ECE 382N-Sec (FA25):

L8: TEE Designs

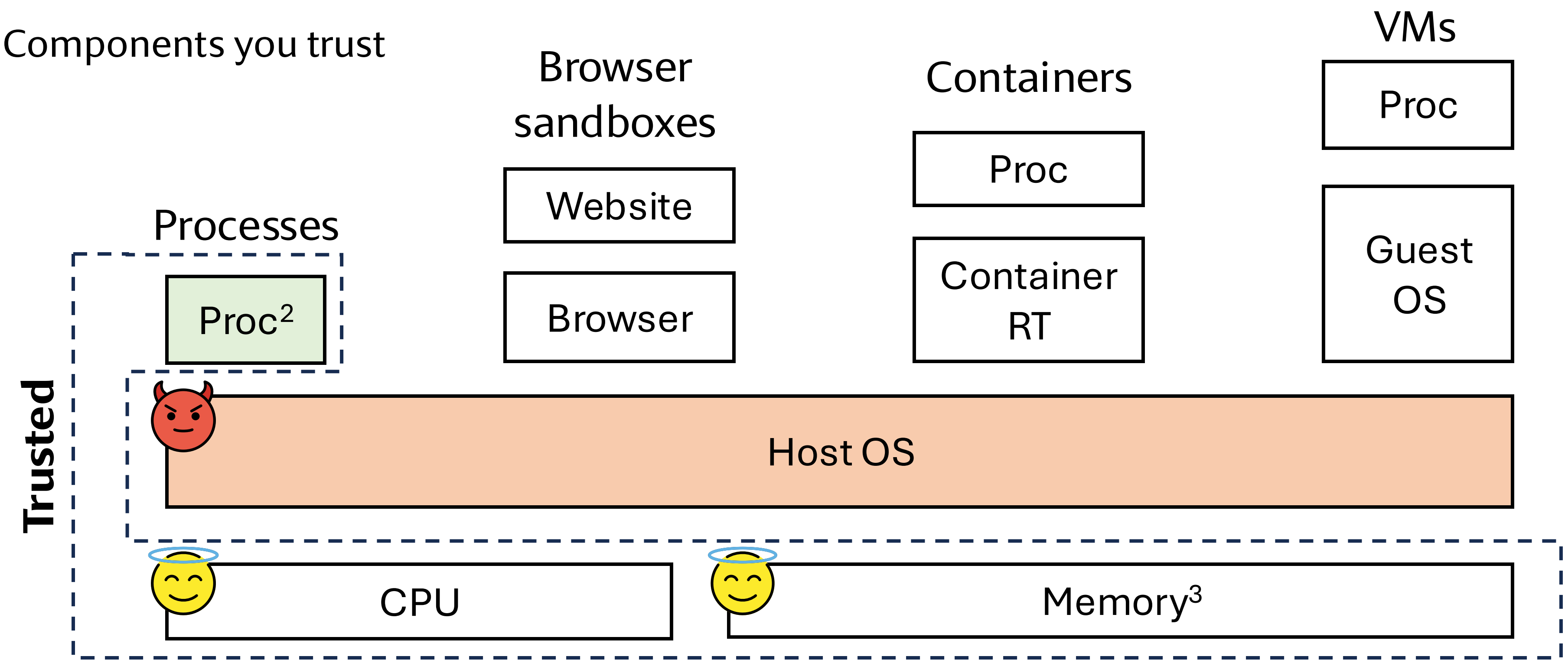
Neil Zhao

neil.zhao@utexas.edu

Trusted-Execution Environments (TEE)[1]

🟩 Your program

😇 Components you trust



[1] TEE is a somewhat overloaded term. We focus on hardware-based TEEs

[2] The process may be divided into trusted and untrusted parts

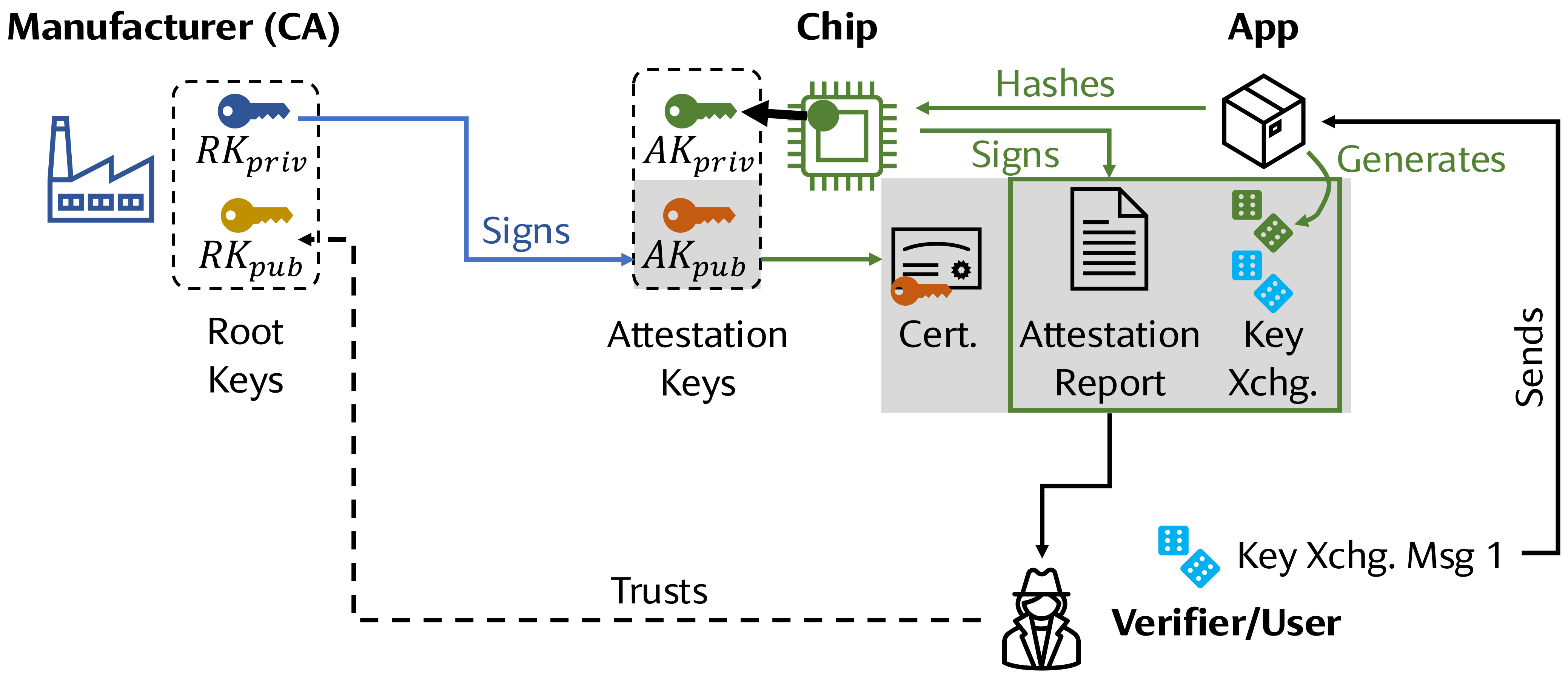
[3] Depending on the memory type and threat model, it may or may not be trusted

(Common*) Security Goals of TEEs

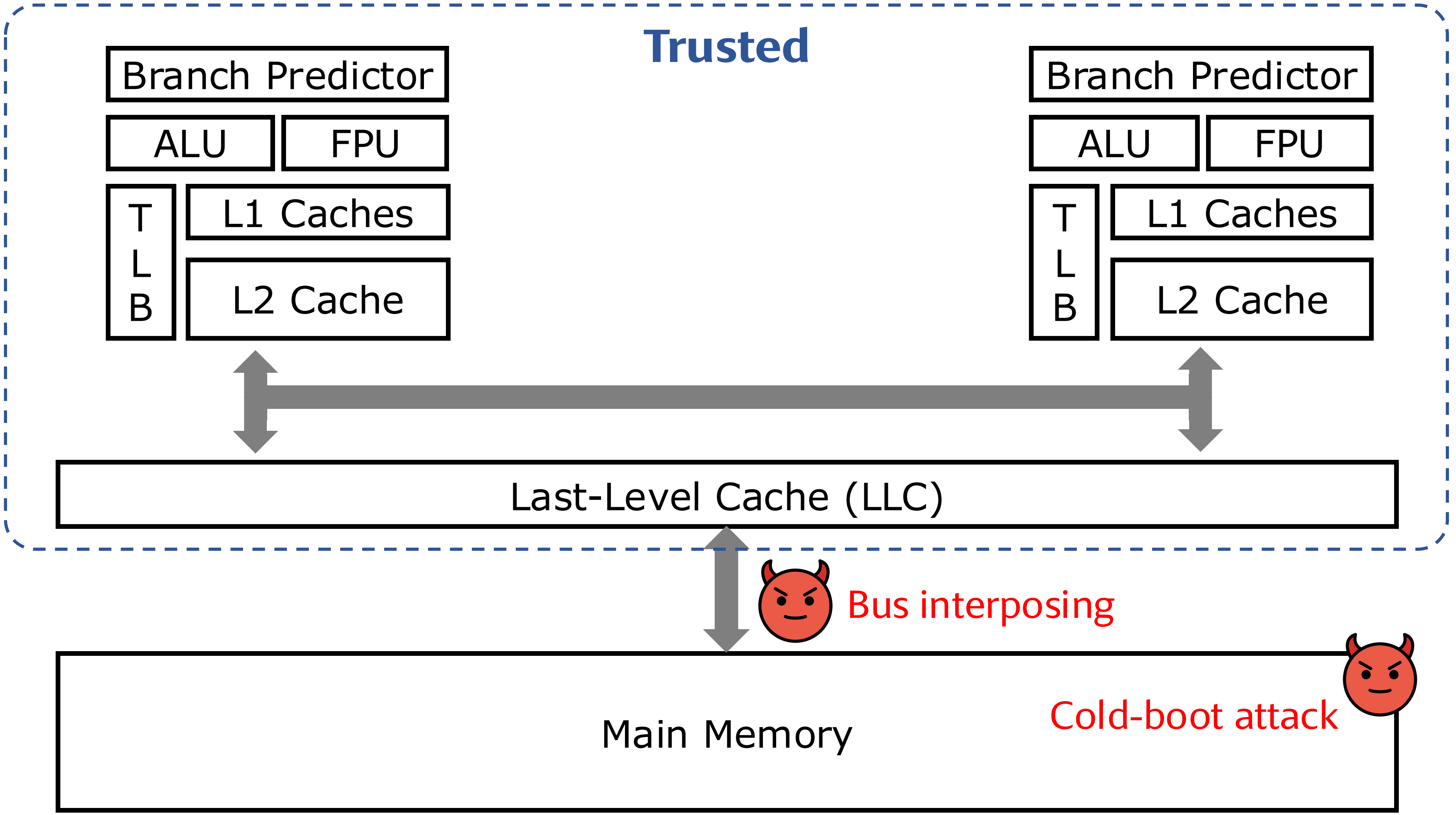
			Example Attacks	
			Software Attack	**Physical Attack**
✓	**Confidentiality**	Attacker cannot directly access my private program states (Side channel? Spectre?)	OS reads my pages	Bus snooping
✓	**Integrity**	Attacker cannot tamper with my program states (**Freshness:** Program state is up-to-date)	OS writes my pages	? Bus spoofing
✗	**Availability**	Attacker refuses to execute or give enough resources to my program	OS allocates no CPU time	Pull the plug

*Many variants exist

Software Attestation

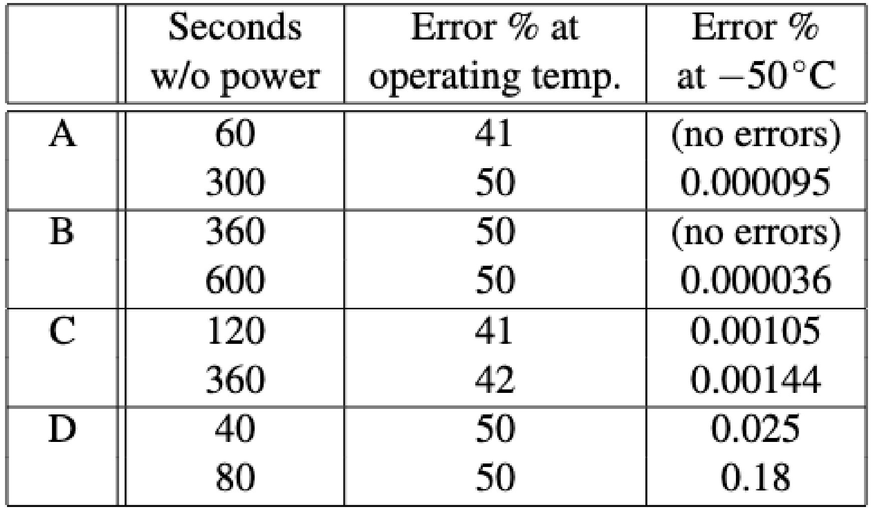


The Need for Memory Encryption and Integrity Protection



Cold-Boot Attack

Observation: Data in DRAM cells can survive for seconds after losing power
⇒ The window can be extended by cooling the DRAM to a low temperature



	Seconds w/o power	Error % at operating temp.	Error % at $-50°$C
A	60	41	(no errors)
	300	50	0.000095
B	360	50	(no errors)
	600	50	0.000036
C	120	41	0.00105
	360	42	0.00144
D	40	50	0.025
	80	50	0.18

Table 2: Effect of cooling on error rates

Source: Halderman et al., "Lest We Remember: Cold Boot Attacks on Encryption Keys," USENIX Sec '08

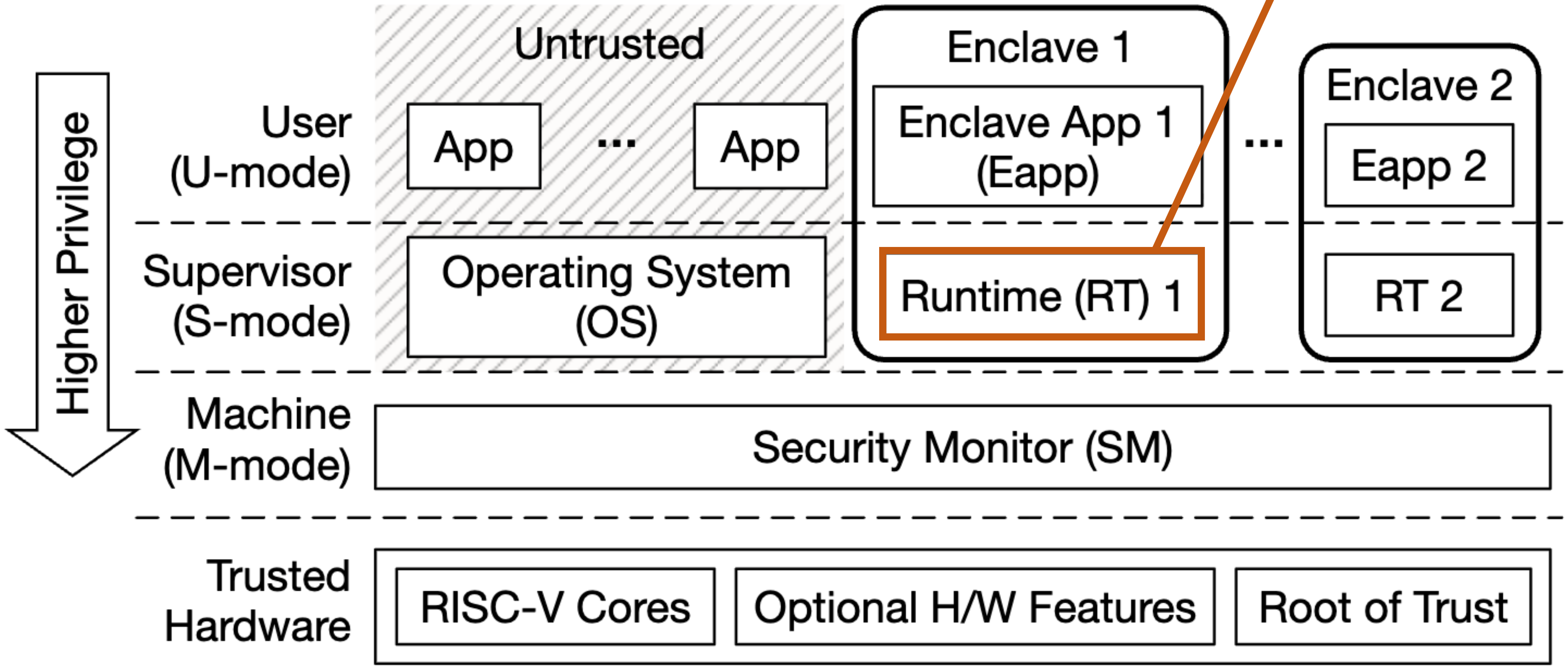
A Typical Attack Process:

- Cool down the DRAM
- Suddenly power off the machine
  ⇒ Take a "snapshot" of the memory
- Boot into a USB drive that contains the program for dumping the memory

Can dump the entire memory, including the disk encryption key found in the memory

