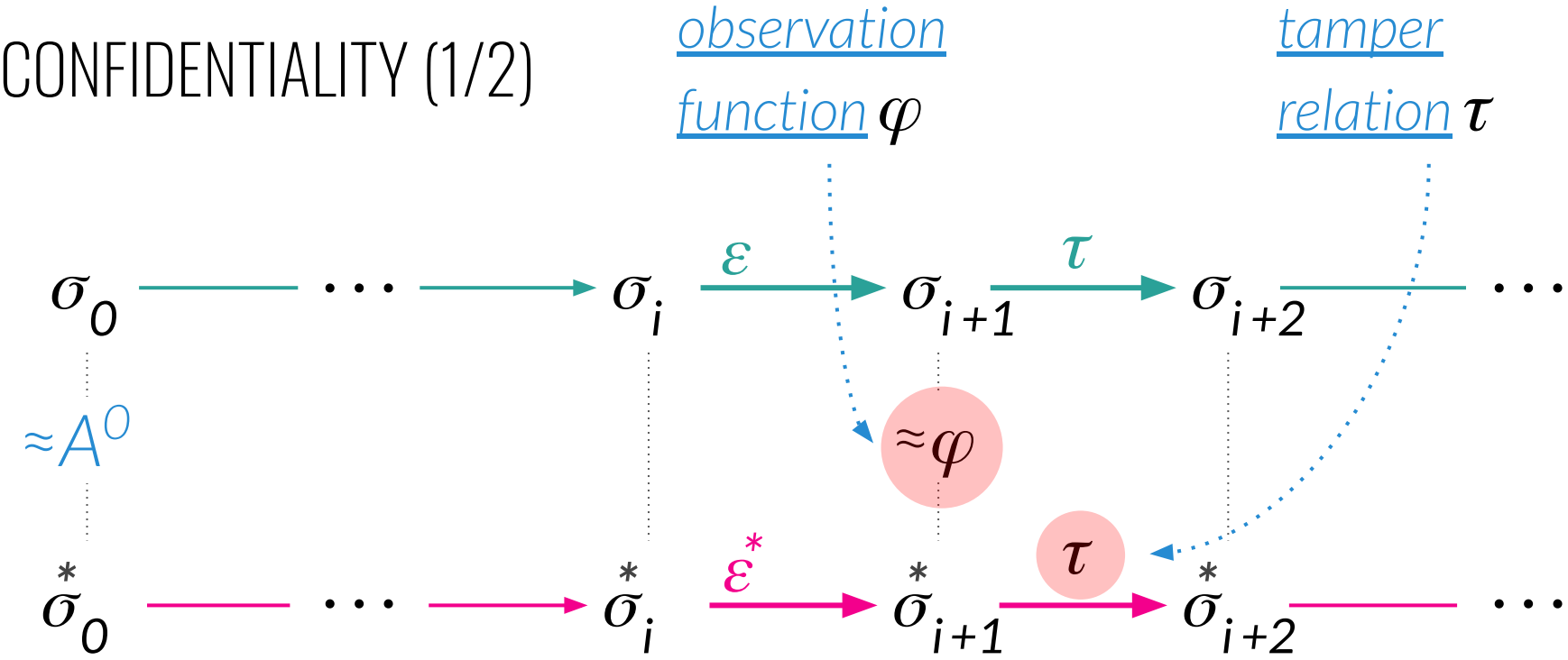
Measurement describes an enclave's initial state

MEASUREMENT (2/2)

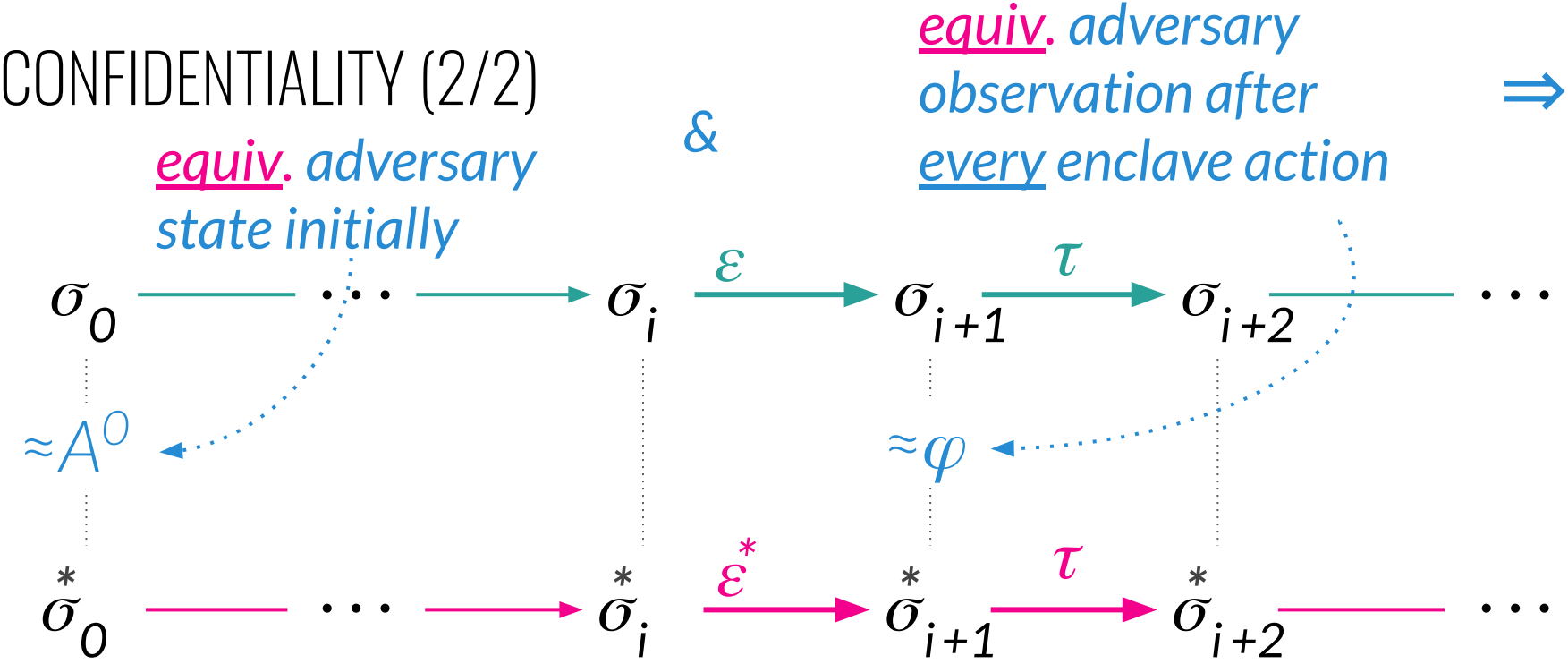


⇒ equiv. states E^x and outputs O^x $\forall$ x

CONFIDENTIALITY (1/2)



CONFIDENTIALITY (2/2)



⇒ equiv. states A^x ∀ x

THREAT MODEL (1/2)

Given (τ, φ):

Integrity
+
Confidentiality
+
Measurement

enclave
$\Rightarrow \mathrm{SRE}_{(\tau,\varphi)}$

Observation function φ

A_e, I_e, O_e
at any time

E_0 *(initial)*

but what else?

Exceptions?
Page tables?
Cache misses?

THREAT MODEL (2/2)

Given (τ, φ) :

Integrity
+
Confidentiality
+
Measurement

enclave
$\Rightarrow \mathrm{SRE}_{(\tau,\varphi)}$

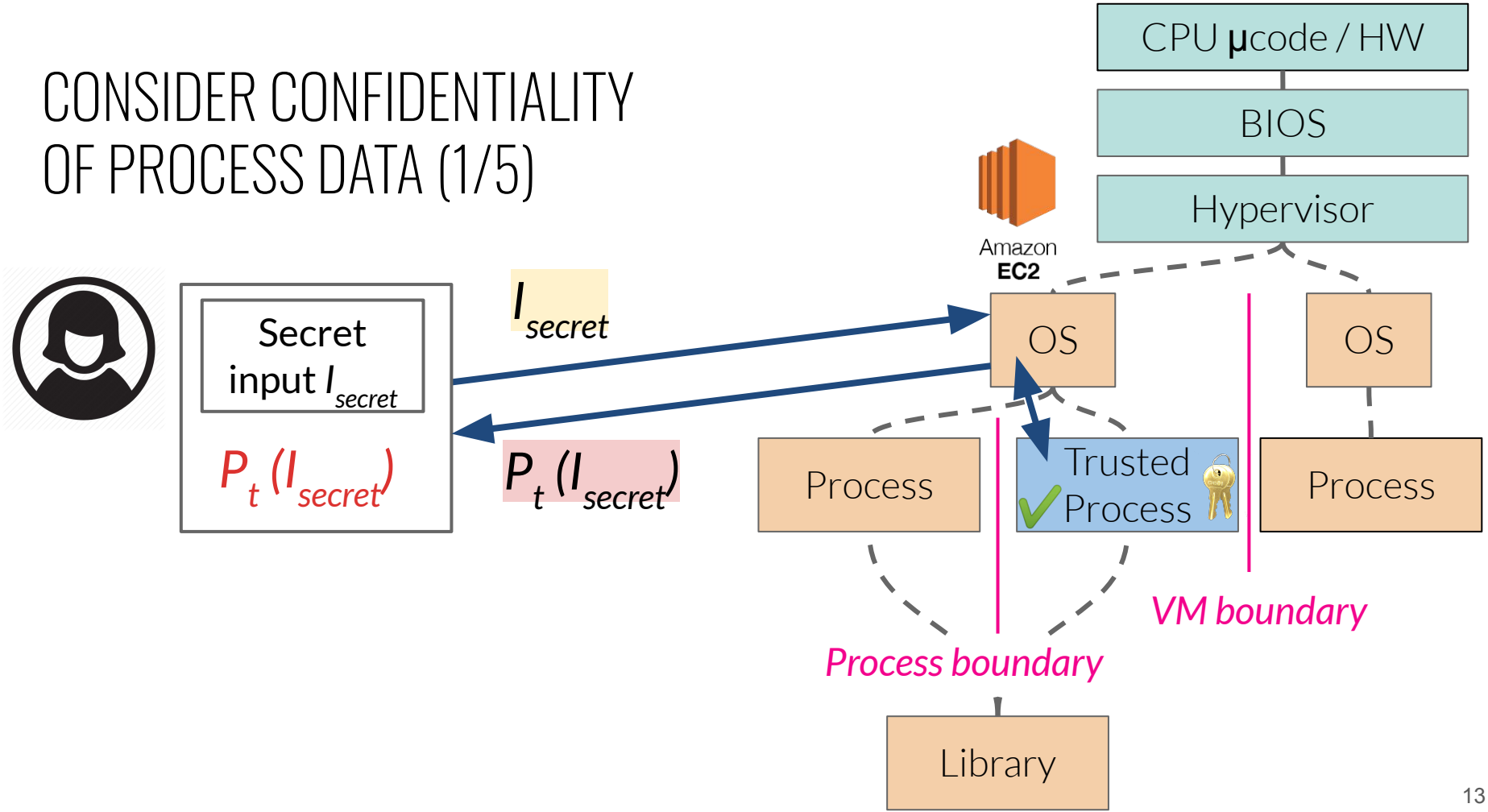
Tamper relation τ

mess with A_e, I_e
Destroy e
create other enclaves

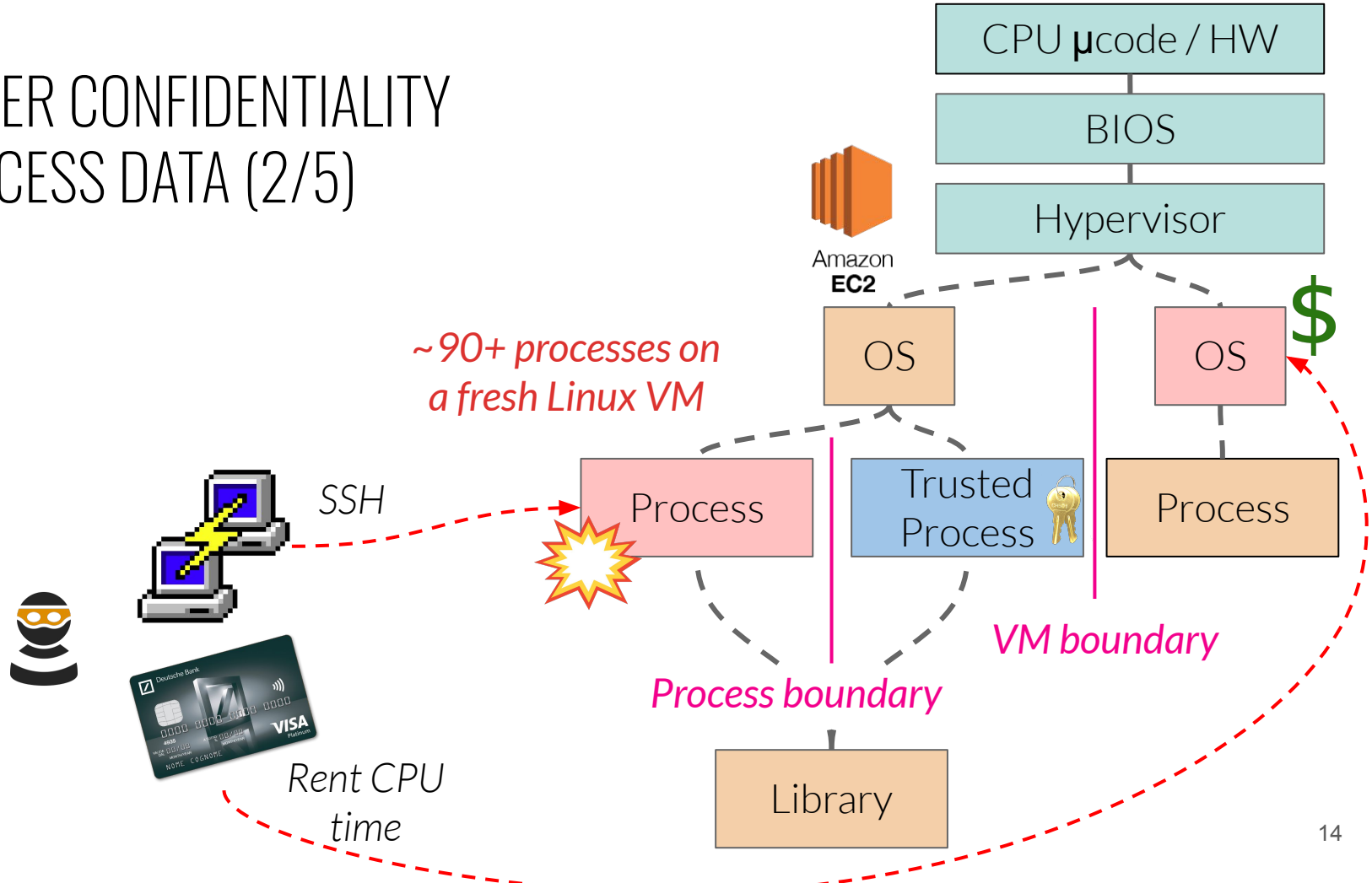
→

Result of state changes:
Cache hit/miss?
Page fault?
etc.

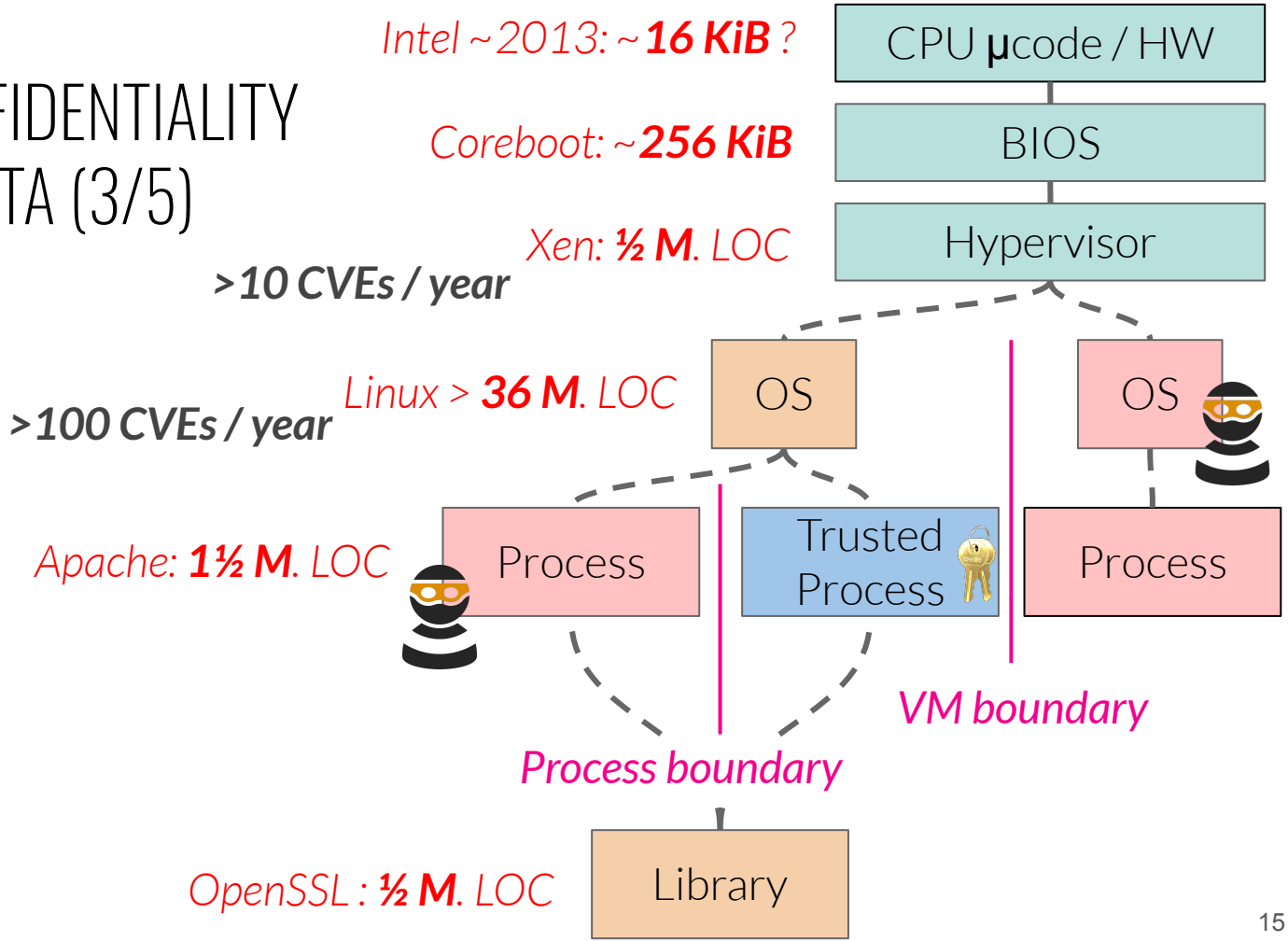
CONSIDER CONFIDENTIALITY OF PROCESS DATA (1/5)



