Cryptographically Attested Secure Hardware

Ilia Lebedev, Christian Wentz
CSAIL@MIT and gradient.tech

Hi!

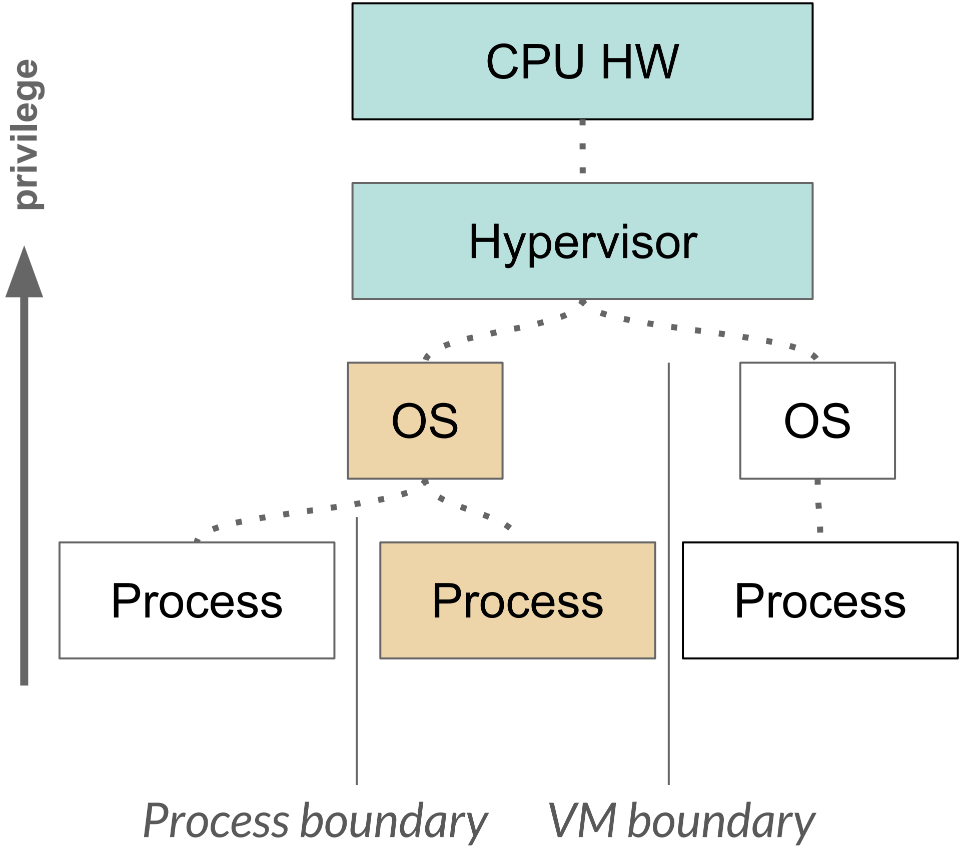
My name is Ilia, let’s small talk:
what are some of your hopes and dreams?

I dream we stop using the word “secure” without context (secure against ___)

Today, privilege implies trust (1/3)

If computing remotely, what is the TCB?

trusted computing base

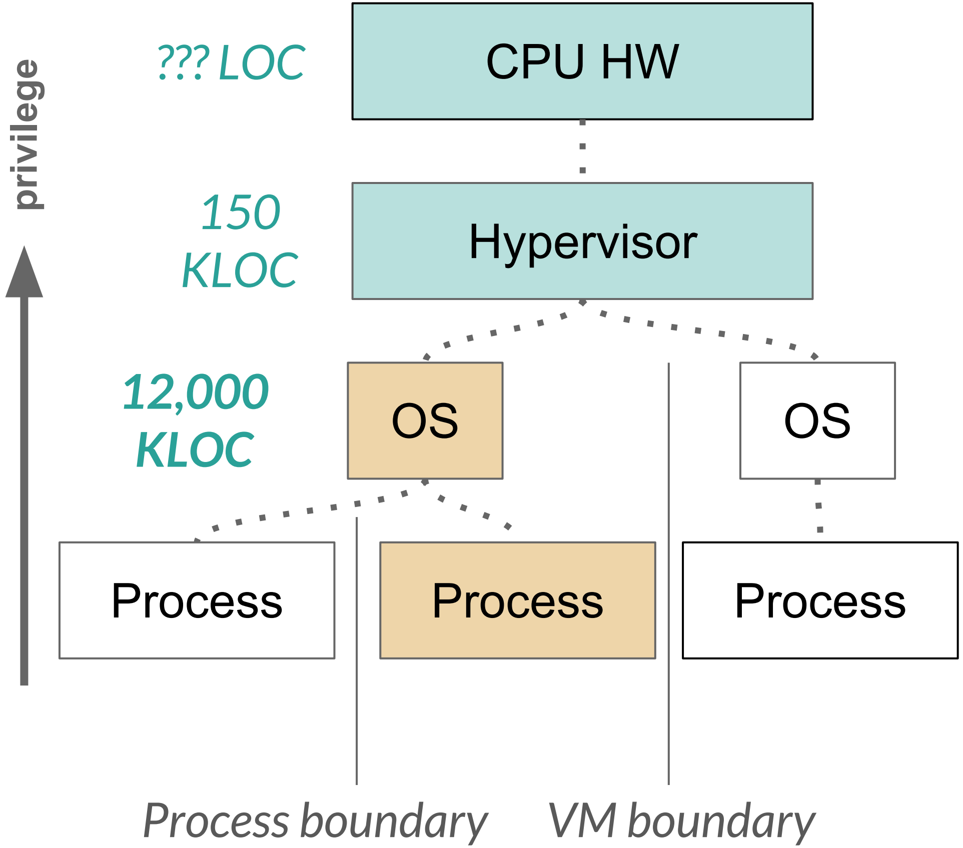


Today, privilege implies trust (2/3)

If computing remotely, what is the TCB?

trusted computing base

Without formal guarantees, large TCBs are buggy, vulnerable, and not trustworthy

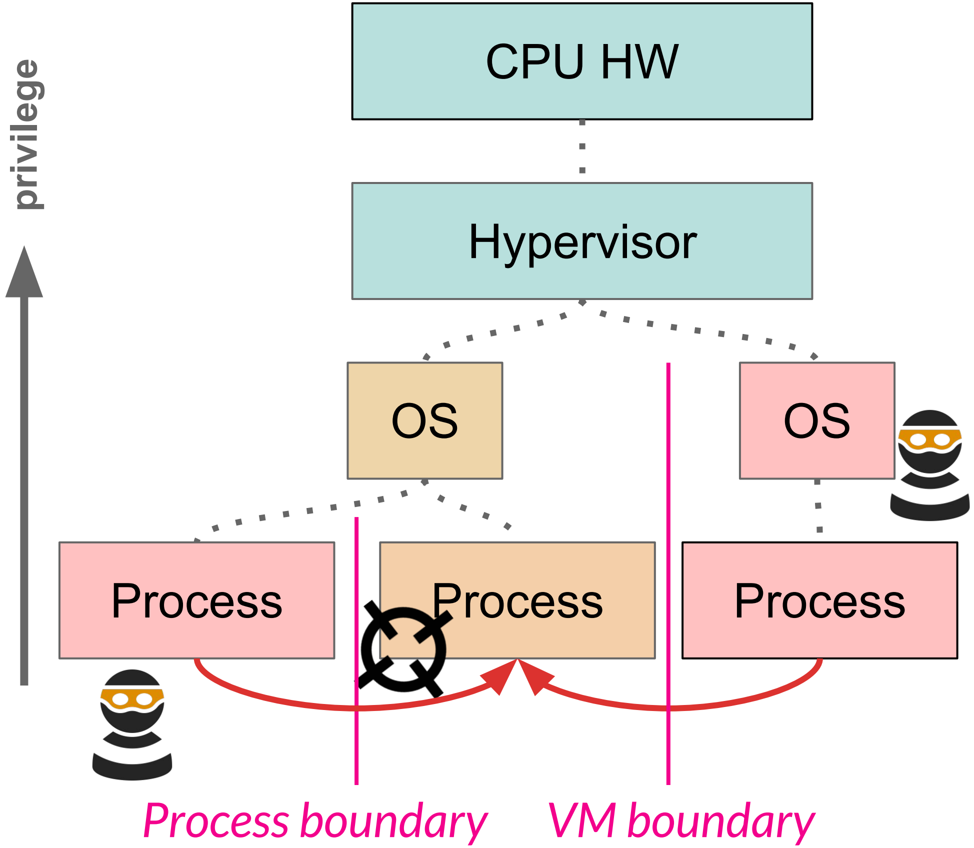


Today, privilege implies trust (3/3)

Leaks via "side channels"
(shared resources)

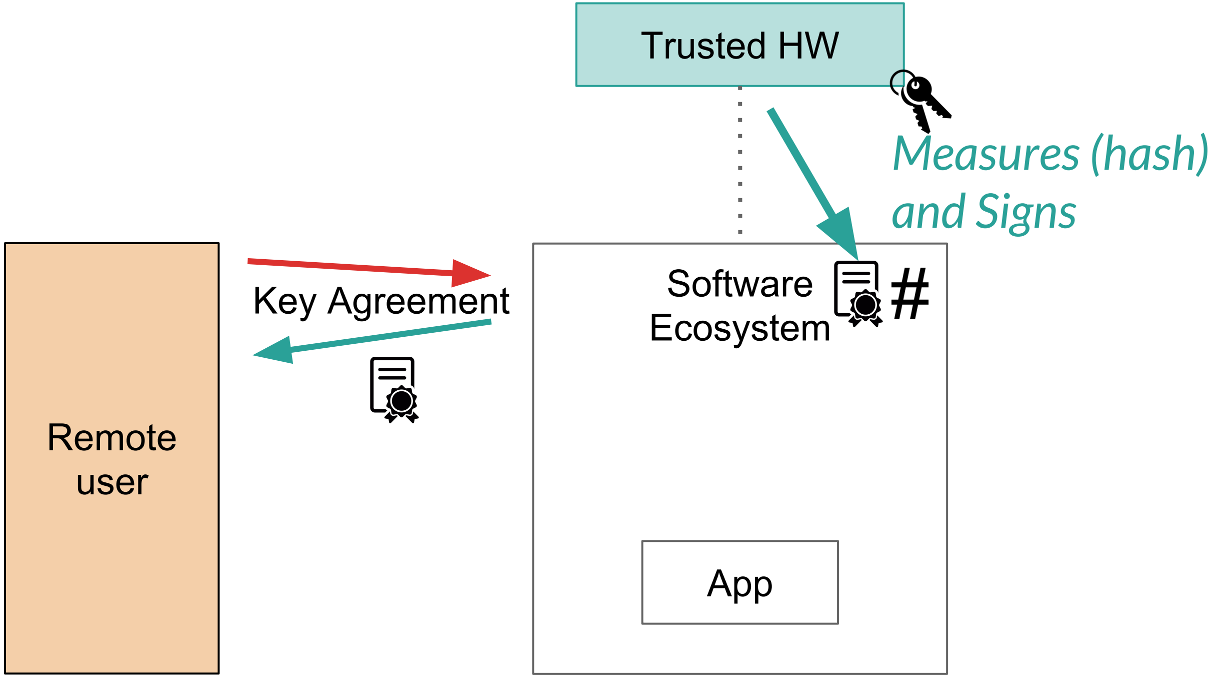
Process abstraction is insufficient.

Separate mutually distrusting entities into isolated protection domains



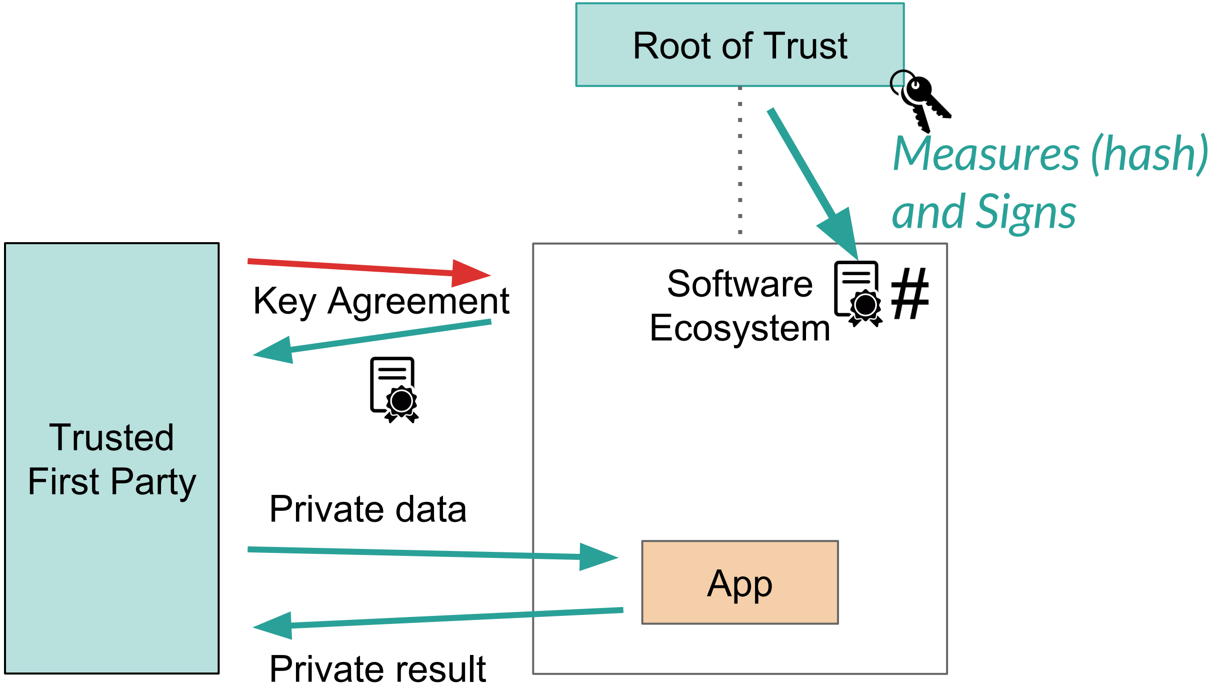
Chapter 1: Remotely Attested Execution

Remote Software Attestation (1 /2)