Demo: https://www.youtube.com/watch?v=XfUlRsE3ymQ

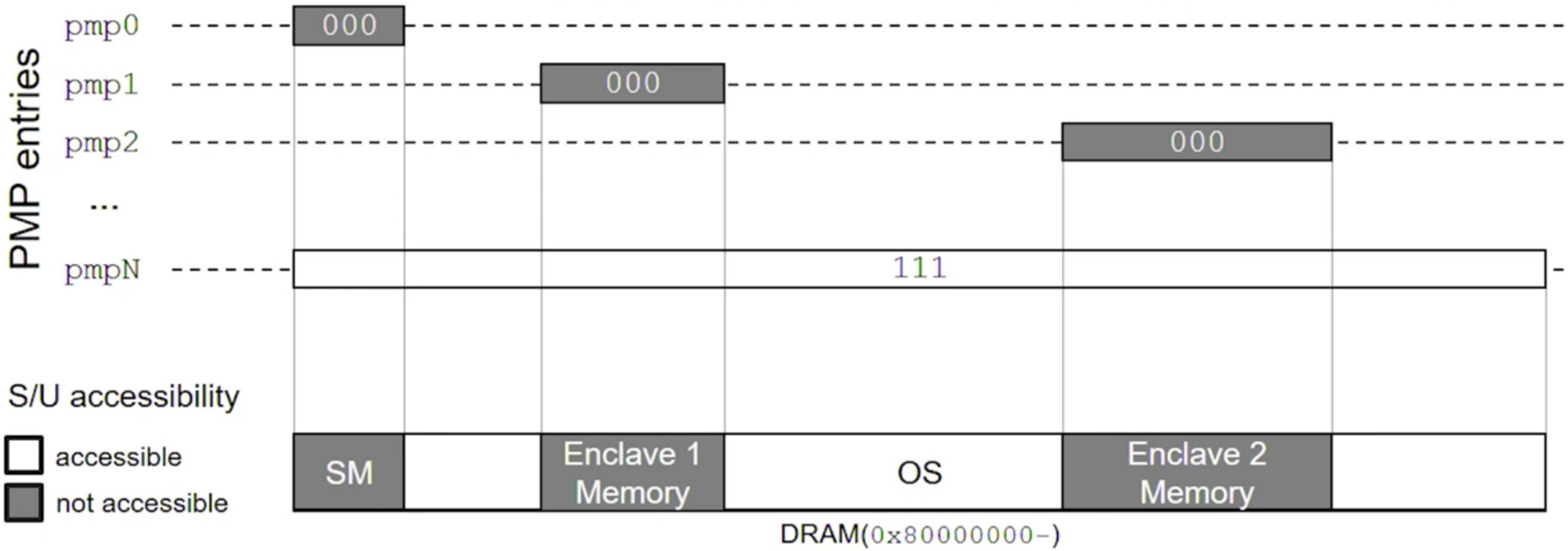
Keystone Enclave



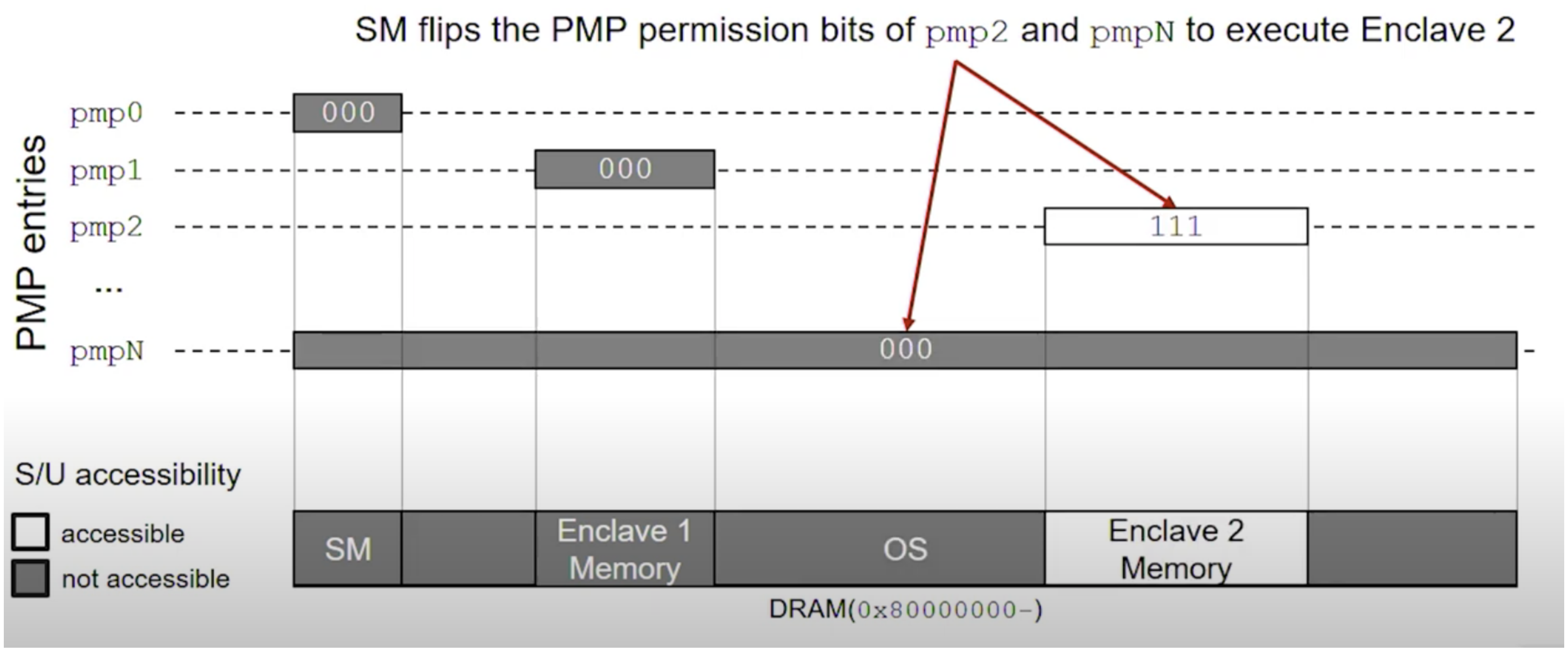
Kernel-like. Handles paging, thread scheduling, ...

Keystone Enclave

The enclave can only exit by an SM SBI call.
The SM flips the permissions before entering the untrusted context.

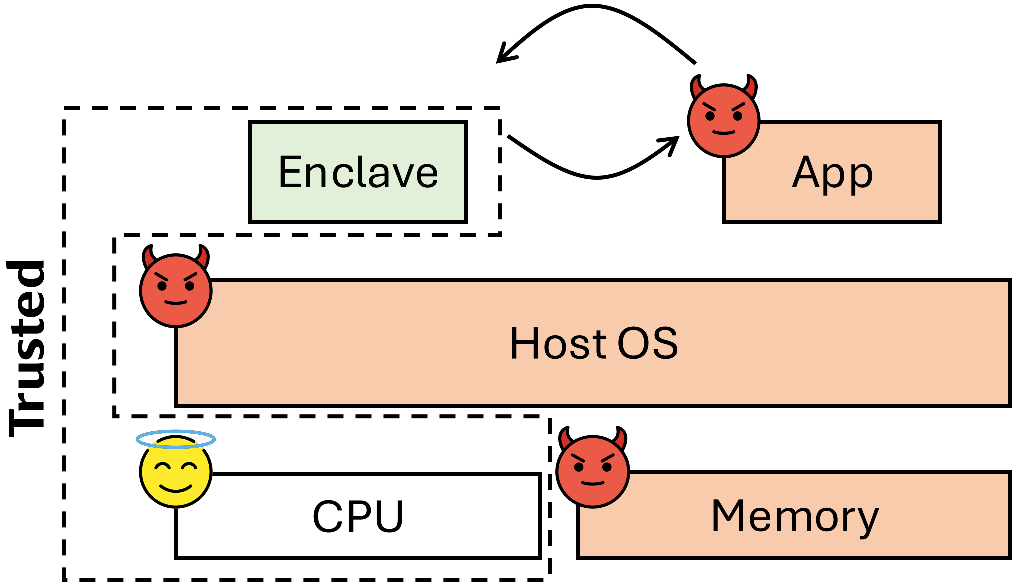


Keystone Enclave



SM flips the PMP permission bits of pmp2 and pmpN to execute Enclave 2

Intel SGX-1 Overview



Resource management is delegated to the untrusted OS, who

- Allocates and frees memory
- Schedules enclave threads
- Serves interrupts
- …

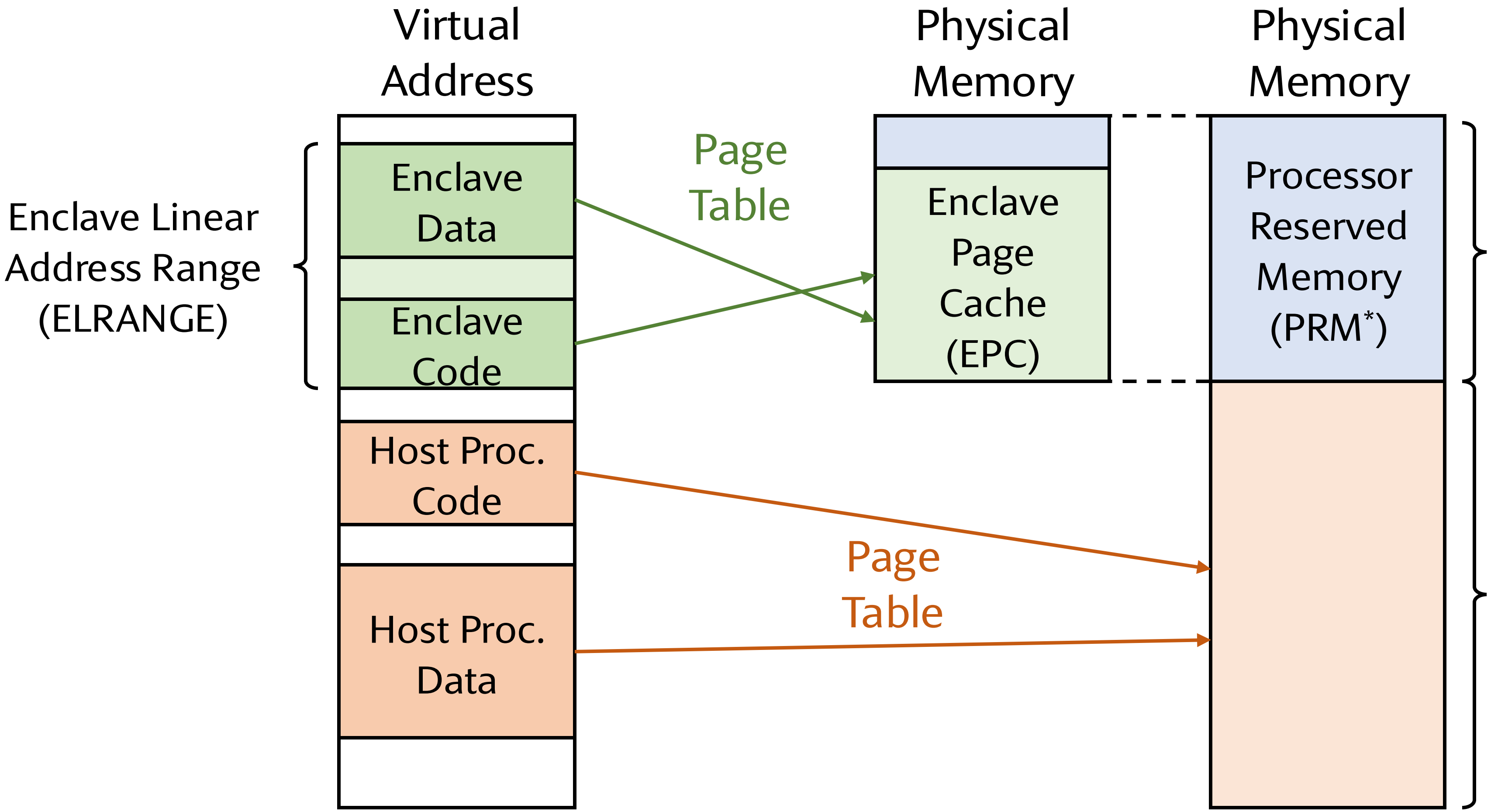
Before We Start

- We focus on the legacy Intel SGX-1
  - Successors: Intel SGX-2, Intel TDX
  - SGX-1 is well studied and serves as a good baseline to learn

- The exact Intel SGX design is complex, full of acronyms, and often undocumented. We simplified our discussion to help you understand the general TEE design challenges and solutions. Please consult Intel's Software Developer's Manual (SDM) on how to properly use it

- Why a certain design point is chosen is often undocumented. Therefore, some explanations are based on educated guesses

Intel SGX-1 Isolation Overview



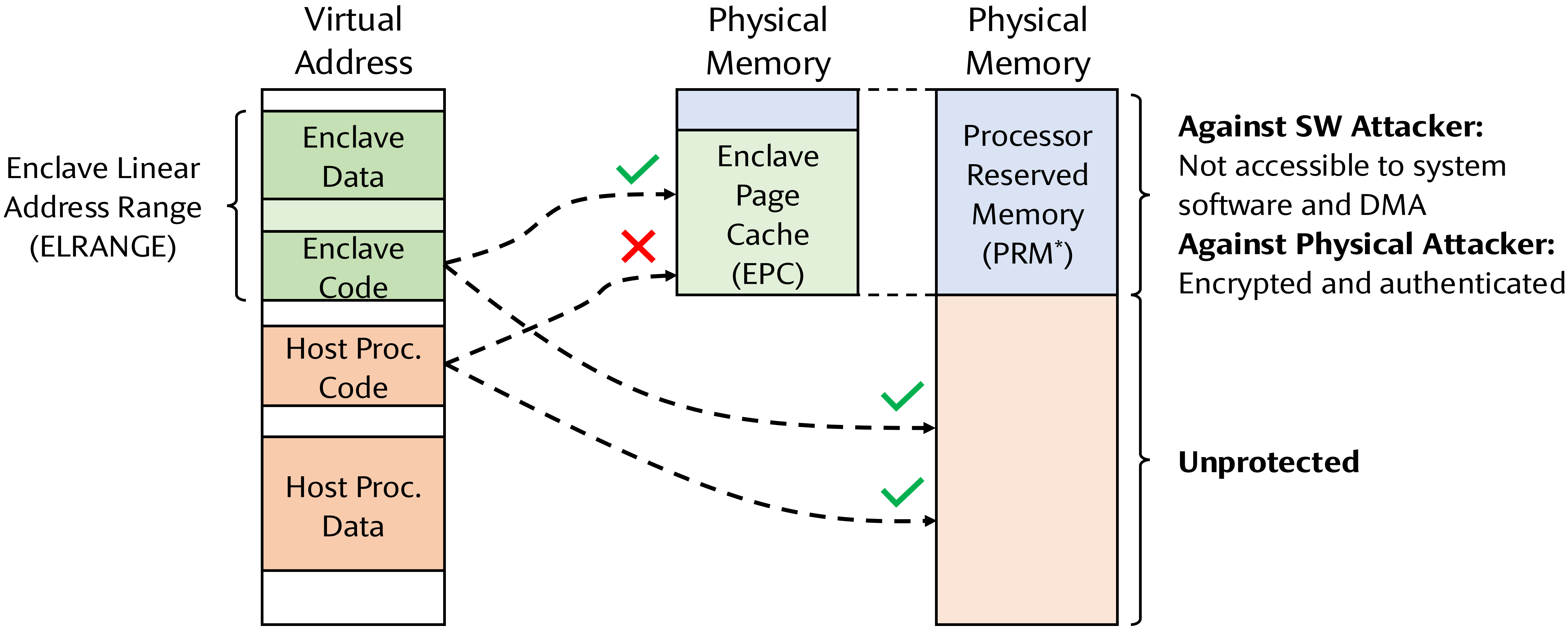
Against SW Attacker:
Not accessible to system software and DMA

Against Physical Attacker:
Encrypted and authenticated

Unprotected

*Not to scale. PRM is often 128MB

Intel SGX-1 Isolation Overview



Virtual Address

Enclave Linear Address Range (ELRANGE)

Enclave Data

Enclave Code

Host Proc. Code

Host Proc. Data

Physical Memory

Enclave Page Cache (EPC)

Physical Memory

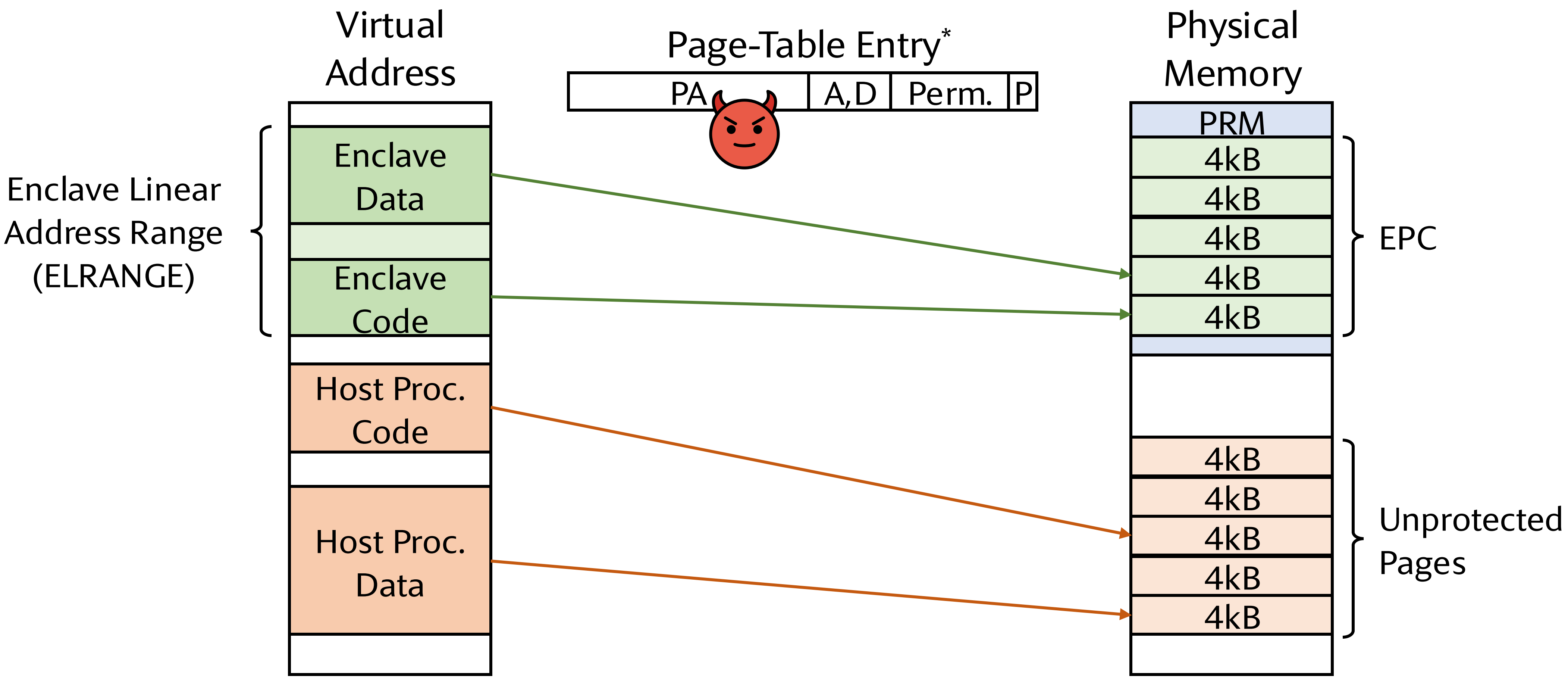
Processor Reserved Memory (PRM*)

Against SW Attacker:
Not accessible to system software and DMA
Against Physical Attacker:
Encrypted and authenticated

