Intel's SGX In-depth Architecture

Syed Kamran Haider
with
Hamza Omar, Masab Ahmad, Chenglu Jin, and Marten van Dijk

UCONN

With the help of:

1. Intel SGX Tutorial (Reference Number: 332680-002) presented at ISCA 2015
2. "*Intel SGX Explained*", Victor Costan and Srinivas Devadas, CSAIL MIT

Warmup Puzzle!

You are one of P recently arrested prisoners. The warden makes the following announcement:

"You may meet together today and plan a strategy, but after today you will be in isolated cells and have no communication with one another.

I have set up a "switch room" which contains a light switch, which is either on or off. The switch is not connected to anything.

Every now and then, I will select one prisoner at random to enter the "switch room". This prisoner may throw the switch (from on to off, or vice-versa), or may leave the switch unchanged. Nobody else except the prisoners will ever enter this room.

Each prisoner will visit the switch room arbitrarily often. More precisely, for any N, eventually each of you will visit the switch room at least N times.

At any time, any of you may declare: "we have all visited the switch room at least once." If the claim is correct, I will set you free. If the claim is incorrect, I will feed all of you to the sharks."

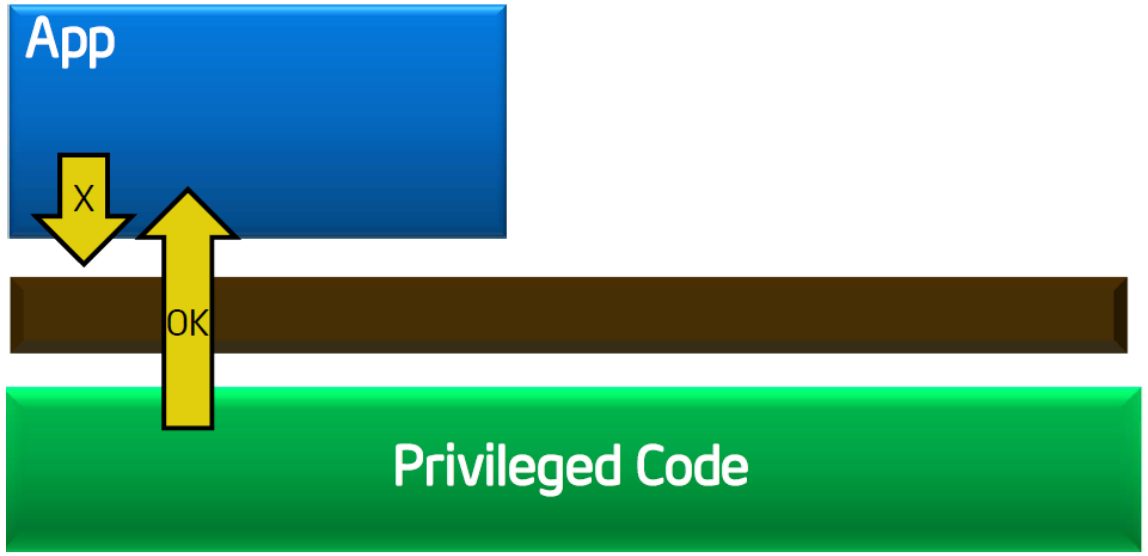
Devise a winning strategy when you know that the initial state of the switch is off.

Outline

- Introduction
- SGX High Level Overview
- SGX Memory Organization
- The Life Cycle of an SGX Enclave
- EPC Page Swapping
- SGX Software Attestation

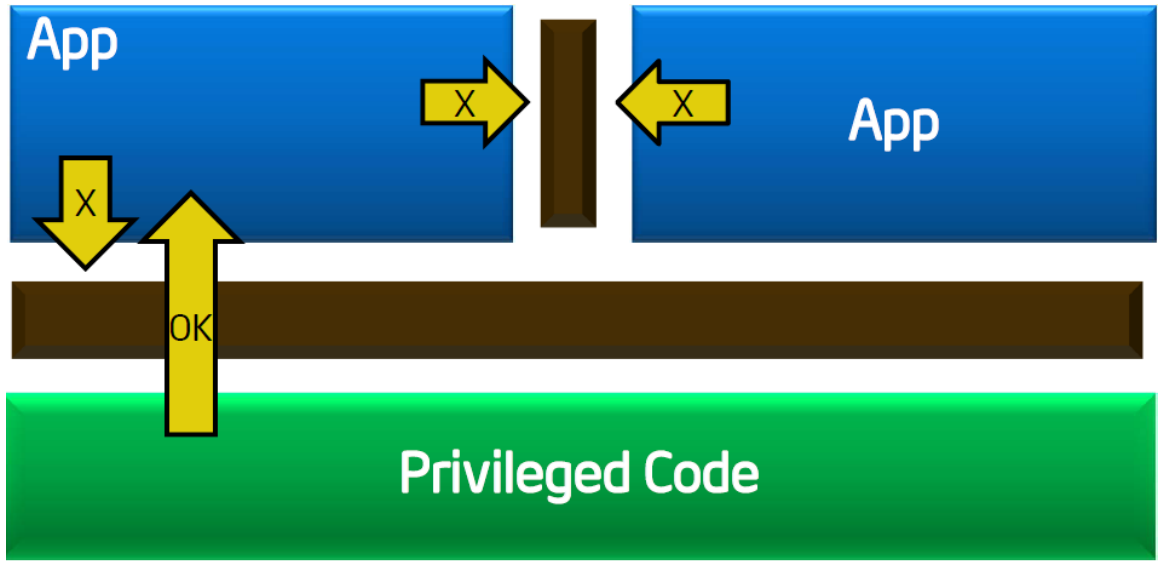
Why Aren't Compute Devices Trustworthy?