Trusted HW creates proof for remote user

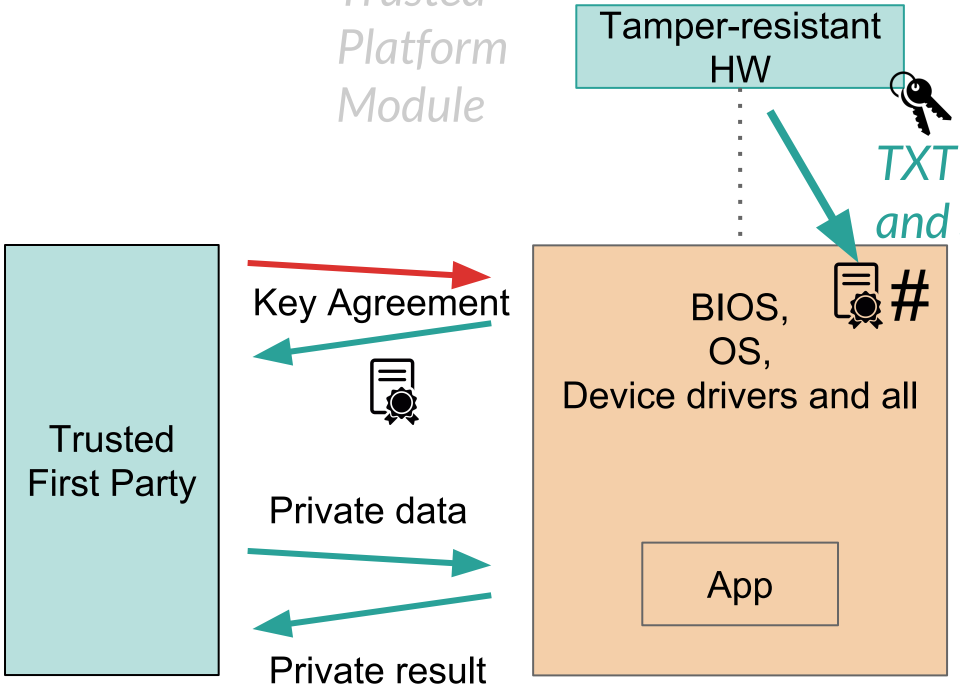
Remote Software Attestation (2 /2)



Remote user decides whether or not to trust certificate

Hardware-Assisted Attestation: TPM+TXT

Trusted Platform Module



Tamper-resistant HW

Trusted First Party

Key Agreement

Private data

Private result

#

BIOS,
OS,
Device drivers and all

App

Trusted HW must keep its keys private!

TXT Measures (hash) and Signs

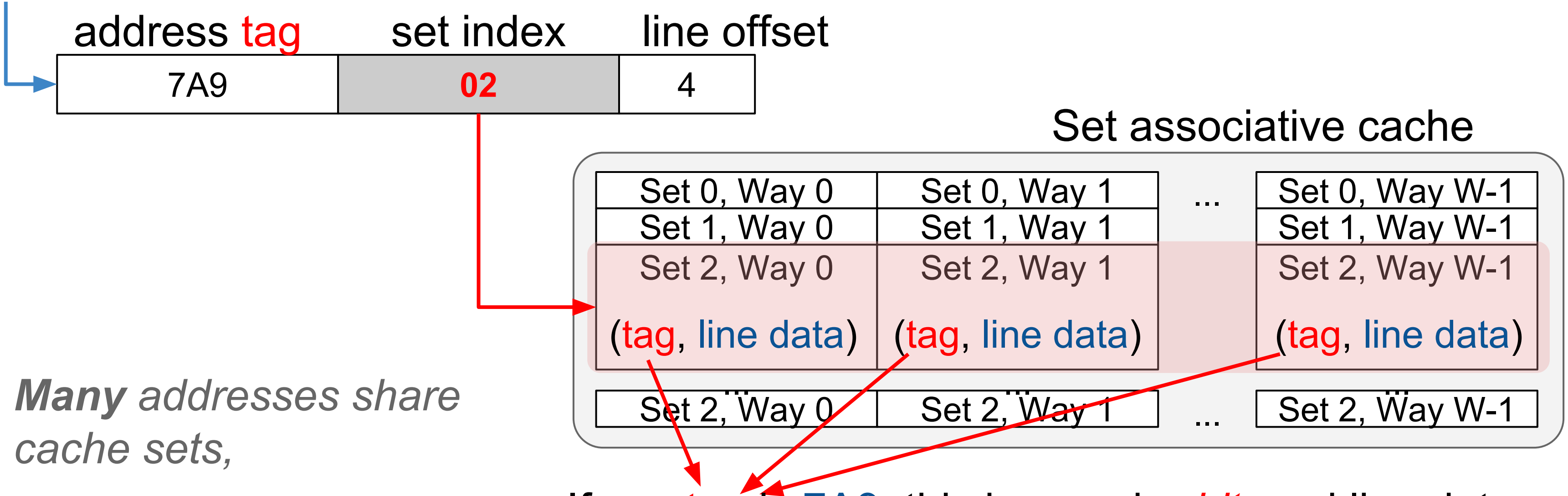
Software ecosystem must not be vulnerable

12,000+ KLOC

Prior work included too much SW in their attestation

Set associative caches share and leak (1/3)

Accessing address
0x7A9024



address tag	set index	line offset
7A9	**02**	4

Set associative cache

Set 0, Way 0	Set 0, Way 1	...	Set 0, Way W-1
Set 1, Way 0	Set 1, Way 1		Set 1, Way W-1
Set 2, Way 0 (tag, line data)	Set 2, Way 1 (tag, line data)		Set 2, Way W-1 (tag, line data)
...	...		...
Set 2, Way 0	Set 2, Way 1	...	Set 2, Way W-1

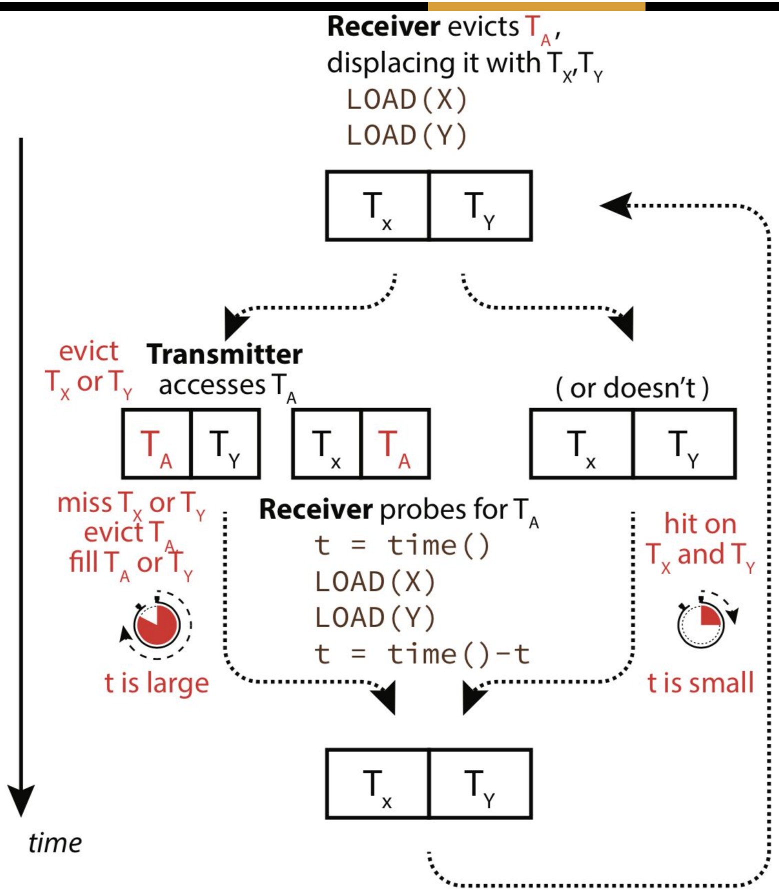
***Many** addresses share cache sets,*

conflict via evictions.

If any tag is 7A9, this is a cache *hit*, and line data is returned / modified.

Else this is a *miss*, and causes a *fill* → ***eviction***

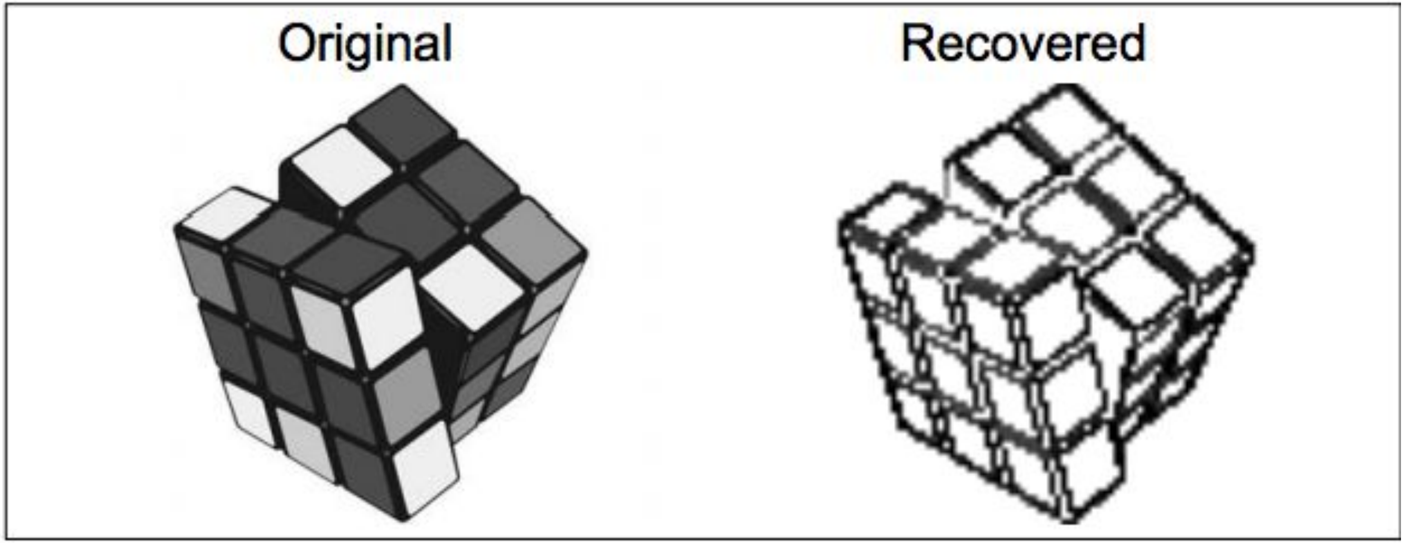
Set associative caches share and leak (2/3)



Page tables also leak (2/2)

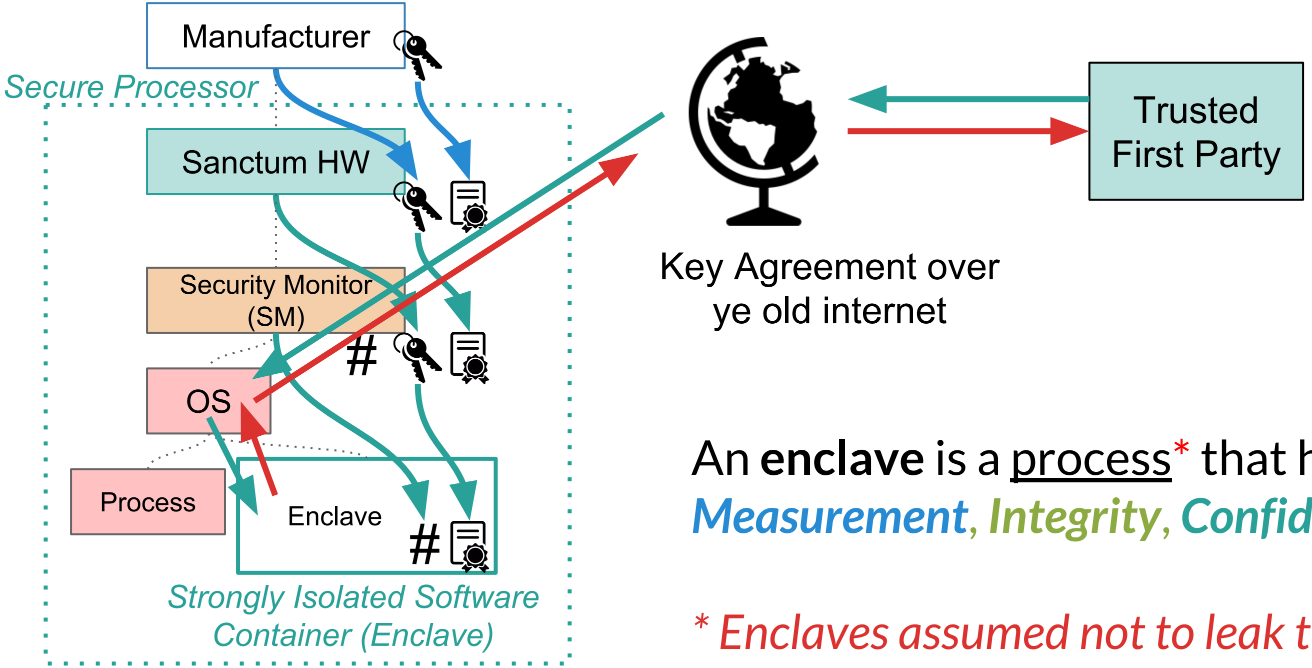
Microsoft Research, IEEE S&P 2015:

Exploit no-noise side channel due to page faults



Encrypted image compared to public images inside enclave

MIT Sanctum Architecture



An **enclave** is a process* that has these properties:
Measurement, ***Integrity***, ***Confidentiality***

** Enclaves assumed not to leak their own private state!*

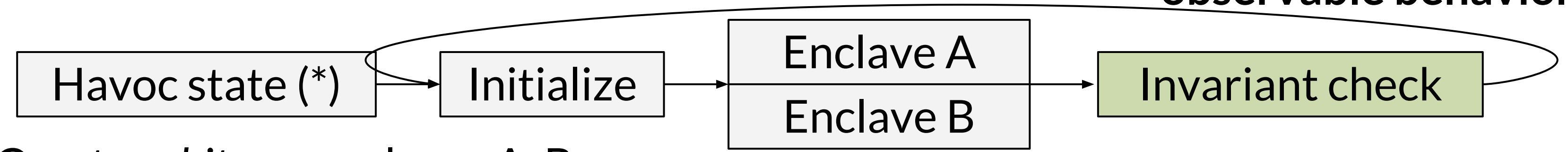
Chapter 2: Enclaves via a Security Monitor