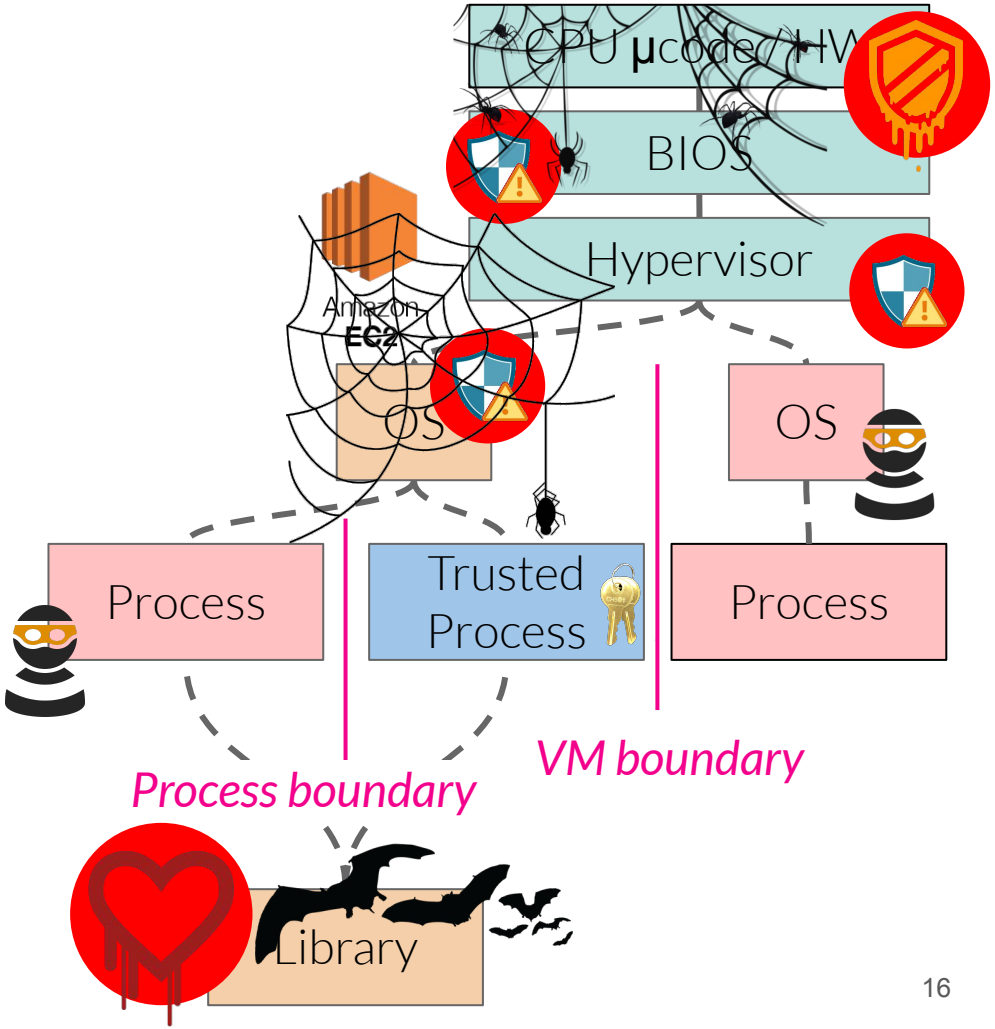
CONSIDER CONFIDENTIALITY OF PROCESS DATA (2/5)



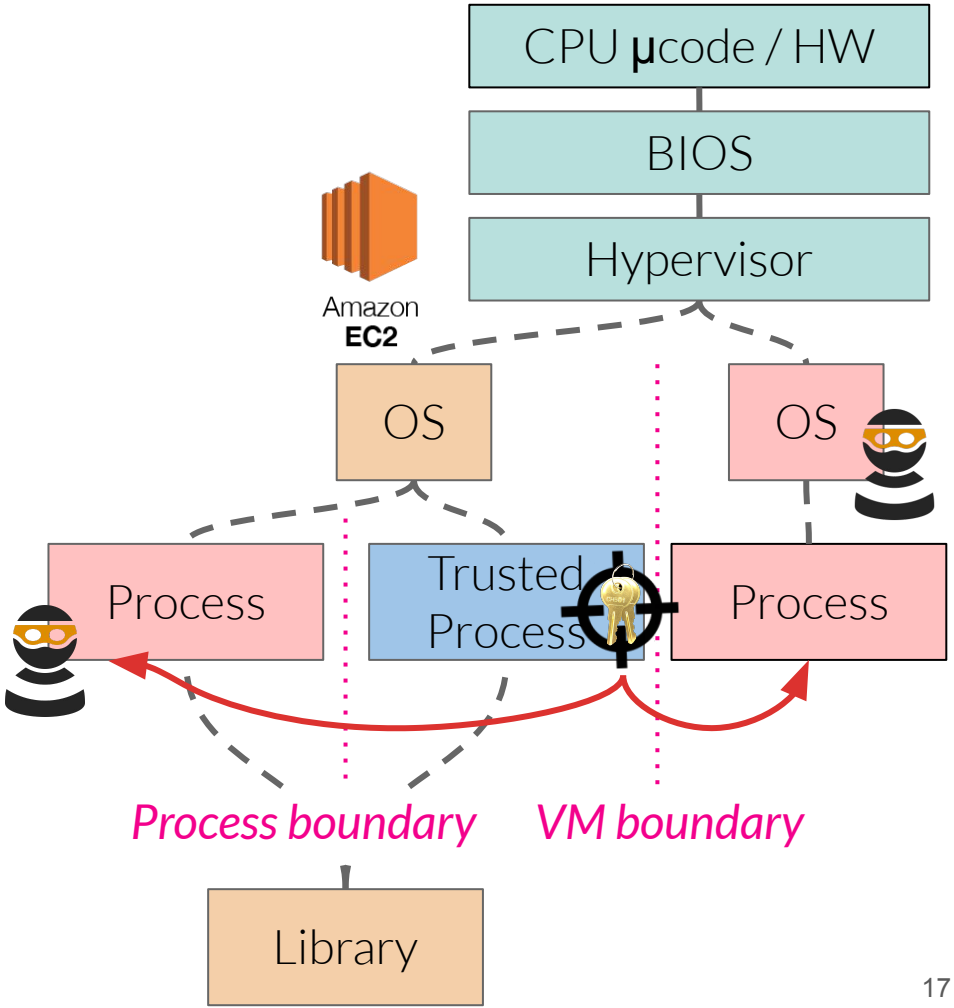
CONSIDER CONFIDENTIALITY OF PROCESS DATA (3/5)



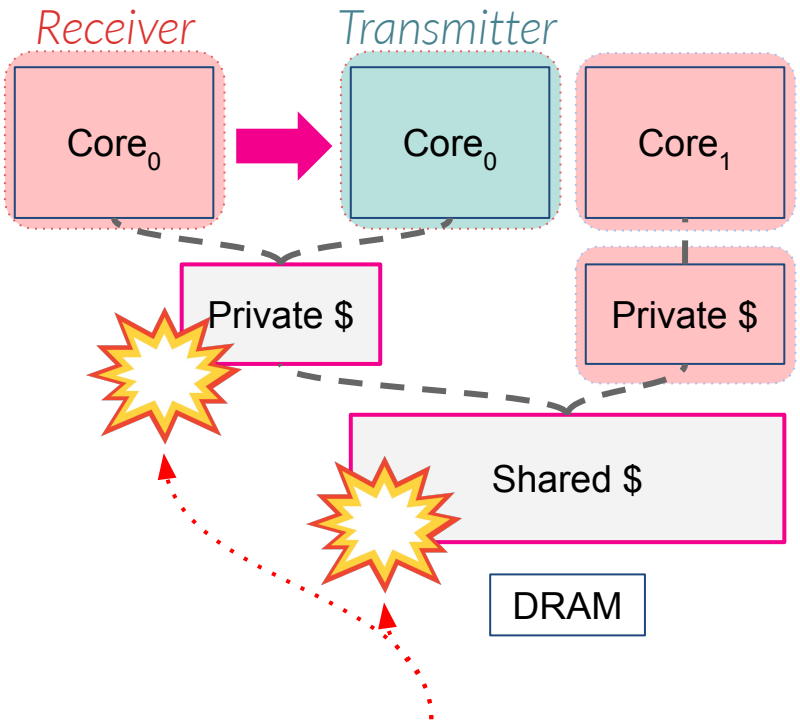
CONSIDER CONFIDENTIALITY OF PROCESS DATA (4/5)



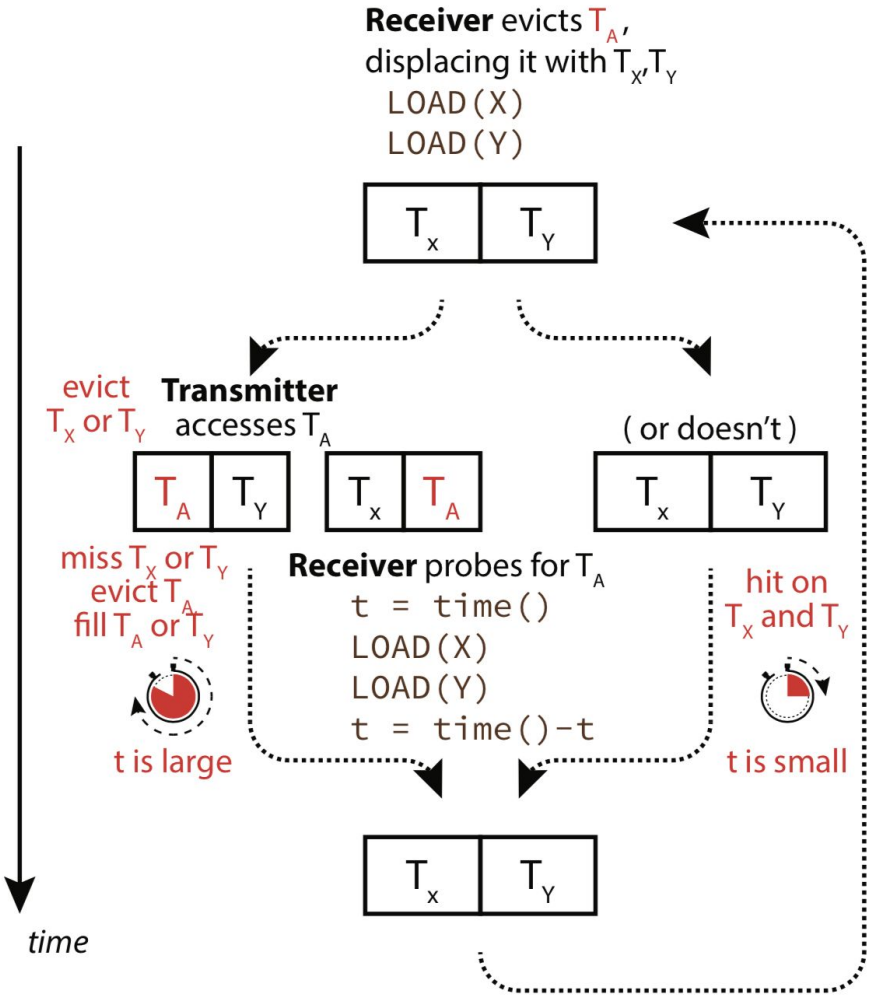
CONSIDER CONFIDENTIALITY OF PROCESS DATA (5/5)



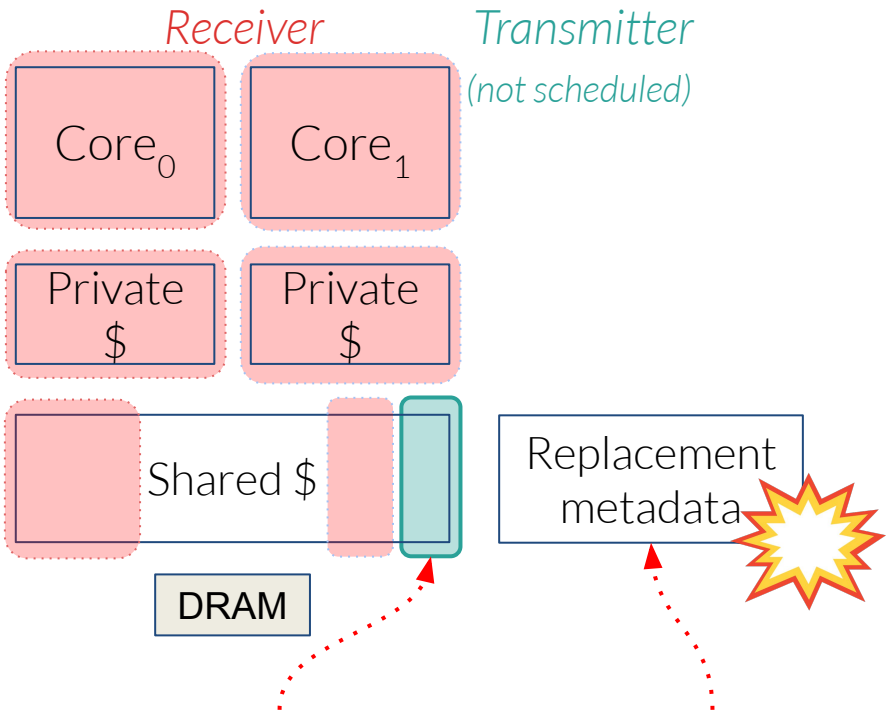
CASE STUDY: CACHES LEAK (1/2)

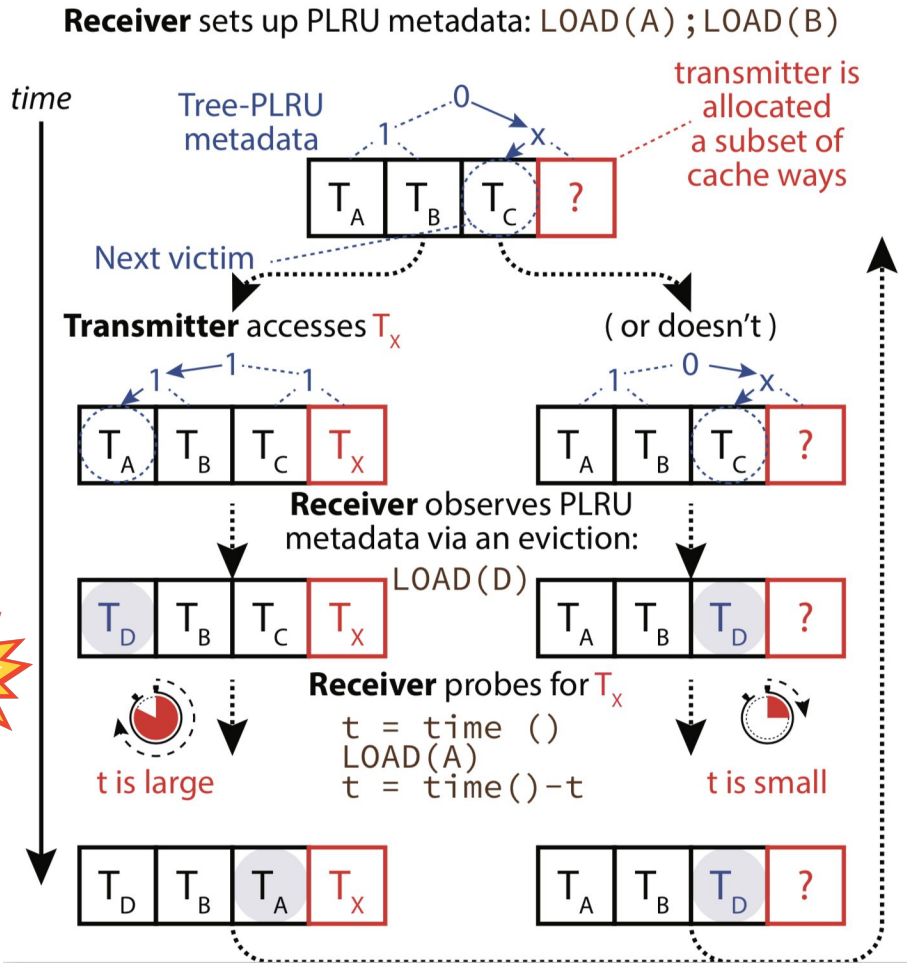


Processes compete for cache lines; contention can be observed

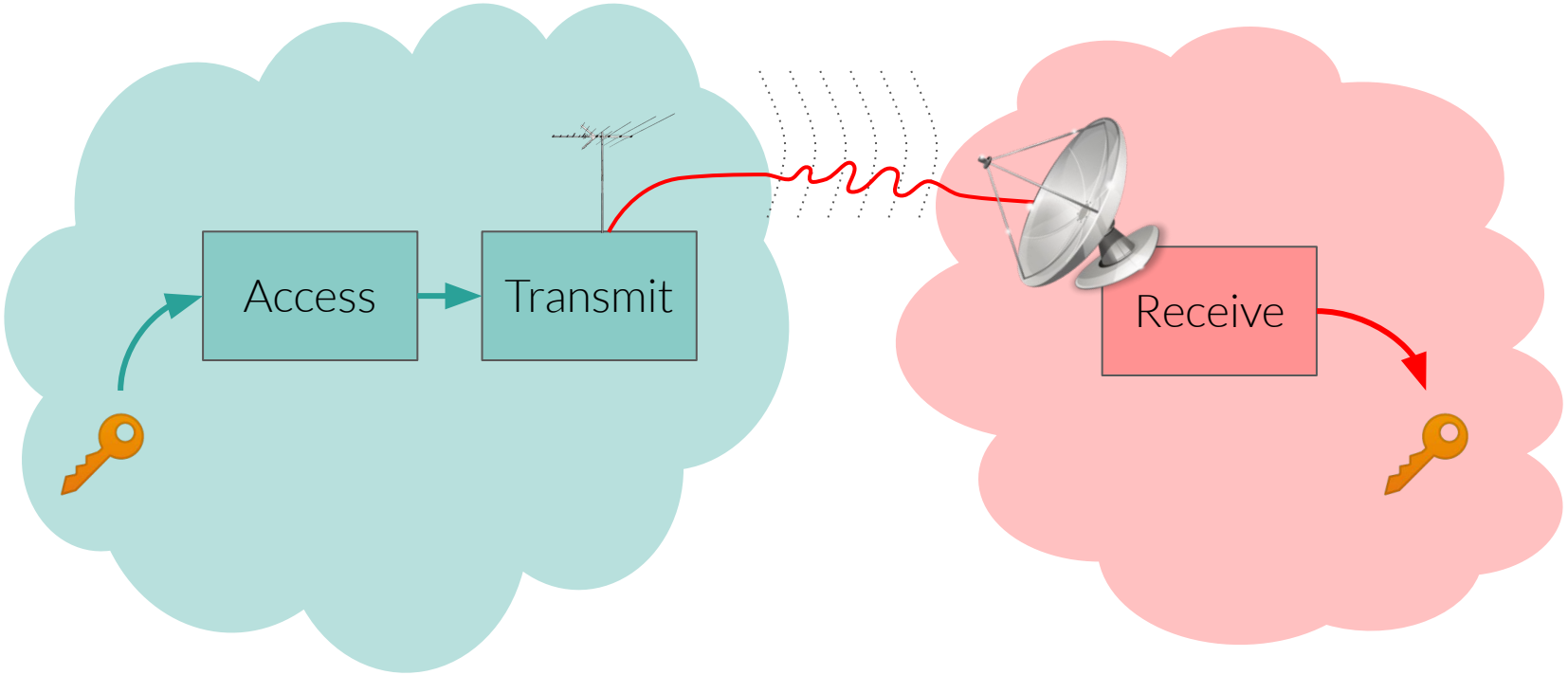


CASE STUDY: CACHES LEAK (2/2)