Unprotected

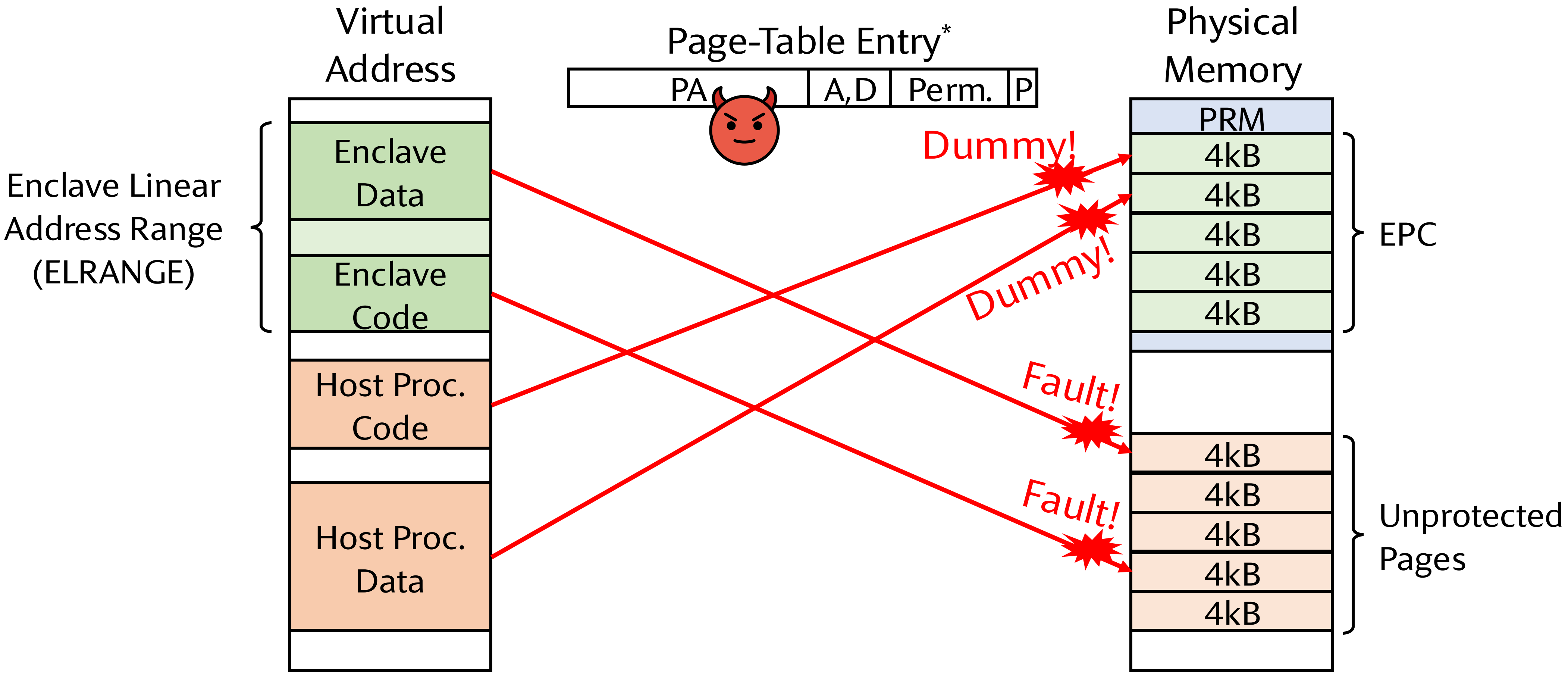
*Not to scale. PRM is often 128MB

The Untrusted OS Manages the Page Mapping



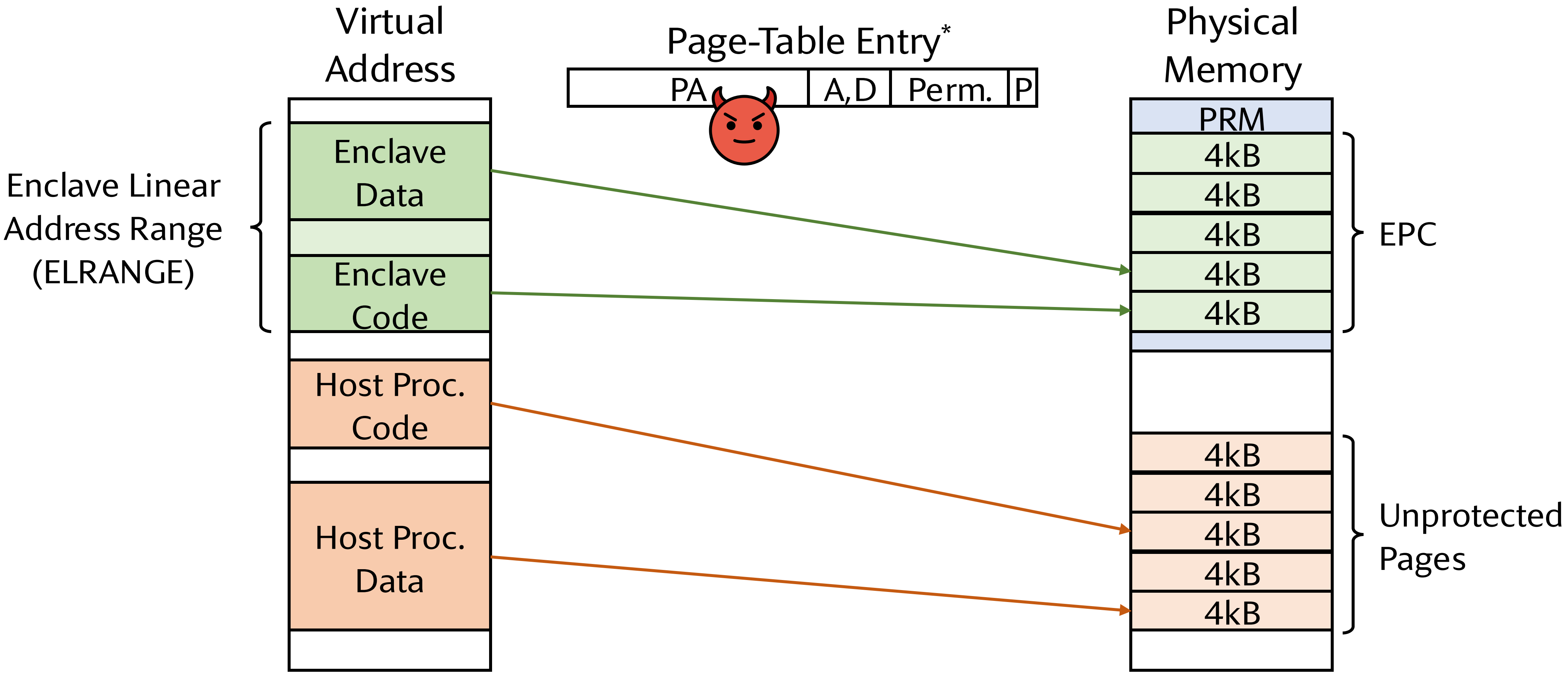
*Many fields omitted

The Untrusted OS Manages the Page Mapping



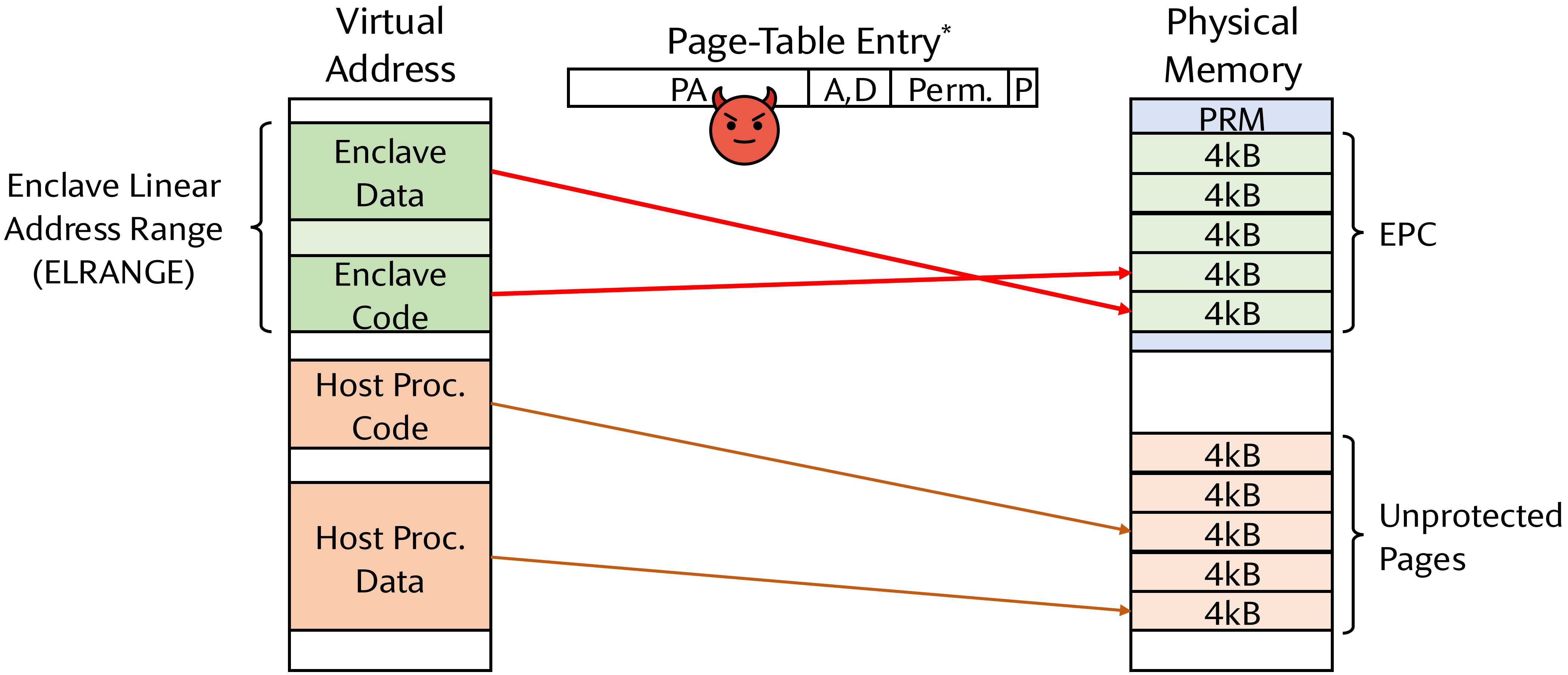
*Many fields omitted

What Else Can Go Wrong?



*Many fields omitted

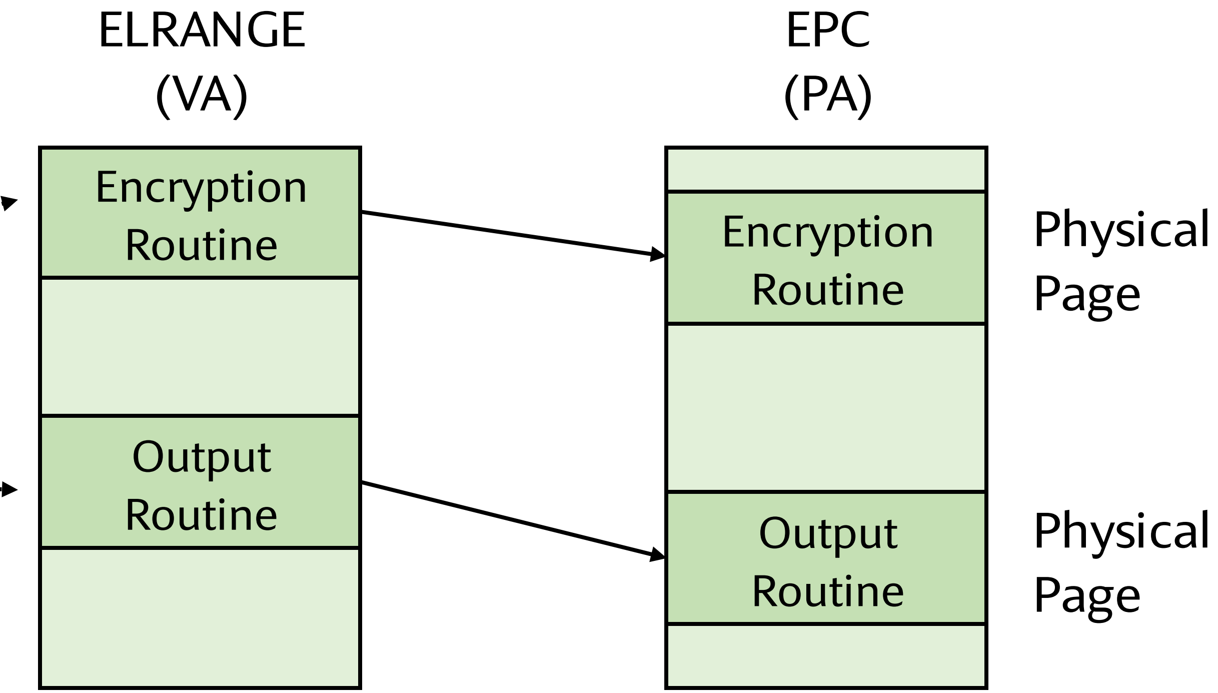
Page Swapping Attack (Similar to Splicing)



*Many fields omitted

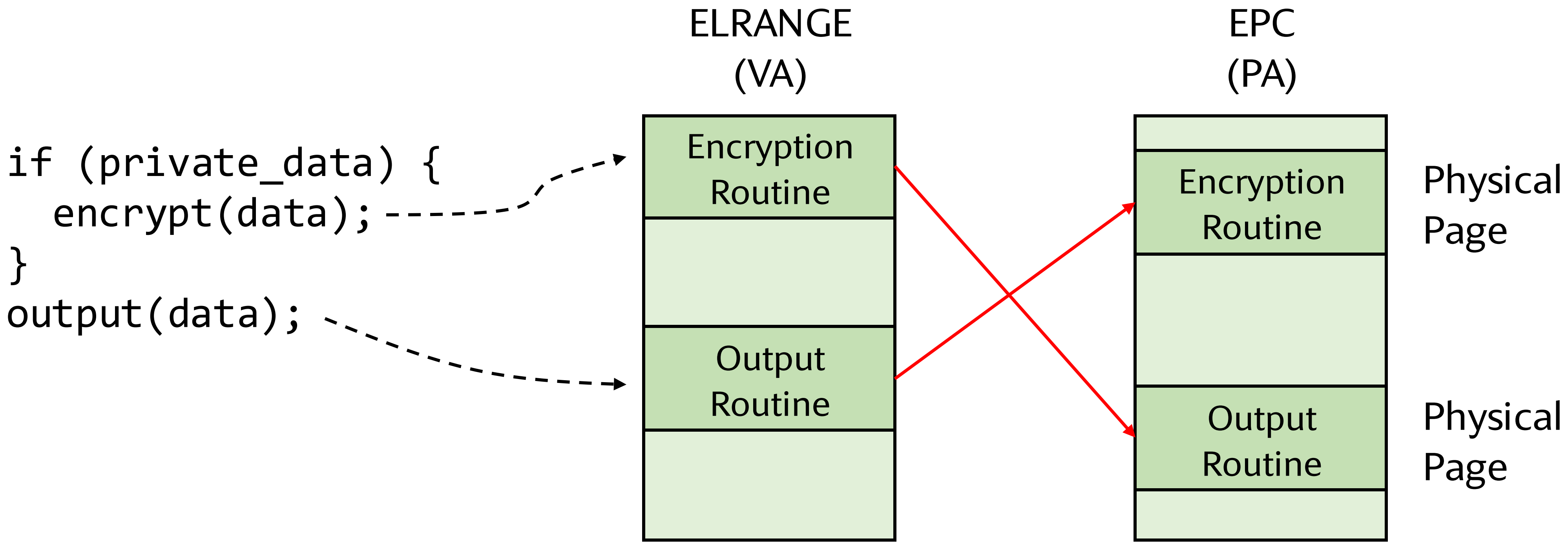
Page Swapping Attack (Similar to Splicing)

```
if (private_data) {
  encrypt(data);
}
output(data);
```



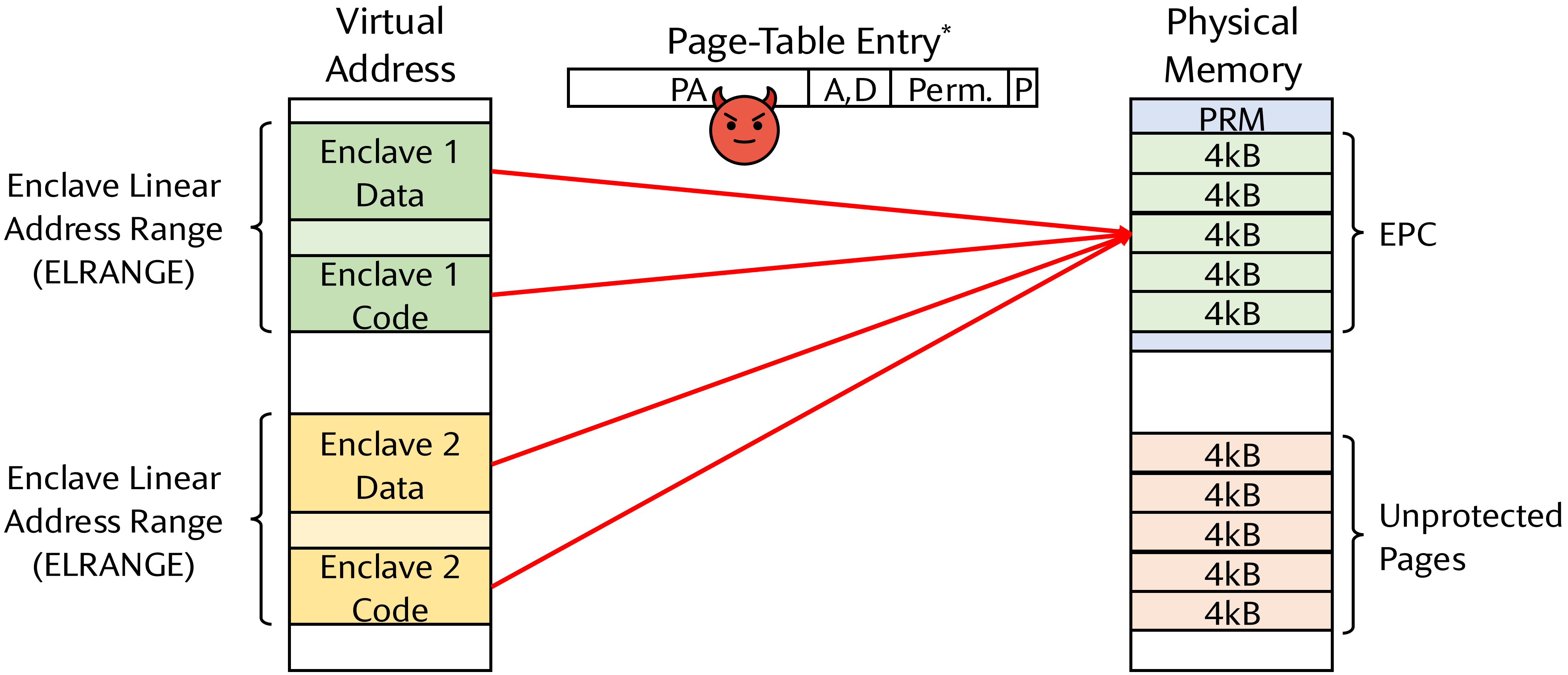
Page Swapping Attack (Similar to Splicing)