Defining properties of an Enclave (1/3)

measurement

Apply action π to both enclaves

enclaves exhibit the same observable behavior



Havoc state (*) → Initialize → Enclave A / Enclave B → Invariant check

Create *arbitrary* enclaves A, B

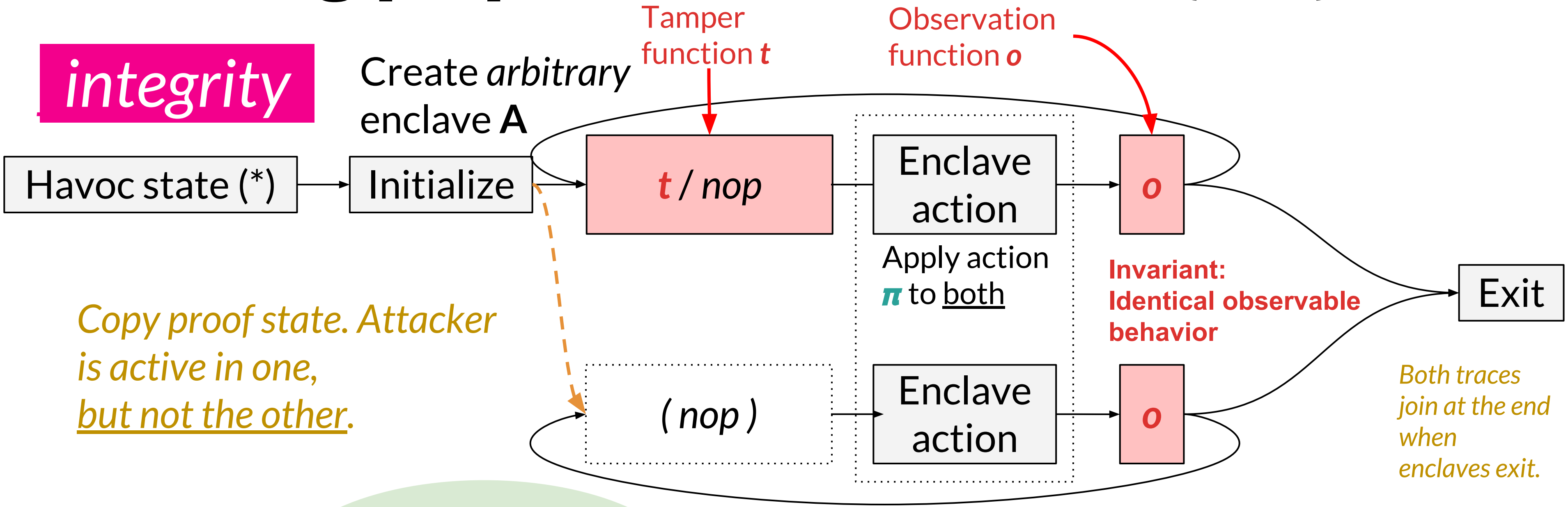
Such that their measurements are **equal**

(☞ﾟヮﾟ)☞
Oh hi! I am authenticated, and you know what to expect from me

Same measurement → same behavior

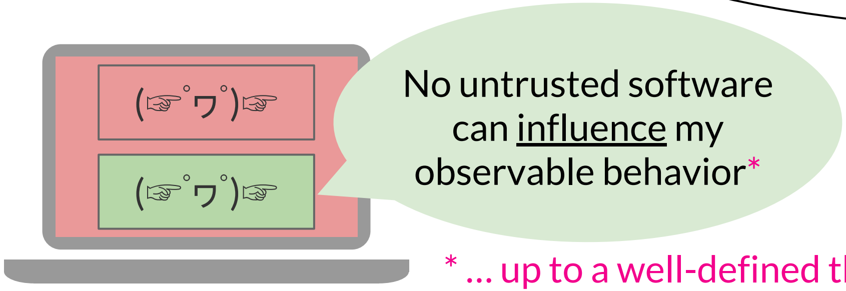
Defining properties of an Enclave (2/3)

integrity


Create *arbitrary* enclave **A**
Tamper function ***t***
Observation function ***o***
Havoc state (*)
Initialize
t / *nop*
Enclave action
Apply action π to both
o
(*nop*)
Enclave action
o
Invariant: Identical observable behavior
Exit

Copy proof state. Attacker is active in one, but not the other.

Both traces join at the end when enclaves exit.



No untrusted software can influence my observable behavior*

Threat model := {Platform API, ***o***, ***t***}

* … up to a well-defined threat model

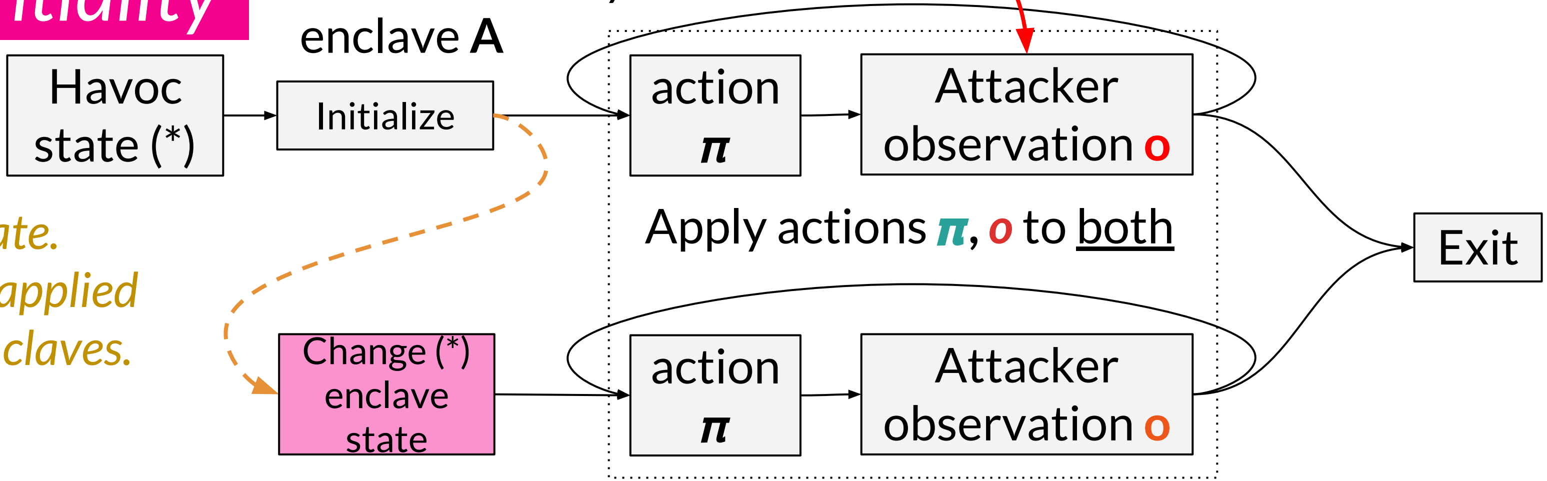
Defining properties of an Enclave (3/3)

confidentiality

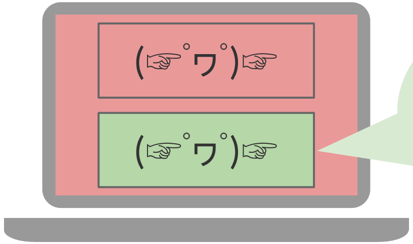
Observation function **o**

Invariant: attacker sees the same observation

Create *arbitrary* enclave **A**


Havoc state (*)
Initialize
action *π*
Attacker observation **o**
Apply actions *π*, *o* to both
Change (*) enclave state
action *π*
Attacker observation **o**
Exit

Copy proof state. Same actions applied to different enclaves.



The observable side effects of my computation are independent of my private state

Threat model := {Platform API, **o**}

Threat Model

(enclaves are correct)

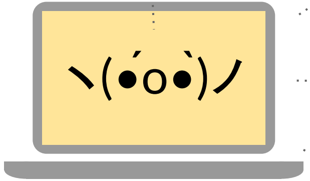
Root of trust software

Hardware

first party

Entities attempting denial of service

No guarantees possible



anonymous remote users

Small “security monitor” firmware

Authenticated remote users

user-provided software (OS, apps. scripts in a web page)

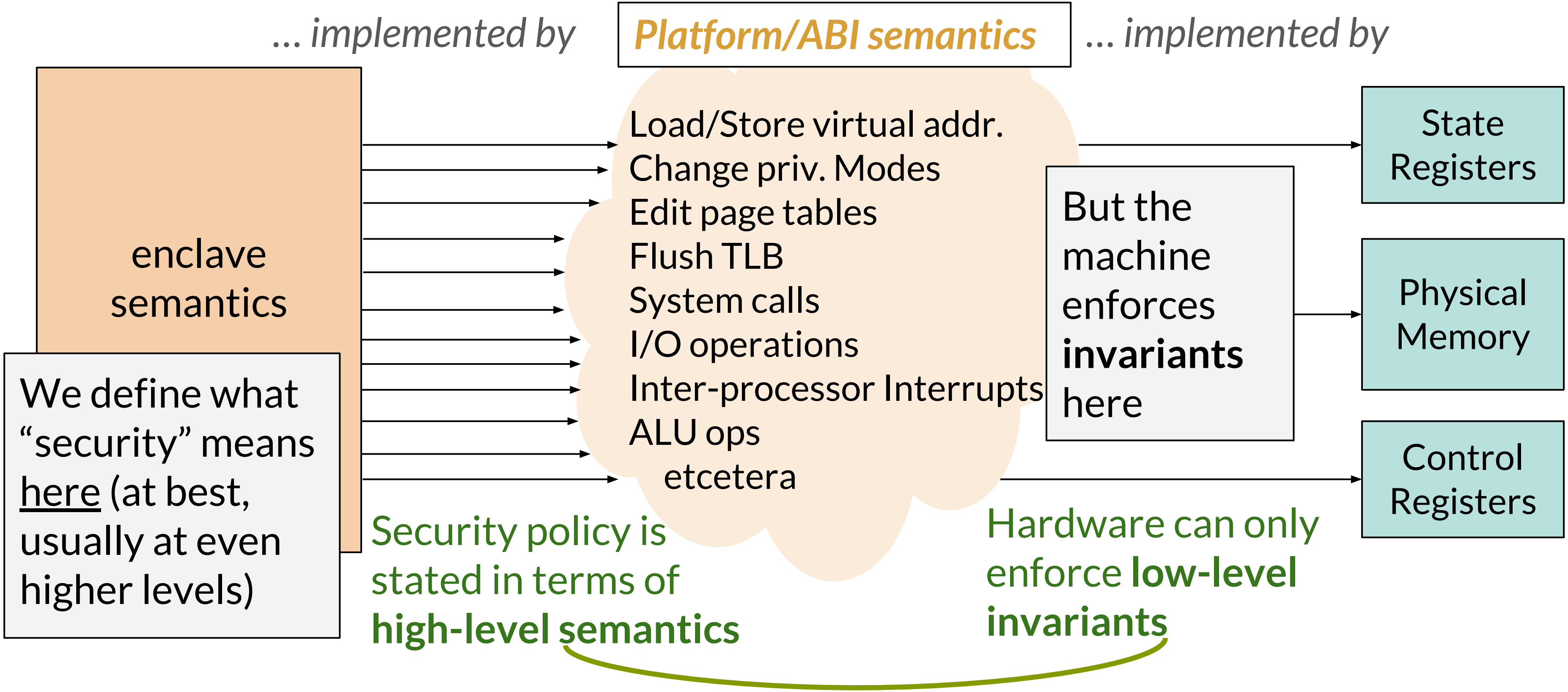
(arbitrary software is compromised)

enclaved software binary

Users with physical access

(no protection offered by Sanctum)

Hardware security is hard



enclave semantics

… *implemented by*

Platform/ABI semantics

Load/Store virtual addr.
Change priv. Modes
Edit page tables
Flush TLB
System calls
I/O operations
Inter-processor Interrupts
ALU ops
etcetera

… *implemented by*

State Registers

Physical Memory

Control Registers

We define what "security" means here (at best, usually at even higher levels)

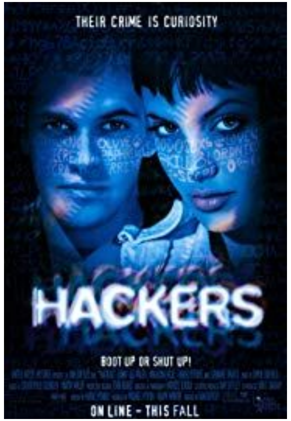
Security policy is stated in terms of **high-level semantics**