- Protected Mode (Privilege Levels i.e., Rings) protects OS from apps …



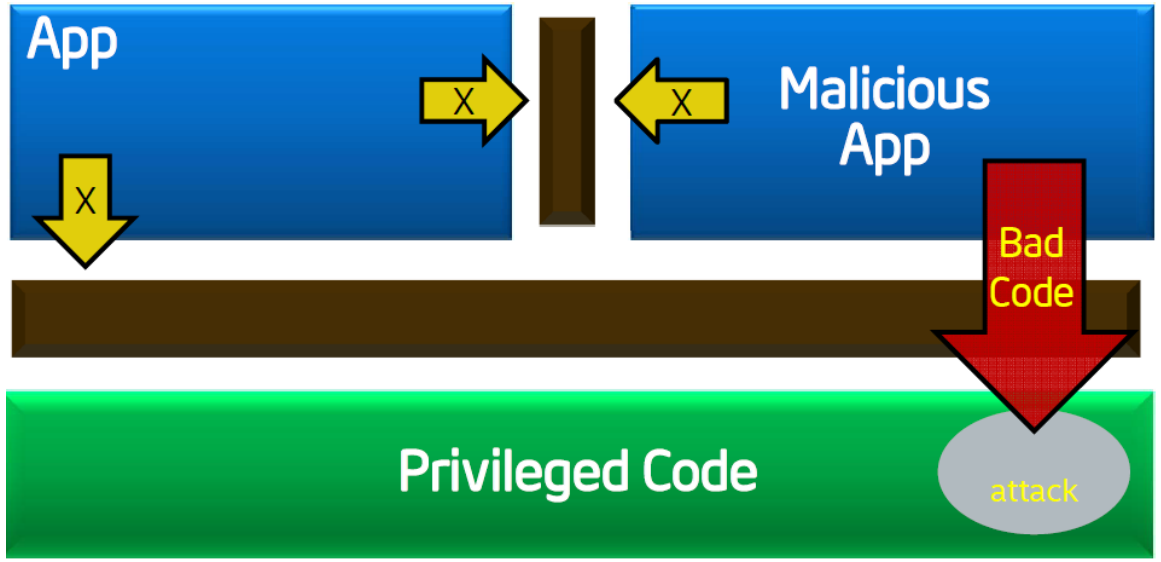
Why Aren't Compute Devices Trustworthy?

- Protected Mode (Privilege Levels i.e., Rings) protects OS from apps …
- … and protects apps from each other …



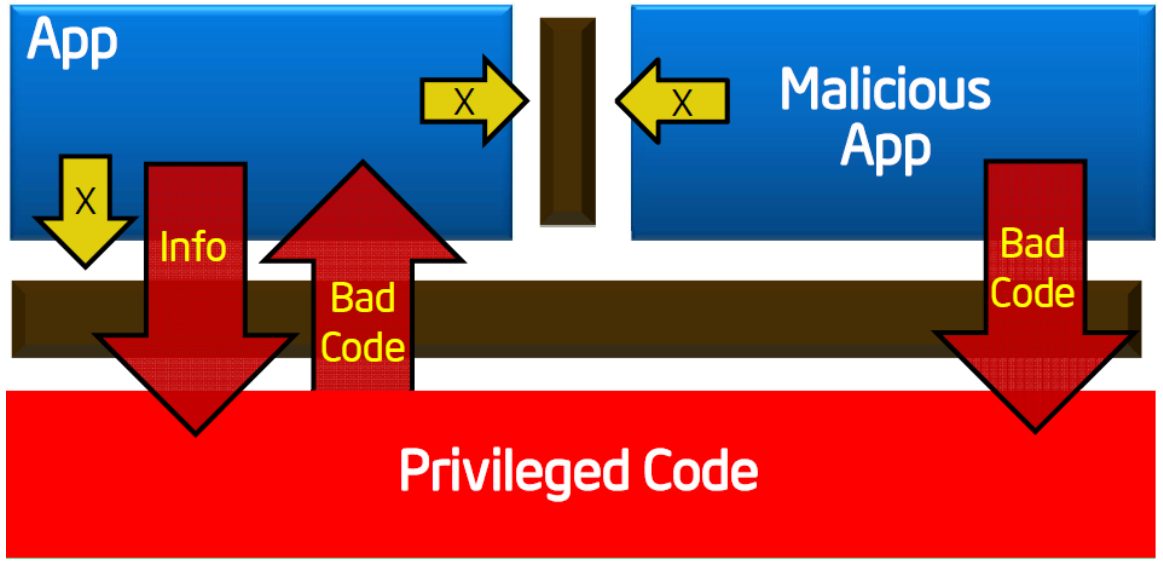
Why Aren't Compute Devices Trustworthy?