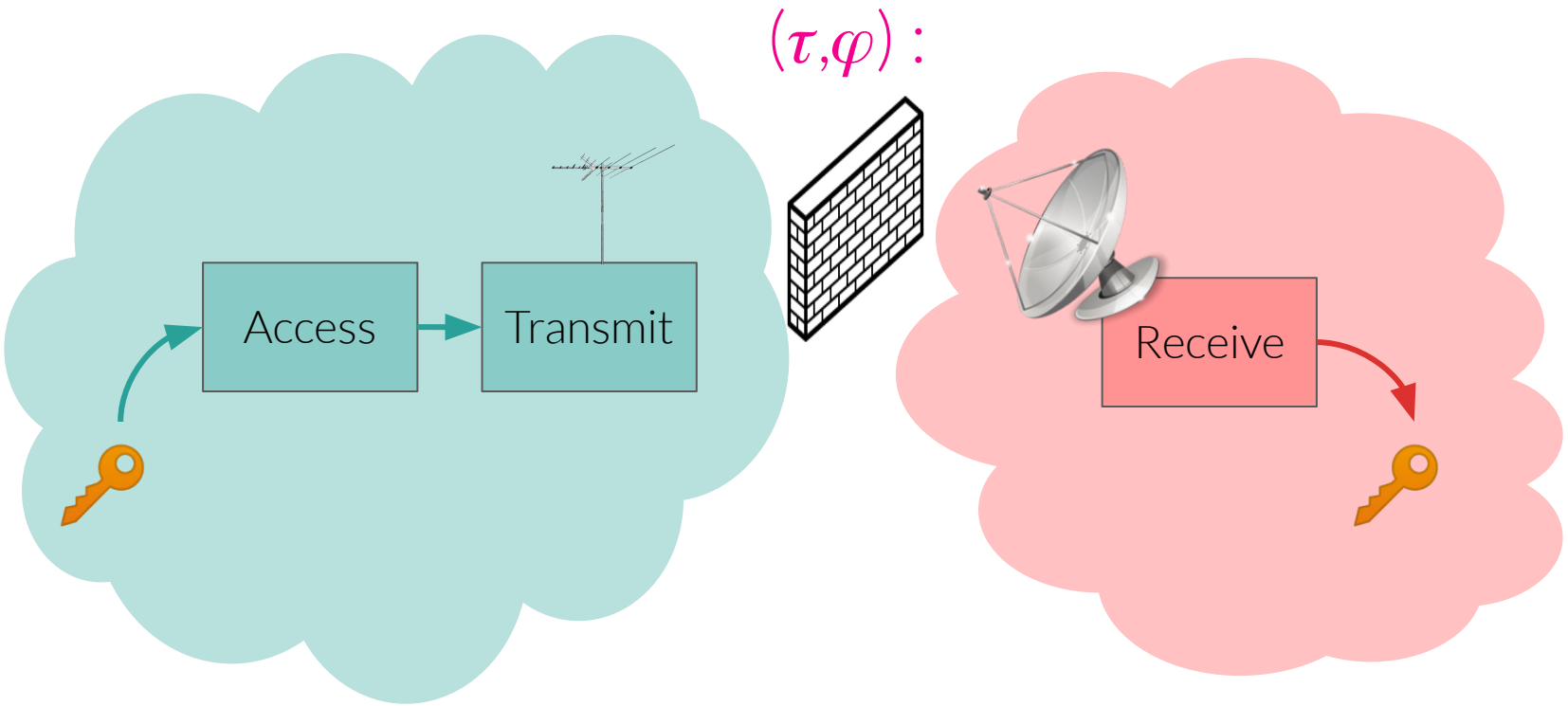




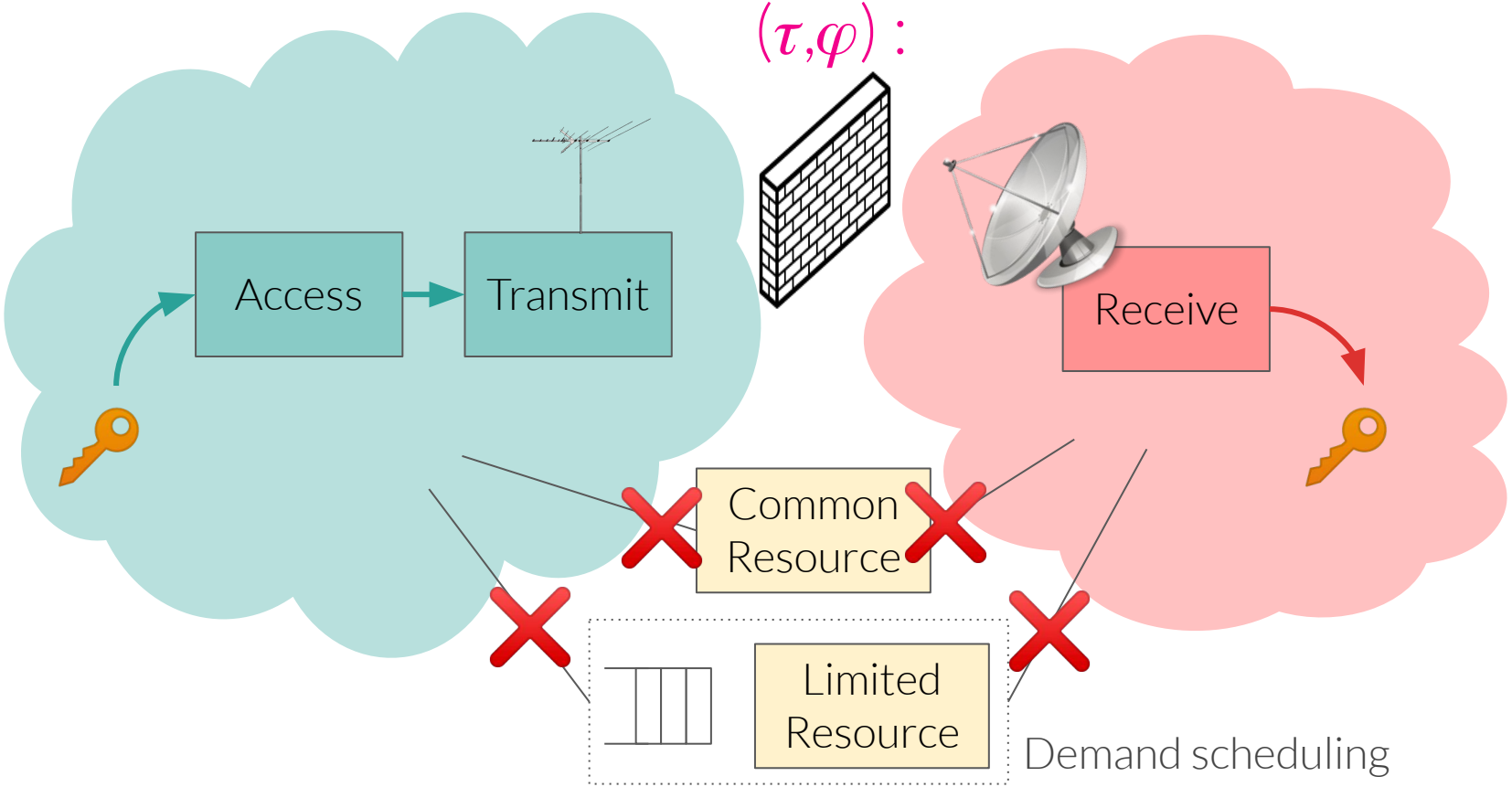
GENERALIZED SCHEMA FOR EXFILTRATION VIA A “SIDE CHANNEL”



A GENERALIZED DEFENSE FOR EXFILTRATION VIA A "SIDE CHANNEL"



ISOLATING MUTUALLY DISTRUSTING SOFTWARE



THREAT MODEL IN A MODERN SYSTEM

Linux > 36 M. LOC
>100 CVEs / year

φ^{OS}: see everything

τ^{OS}: change anything

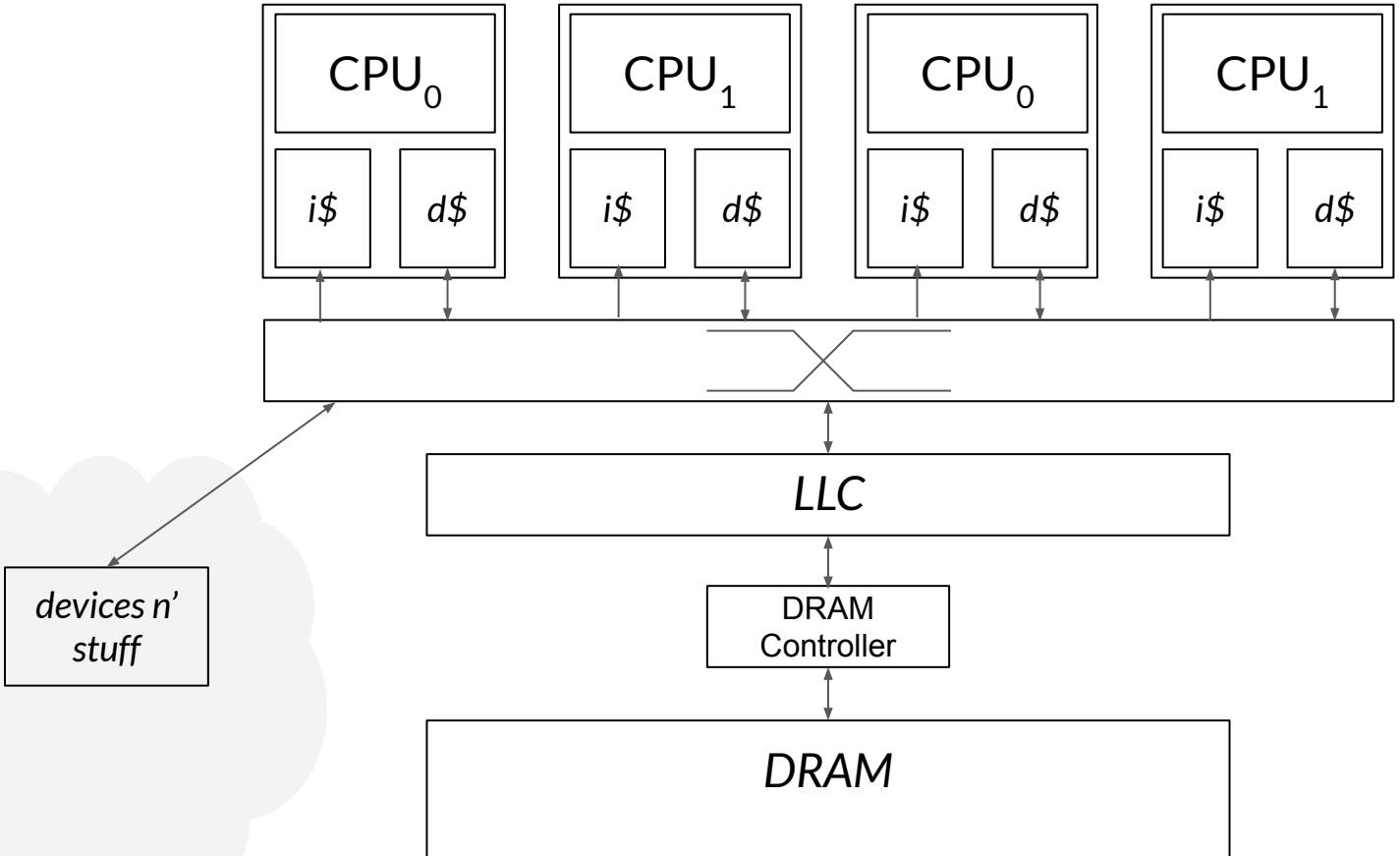
But the OS is trusted not to do this

The OS may be trusted but is it *trustworthy*?

φ^{PROC}: - see own state
- see shared state

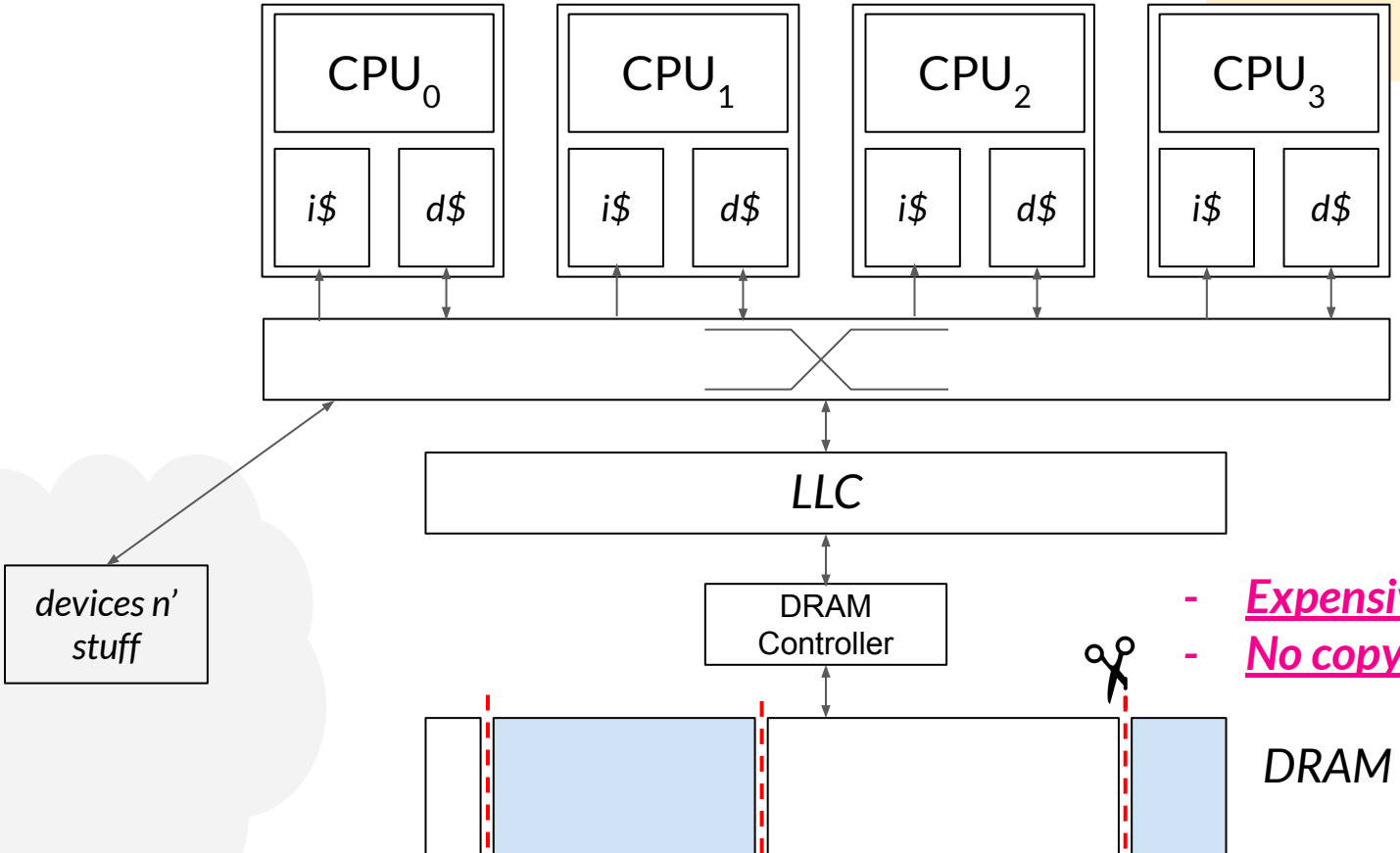
τ^{PROC}: change own state
- Change shared mem
- Syscalls to OS
- Use shared resources

PROTECTION DOMAINS (1/5)



PROTECTION DOMAINS (2/5)

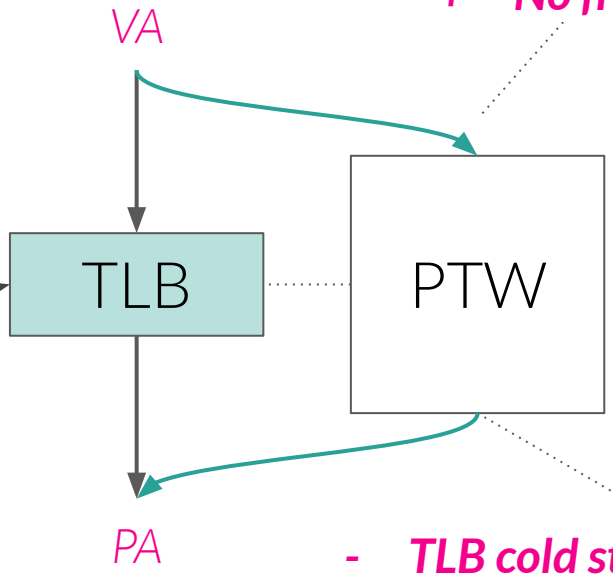
Protection domain := *isolated set of machine resources*