But the machine enforces **invariants** here

Hardware can only enforce **low-level invariants**

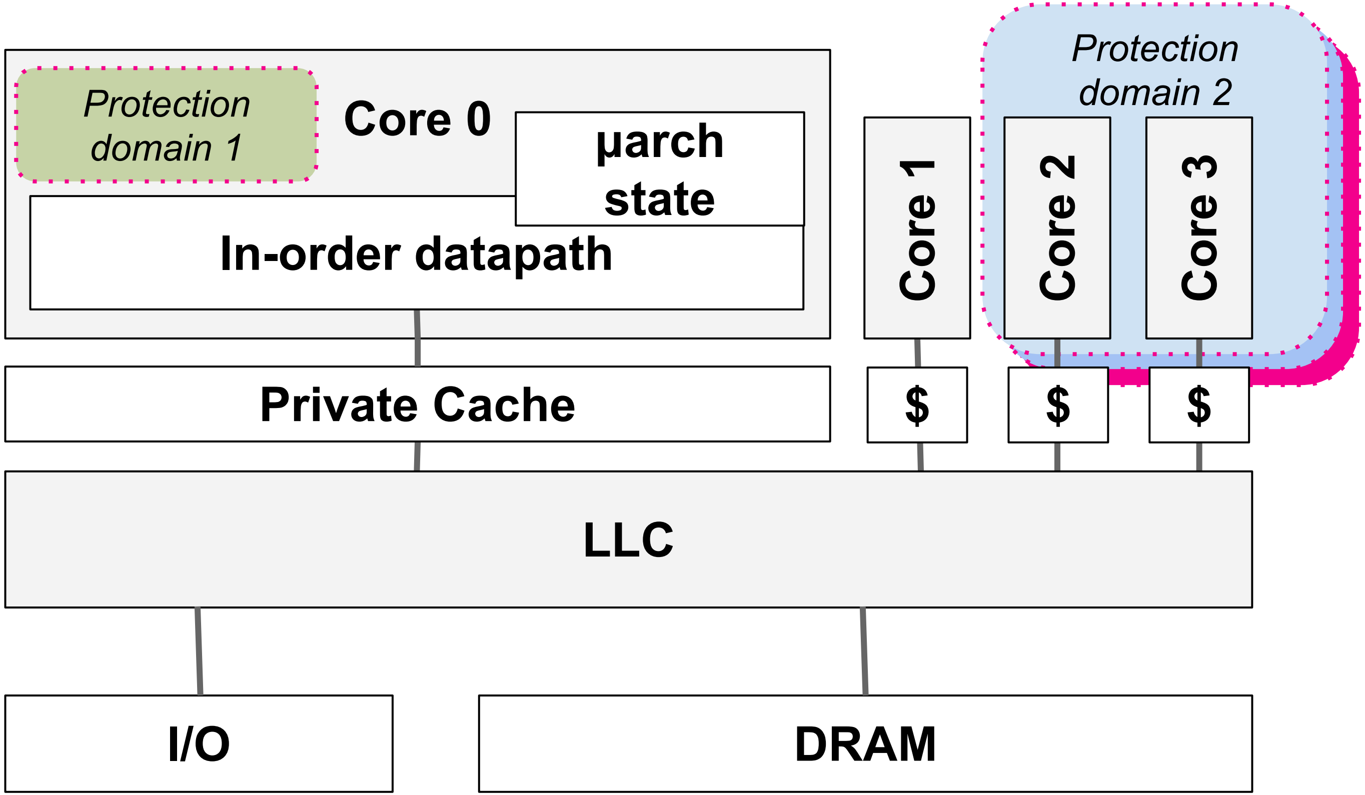
Use formal verification to prove equivalence!

Chapter 3: Strong Microarch. Isolation of Protection Domains

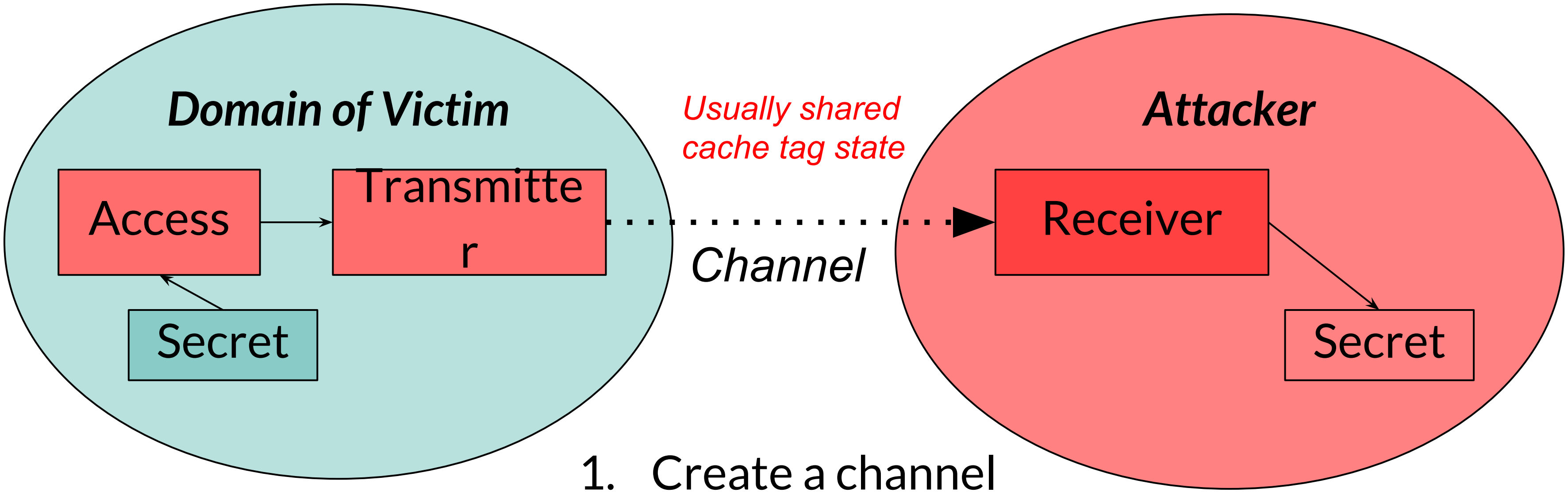


“RISC architecture is gonna change everything”

Sharing resources in a simple processor system

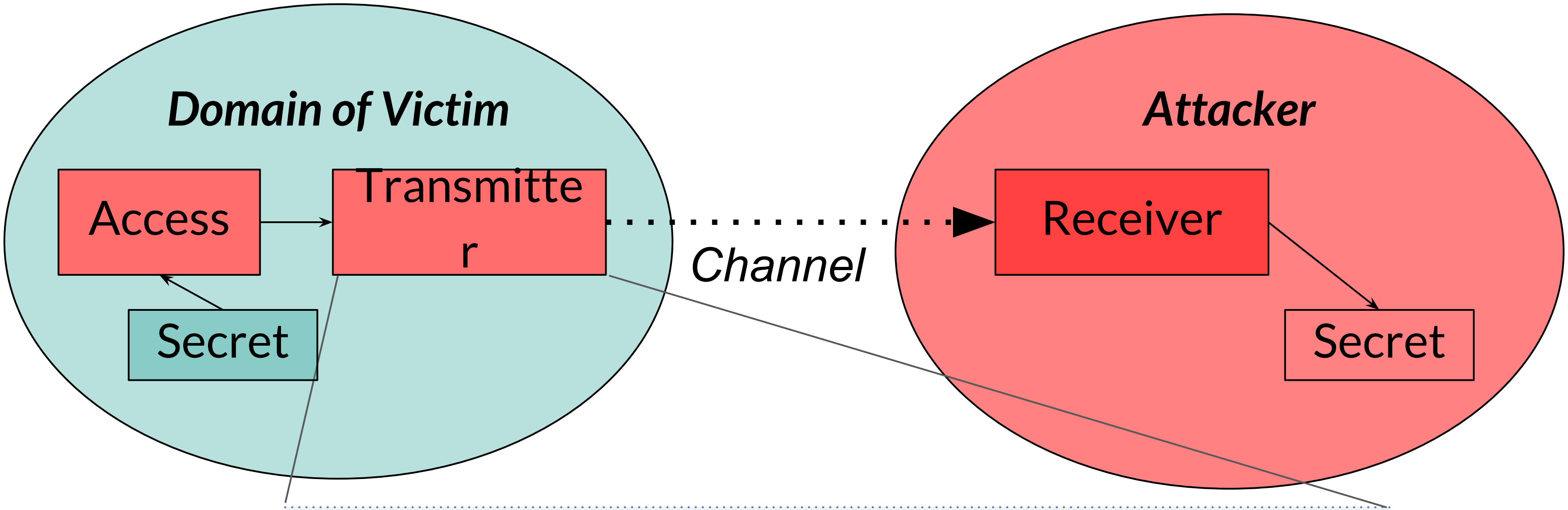


Attack Schema (1/2)



1. Create a channel
2. Create the transmitter
3. Launch the transmitter
4. Access the secret

Attack Schema (2/2)

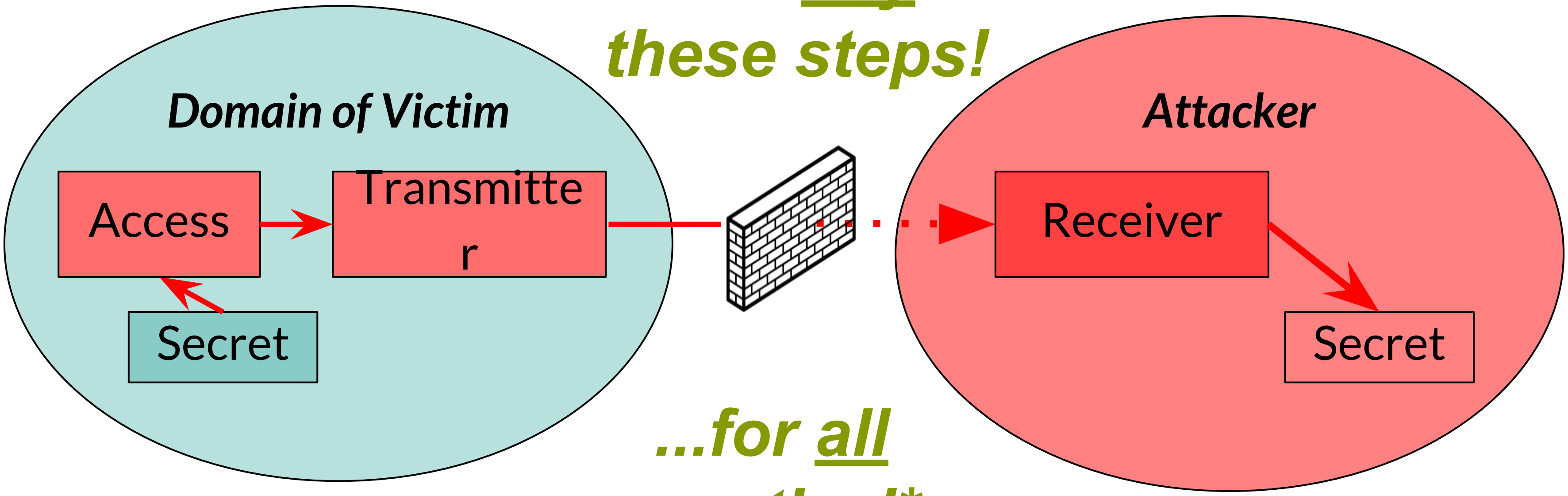


-Pre-existing (classic RSA mod-exp cache leak)
- Written by attacker (Meltdown)
- Assembled from victim code by attacker (Spectre)

Defense Schema

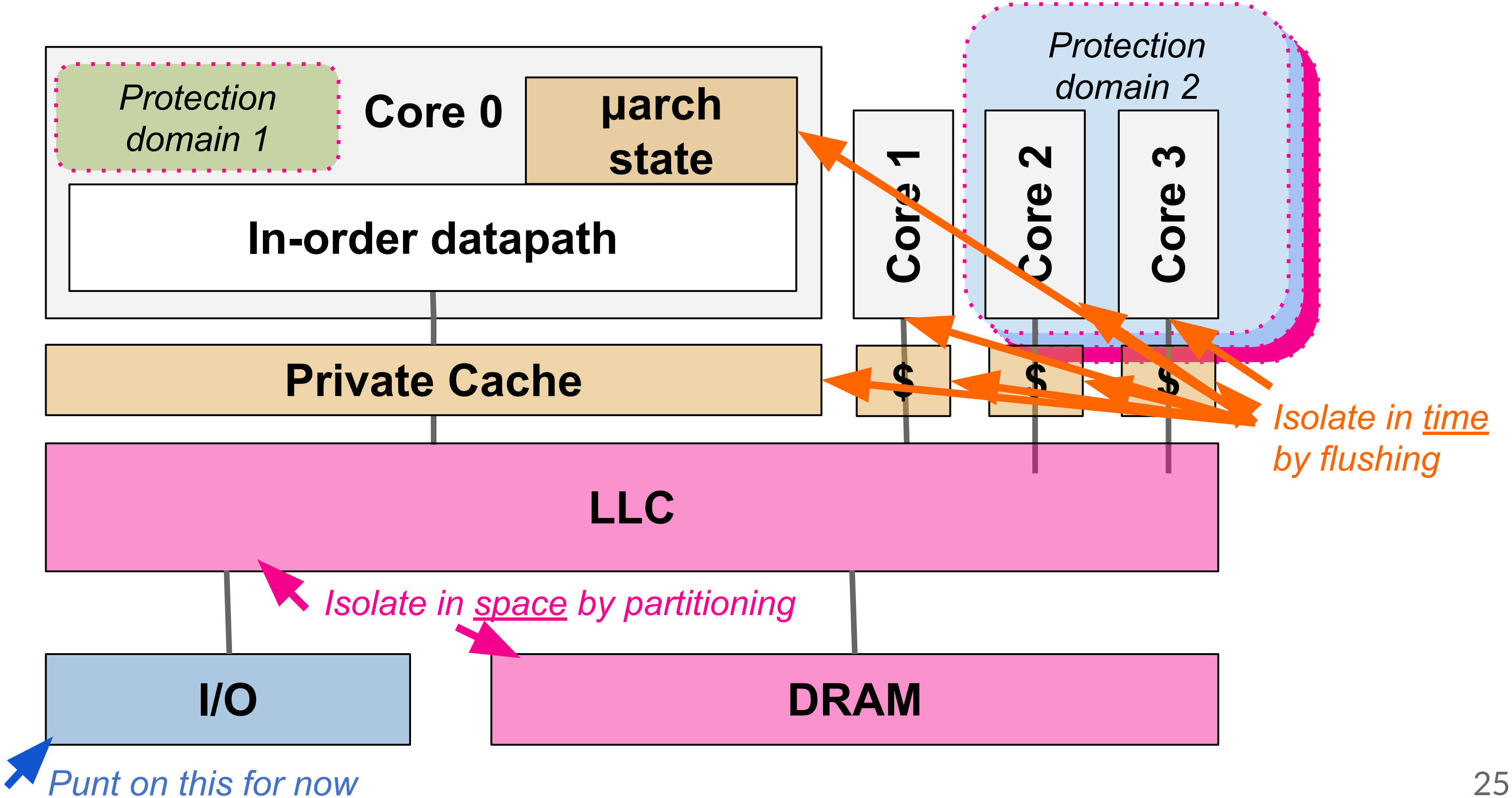
If cannot prevent infiltration,

Block any of these steps!