Private data are leaked without encryption



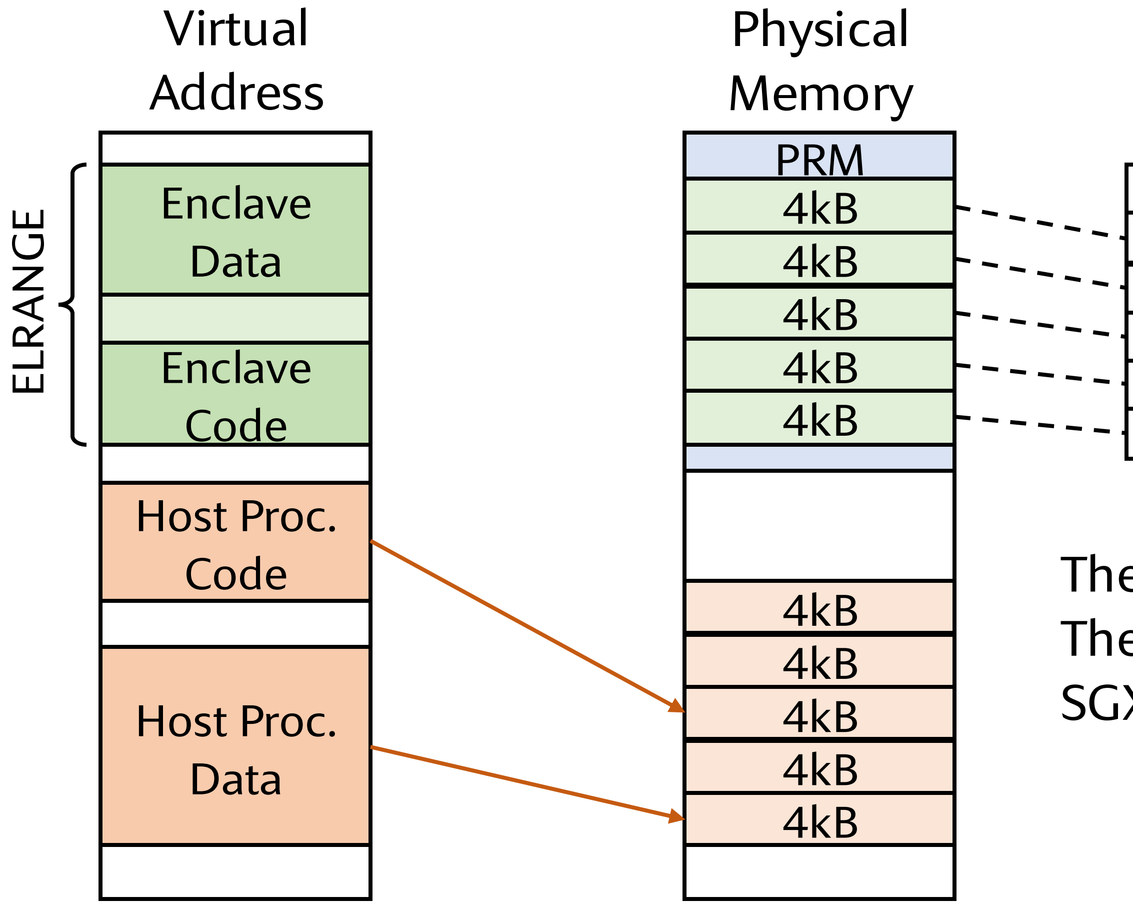
```
if (private_data) {
  encrypt(data);
}
output(data);
```

Intra- and Inter-Enclave Aliasing



*Many fields omitted

EPC Metadata (EPCM)

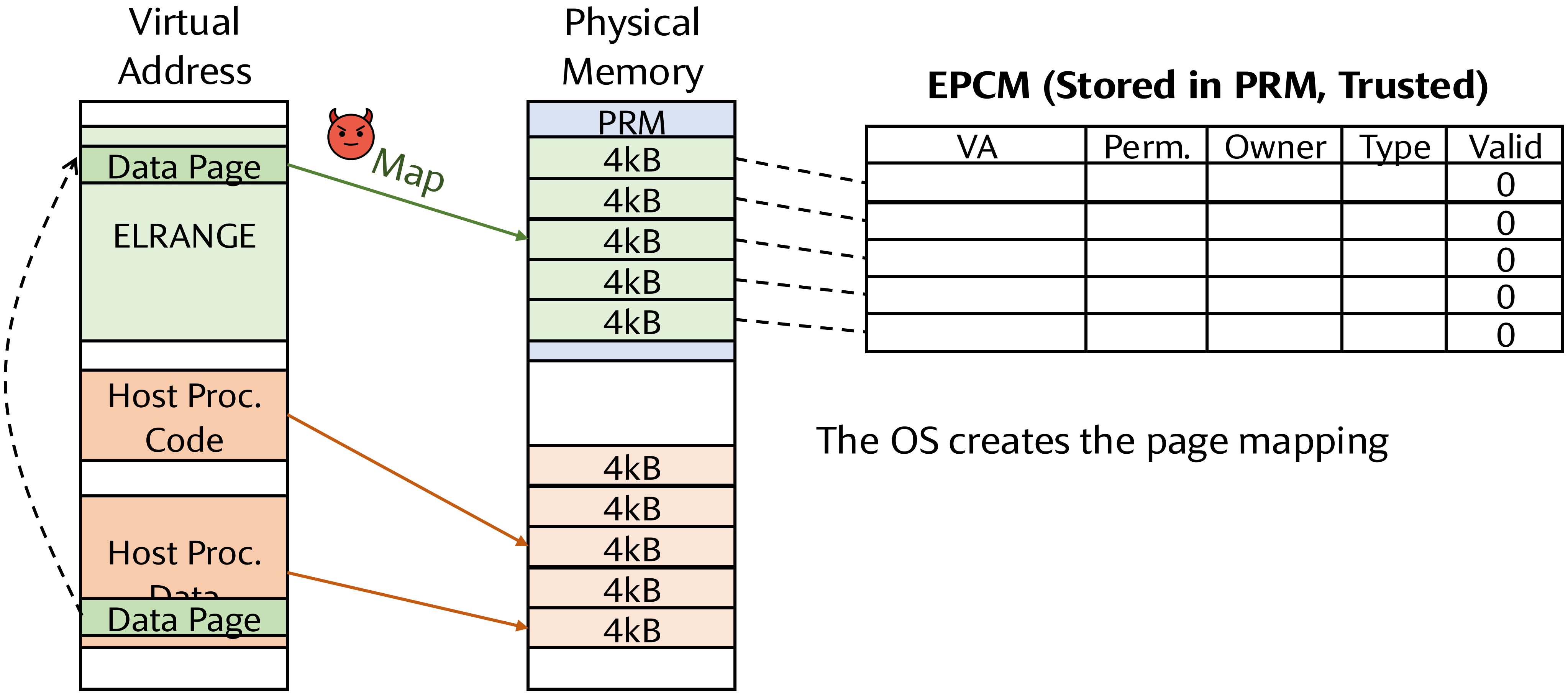


EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid
				0
				0
				0
				0
				0

The OS maintains its own EPC page free list.
The OS decides which “free” page to use.
SGX validates the OS’s decisions

Allocating Memory and Enclave Initialization

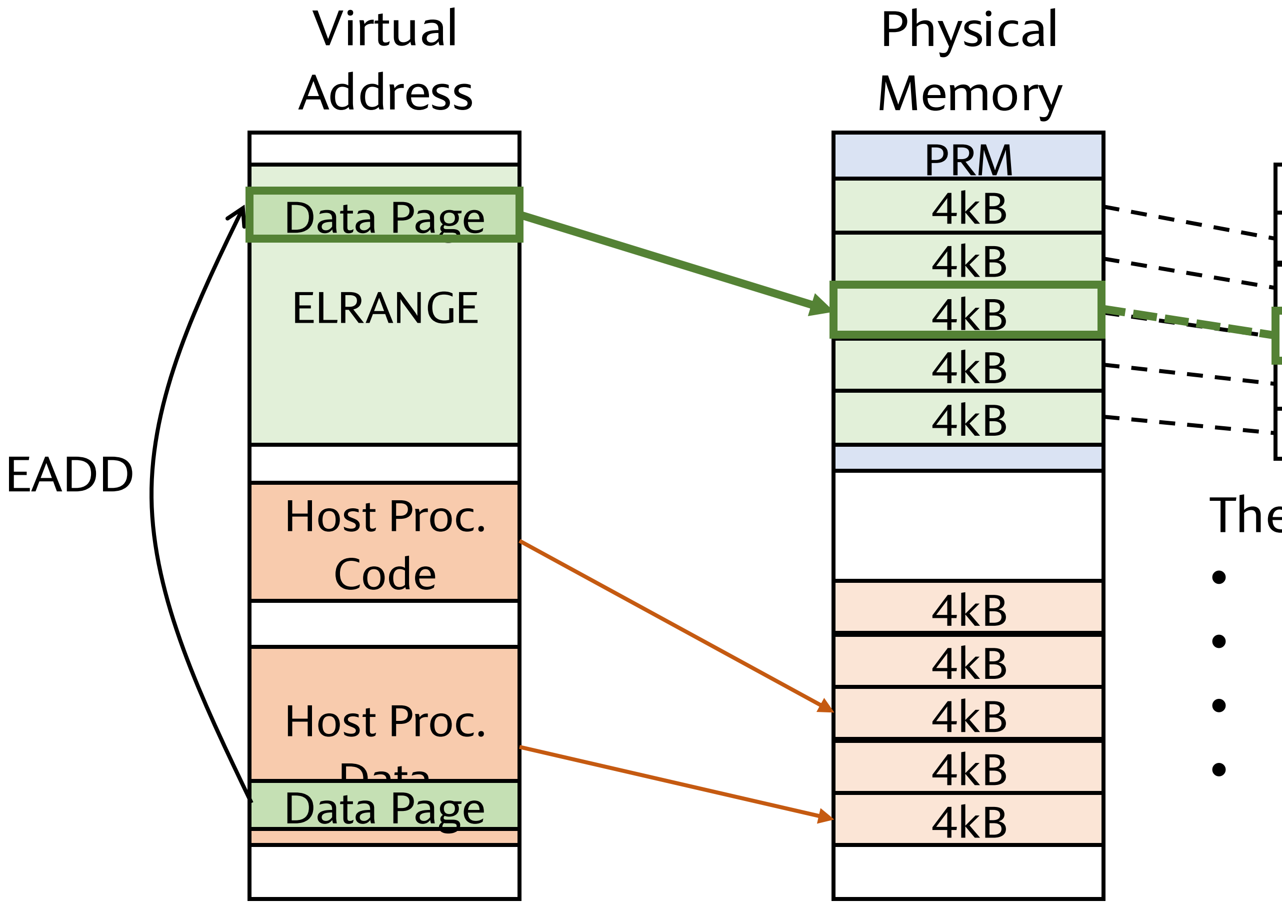


EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid
				0
				0
				0
				0
				0

The OS creates the page mapping

Allocating Memory and Enclave Initialization



EPCM (Stored in PRM, Trusted)

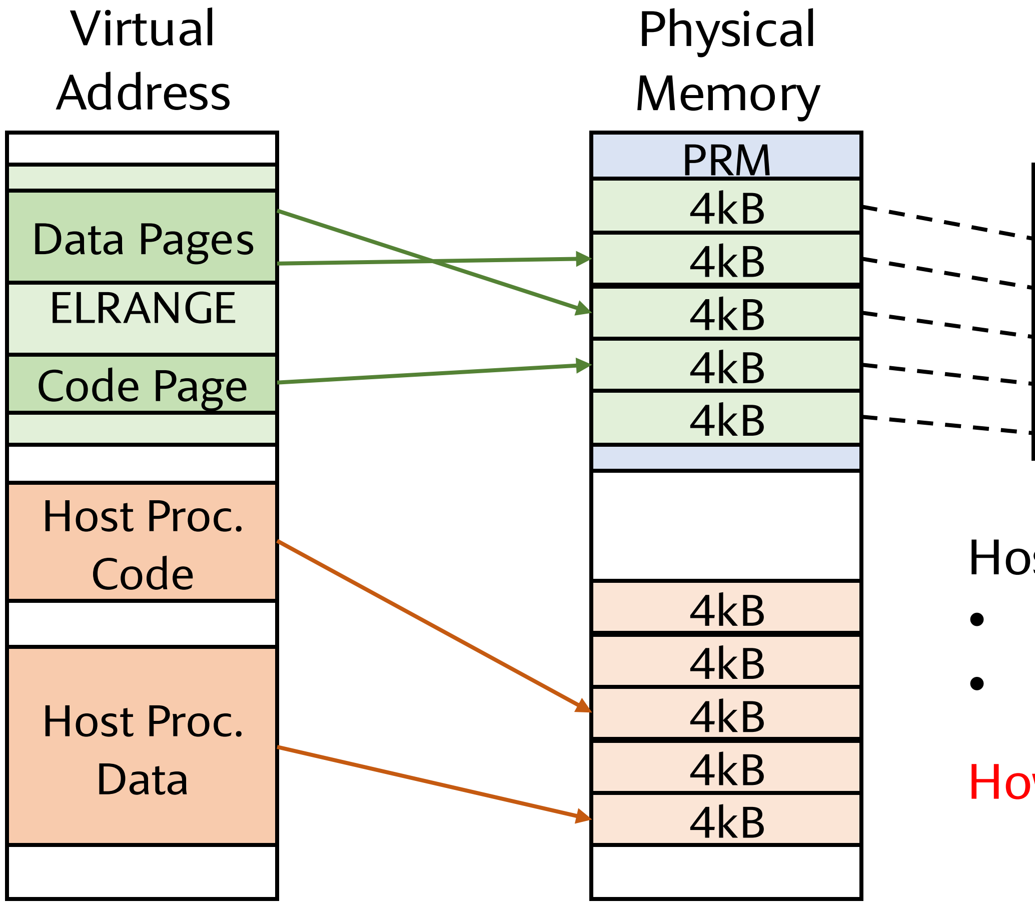
VA	Perm.	Owner	Type	Valid
				0
				0
0x1234000	RW	E1	REG	0→1
				0
				0

The OS executes EADD to copy the data:

- CPU translates the address
- CPU finds the EPCM entry
- **CPU checks if the page is allocated**
- If not CPU updates the EPCM entry and copies the data

Host executes EEXTEND to hash the content

Allocating Memory and Enclave Initialization



EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid
				0
0x1235000	RW	E1	REG	1
0x1234000	RW	E1	REG	1
0x40000	RX	E1	REG	1
				0

Host executes EINIT to finalize the enclave

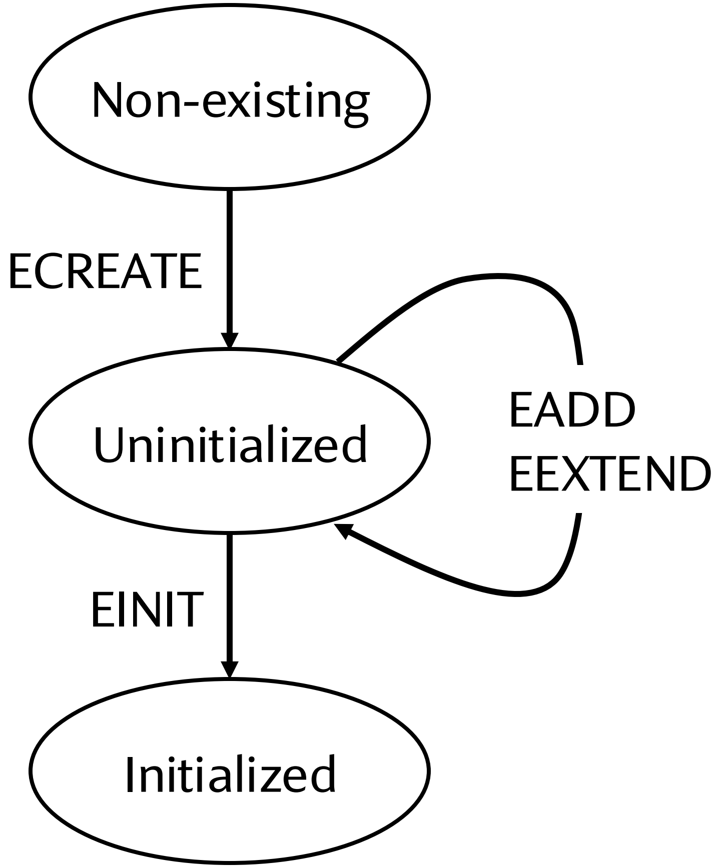
- No more EADDs and EEXTENDs
- Finalize the hash/measurement

How do we validate EPCM entries?