- ***Expensive I/O***
- ***No copy-on-write***

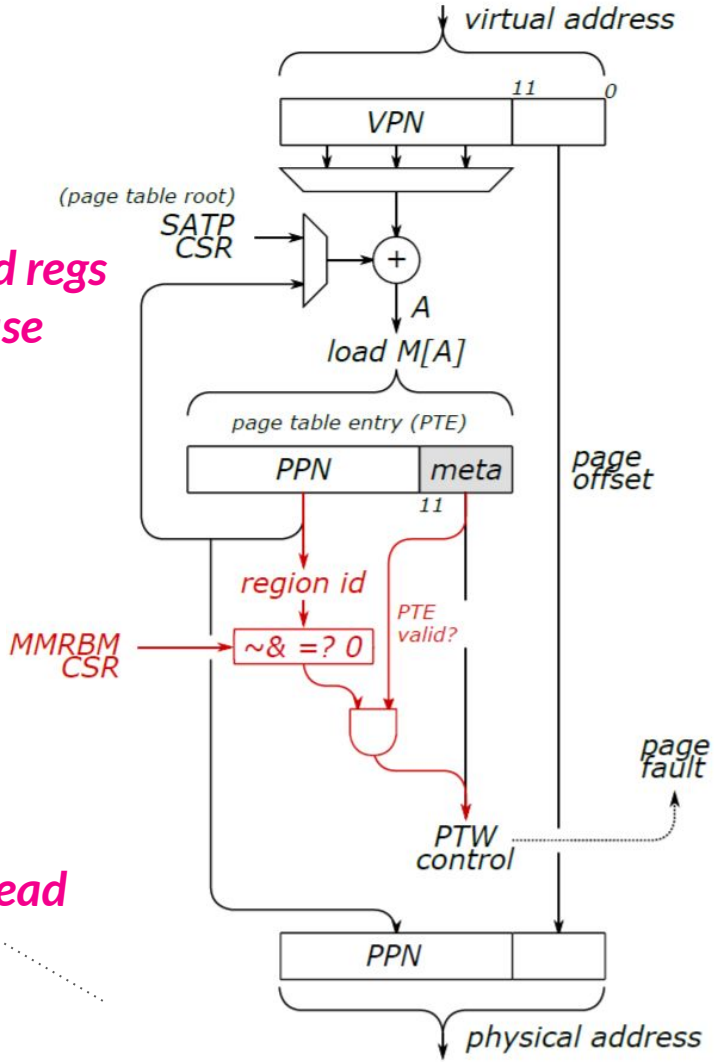
ASIDE : TLB INVARIANT

+ *A few gates and regs*
+ *No freq. decrease*

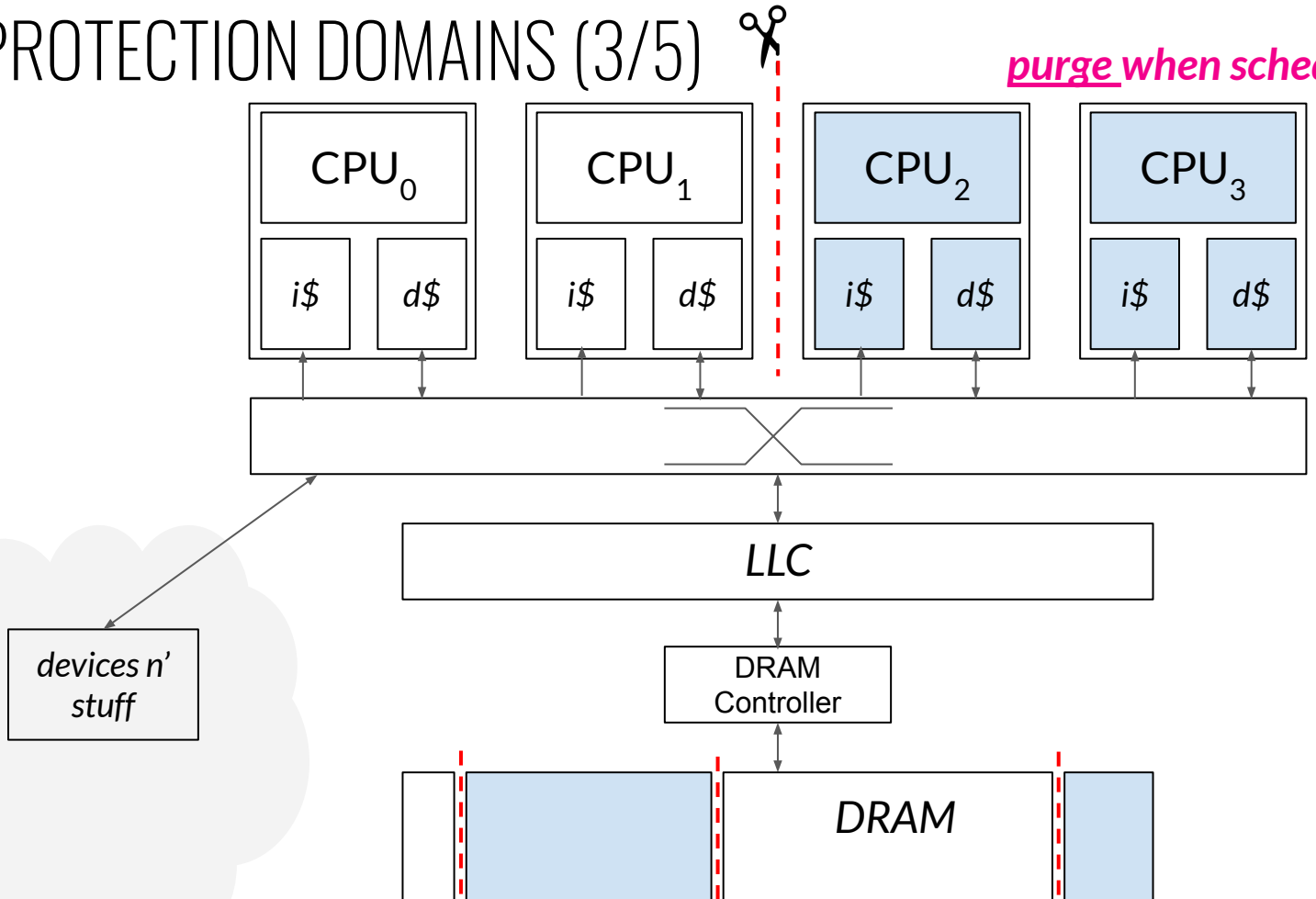


- *TLB cold start overhead*

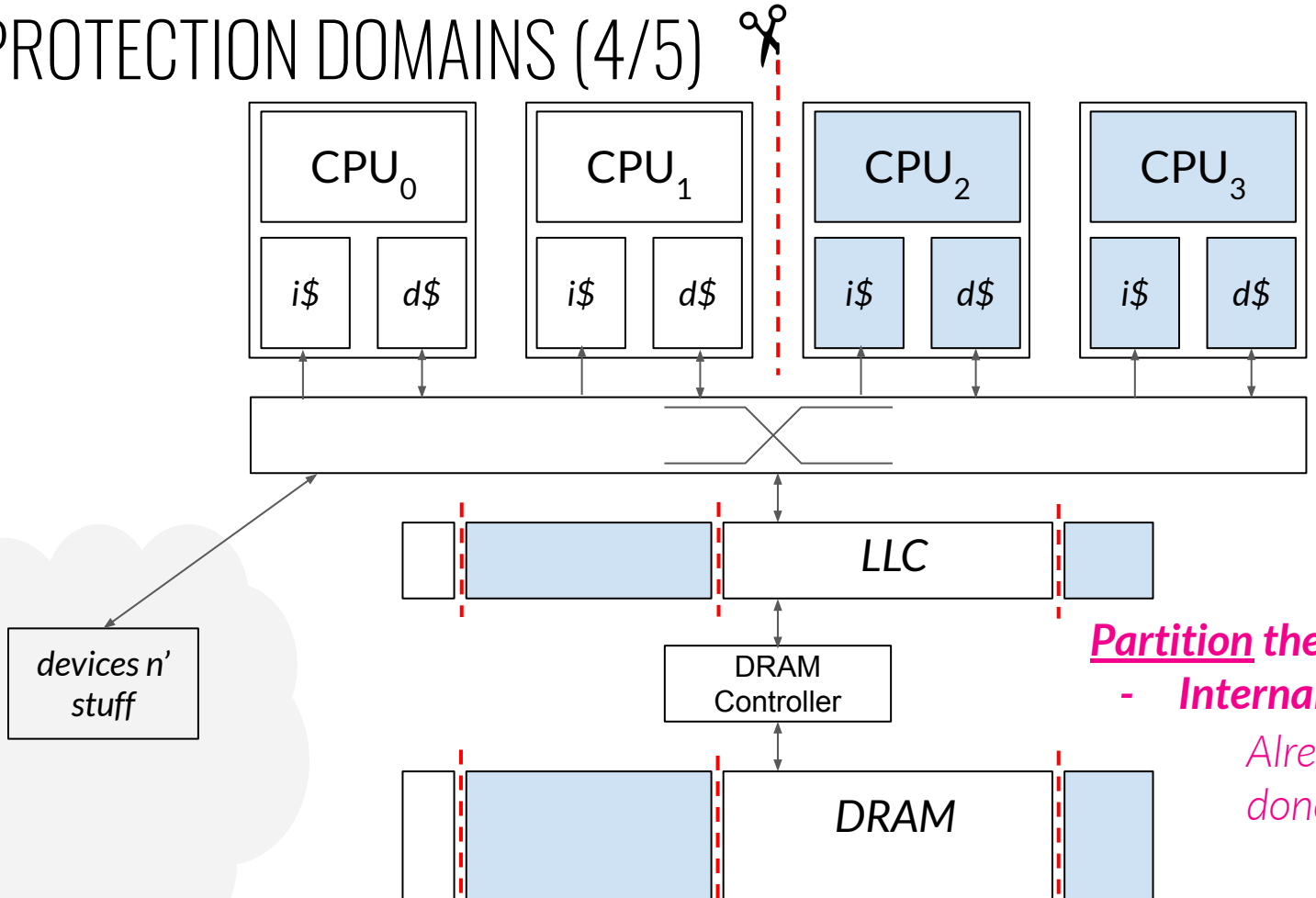
Remove cached translations when policy changes



PROTECTION DOMAINS (3/5)



PROTECTION DOMAINS (4/5)

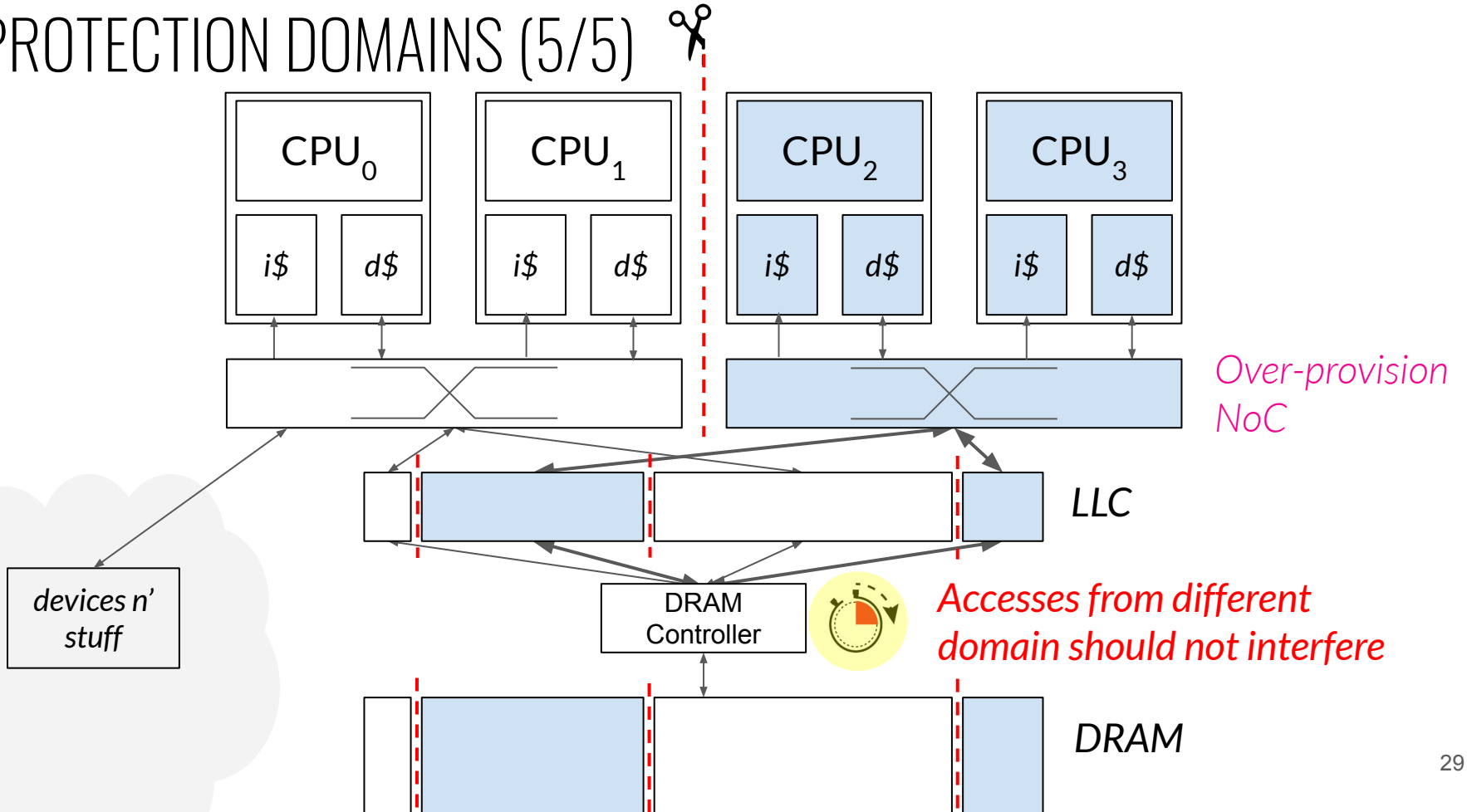


***Partition* the LLC**

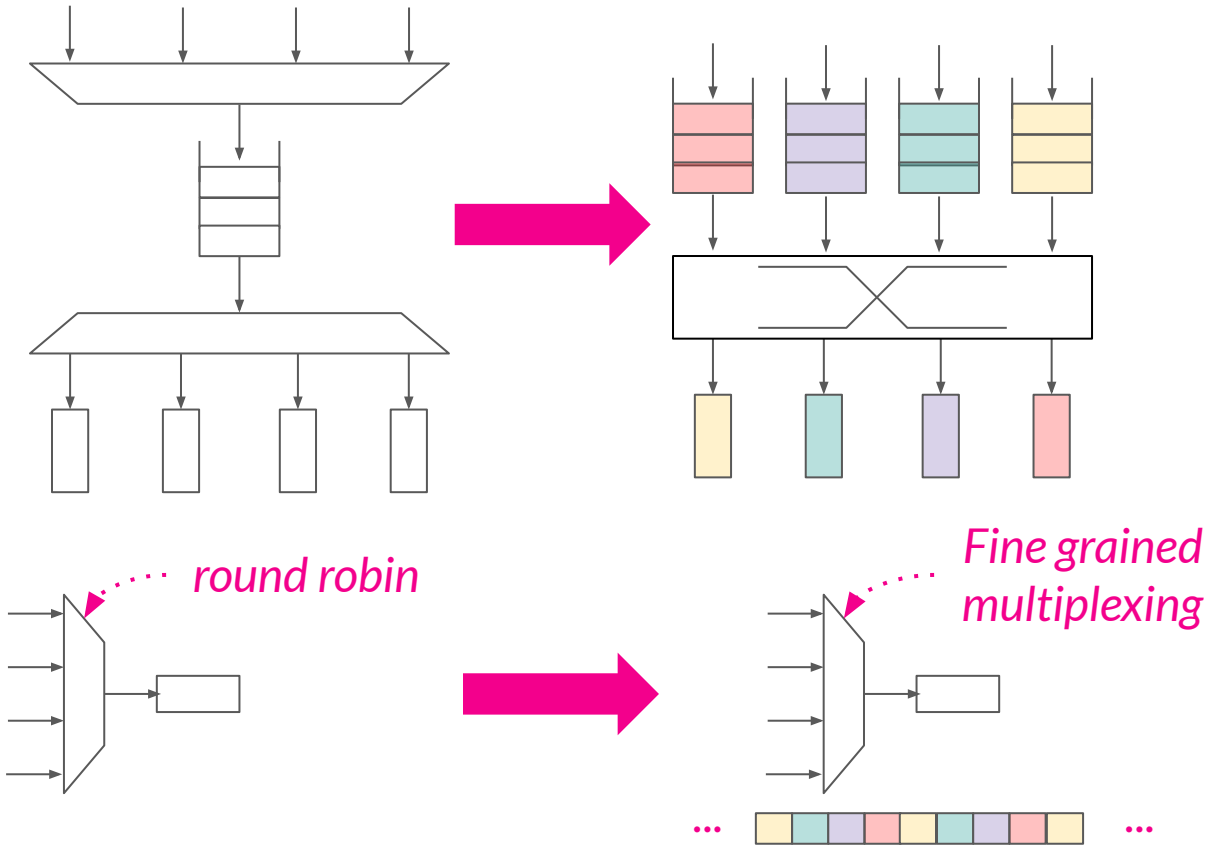
- ***Internal fragmentation***

Already often done for QoS

PROTECTION DOMAINS (5/5)

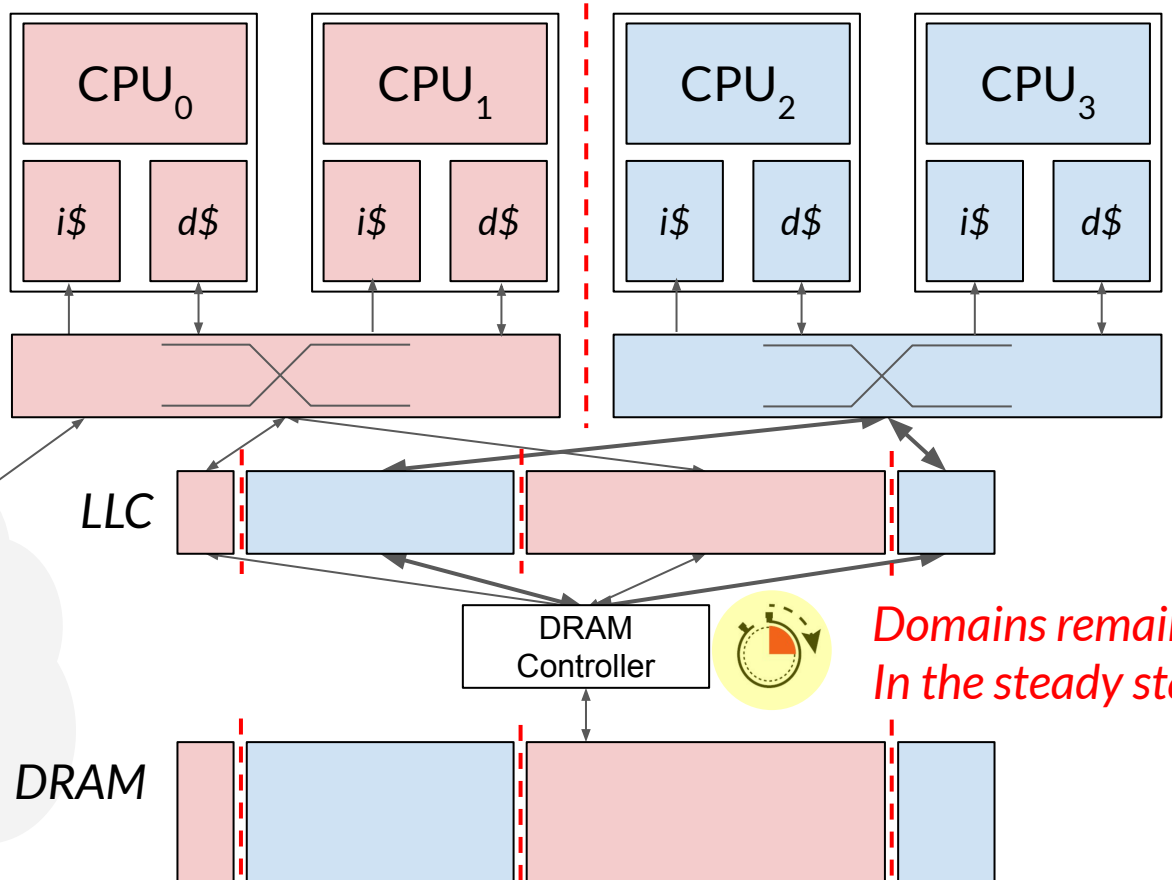


ASIDE : PARTITIONING ARBITERS / NETWORKS



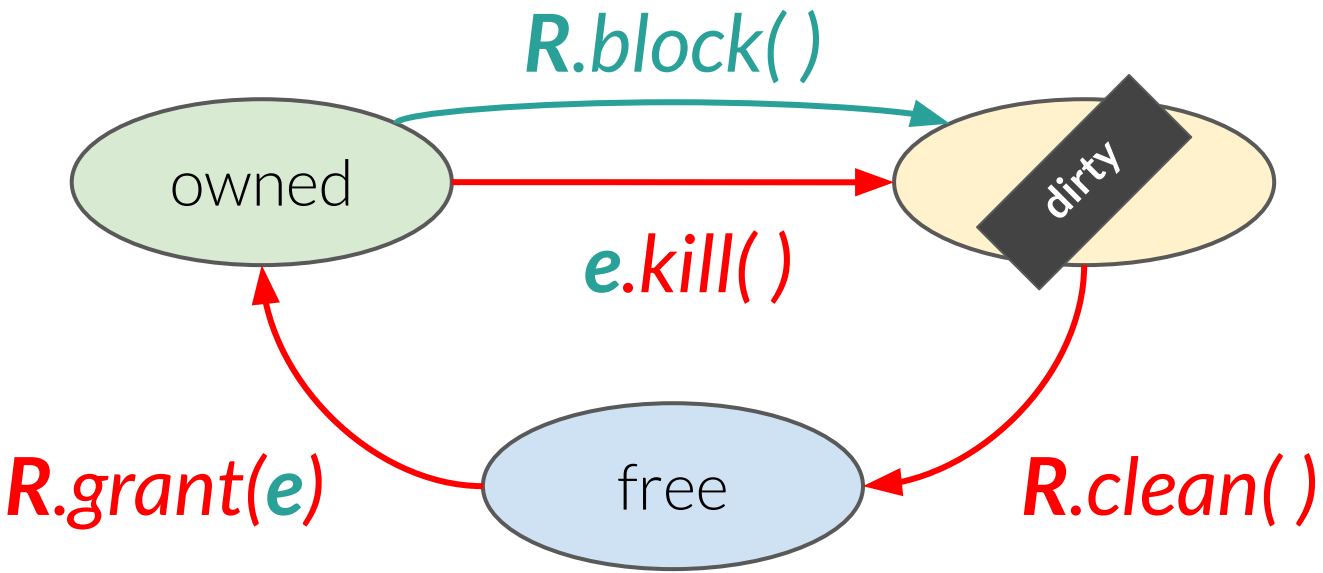
NON-INTERFERENCE BETWEEN PROTECTION DOMAINS

(τ, φ)