...for all practical channels*

Sharing resources in a simple processor system



Isolate in time by flushing

Isolate in space by partitioning

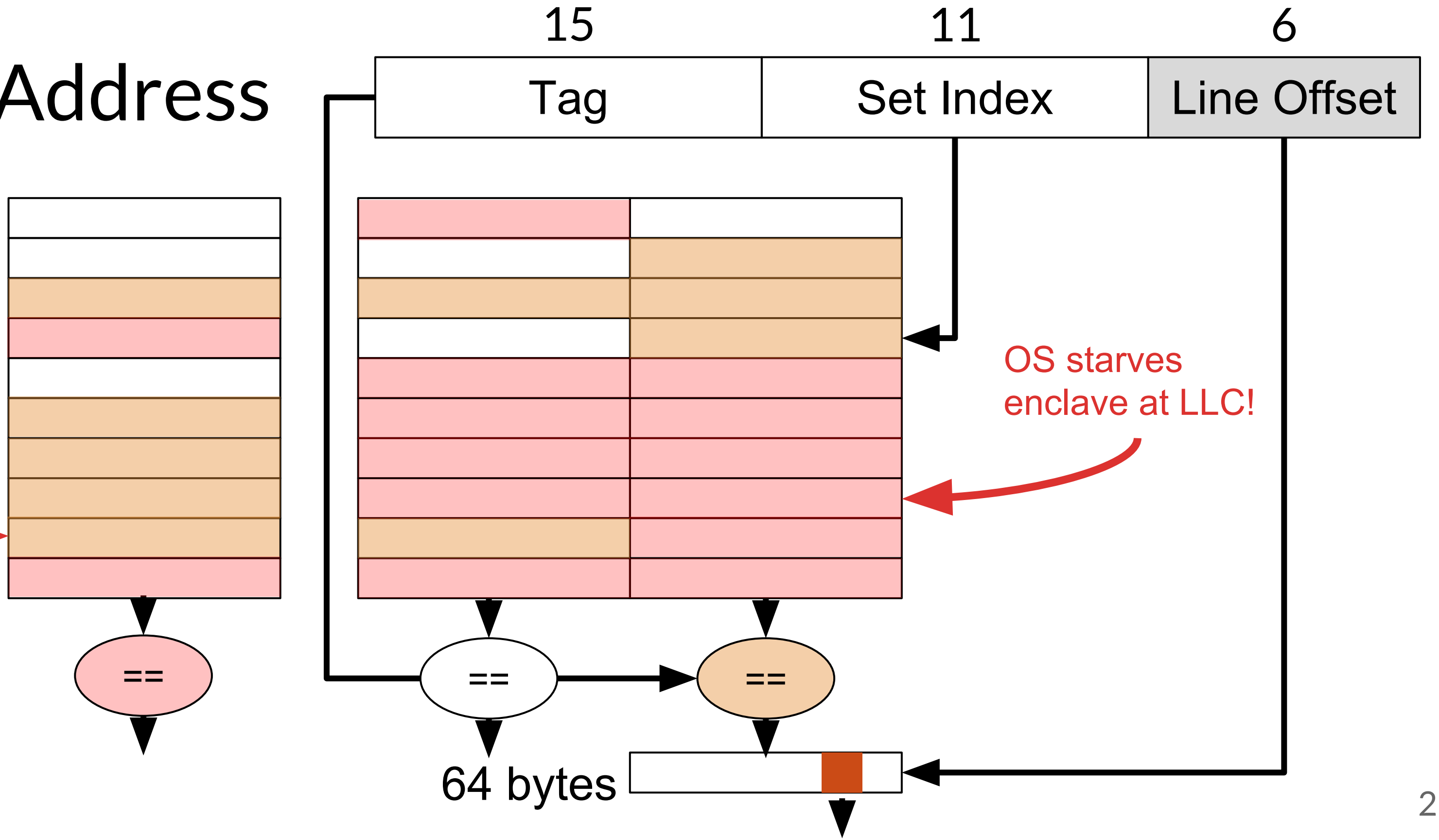
Punt on this for now

Isolating in the LLC (2/8)

LLC sharing leaks privacy!

Physical Address

LLC



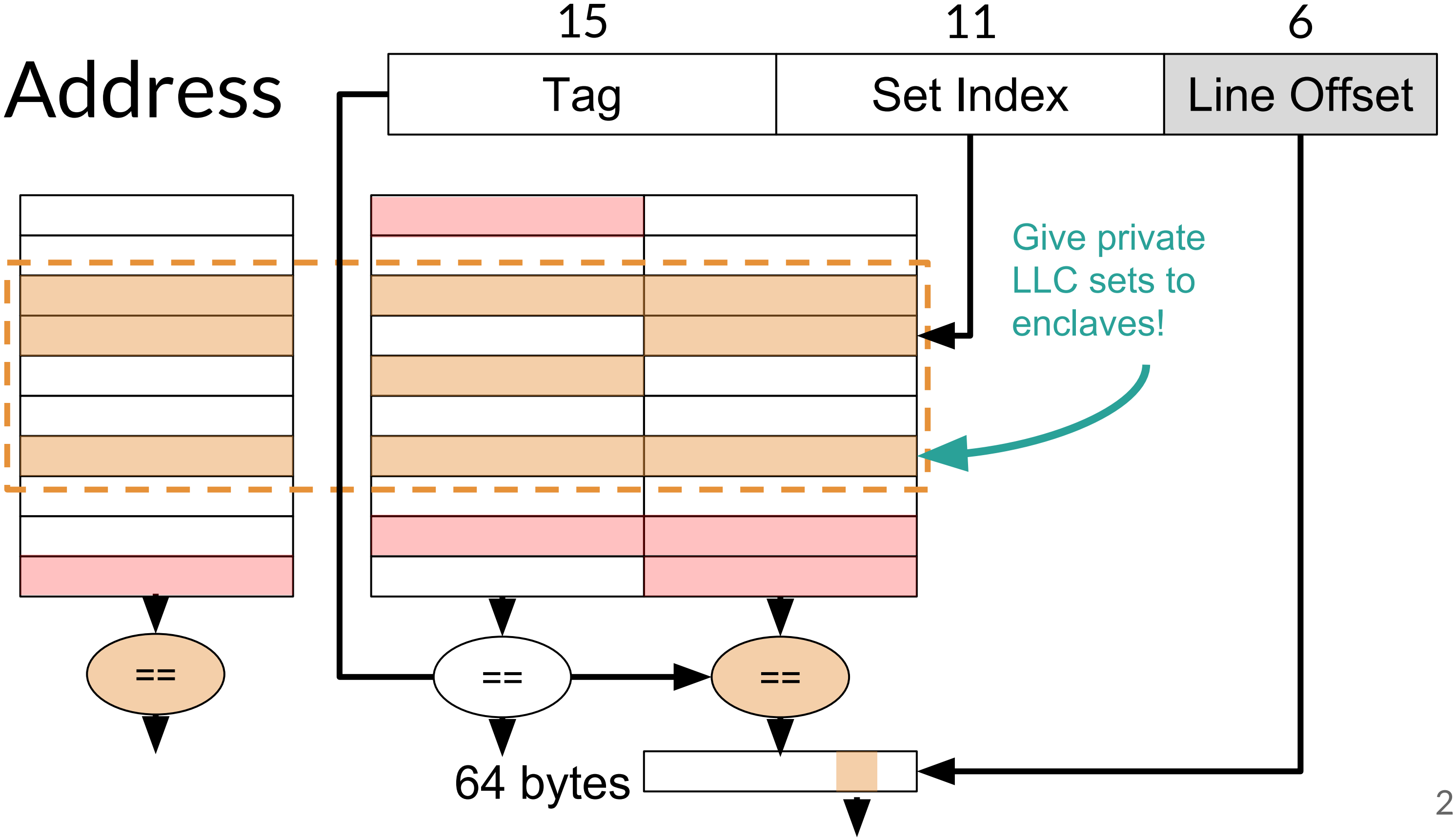
OS starves enclave at LLC!

Availability of this set leaks privacy

Isolating in the LLC (2/8)

LLC sharing leaks privacy!

Physical Address

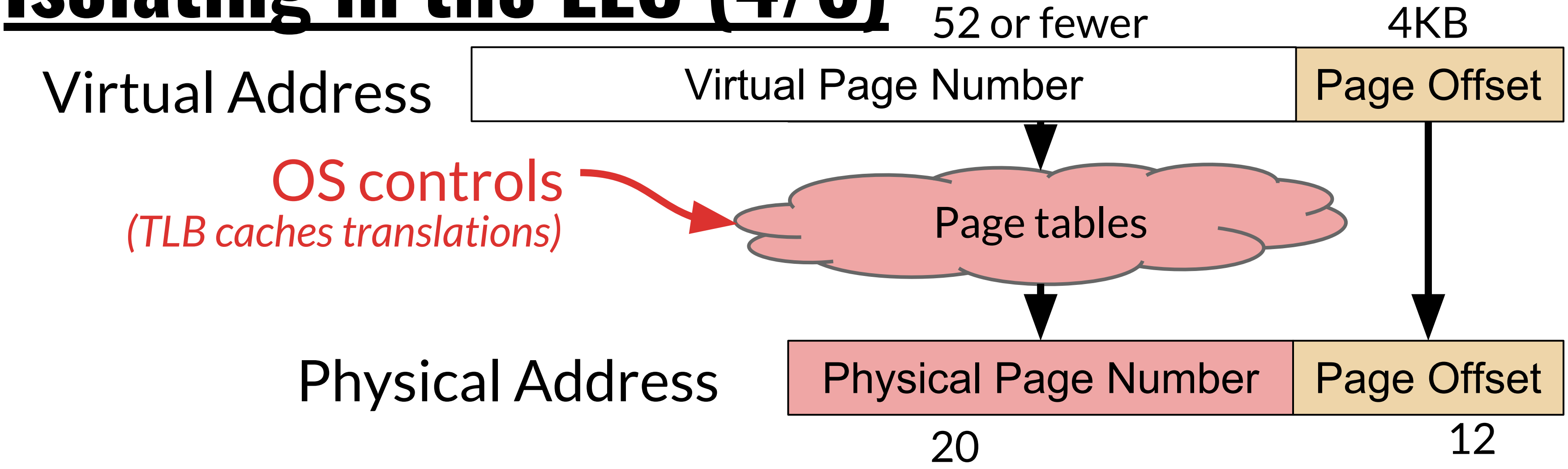


Give private LLC sets to enclaves!

LLC

Isolating in the LLC (4/8)