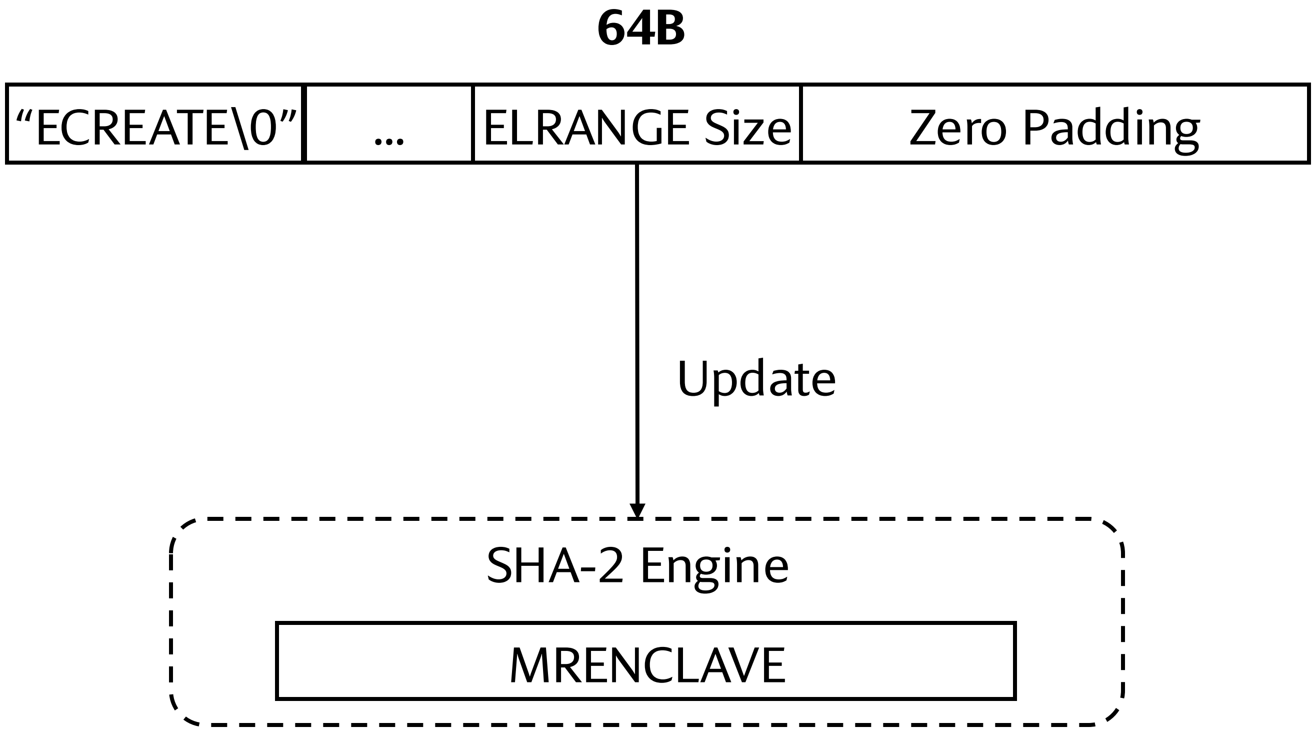
SGX Enclave Measurement

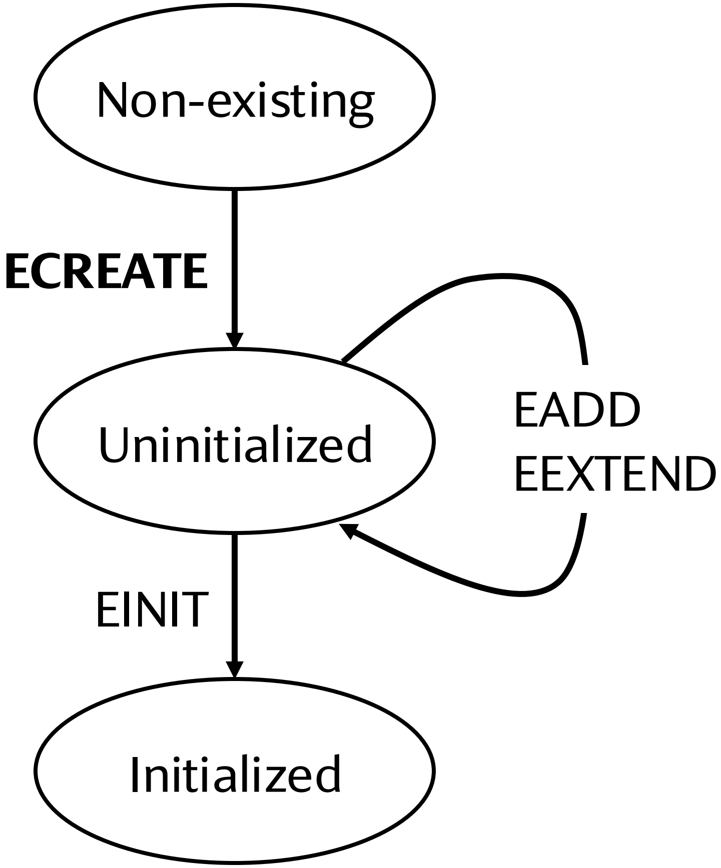
The sequence and operands of ECREATE, EADD, EEXTEND are recorded and then hashed/measured
⇒ Different execution sequence → Different measurement

- SGX uses 256-bit SHA-2 hash function (in=64B, out=32B)
- The measurement is stored inside **MRENCLAVE**

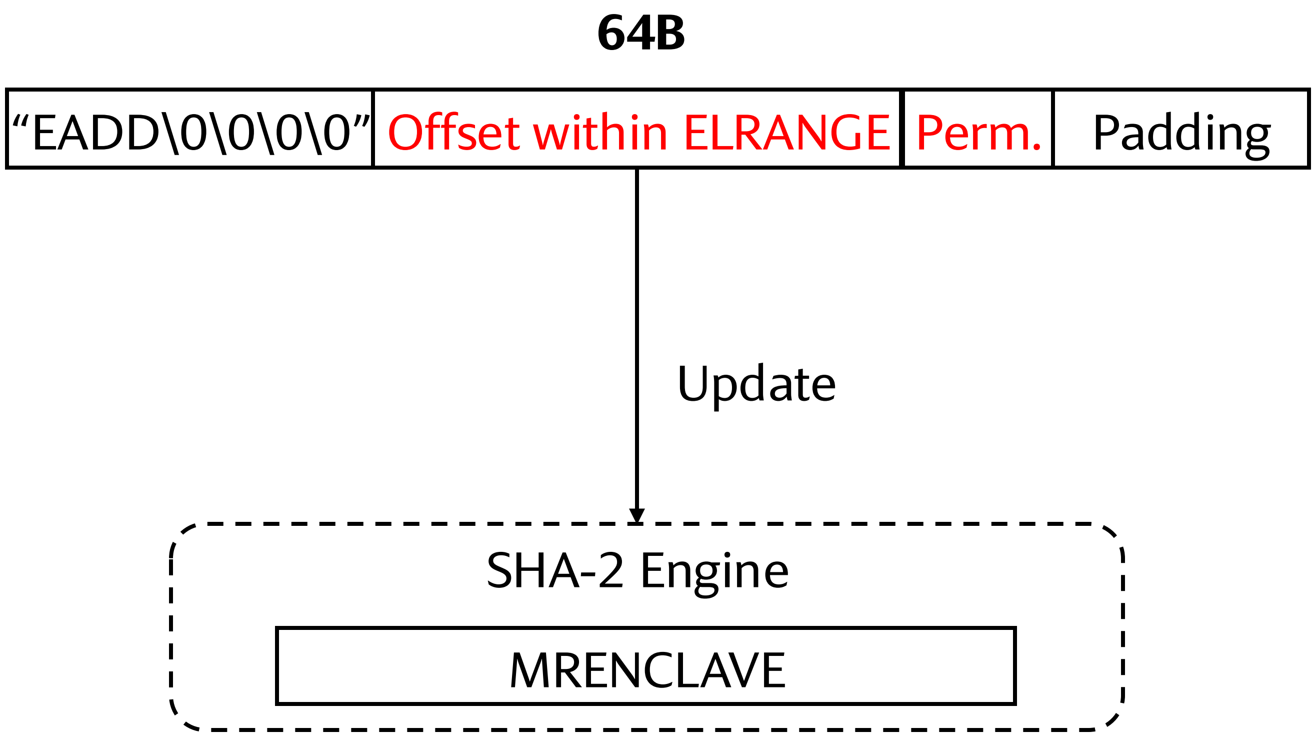


SGX Enclave Measurement - ECREATE





SGX Enclave Measurement - EADD



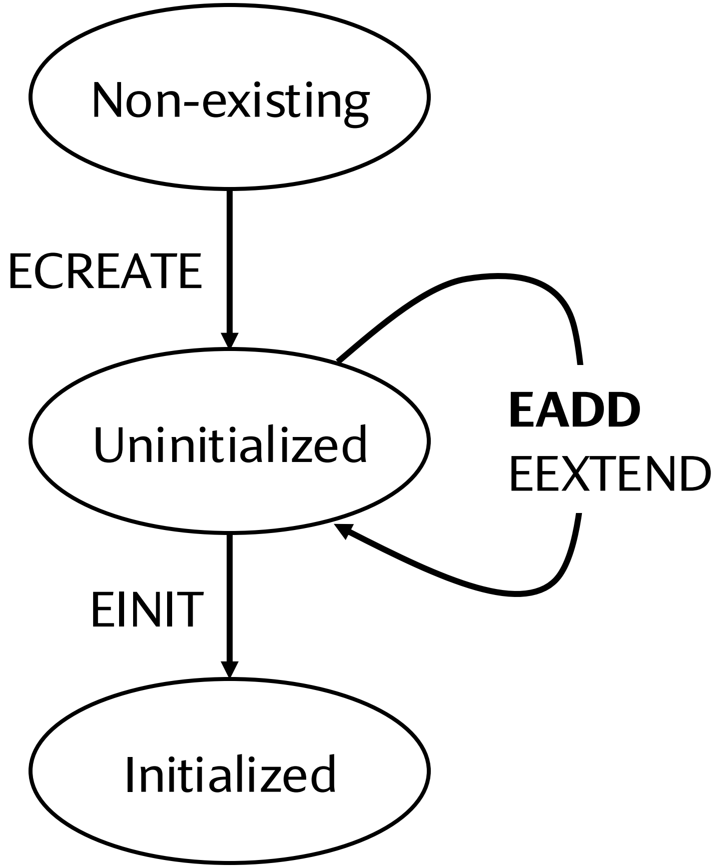
64B

"EADD\0\0\0\0" | Offset within ELRANGE | Perm. | Padding

Update

SHA-2 Engine

MRENCLAVE



Non-existing

ECREATE

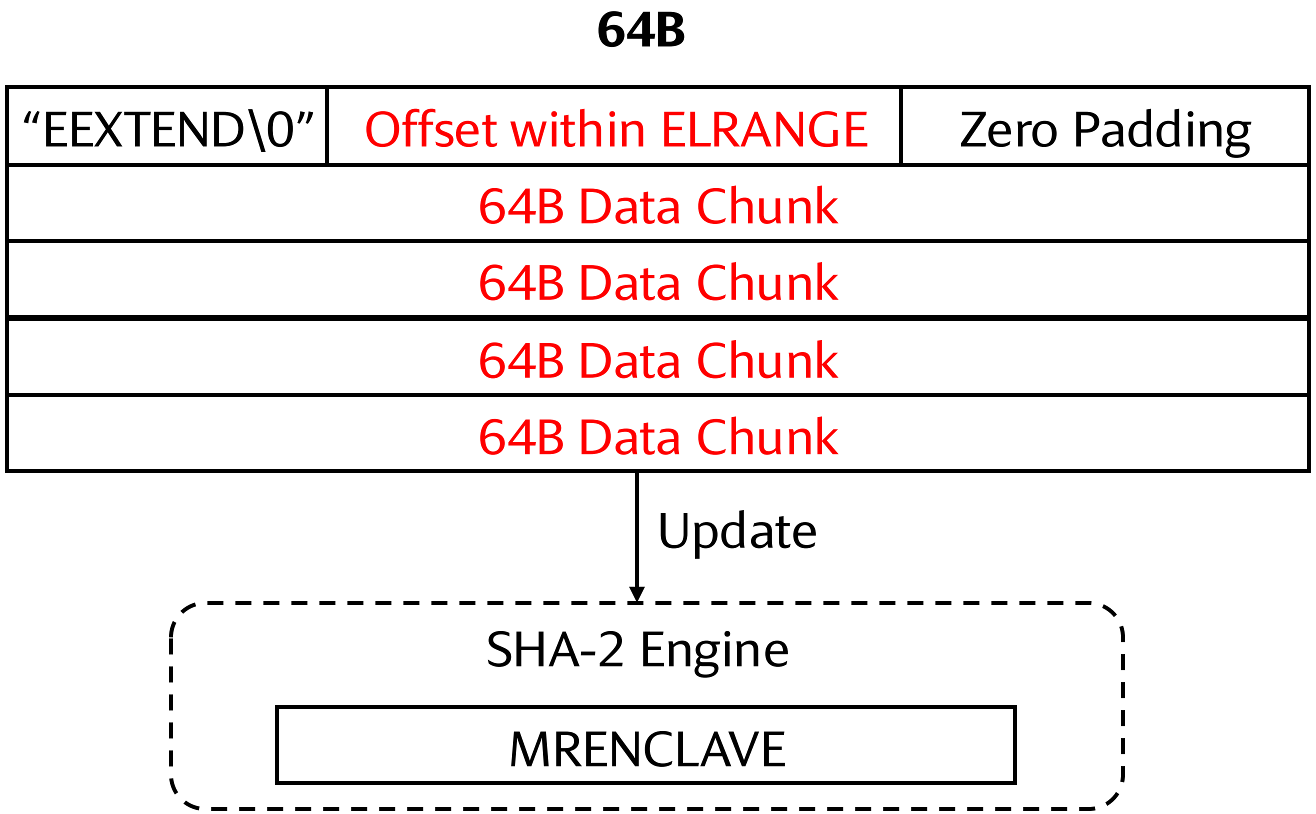
Uninitialized

EADD
EEXTEND

EINIT

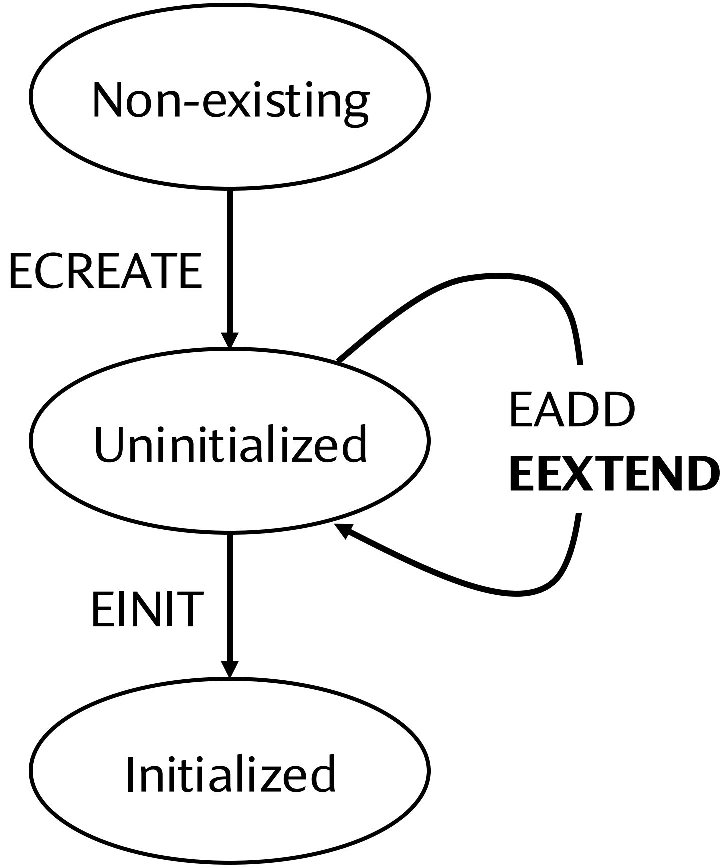
Initialized

SGX Enclave Measurement - EEXTEND



64B

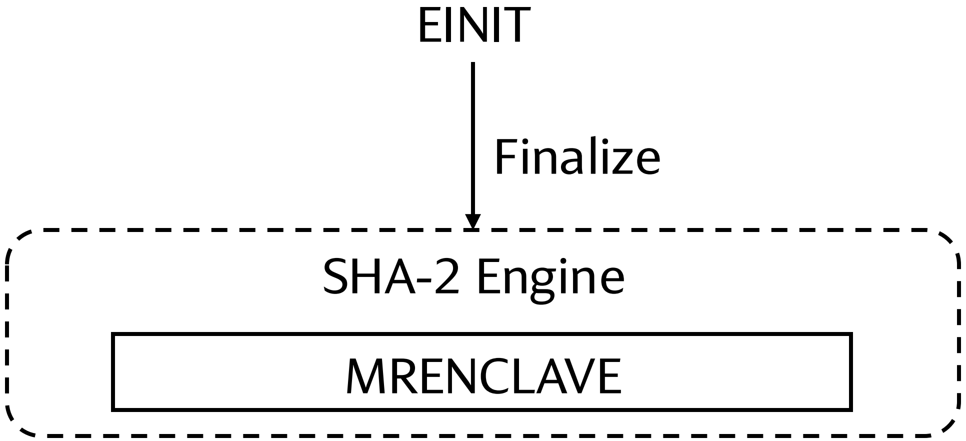
"EEXTEND\0"	Offset within ELRANGE	Zero Padding
64B Data Chunk		
64B Data Chunk		
64B Data Chunk		
64B Data Chunk		