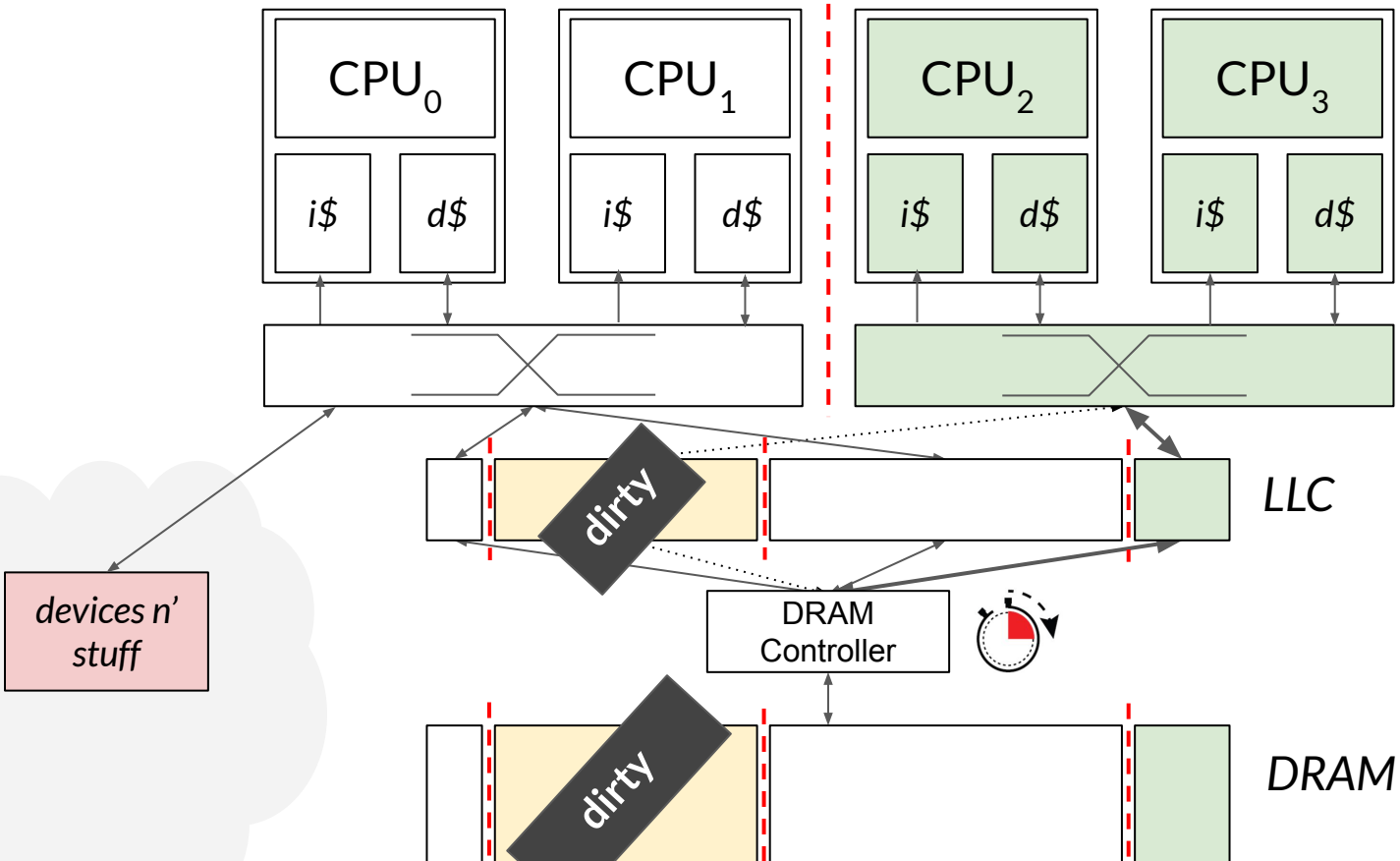


Domains remain isolated
In the steady state,

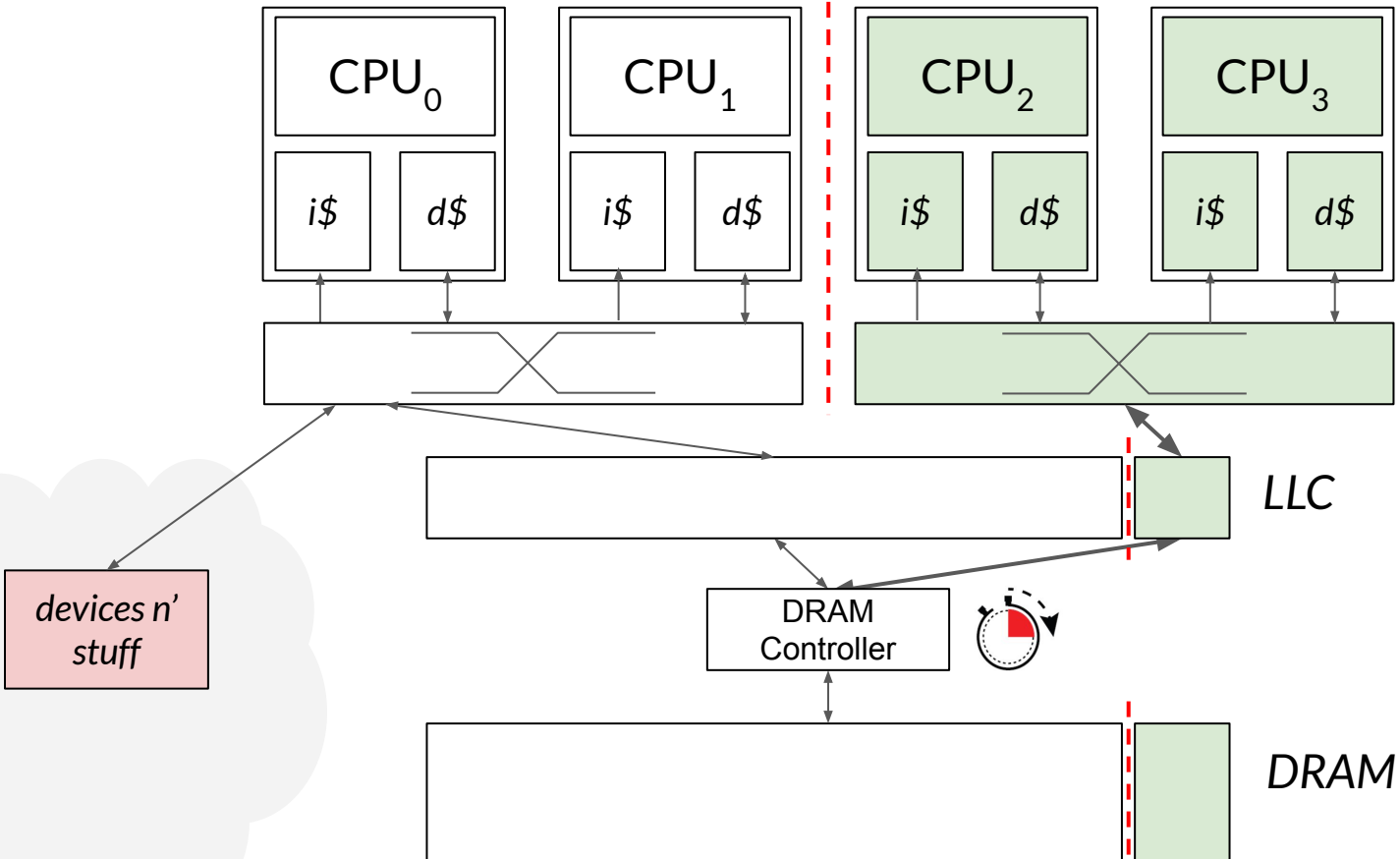
PROTECTION DOMAINS ARE NOT STATIC



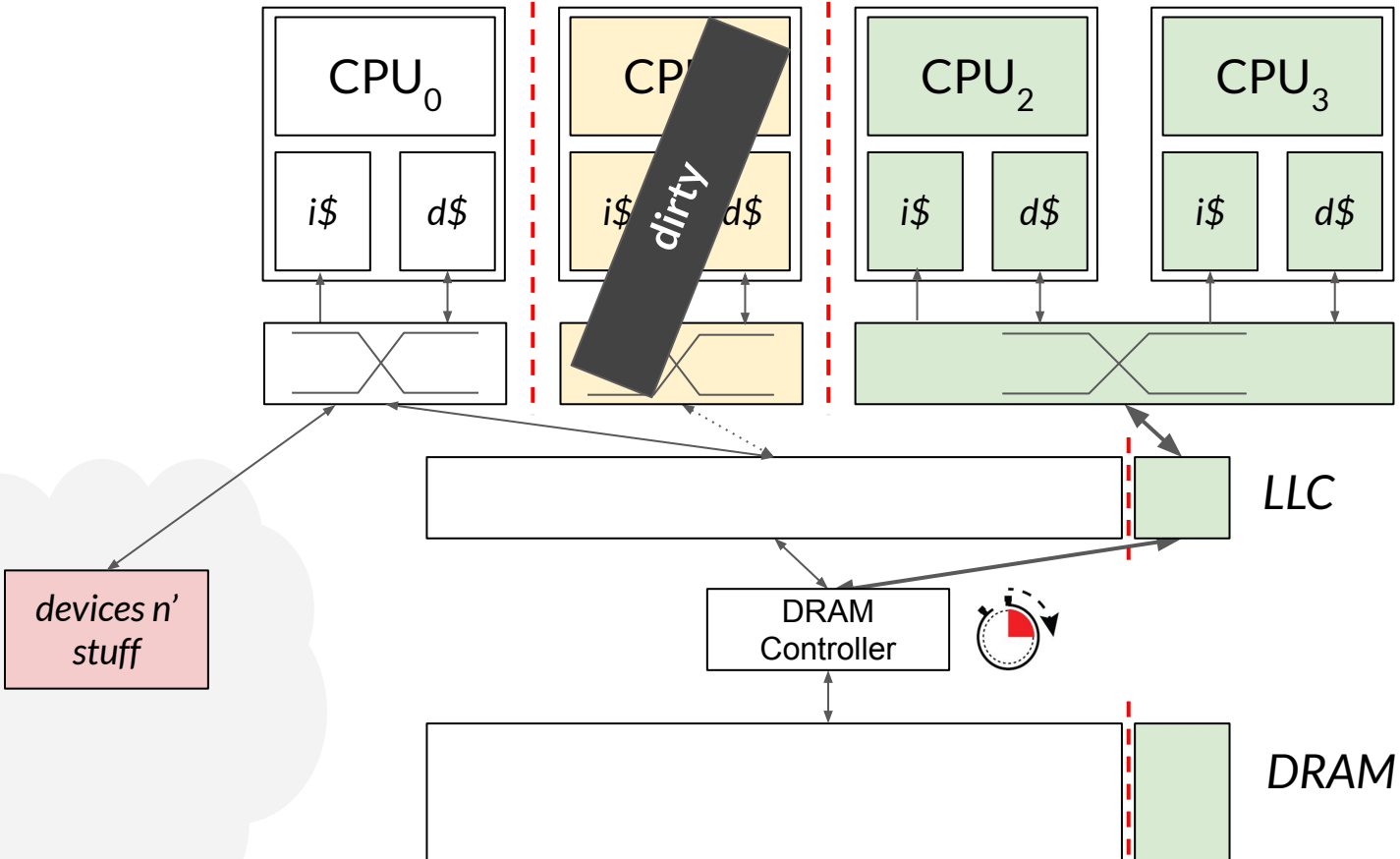
PROTECTION DOMAINS TRANSITIONS (1/4)



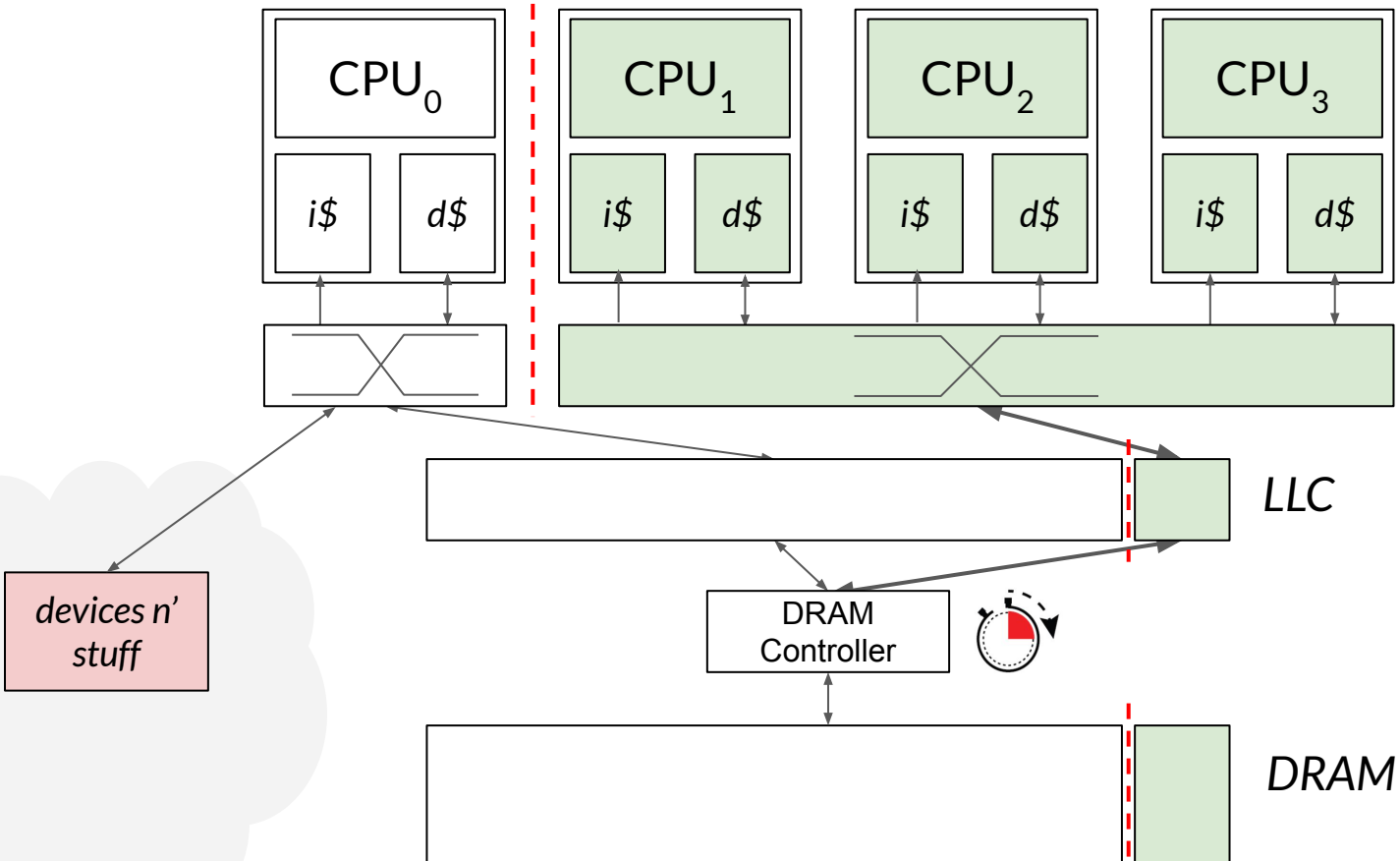
PROTECTION DOMAINS TRANSITIONS (2/4)



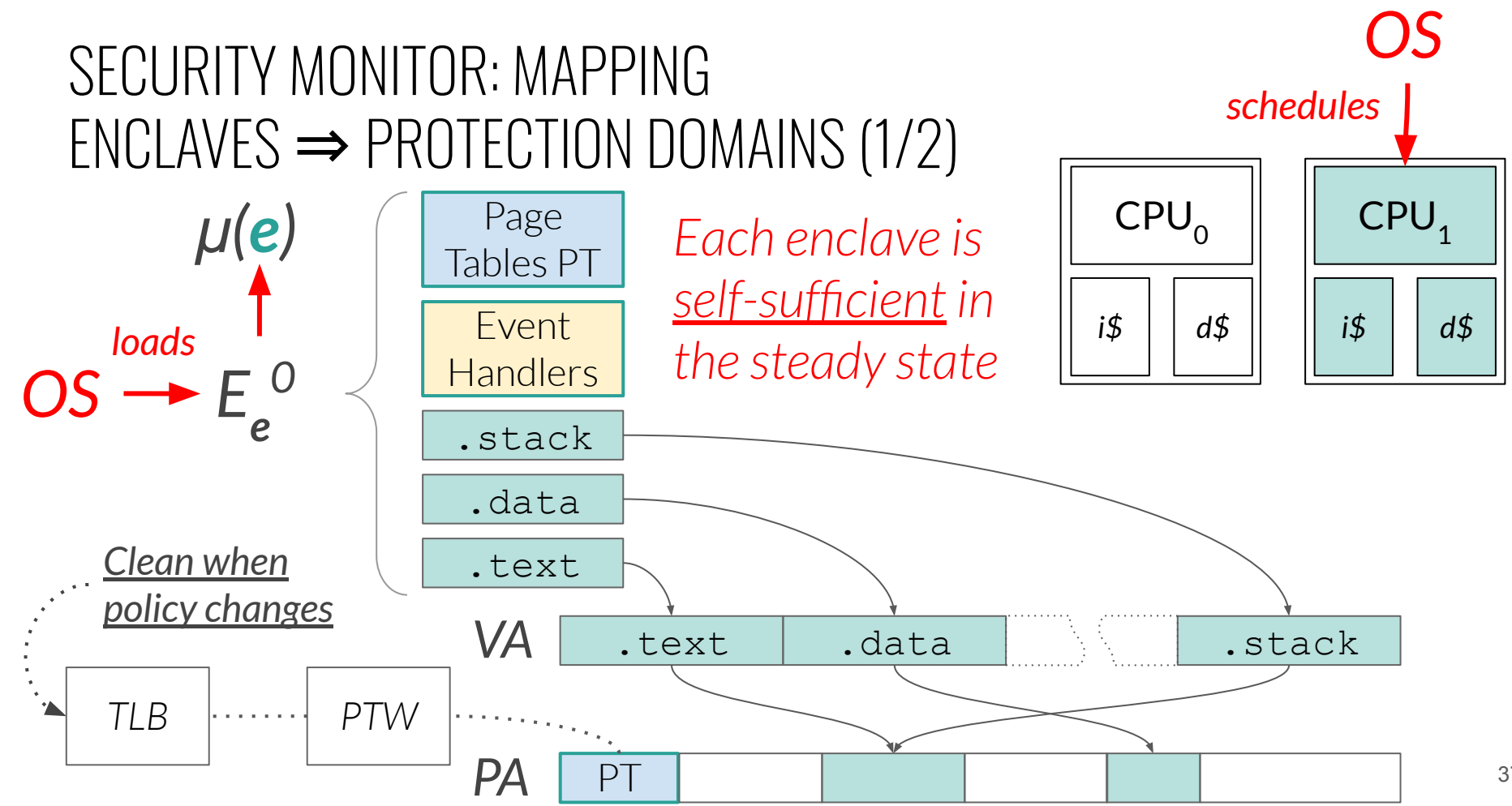
PROTECTION DOMAINS TRANSITIONS (3/4)