Virtual address translation

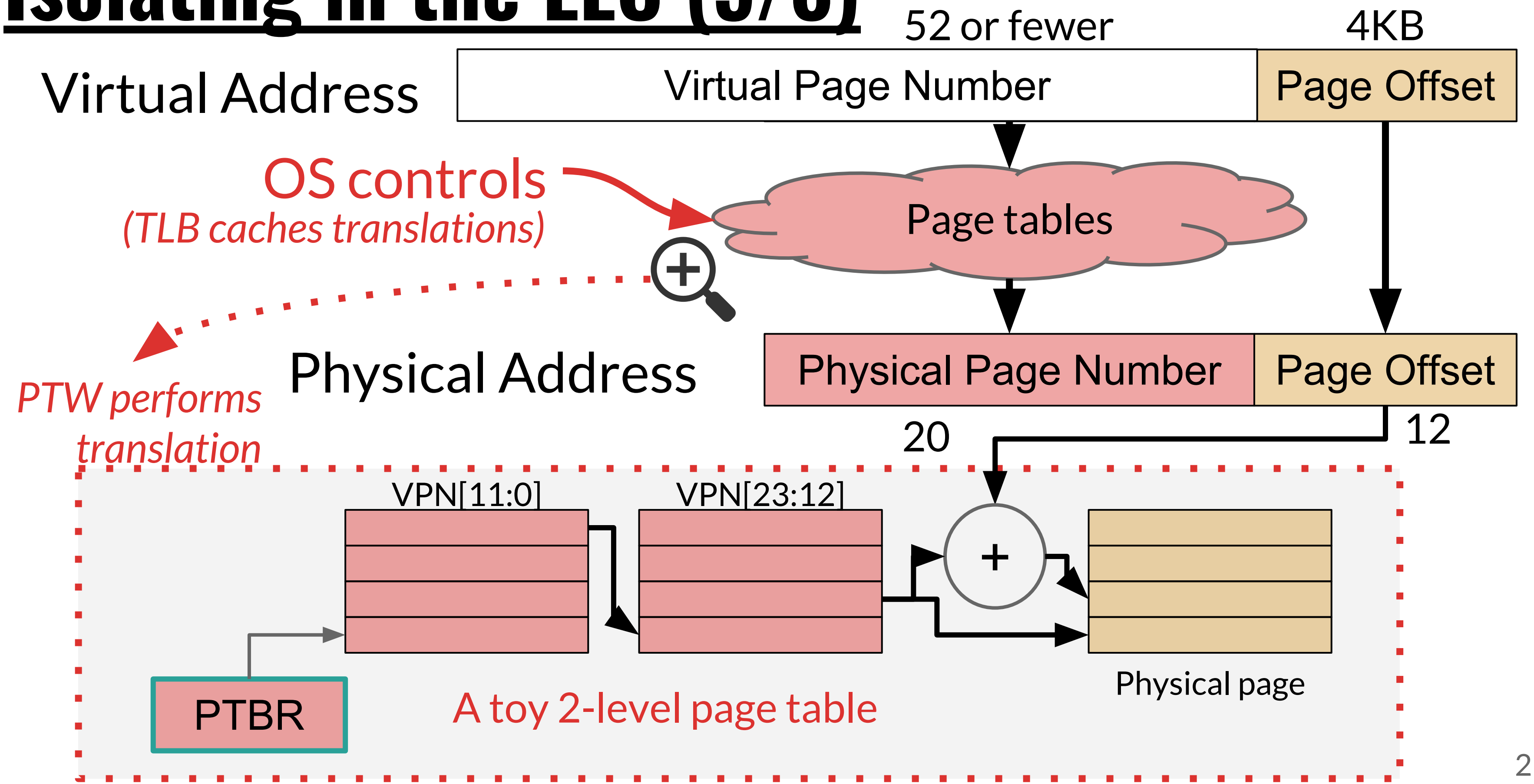


OS controls

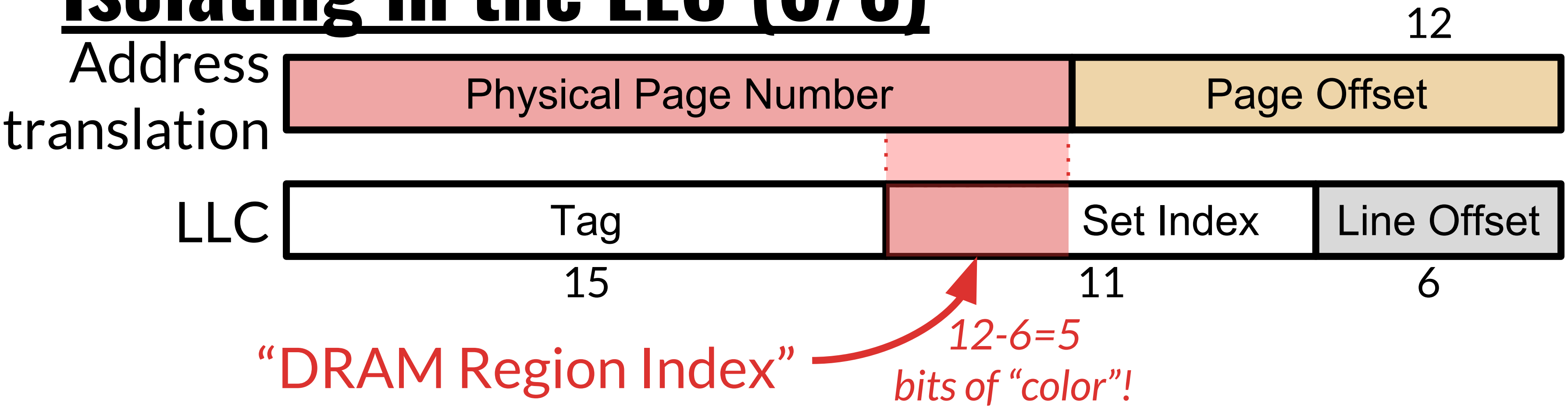
(TLB caches translations)

Isolating in the LLC (5/8)

Virtual address translation

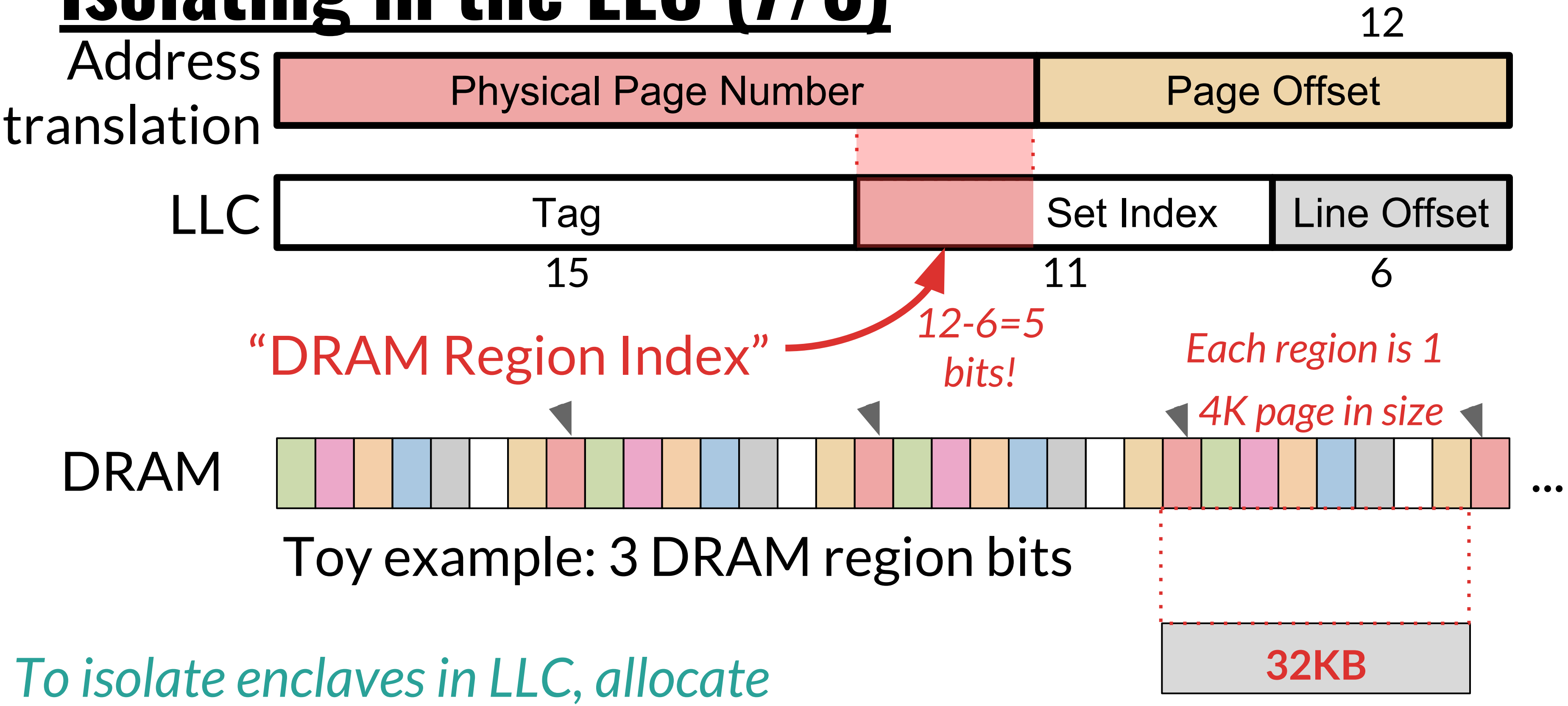


Isolating in the LLC (6/8)



To isolate enclaves in LLC, allocate exclusively, at region granularity!

Isolating in the LLC (7/8)



"DRAM Region Index"

12-6=5 bits!

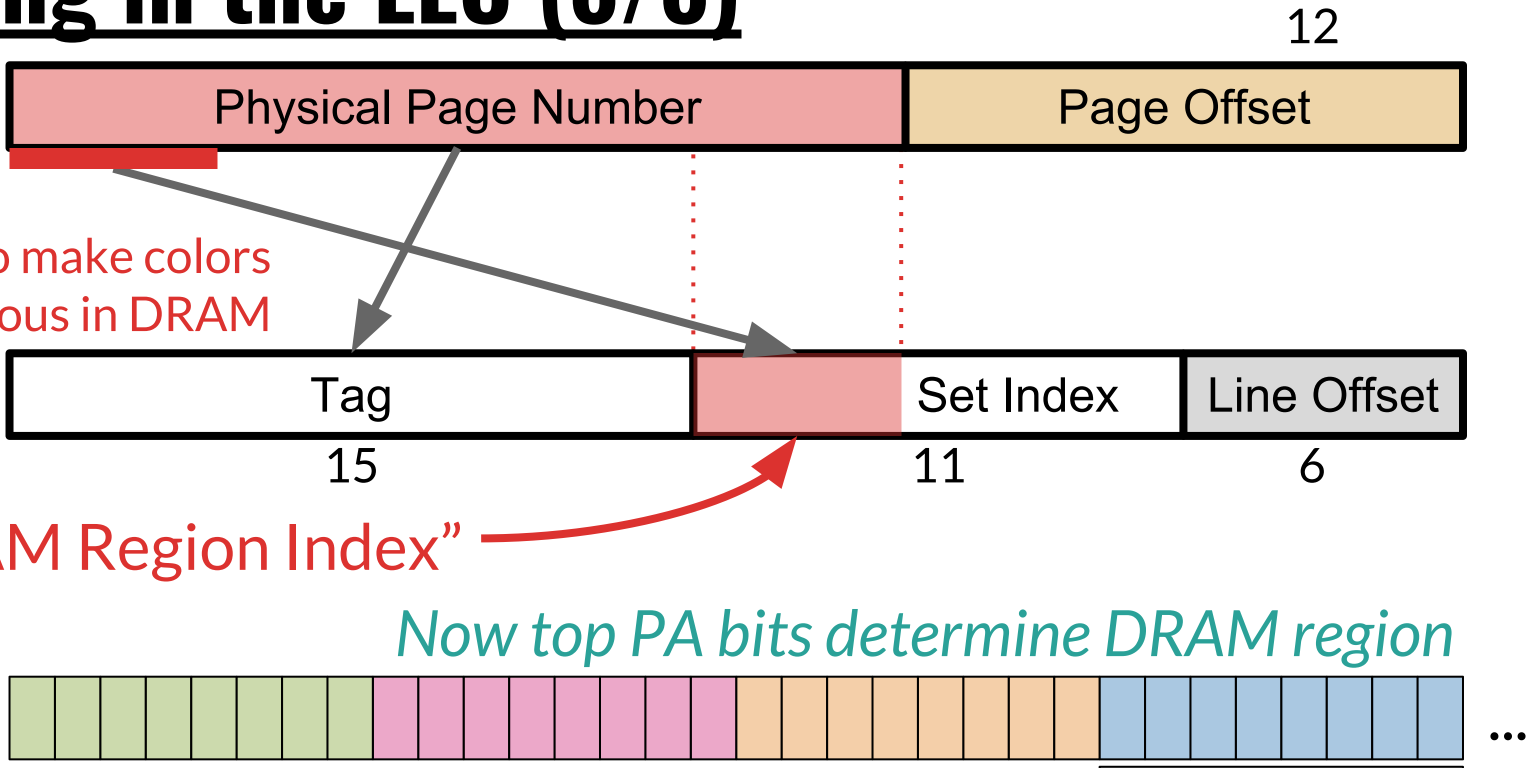
Each region is 1 4K page in size

Toy example: 3 DRAM region bits

To isolate enclaves in LLC, allocate exclusively, at region granularity!

Small problem : A DMA buffer

Isolating in the LLC (8/8)



Now top PA bits determine DRAM region

Toy example: 3 DRAM region bits

Each region is contiguous, 32KB

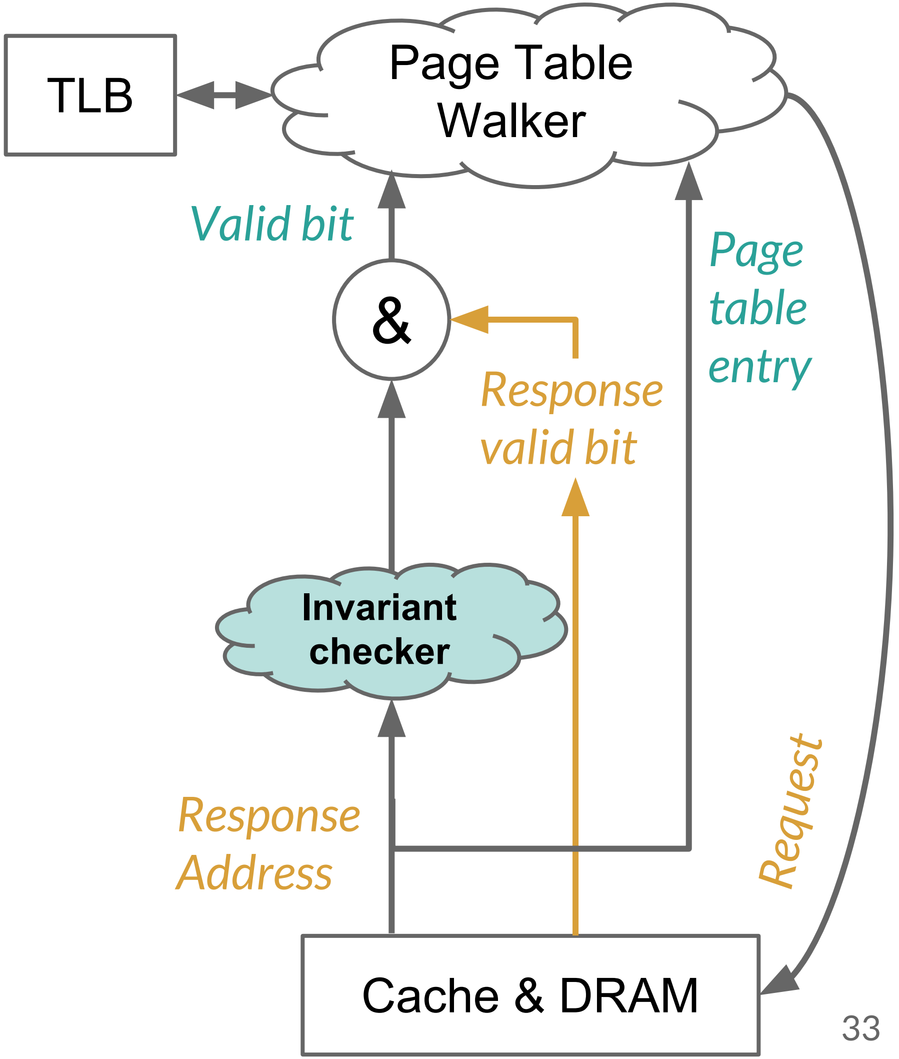
32KB

Hardware-assisted Isolation

Maintain an invariant:
TLB entries are safe!