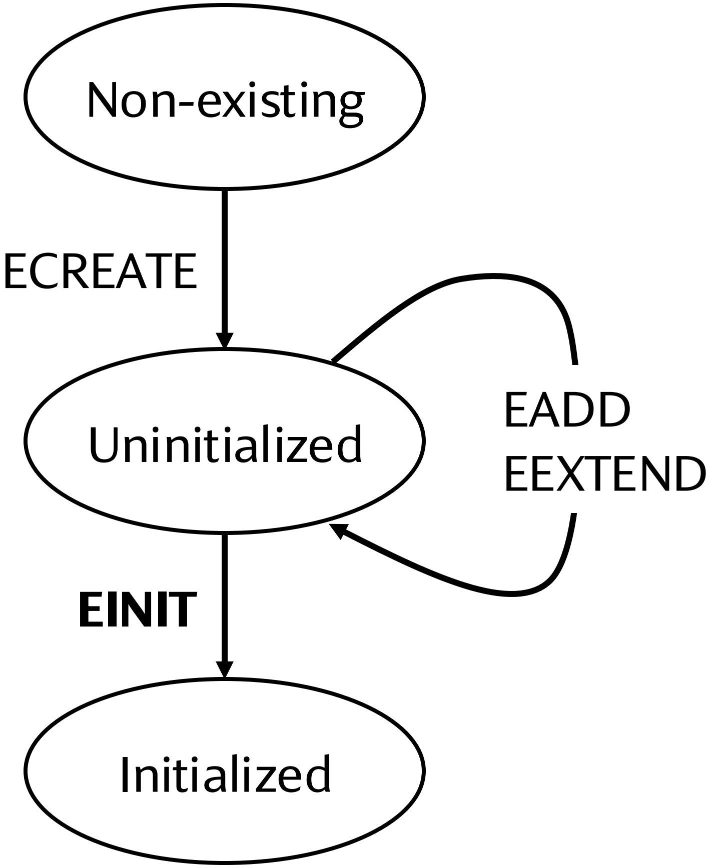


SGX Enclave Measurement - EINIT

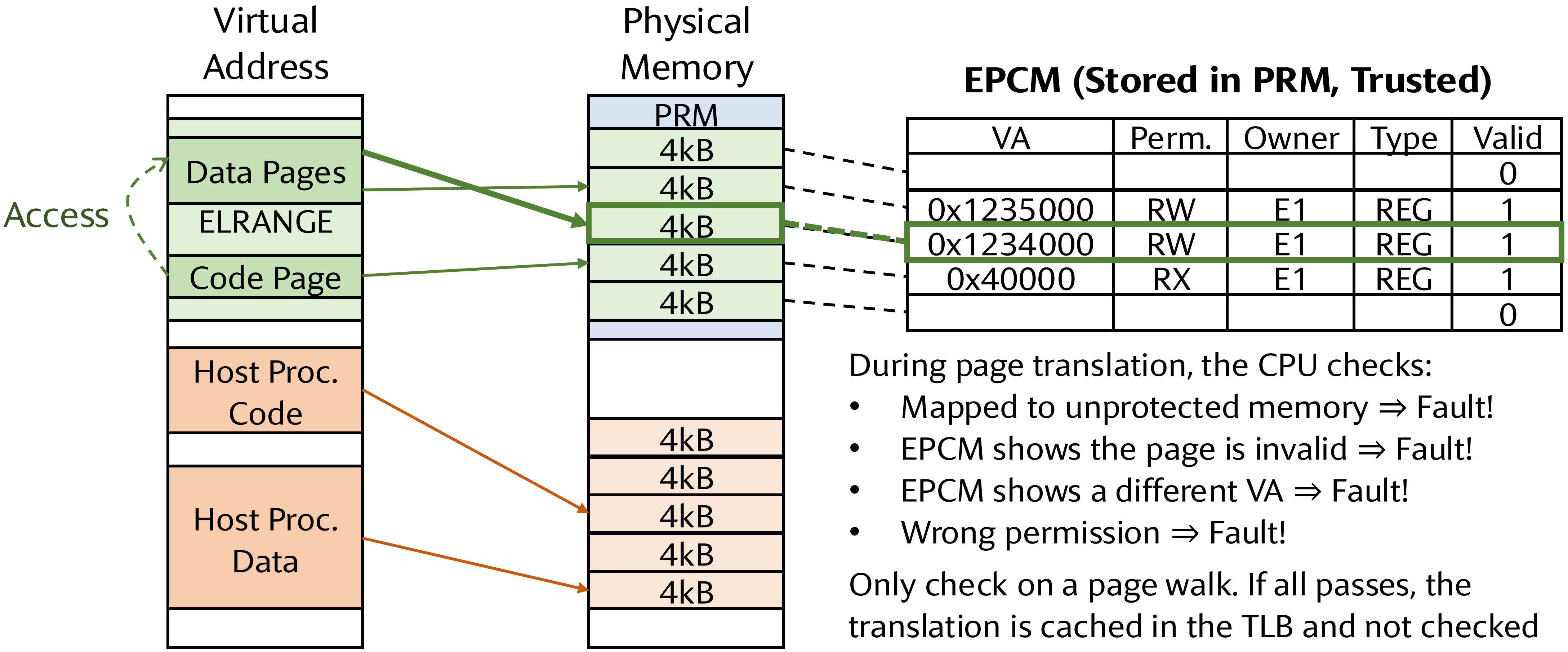
After EINIT, the enclave is initialized

- No more EADDs and EEXTENDs are allowed
- MRENCLAVE reflects the memory layout and content of the enclave
- EINIT validates MRENCLAVE against an author-supplied reference value





Runtime Checks Enforced by EPCM



EPCM (Stored in PRM, Trusted)

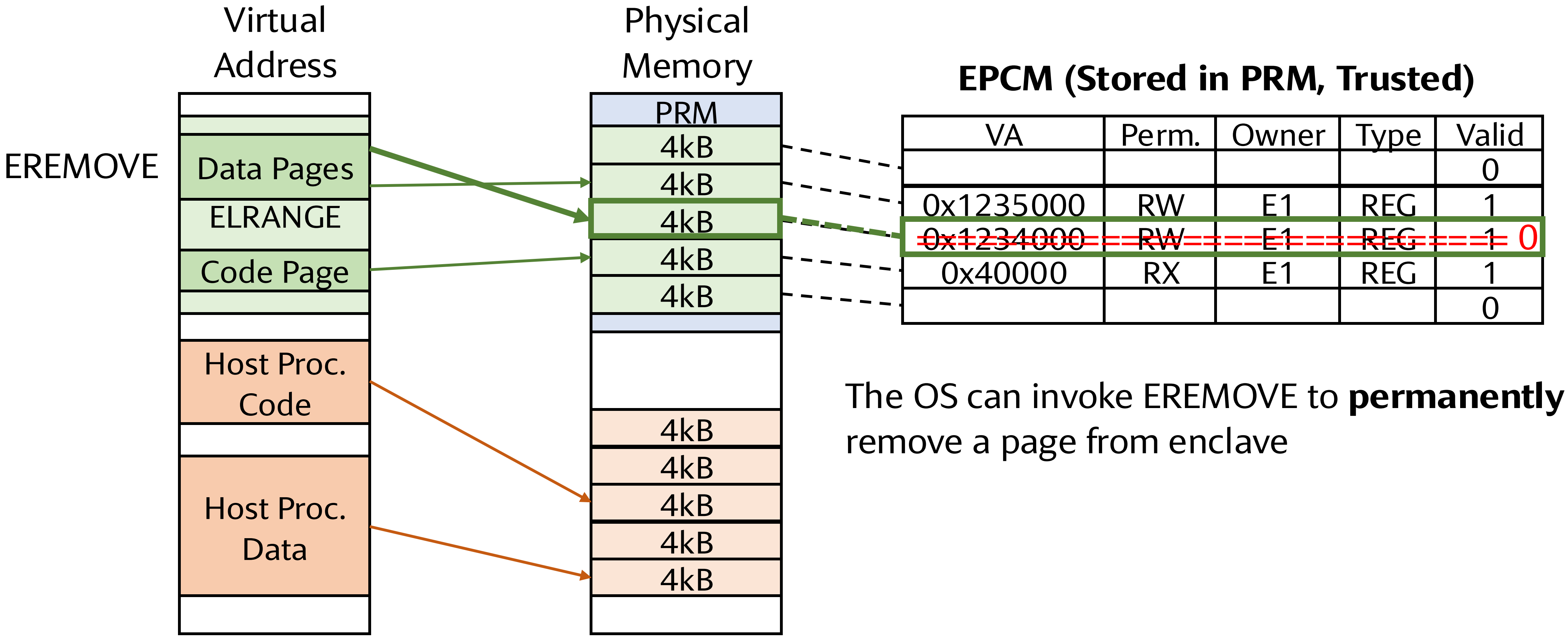
VA	Perm.	Owner	Type	Valid
				0
0x1235000	RW	E1	REG	1
0x1234000	RW	E1	REG	1
0x40000	RX	E1	REG	1
				0

During page translation, the CPU checks:

- Mapped to unprotected memory ⇒ Fault!
- EPCM shows the page is invalid ⇒ Fault!
- EPCM shows a different VA ⇒ Fault!
- Wrong permission ⇒ Fault!

Only check on a page walk. If all passes, the translation is cached in the TLB and not checked until the next page walk. Why secure?

Tearing Down an Enclave

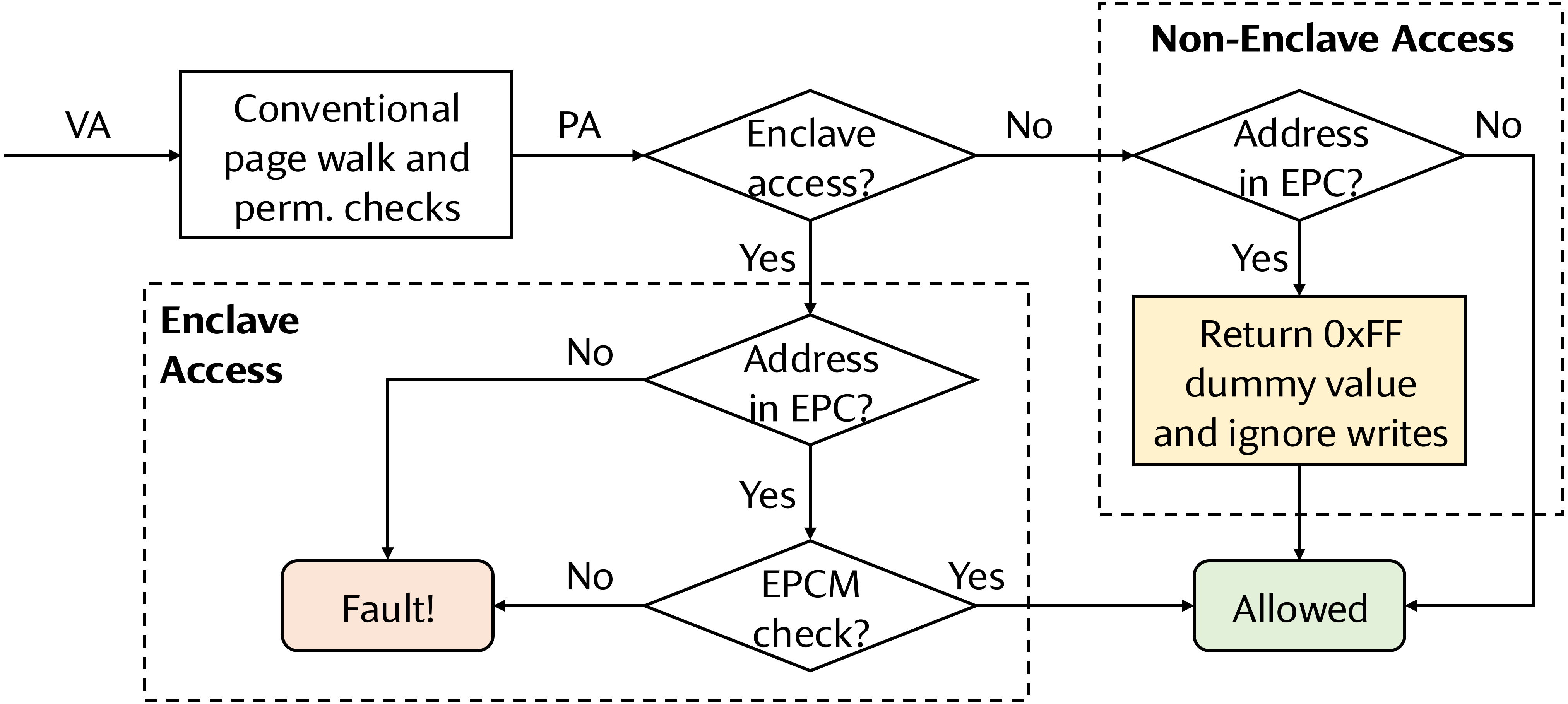


EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid
				0
0x1235000	RW	E1	REG	1
~~0x1234000~~	~~RW~~	~~E1~~	~~REG~~	~~1~~ 0
0x40000	RX	E1	REG	1
				0

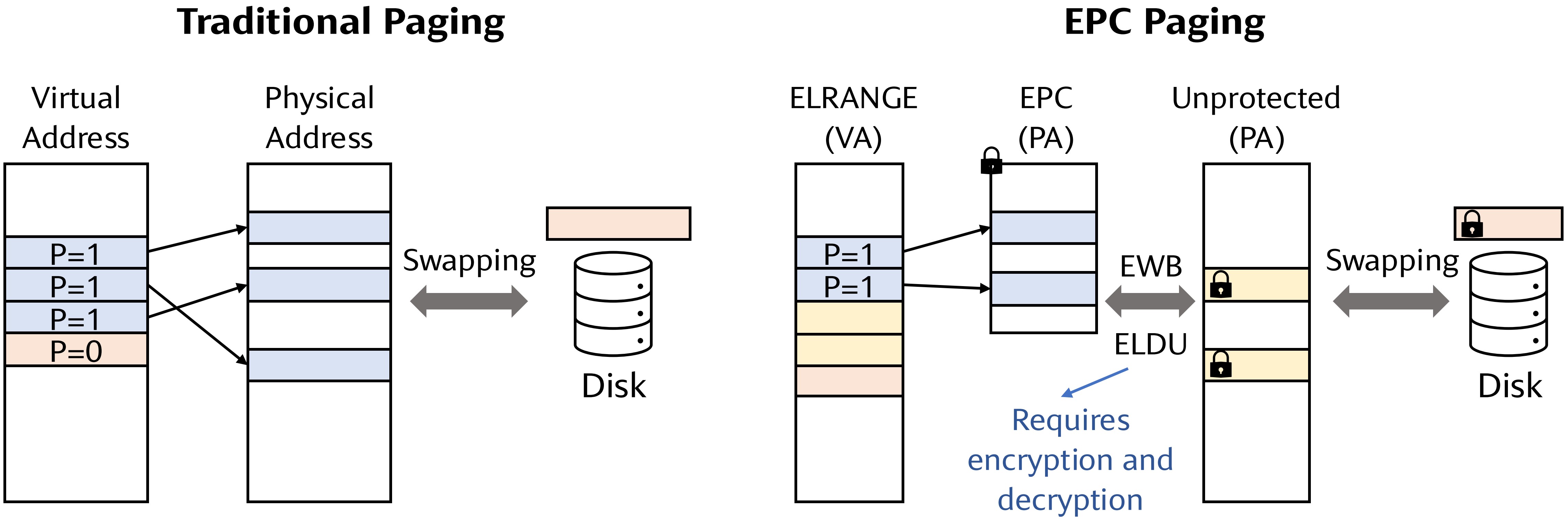
The OS can invoke EREMOVE to **permanently** remove a page from enclave

SGX-1 Access Control

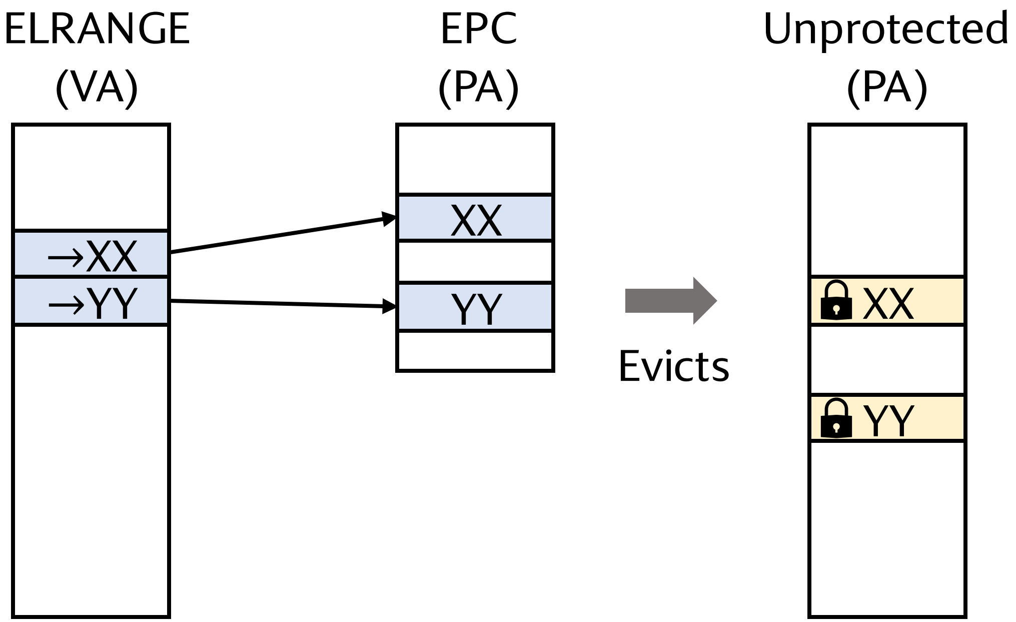


EPC Page Eviction

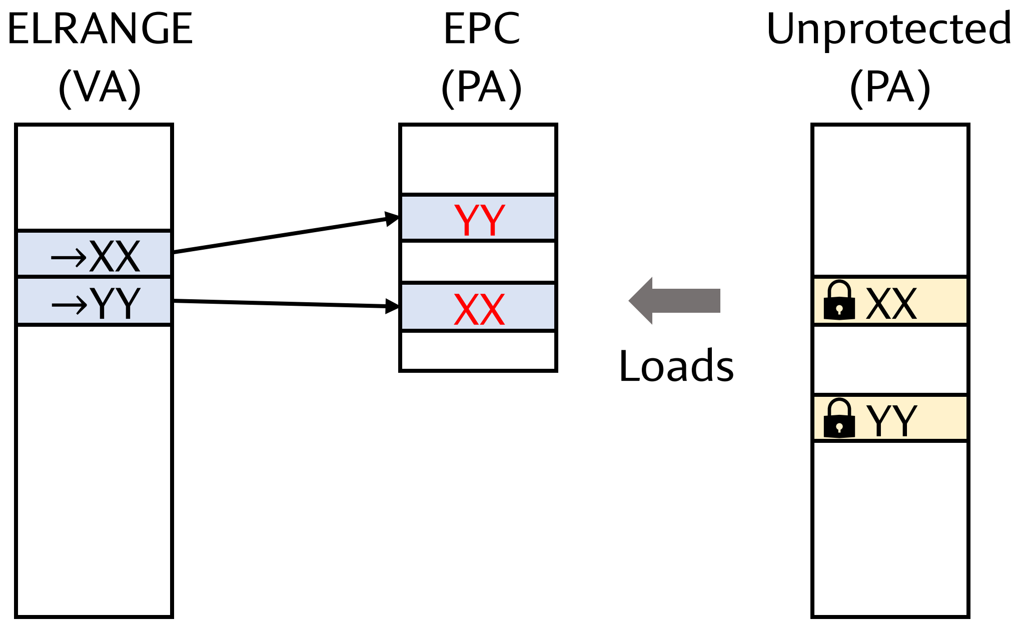
The maximum size of PRM is 128MB (SGX-1)



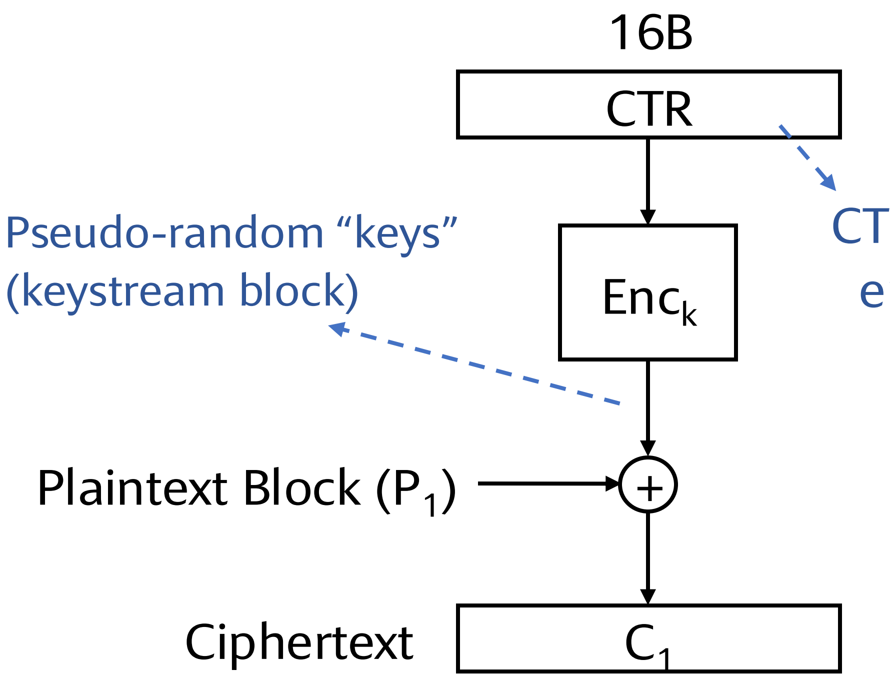
Switcheroo (or Page Swapping) Keeps Trying to Sneak Back!