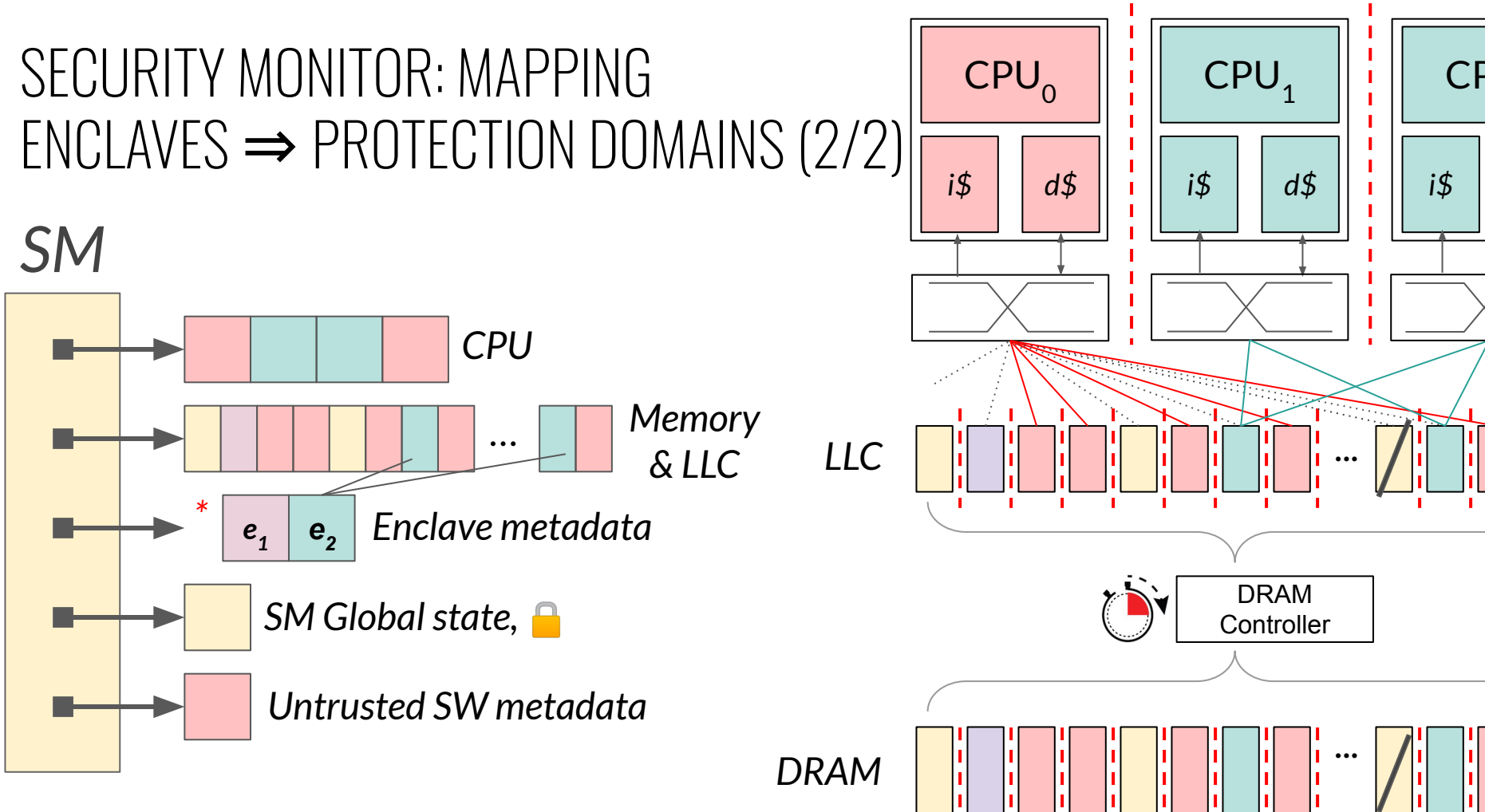
PROTECTION DOMAINS TRANSITIONS (4/4)



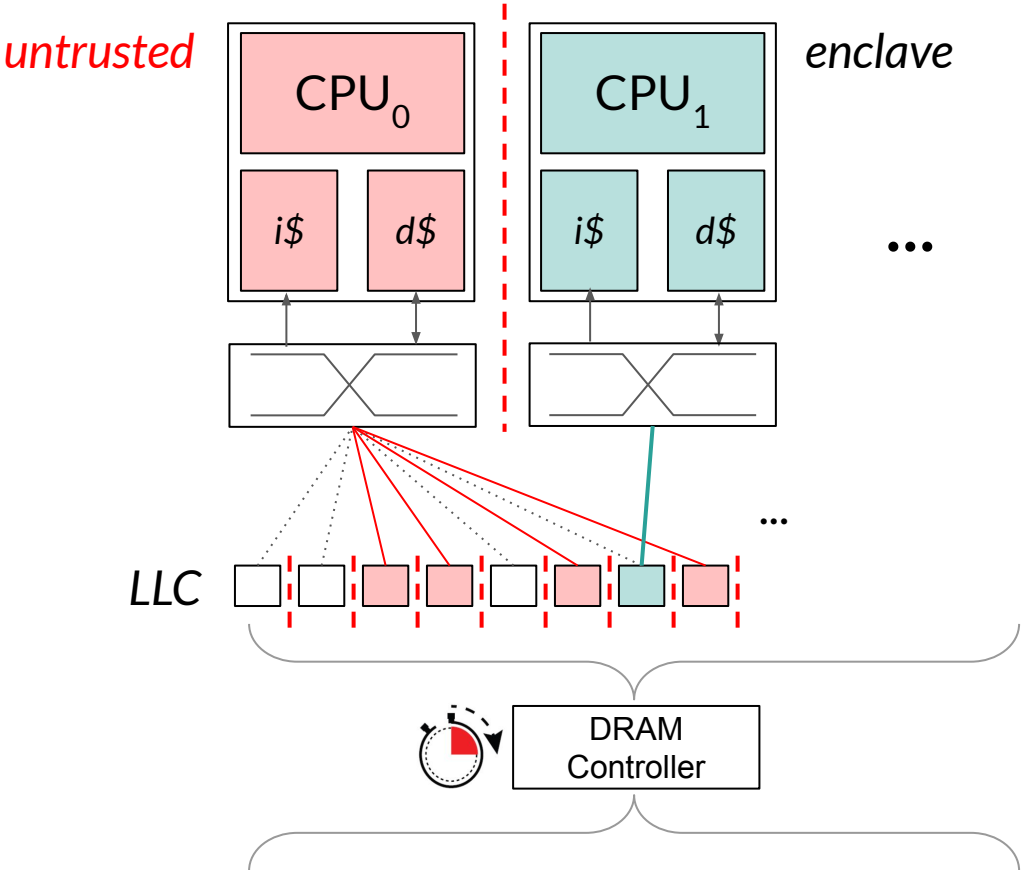
SECURITY MONITOR: MAPPING ENCLAVES ⇒ PROTECTION DOMAINS (1/2)



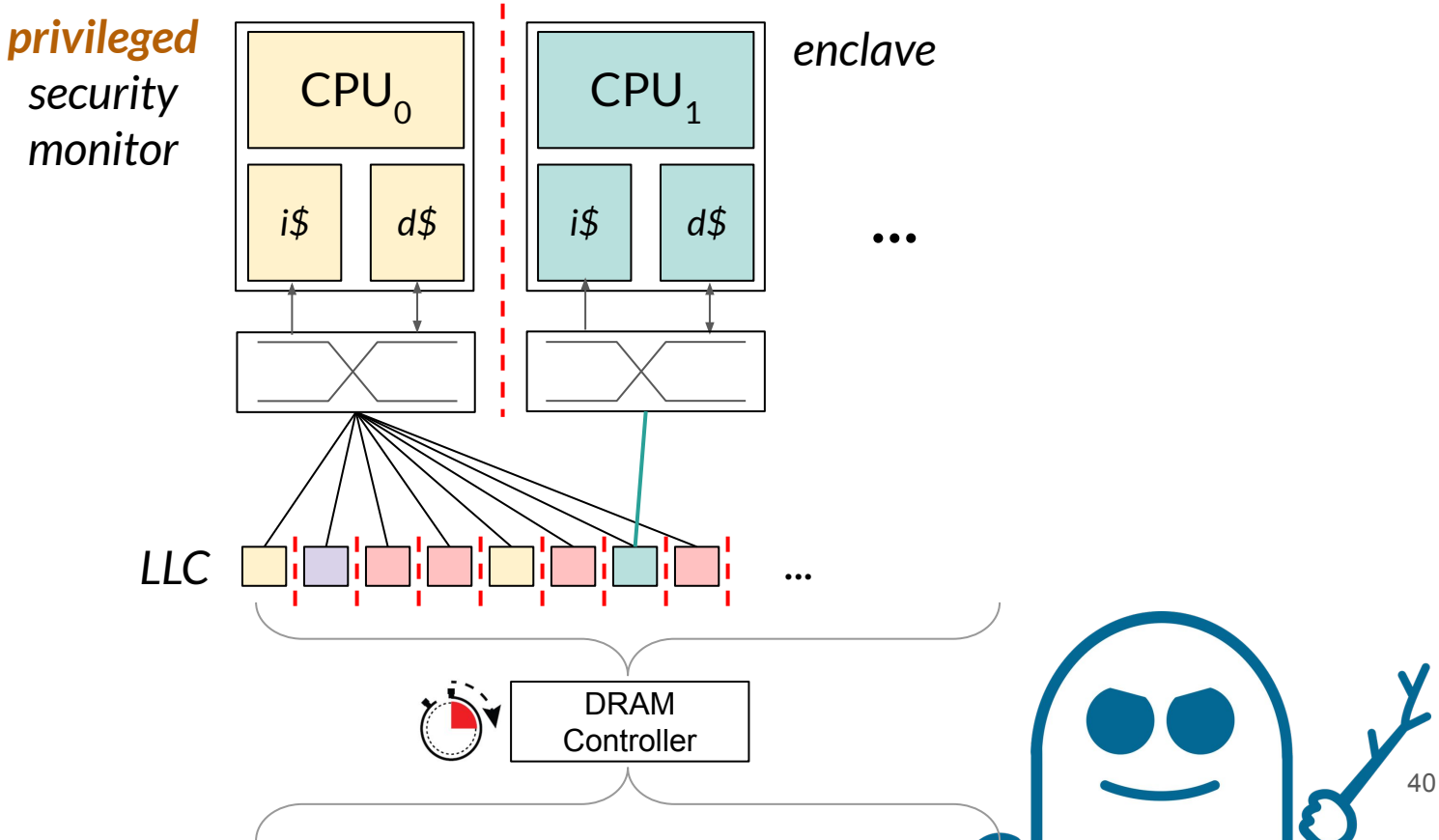
SECURITY MONITOR: MAPPING ENCLAVES ⇒ PROTECTION DOMAINS (2/2)



SPECULATIVE EXECUTION + PROTECTION DOMAINS = ? (1/2)



SPECULATIVE EXECUTION + PROTECTION DOMAINS = ? (2/2)



SPECULATIVE EXECUTION + PROTECTION DOMAINS = ? (3/3)

e / A ⇒ *isolated*

- TLB Invariant
- $ Partitioning
- NoC Partitions
- Exclusive memory

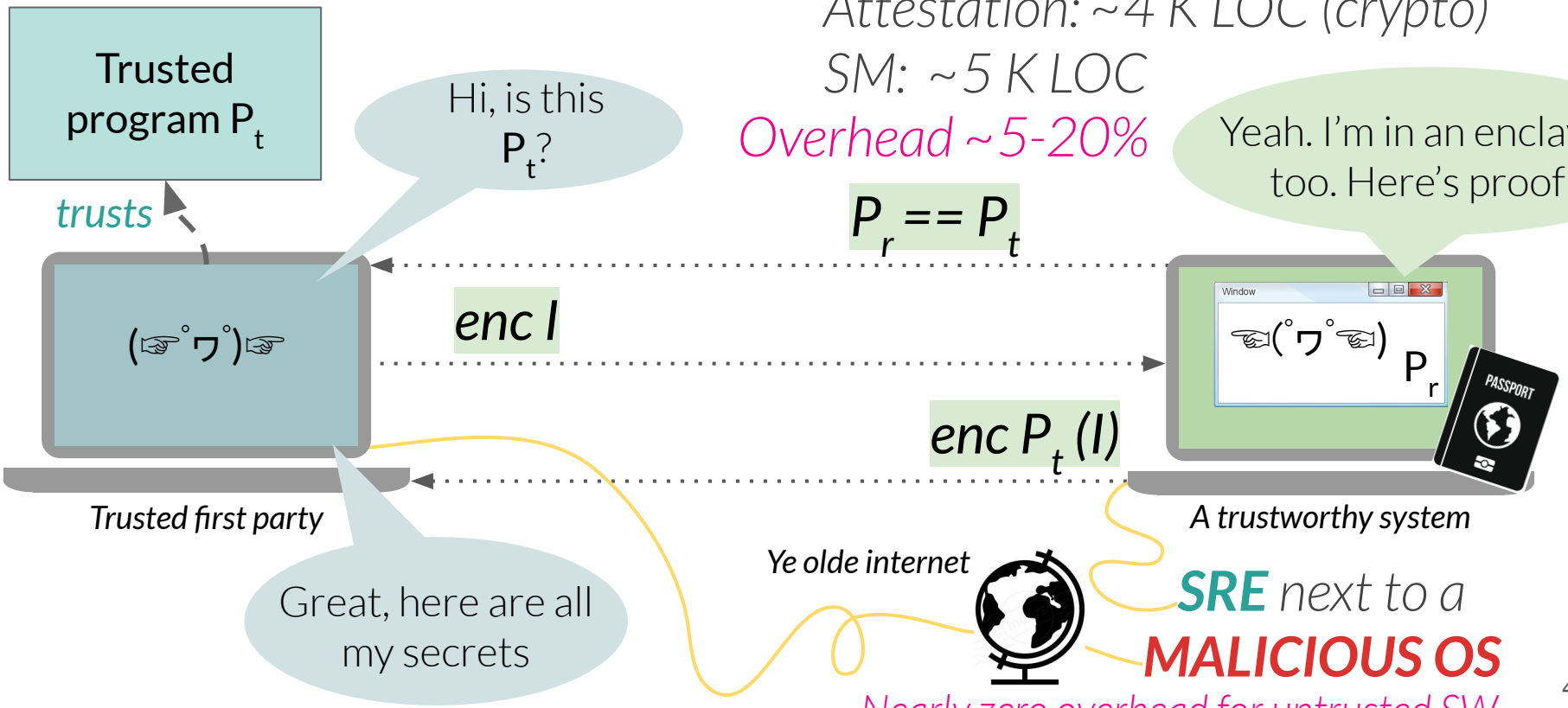
safe to speculate freely

SM ⇒ *not isolated*

- Only fetch in SM
- Serialize every load instruction

disable speculation

PUTTING IT ALL TOGETHER



TRUSTED CODE:

Witness: ~4K LOC (200 + crypto)

Attestation: ~4 K LOC (crypto)

SM: ~5 K LOC

Overhead ~5-20%

***SRE** next to a*

MALICIOUS OS

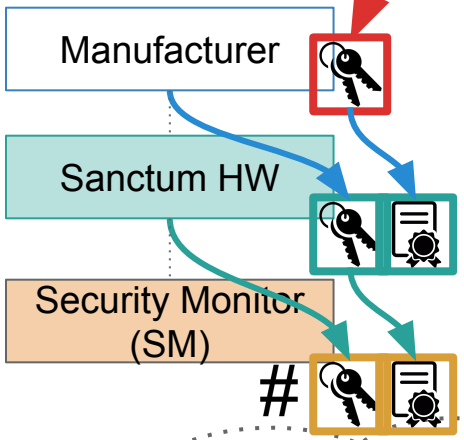
Nearly zero overhead for untrusted SW

THIS PAGE IS INTENTIONALLY LEFT BLANK

AUTHENTICATED KEY AGREEMENT

Diffie Hellman to establish a private channel with remote enclave
(discrete log crypto, or elliptic curve where $\{g^A, g^B\} \rightarrow G^{AB}$ is hard.)

Private key corresponding to a well-known public key



Sanctum computer

Remote user

Select *primes* **p, g.**

Generate random **A**

Compute $(g^A \bmod p)$

Send
$p, g,$
$(g^A \bmod p)$

Does remote user trust **metadata**, ?

Send
M

Generate random **B**, compute g^B
M = {g^A, g^B, **metadata**, }, signed with with *.