HW enforces invariants
at page walks

SW updates invariants and causes TLB shootdowns



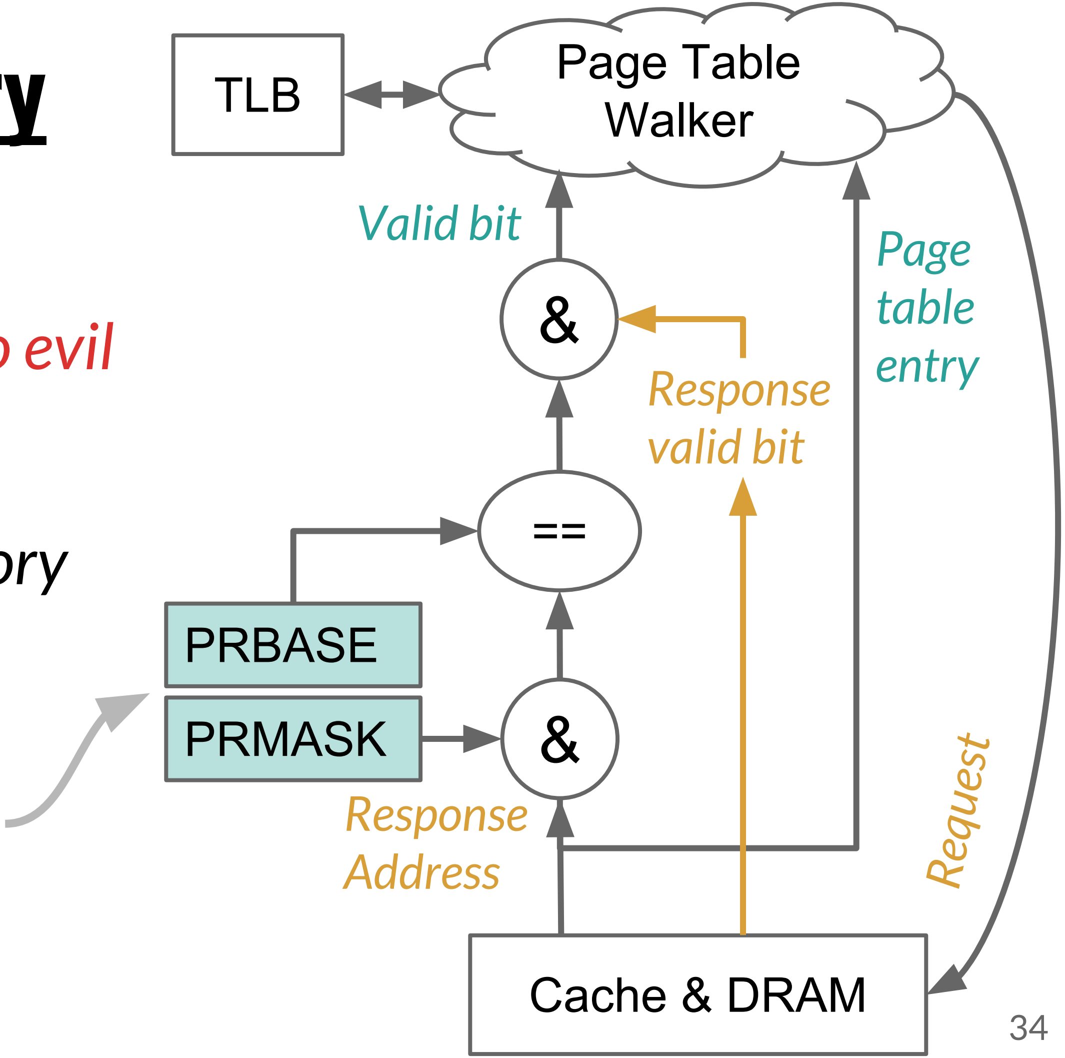
Protect SM memory from everyone

OS could rewrite S.M. code, do evil

fix by...

Never map VAddr to SM memory

SM sanitizes mode switch



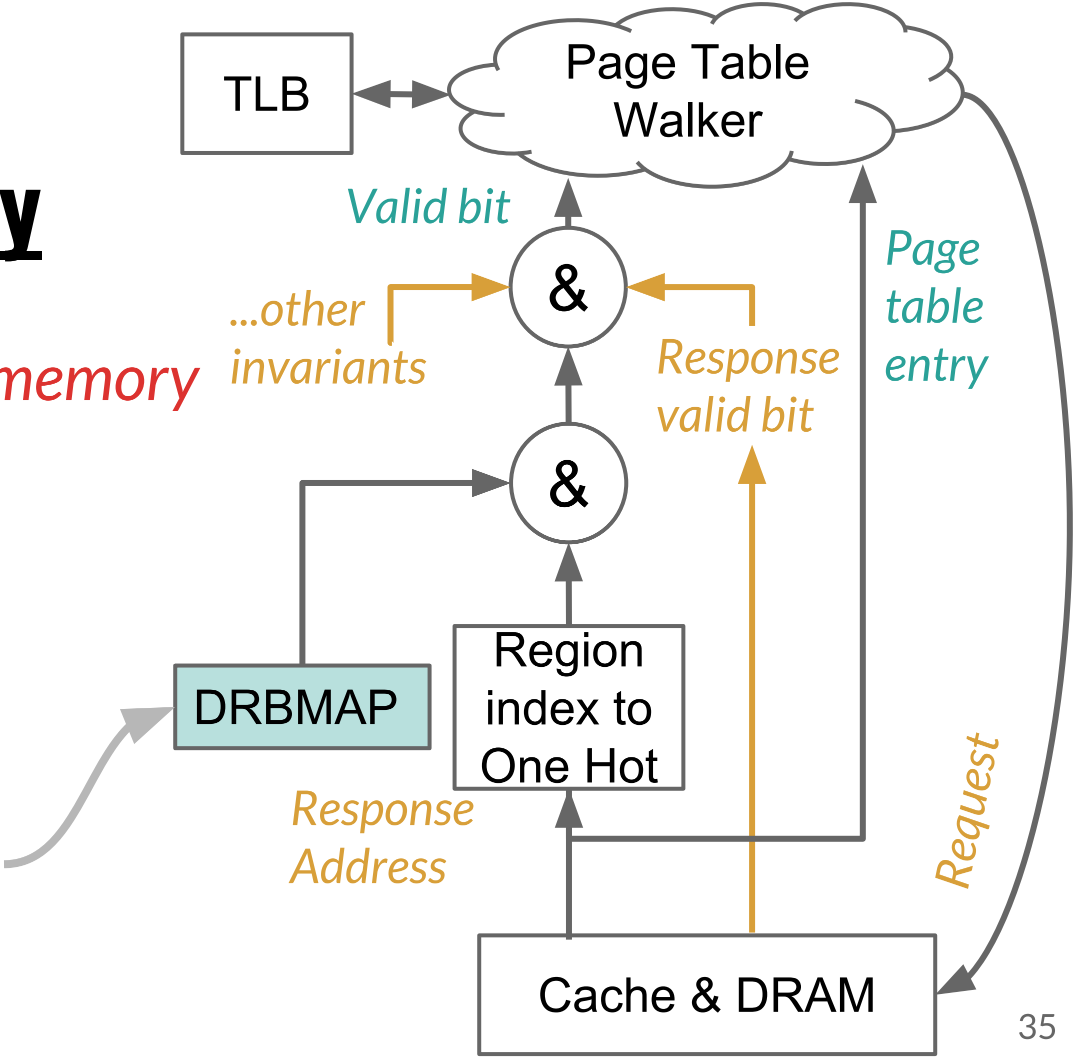
Isolating Enclaves in Physical Memory

OS could read/write Enclave memory

fix by...

Enforce DRAM Region permissions to at page walk

S.M. updates permissions when scheduling enclaves



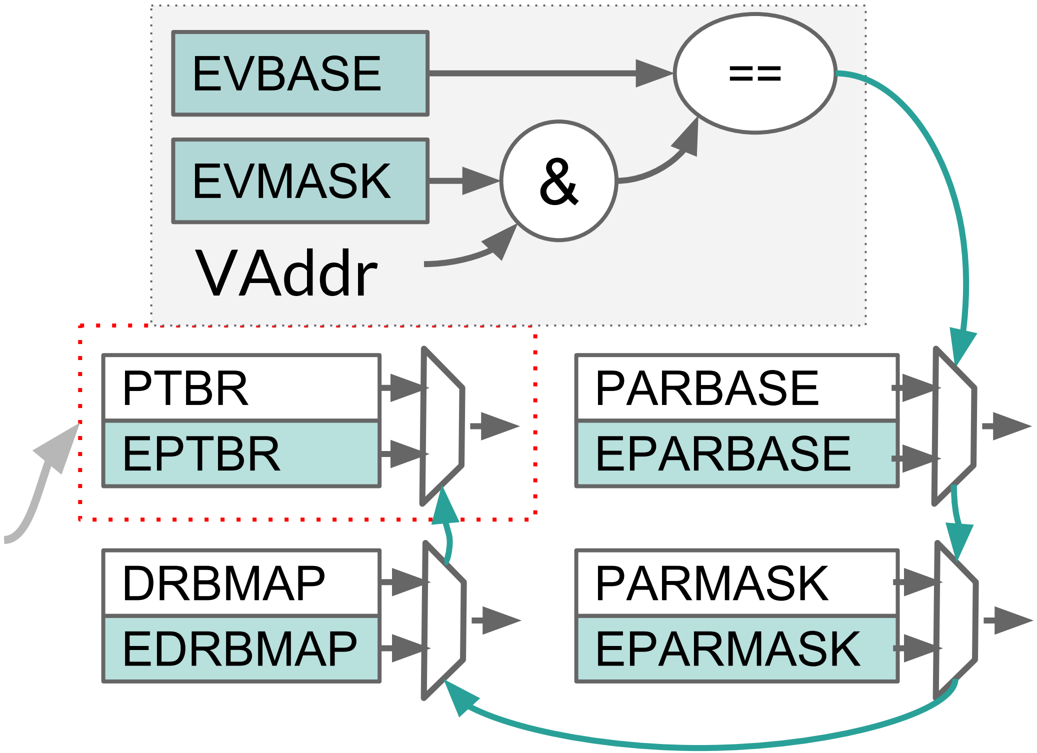
Isolating enclave page tables

OS could spy on enclave's page table entries

fix by...

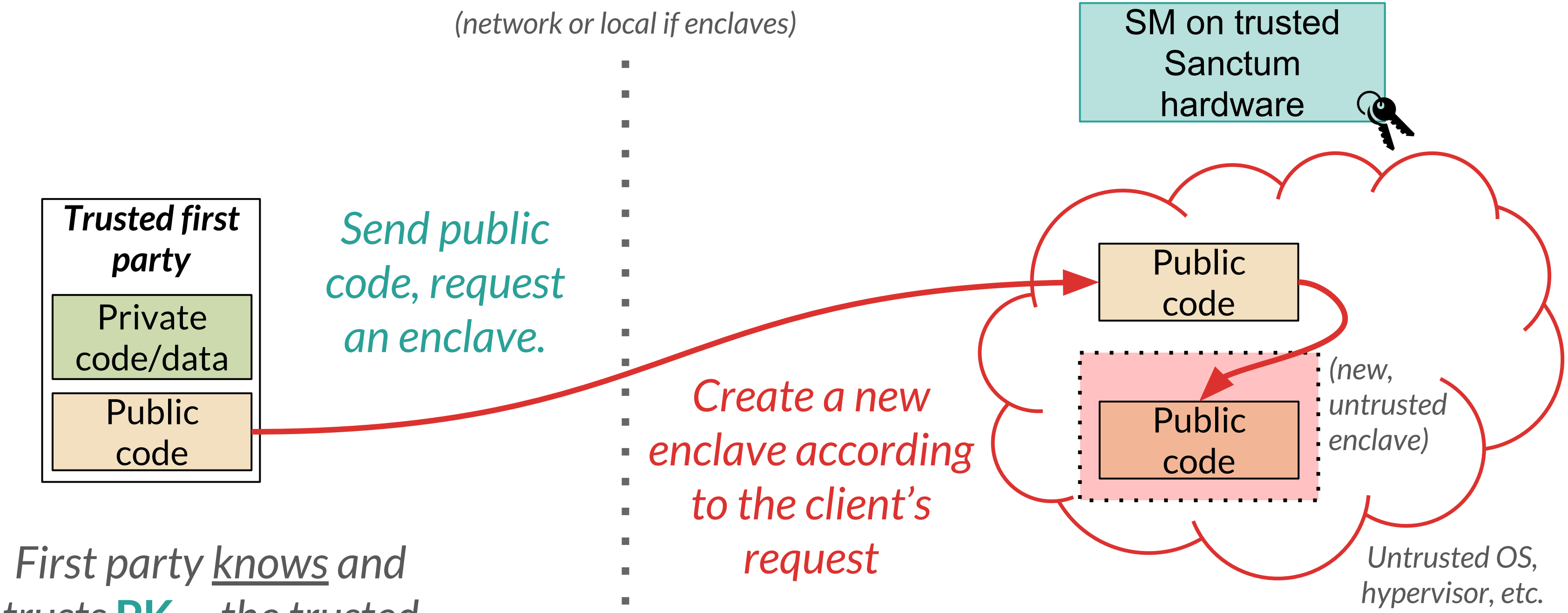
Implement <u>enclave-private page tables</u>

Should this VA use enclave's tables?



this slide is intentionally left blank

Remote attestation of enclaves (1/5)



(network or local if enclaves)

SM on trusted Sanctum hardware

Trusted first party

Private code/data

Public code

Send public code, request an enclave.