Switcheroo (or Page Swapping) Keeps Trying to Sneak Back!

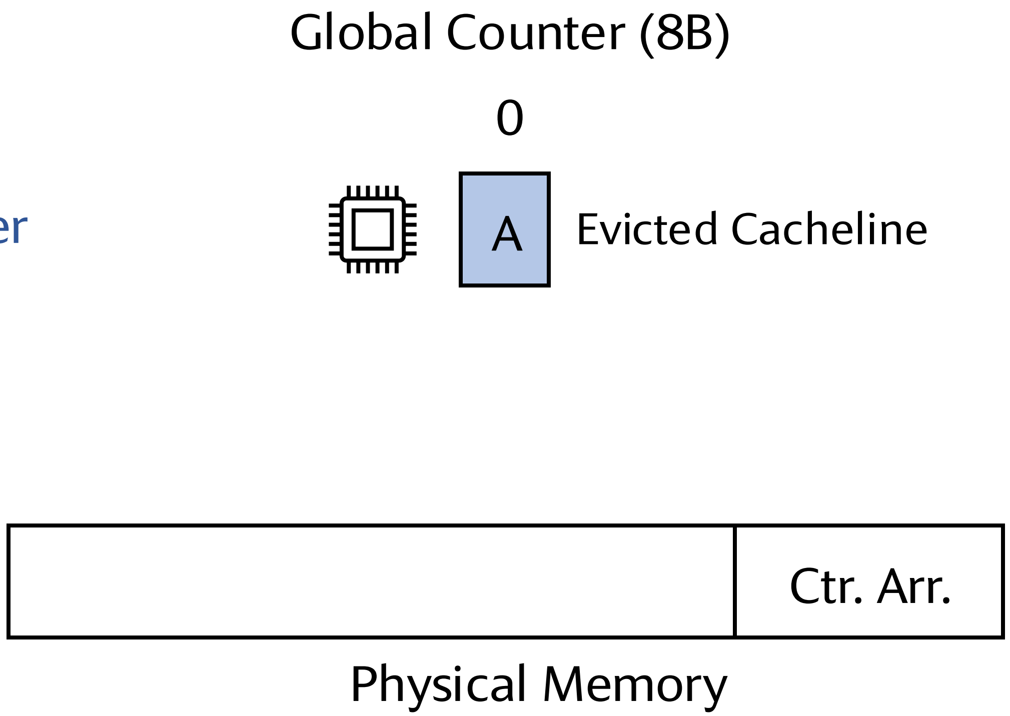


Counter (CTR) Mode



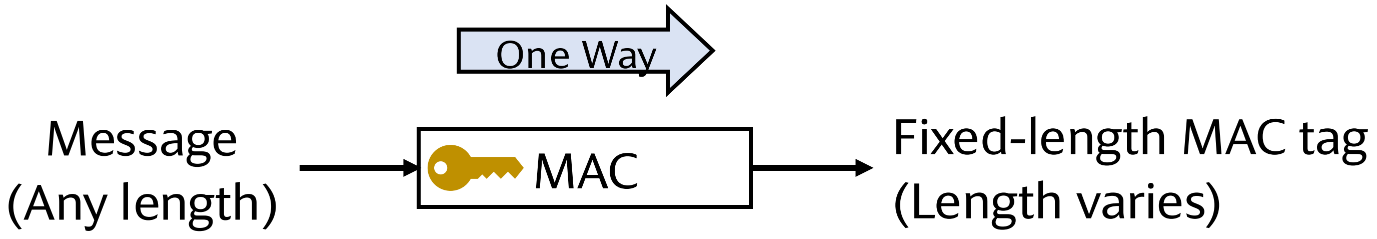
Pseudo-random “keys” (keystream block)

CTR increments after encrypting a block



$C = \mathrm{Enc}_k(CTR) \oplus P$ $\quad$ $P = \mathrm{Enc}_k(CTR) \oplus C$

Hammer 5: Message Authentication Code (MAC)



Properties:

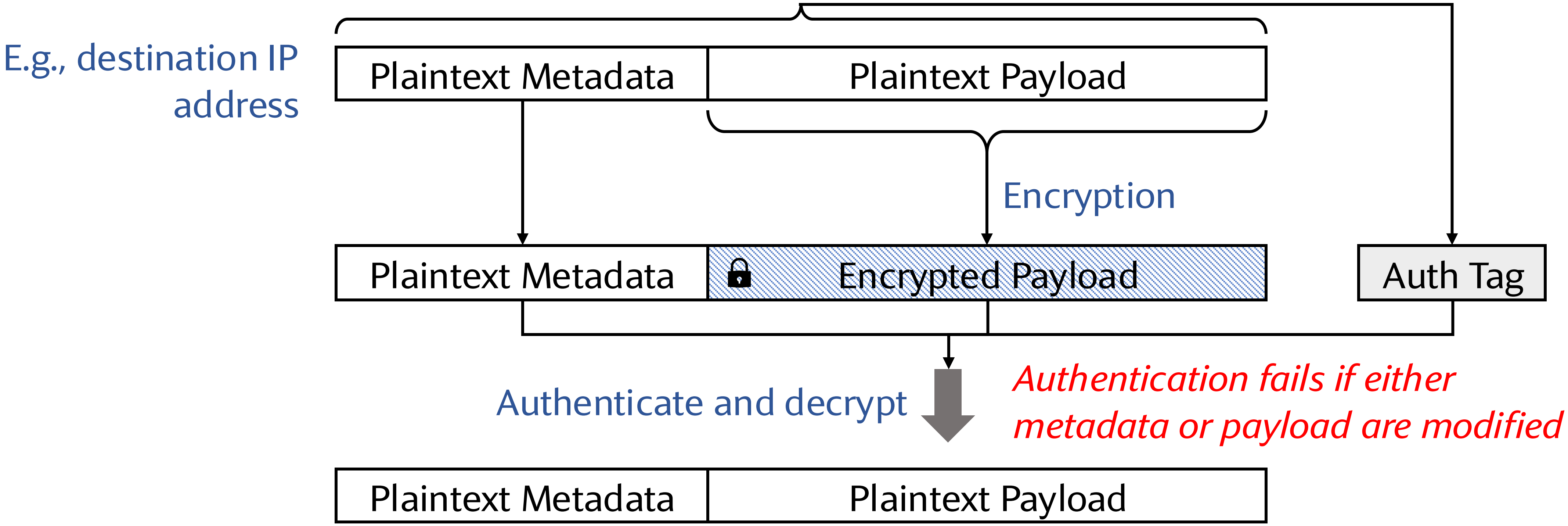
- Verifier has the same key
- Only the person who has the key can produce the correct MAC tag
  ⇒ Correct MAC: The message is authentic

Examples:

- Hash-based MAC (HMAC): Turns a crypto hash function into a MAC construction (e.g., HMAC-SHA256)
- Poly1305: A dedicated MAC design by DJB. Commonly used with ChaCha20, a stream cipher

Hammer 7: Authenticated Encryption with Associated Data (AEAD)

An alternative to cipher + MAC



E.g., destination IP address

Plaintext Metadata

Plaintext Payload

Encryption

Plaintext Metadata

Encrypted Payload

Auth Tag

Authenticate and decrypt

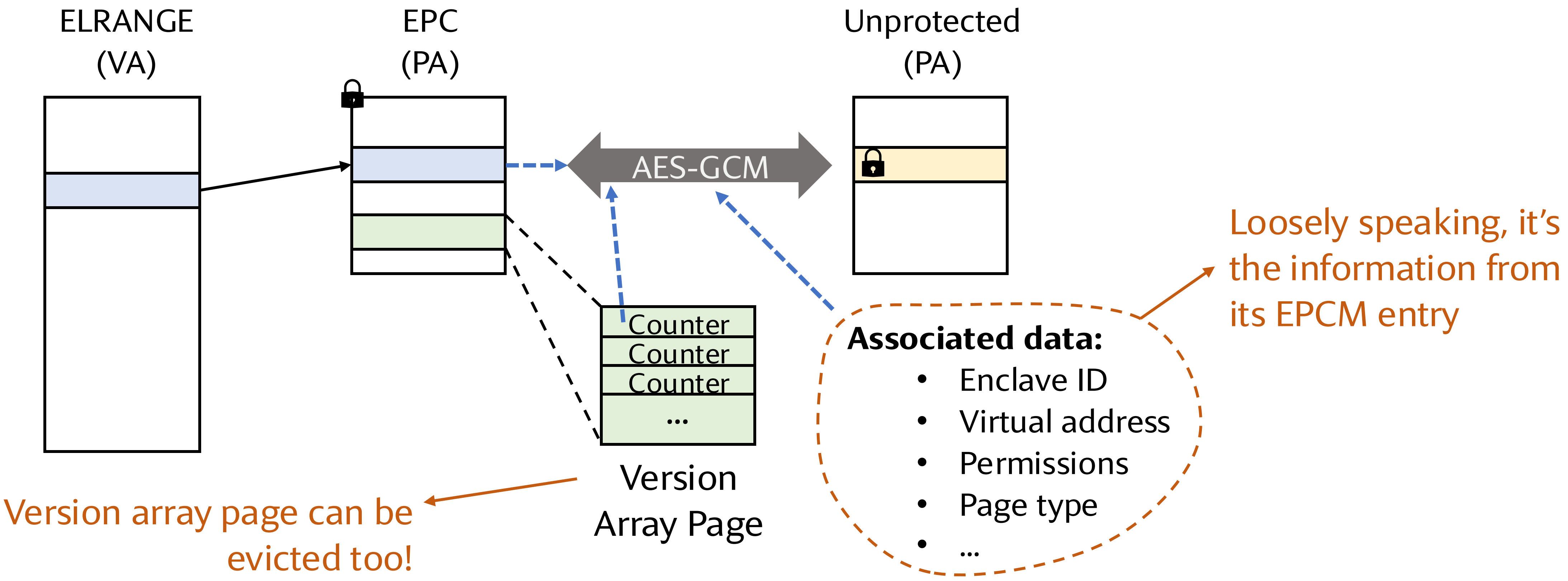
Authentication fails if either metadata or payload are modified

Plaintext Metadata

Plaintext Payload

Example AEAD: AES-GCM (= AES-CTR + GMAC, loosely speaking)

EPC Page Eviction



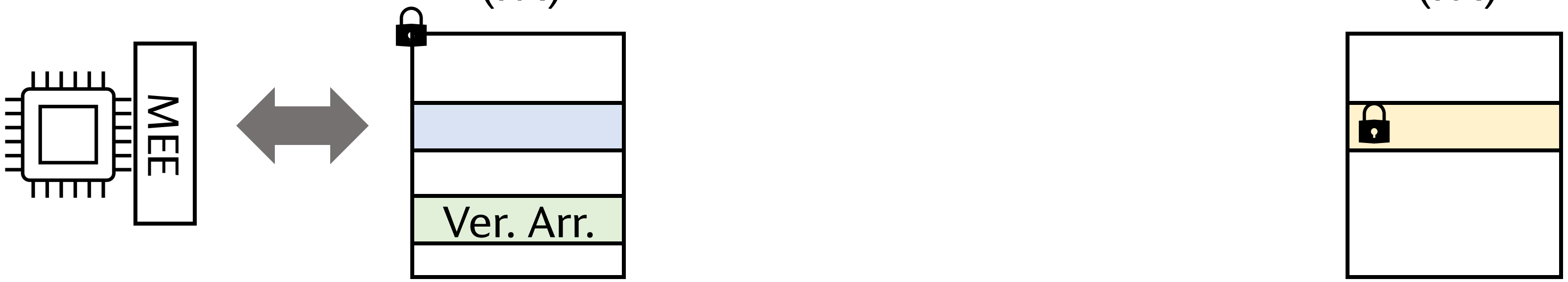
Associated data:

- Enclave ID
- Virtual address
- Permissions
- Page type
- …

Loosely speaking, it's the information from its EPCM entry

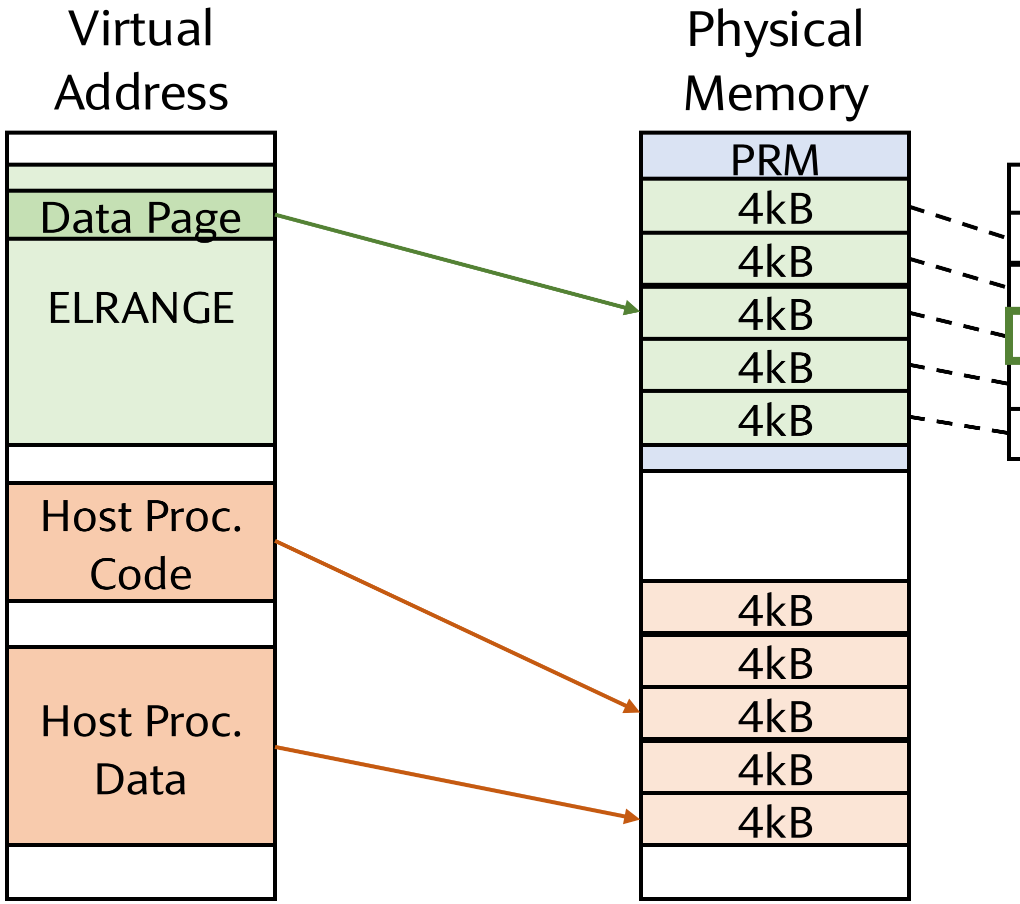
Version array page can be evicted too!