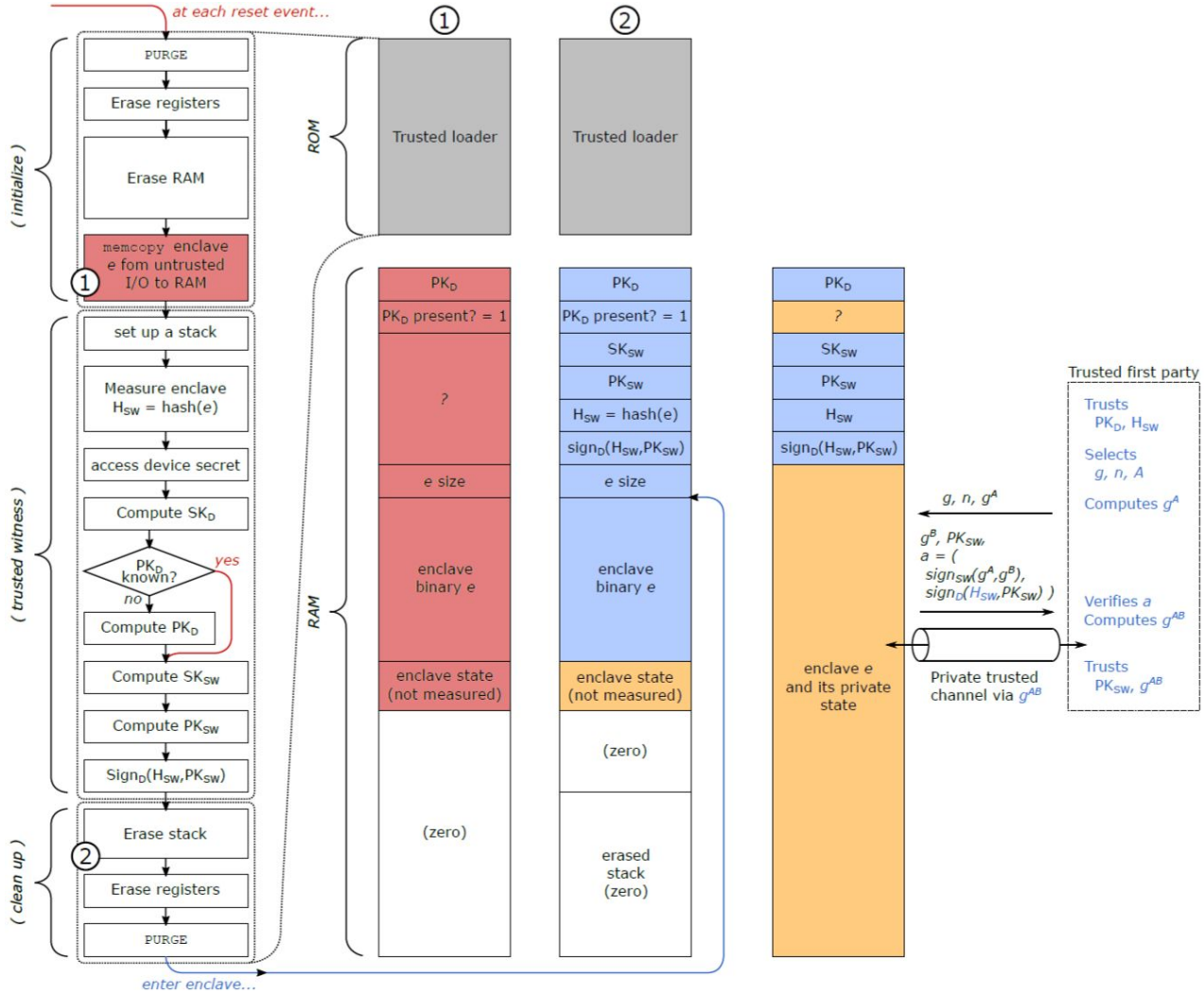
enclave

Compute symmetric key
K = $(g^B)^A \bmod p$

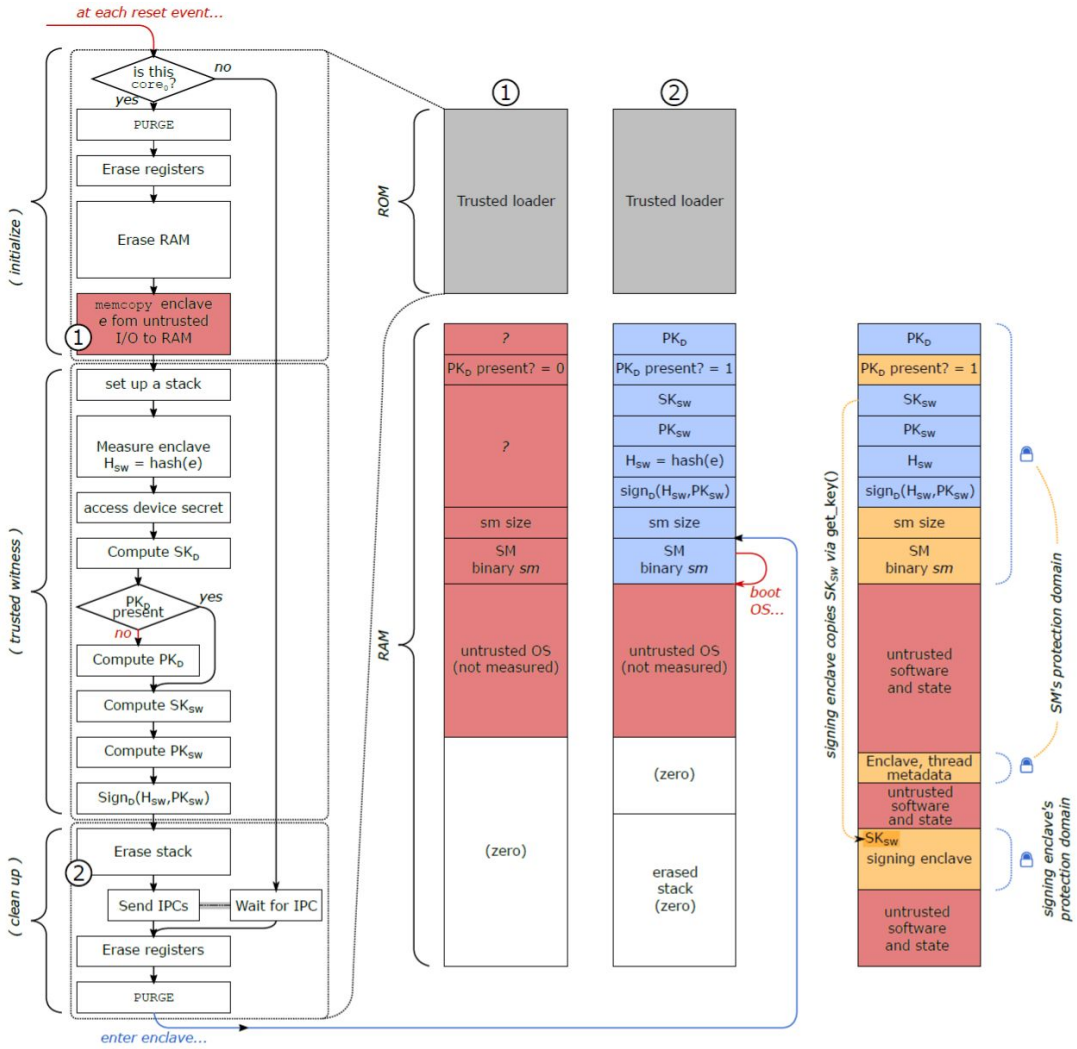
Both parties now share a secret key: K

Compute symmetric key
K = $(g^A)^B \bmod p$

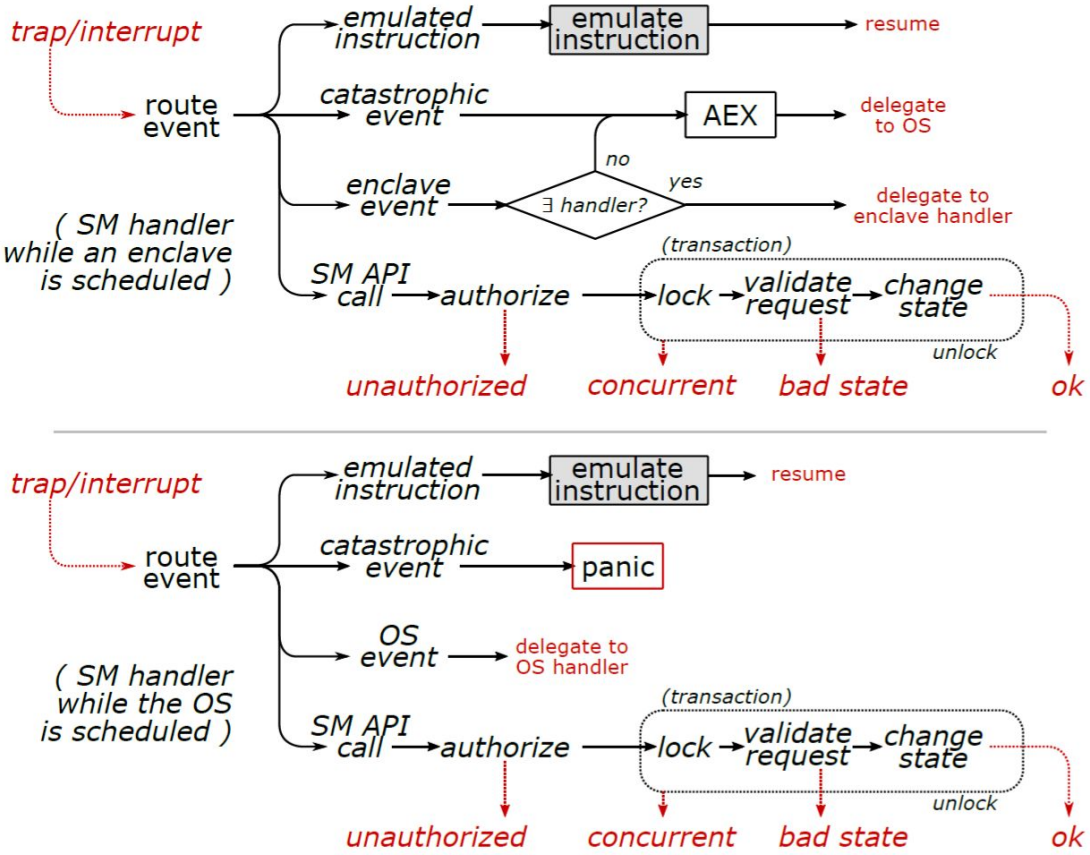
EMBEDDED ENCLAVE



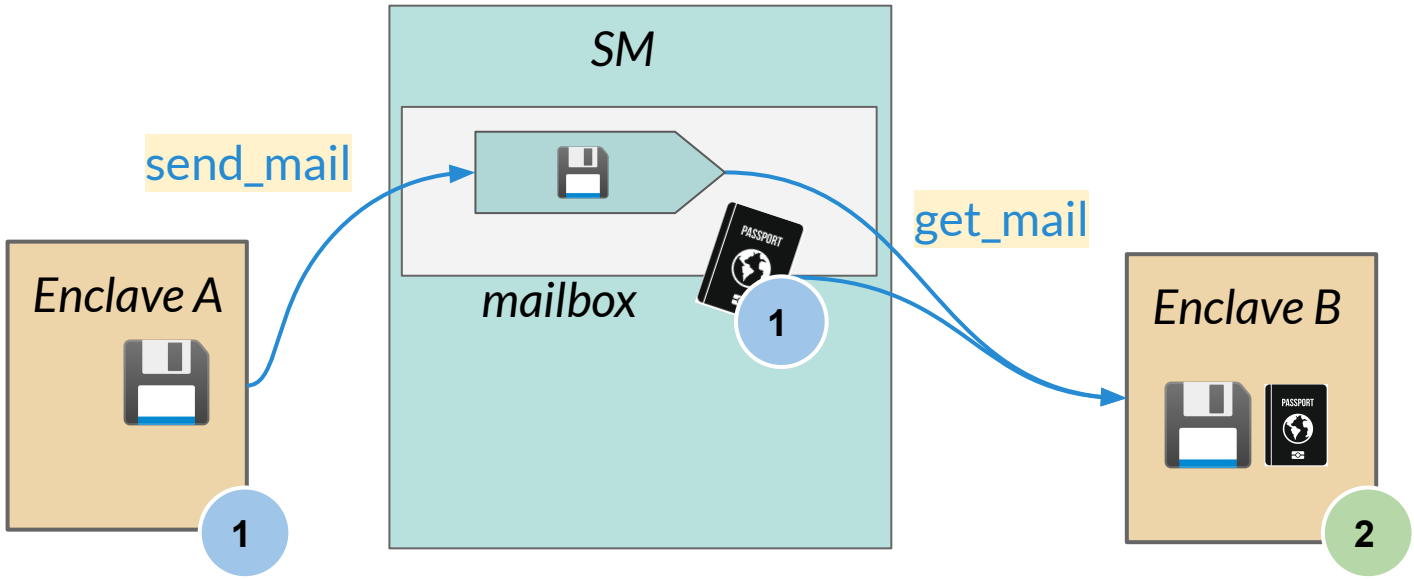
SM LOADED AT BOOT



SECURITY MONITOR COMMANDS

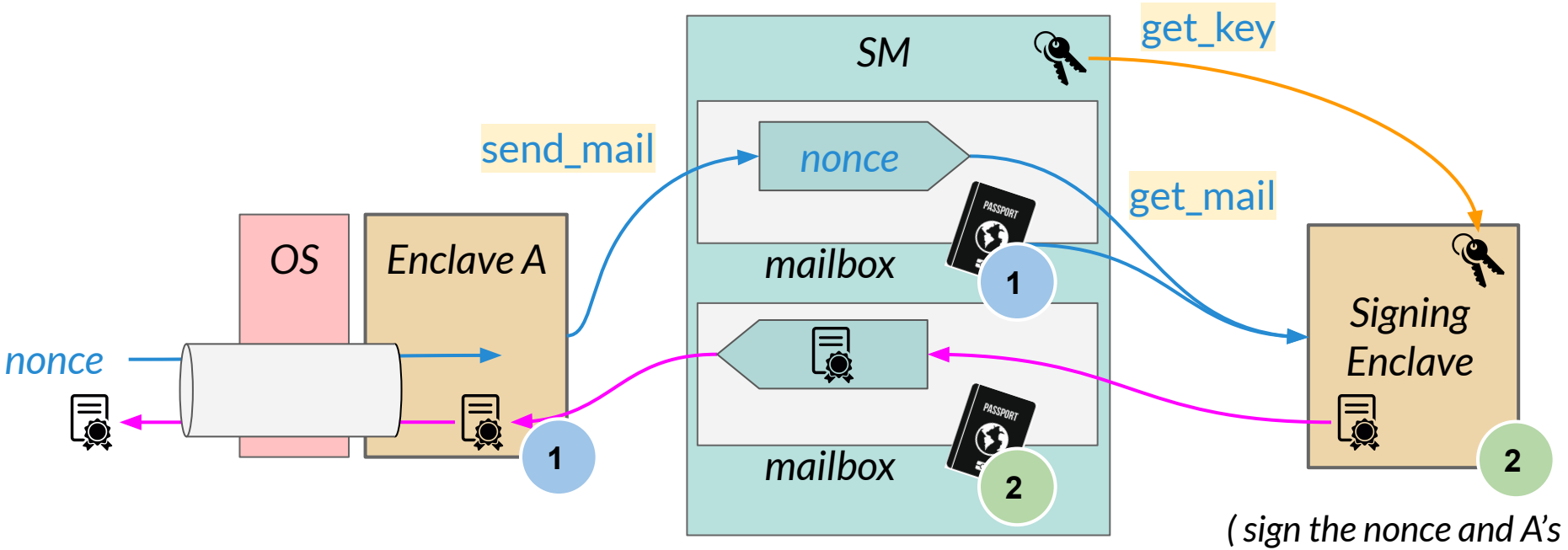


LOCAL ATTESTATION (1/2)



*(The SM's **authority** allows for local attestation*
***without cryptographic signatures**)*

LOCAL ATTESTATION (2/2)



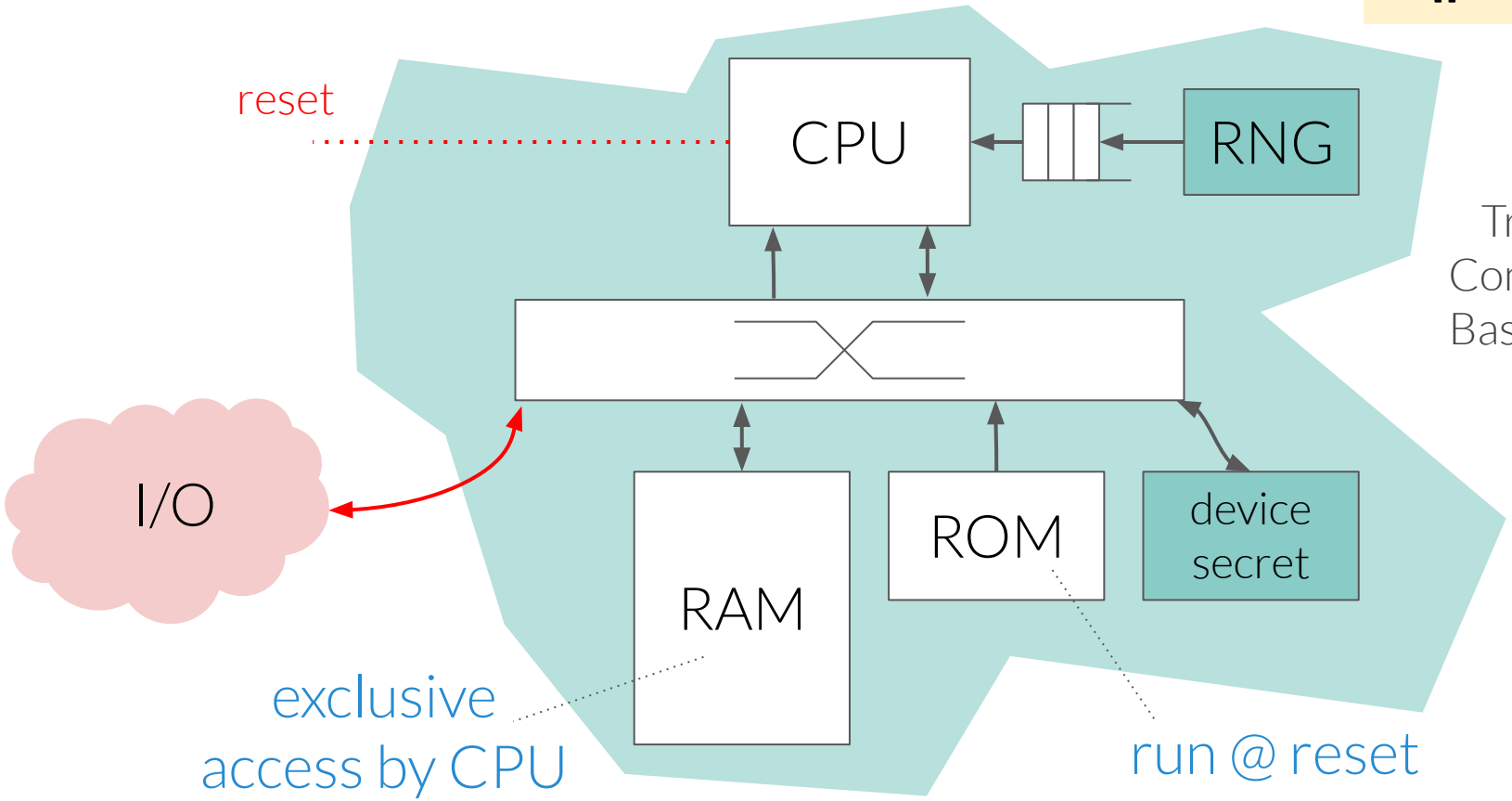
(Use a special enclave for attestation → SM does not require rigorous privacy)

(sign the nonce and A's measurement)