Create a new enclave according to the client's request

Public code

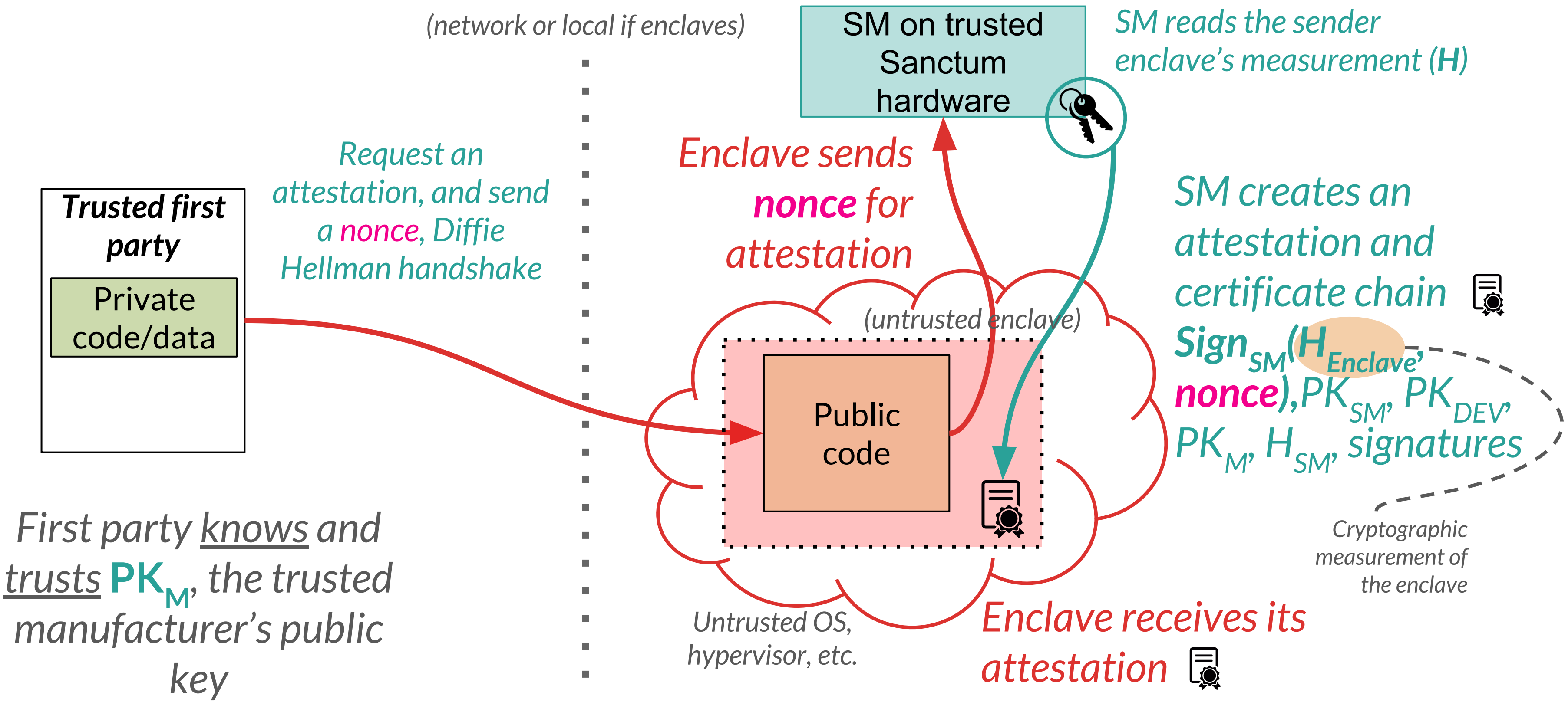
Public code

(new, untrusted enclave)

Untrusted OS, hypervisor, etc.

First party knows and trusts PK_M*, the trusted manufacturer's public key*

Remote attestation of enclaves (2/5)



(network or local if enclaves)

SM on trusted Sanctum hardware

SM reads the sender enclave's measurement (**H**)

Request an attestation, and send a nonce, Diffie Hellman handshake

Trusted first party

Private code/data

Enclave sends **nonce** for attestation

SM creates an attestation and certificate chain

Sign$_{SM}$(**H**$_{Enclave}$, **nonce**), PK$_{SM}$, PK$_{DEV}$, PK$_{M}$, H$_{SM}$, signatures

(untrusted enclave)

Public code

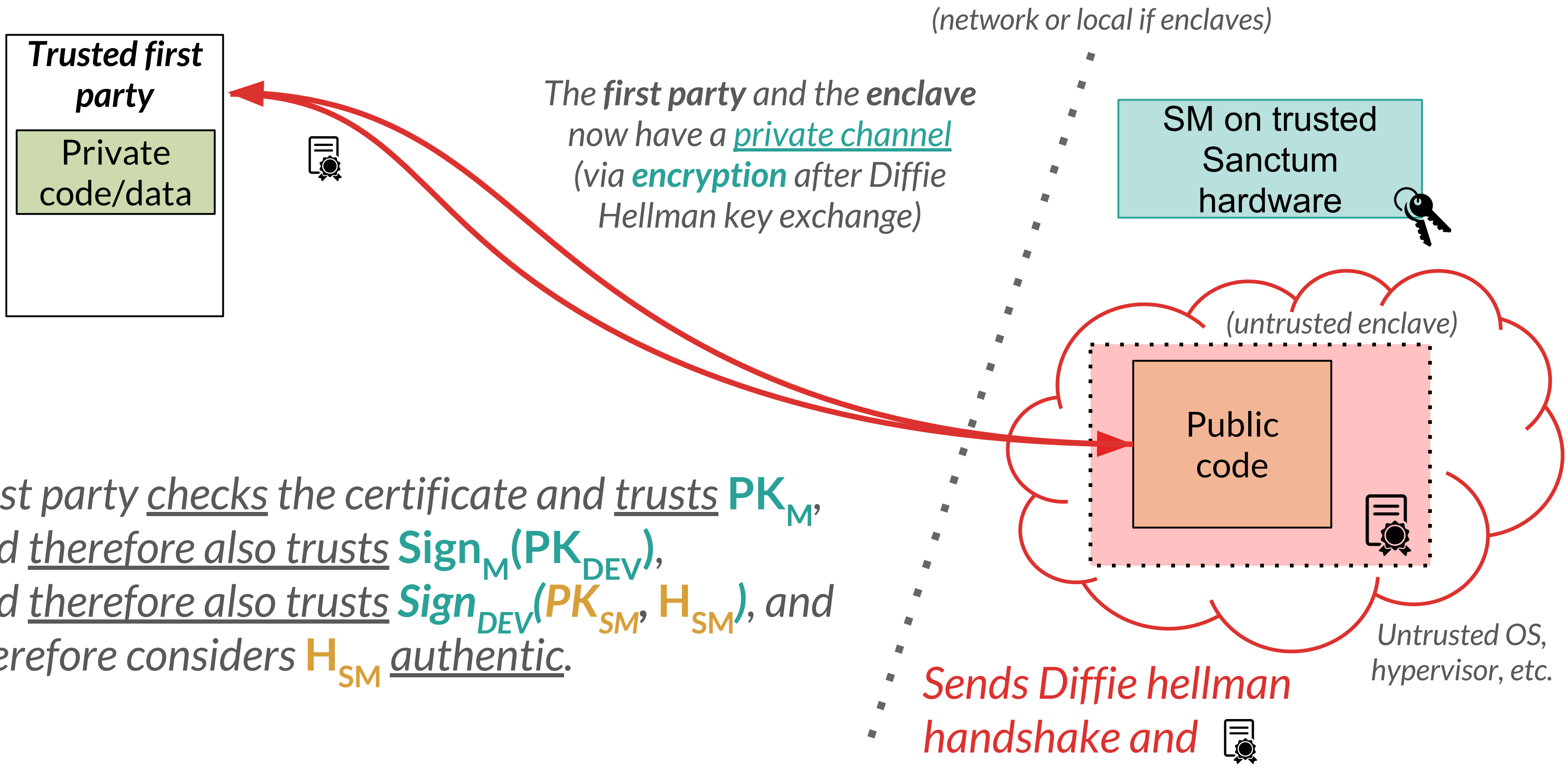
Cryptographic measurement of the enclave

Untrusted OS, hypervisor, etc.

Enclave receives its attestation

First party knows and trusts **PK**$_{M}$, the trusted manufacturer's public key

Remote attestation of enclaves (3/5)

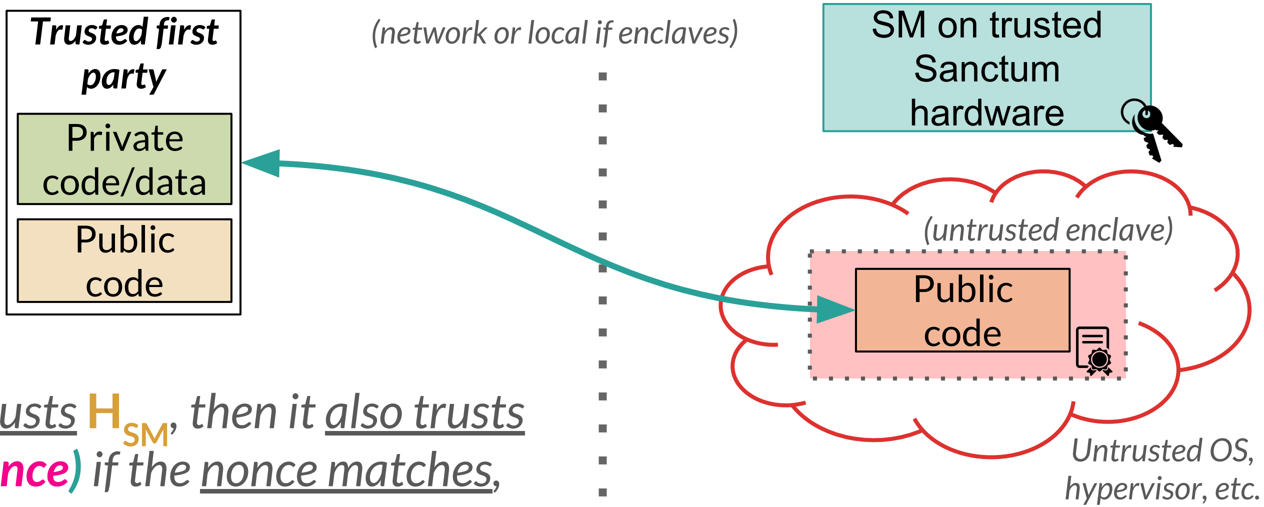

(network or local if enclaves)
Trusted first party
Private code/data
SM on trusted Sanctum hardware
(untrusted enclave)
Public code
Untrusted OS, hypervisor, etc.

*The **first party** and the **enclave** now have a private channel (via **encryption** after Diffie Hellman key exchange)*

Sends Diffie hellman handshake and

First party checks the certificate and trusts **PK_M**, *and therefore also trusts* **$Sign_M(PK_{DEV})$**, *and therefore also trusts* **$Sign_{DEV}(PK_{SM}, H_{SM})$**, *and therefore considers* **H_{SM}** *authentic.*

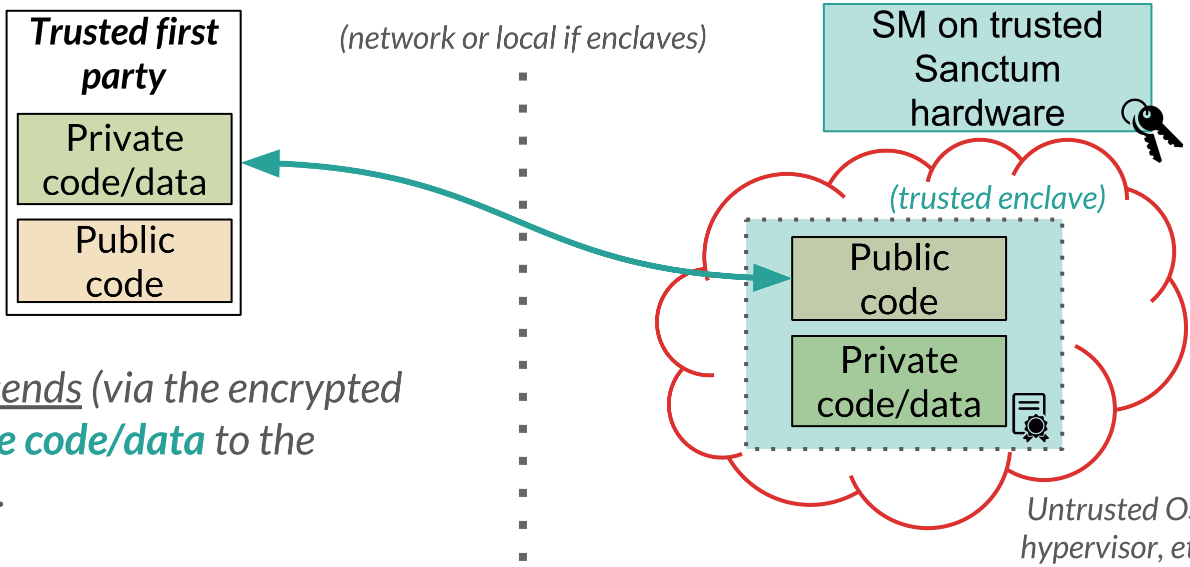
Remote attestation of enclaves (4/5)


Trusted first party
Private code/data
Public code
(network or local if enclaves)
SM on trusted Sanctum hardware
(untrusted enclave)
Public code
Untrusted OS, hypervisor, etc.

If the first party trusts H_{SM}, then it also trusts $Sign_{SM}(H_{Enclave}, nonce)$ if the nonce matches, and therefore considers $H_{Enclave}$ authentic.

*If $H_{Enclave}$ matches the expected value, then the first party can **trust the enclave**.*

Remote attestation of enclaves (5/5)



The first party sends (via the encrypted channel) ***private code/data*** *to the trusted enclave.*

The enclave's ***initial state*** *and* ***isolation*** *are authenticated (and trusted).*

The enclaved application ***must not have leaks or vulnerabilities;***

The enclave performs its computation (which may communicate with the OS or other parties, use other data, send results to the first party, etc.).

The SM guarantees it remains isolated.

Detail: attestation in Sanctum

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.)

Remote user

Select *primes* p, g.

Generate random A

Compute (g^A *mod p*)

Send

p, *g*,

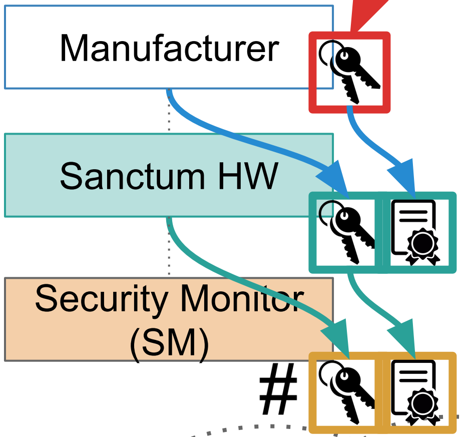
(g^A *mod p*)

Does remote user trust

metadata, 📜, 📜 ?

Send

M



Private key corresponding to a well-known public key

Sanctum computer

Generate random **B**, compute $\mathbf{g^B}$

M = {g^A, g^B, **metadata**, 📜 📜 },

signed with with 🔑*.

enclave

Compute symmetric key

K = $(g^B)^A$ mod p

Both parties now share a secret key: **K**

Compute symmetric key

K = $(g^A)^B$ mod p