Memory Protection Comparison



	EPC (PA)	Unprotected (PA)
View of software running on CPU	Oblivious to the encryption. Sees only plaintext. Illegal access prevented via access control	Privileged software can access evicted pages, but only see ciphertext
View of a physical attacker (w/ a probe)	Fully encrypted	Selectively encrypted
Encryption mode	Tweaked counter mode where the “counter” depends on the PA	AEAD. AD includes enclave ID, VA, permission bits, type, etc

Workflow of EPC Page Eviction



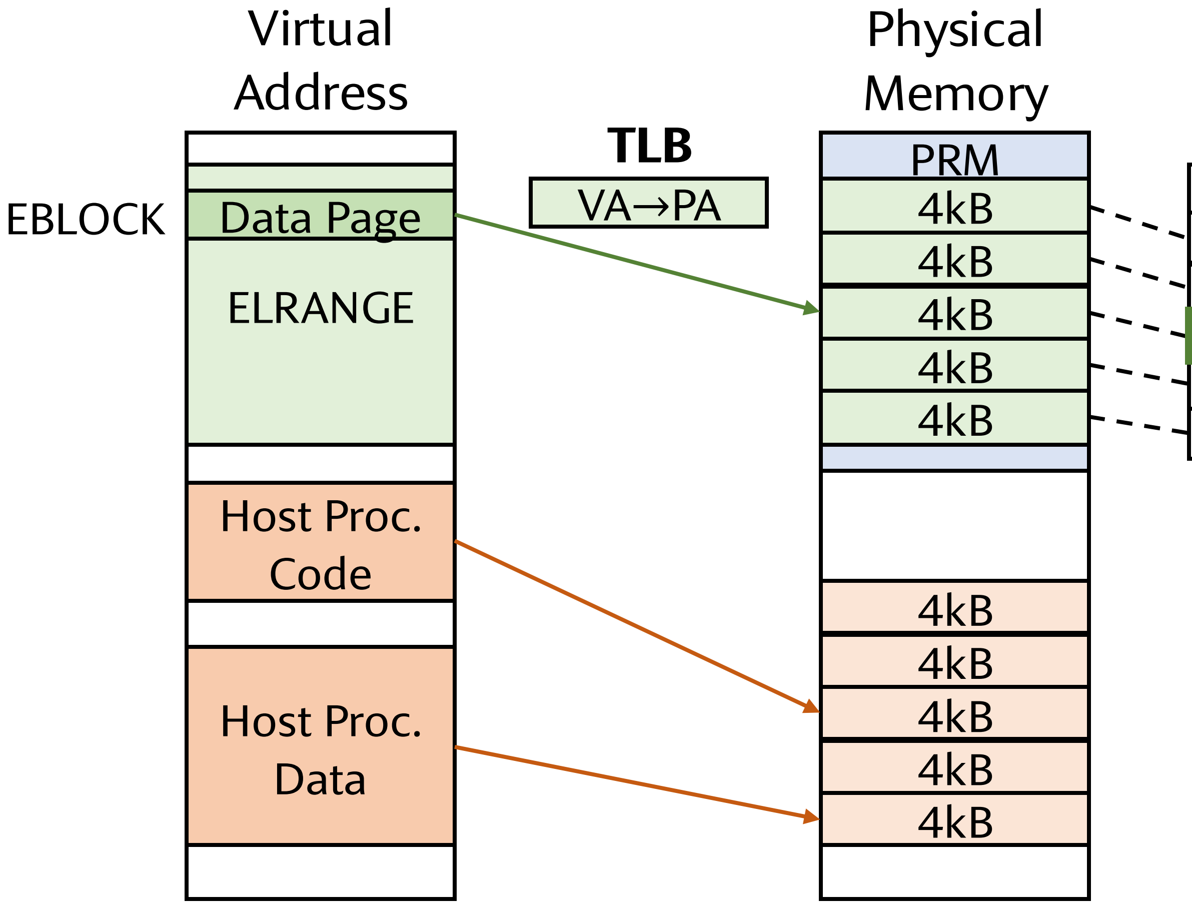
EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid
				0
				0
0x2000	RW	E1	REG	1
				0
				0

Access control results are cached in the TLB
We need to flush the translation before eviction

How to guarantee the TLB entry is flushed by the untrusted OS?

Workflow of EPC Page Eviction

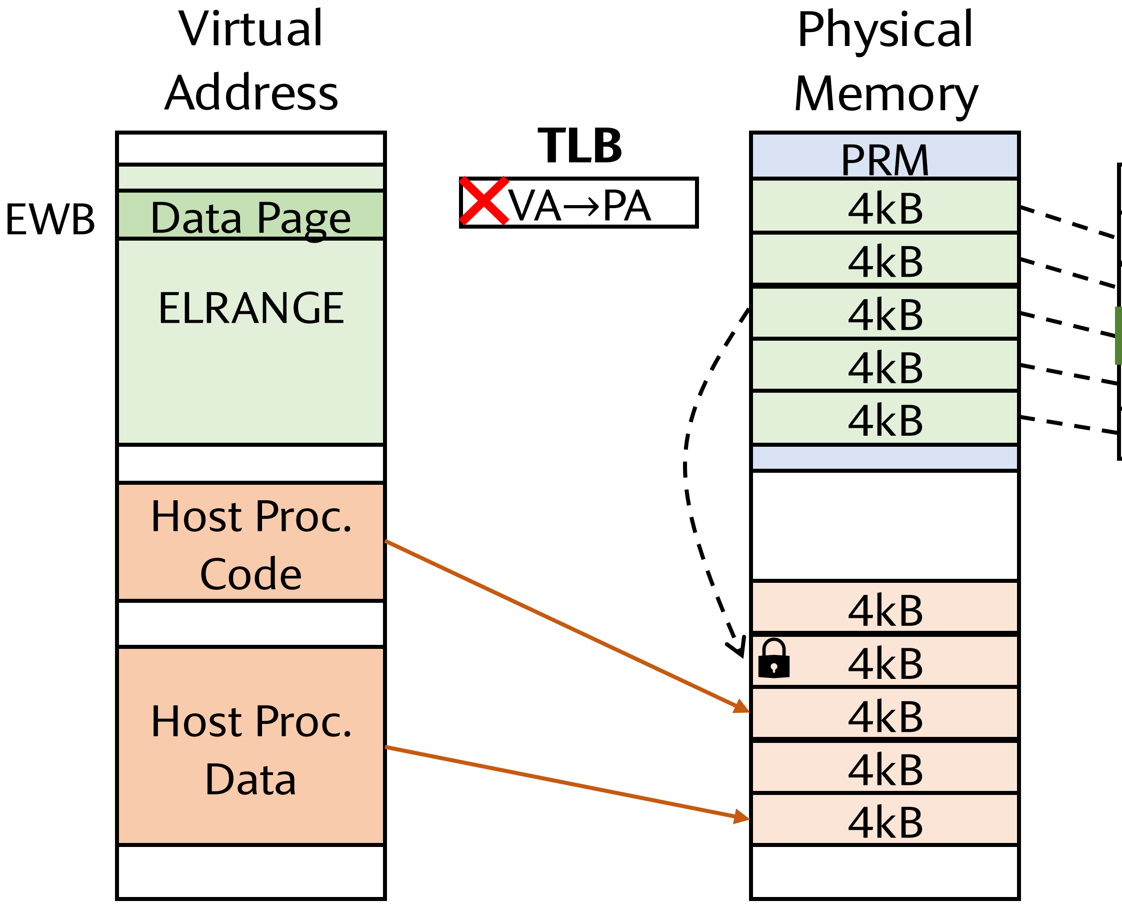


EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid	**Blocked**
				0	0
				0	0
0x2000	RW	E1	REG	1	~~0~~ 1
				0	0
				0	0

- OS executes EBLOCK targeting the page
    - Blocked pages are not accessible
    - ⇒ No new TLB entries can be created
- OS executes ETRACK
    - Notify SGX to track if translations are flushed

Workflow of EPC Page Eviction

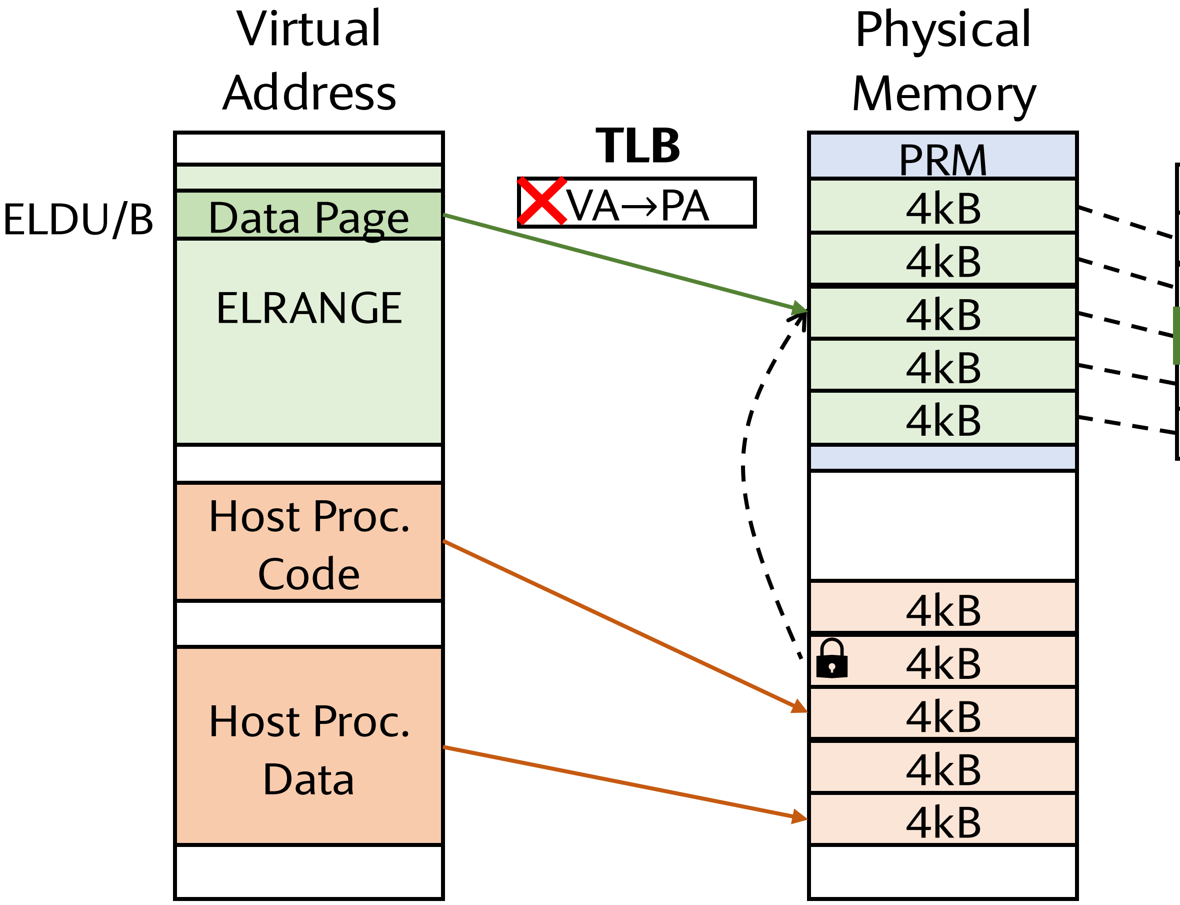


EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid	**Blocked**
				0	0
				0	0
~~0x2000~~	~~RW~~	~~E1~~	~~REG~~	~~1~~ 0	~~0~~ 0
				0	0
				0	0

- OS executes EBLOCK targeting the page
  - Blocked pages are not accessible
  - ⇒ No new TLB entries can be created
- OS executes ETRACK
  - Notify SGX to track if translations are flushed
- OS flushes the TLB entries (e.g., via interrupts)
- OS executes EWB to evict the page
- SGX validates TLB entries are flushed and invalidates the EPCM entries

Workflow of EPC Page Eviction

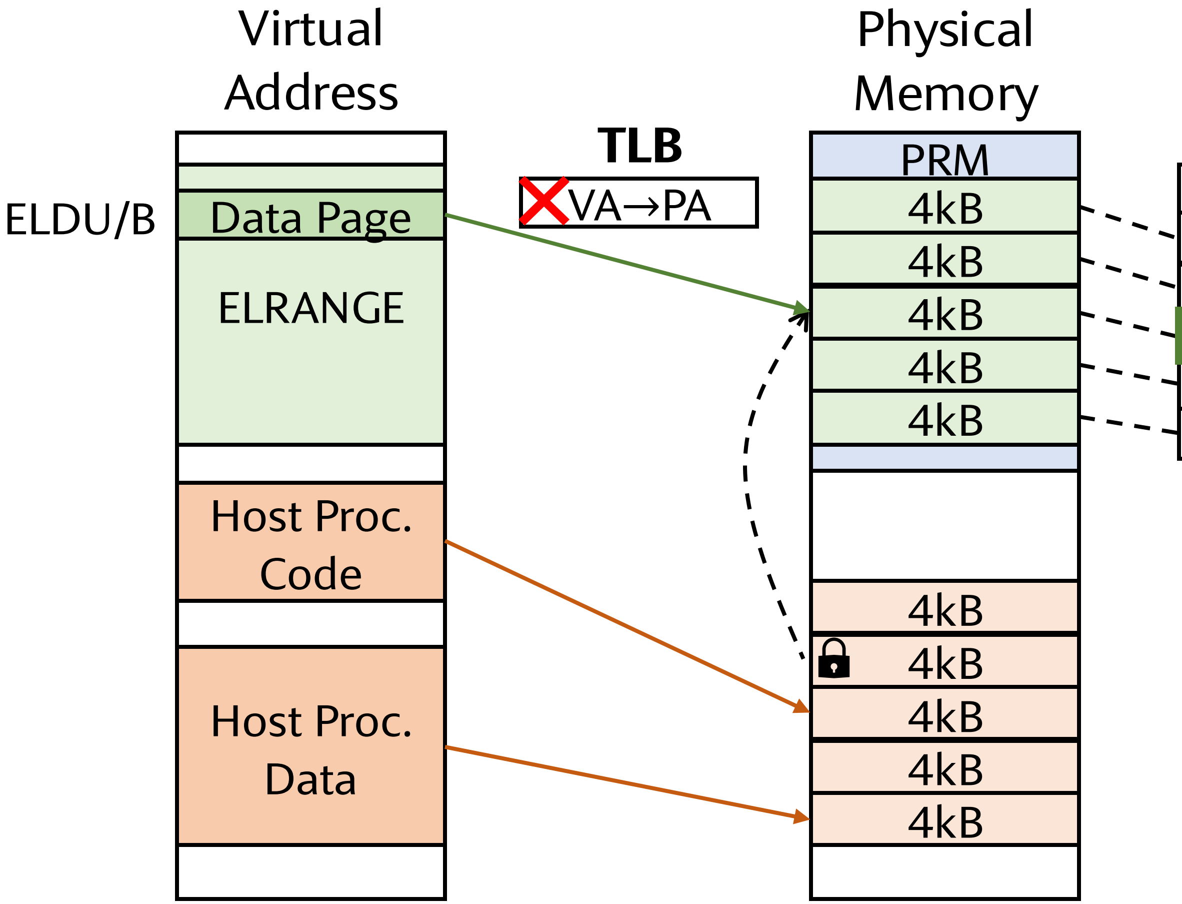


EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid	**Blocked**
				0	0
				0	0
0x2000	RW	E1	REG	1	0
				0	0
				0	0

- OS executes ELDU/ELDB
  - SGX authenticates the evicted page
  - SGX decrypts the evicted page into the free EPC page
  - SGX updates/restores the EPCM entry

Workflow of EPC Page Eviction

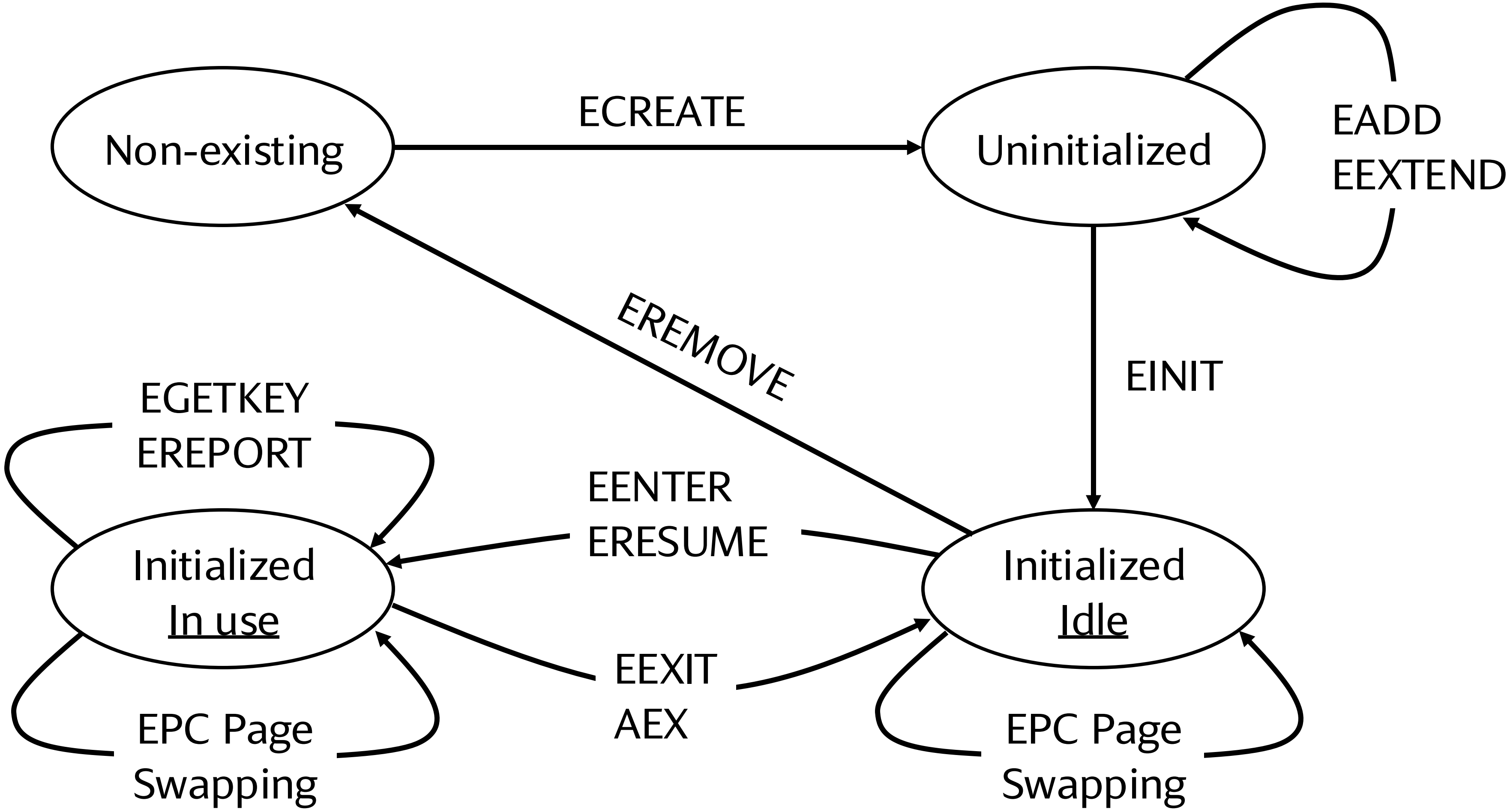


EPCM (Stored in PRM, Trusted)

VA	Perm.	Owner	Type	Valid	**Blocked**
				0	0
				0	0
0x2000	RW	E1	REG	1	0
				0	0
				0	0

- OS executes ELDU/ELDB
    - SGX authenticates the evicted page
    - SGX decrypts the evicted page into the free EPC page
    - SGX updates/restores the EPCM entry

SGX Life Cycle


Non-existing
ECREATE
Uninitialized
EADD
EEXTEND
EREMOVE
EINIT
EGETKEY
EREPORT
EENTER
ERESUME
Initialized
In use
Initialized
Idle
EEXIT
AEX
EPC Page
Swapping
EPC Page
Swapping