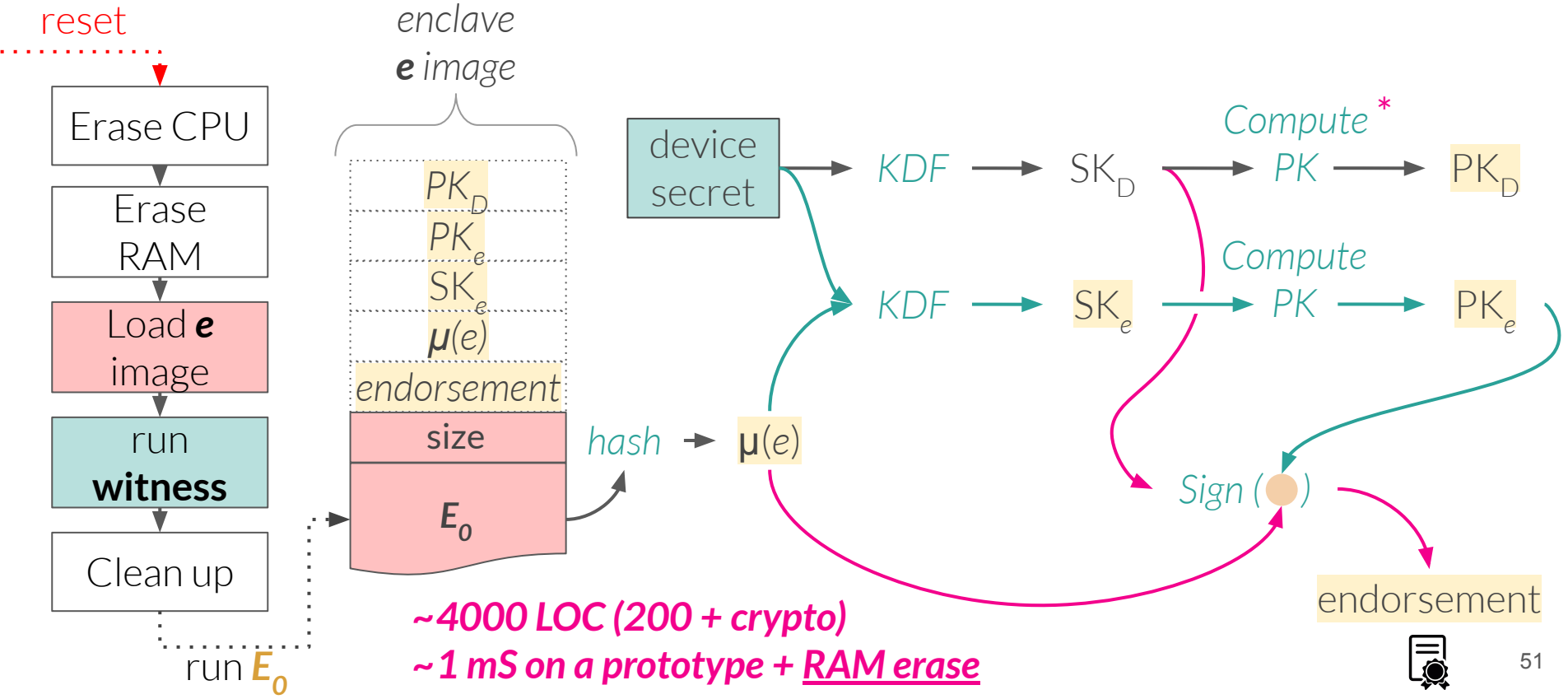
“THE WITNESS” : IMPLEMENTING MEASUREMENT (1/2)

This looks like a toy, but:

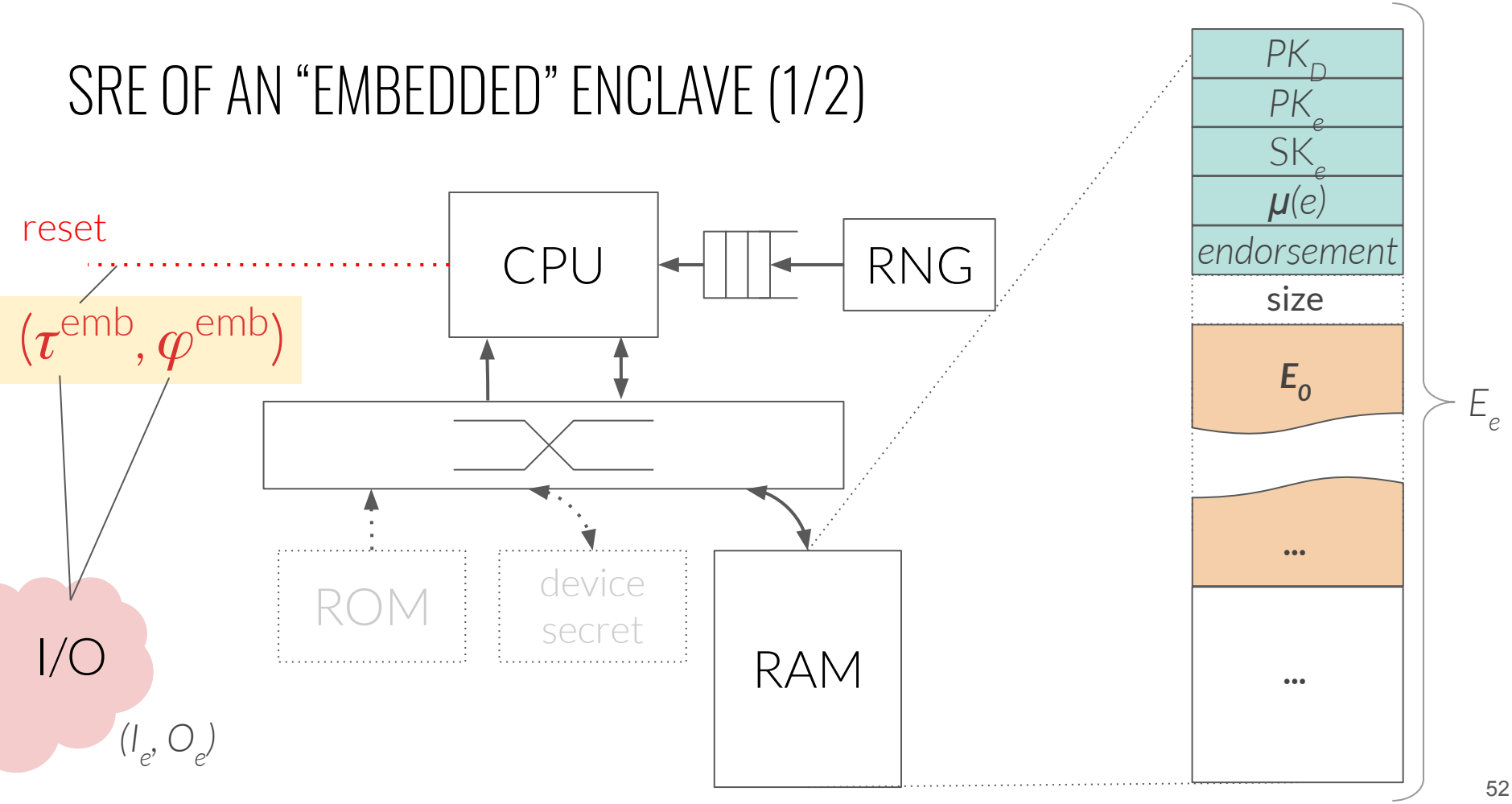
opentitan



"THE WITNESS" : IMPLEMENTING MEASUREMENT (2/2)

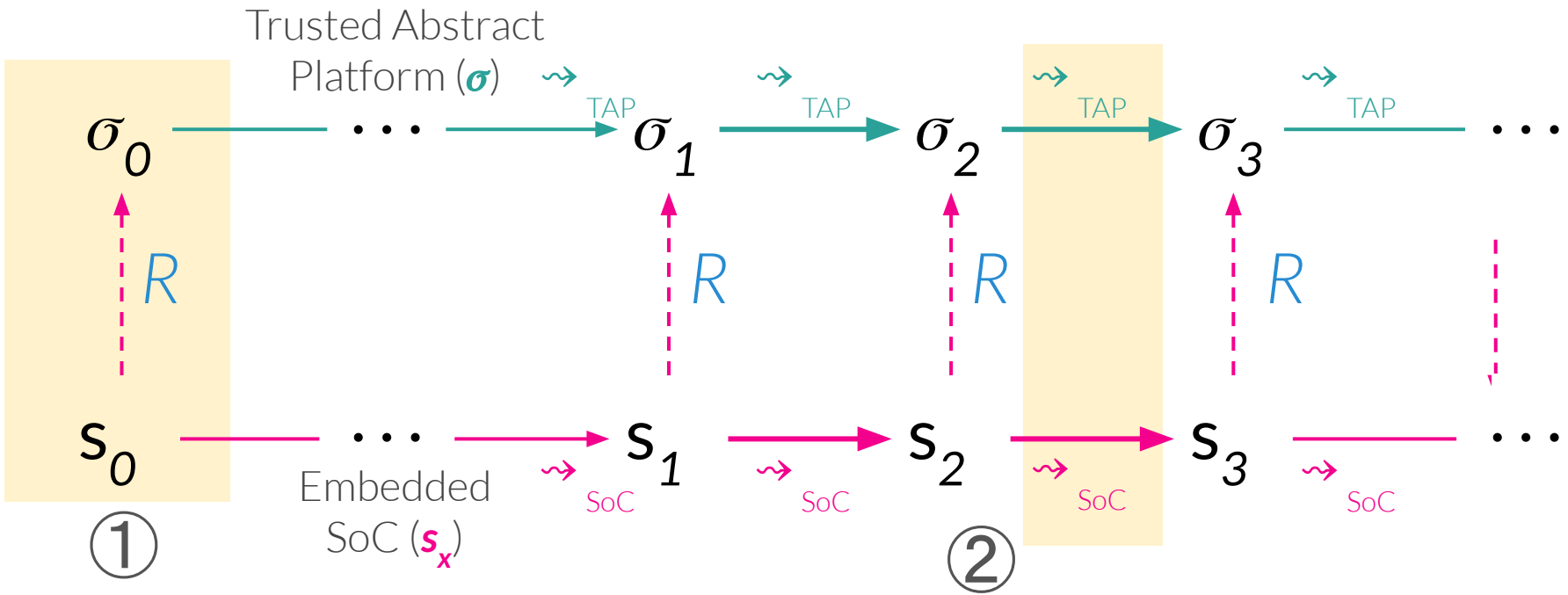


SRE OF AN "EMBEDDED" ENCLAVE (1/2)



SRE OF AN "EMBEDDED" ENCLAVE (2/2)

"REFINEMENT"



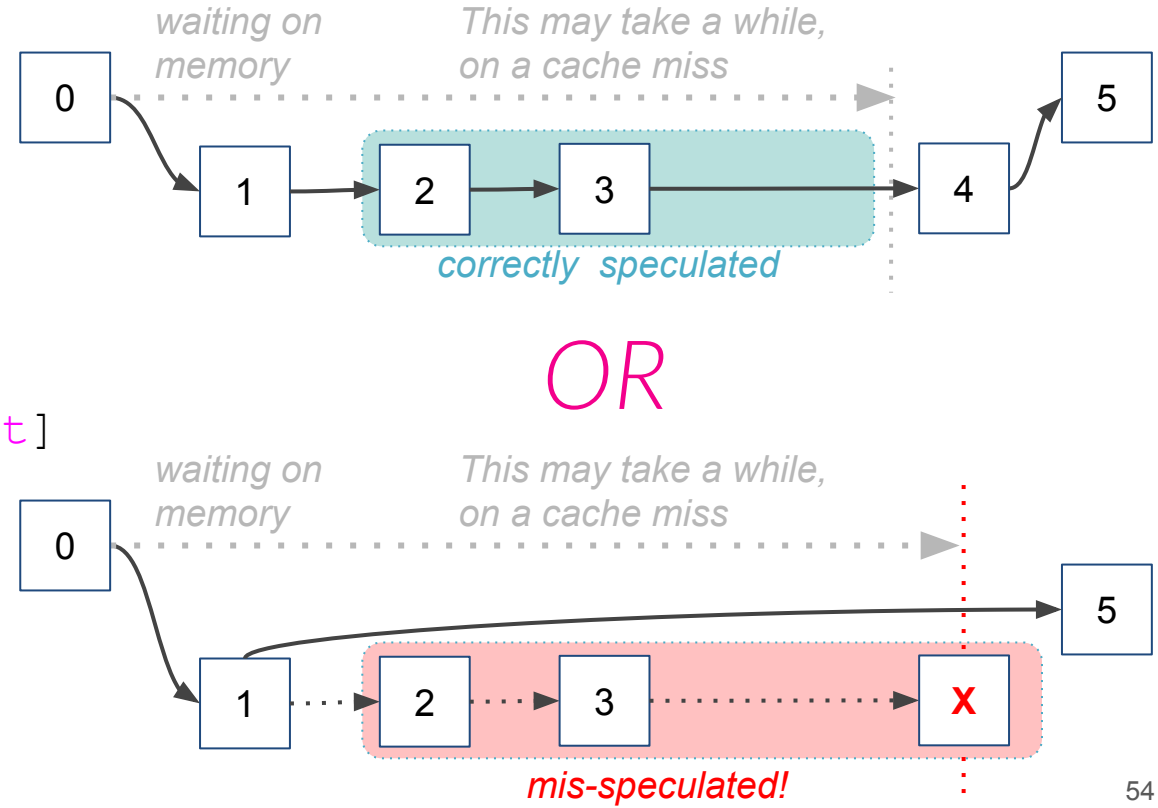
$s_0 \xrightarrow{R} \sigma_0$

$\rightsquigarrow_{SoC} \xrightarrow{R} \{\rightsquigarrow_{TAP}, \emptyset\}$

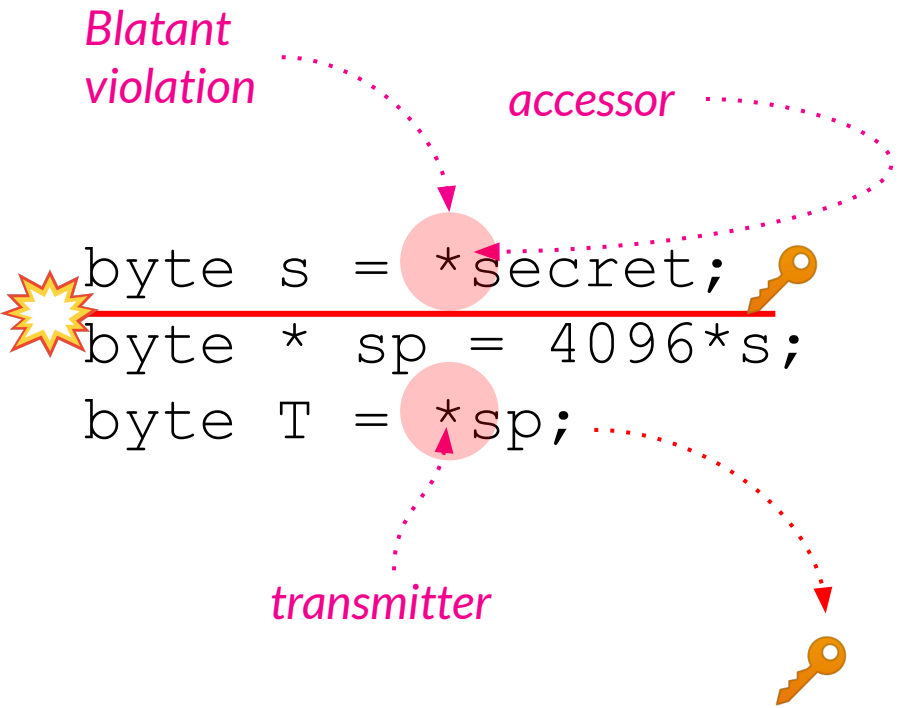
SPECULATION UNDERMINES INTEGRITY (1/4)

consider this (*very*) hypothetical program:

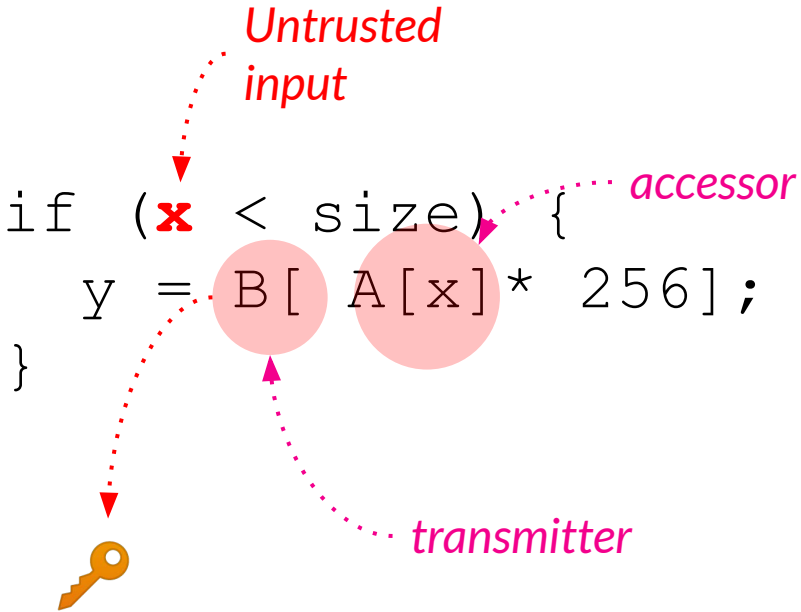
```
0: a ← memory[ptr*]
1: if a != 0, goto 5
2:    secret ← 42
3:    a ← memory[secret]
4:    a ← a+1
5: lfence
wait ·····
   ...
```



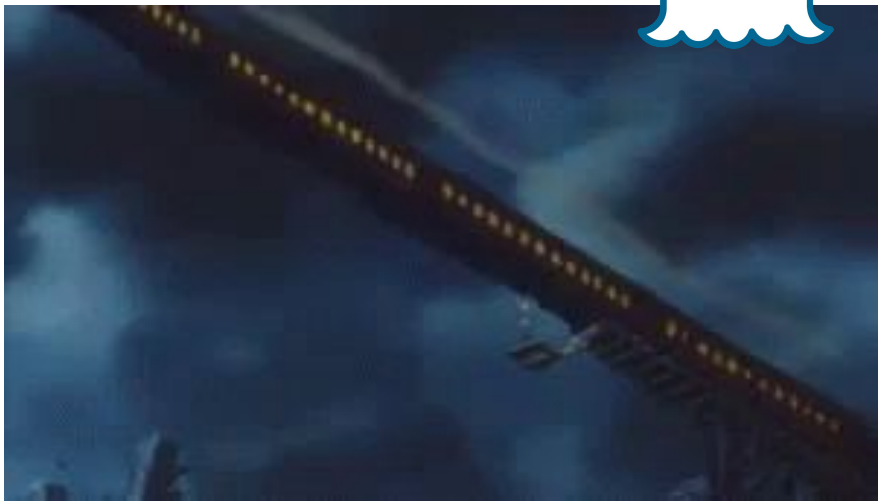
SPECULATION UNDERMINES INTEGRITY (2/4)



SPECULATION UNDERMINES INTEGRITY (3/4)



SPECULATION UNDERMINES INTEGRITY (4/4)



```
// program
// with an
indirect_branch()
// keeps on
// going
```

Mis-predict target

```
// attacker-owned snippet
```

accessor

transmitter