- Protected Mode (Privilege Levels i.e., Rings) protects OS from apps …
- … and protects apps from each other …
- … UNTIL a malicious app exploits a flaw to gain full privileges and then tampers with the OS or other apps.
- **Apps are typically not protected from privileged code attacks**



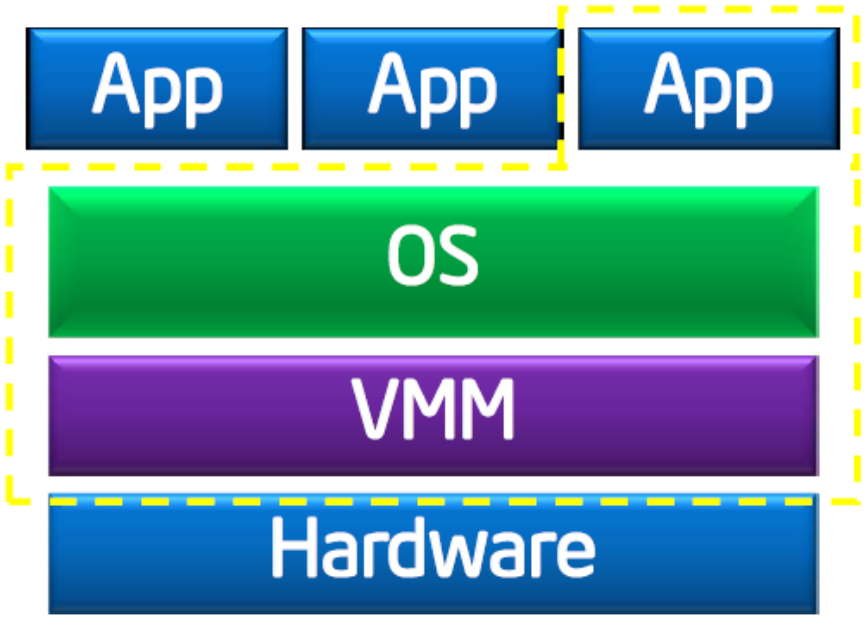
Why Aren't Compute Devices Trustworthy?

- Protected Mode (Privilege Levels i.e., Rings) protects OS from apps …
- … and protects apps from each other …
- … UNTIL a malicious app exploits a flaw to gain full privileges and then tampers with the OS or other apps.
- **Apps are typically not protected from privileged code attacks**



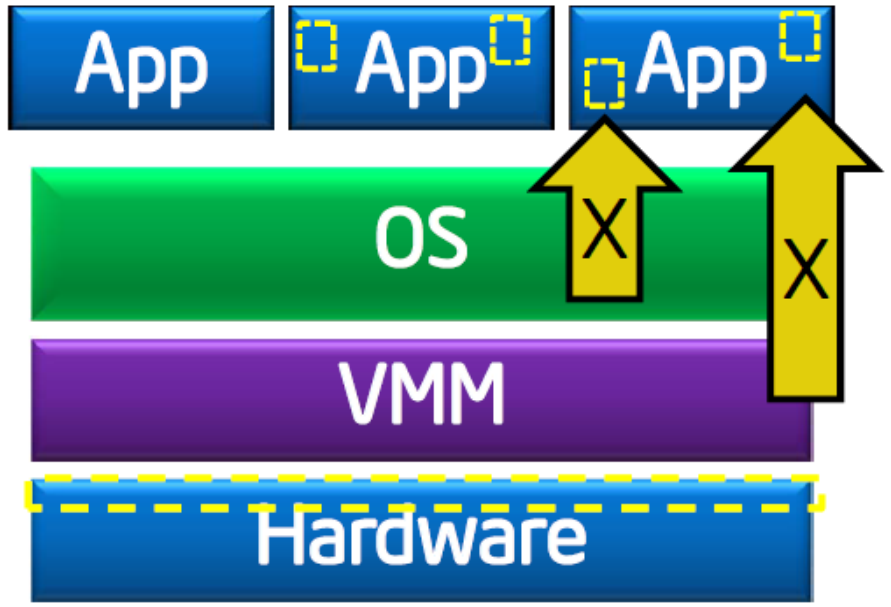
Attack surface today…

- In current systems, a large code base constitutes the part that can be exploited…
  - Application Codes
  - OS code
  - Virtual Machine Manager code
- Hence, millions of lines of code need to be inspected for exploitable bugs etc…



Reduced attack surface with SGX Enclaves

- With SGX, Application gains ability to defend its own secrets
  - Smallest attack surface (App + processor)
  - Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets



Attack Surface

What is SGX?

The key concept behind Intel's Software Guard Extensions (SGX) is an Enclave.

Enclave:

A protected environment that contains the code and data of a security-sensitive computation.

There can be many enclaves in the system at a time!

What is SGX?

SGX enabled processors offer the following two crucial properties:

Isolation

Each enclave's environment is isolated from the untrusted software outside the enclave, as well as from other enclaves.

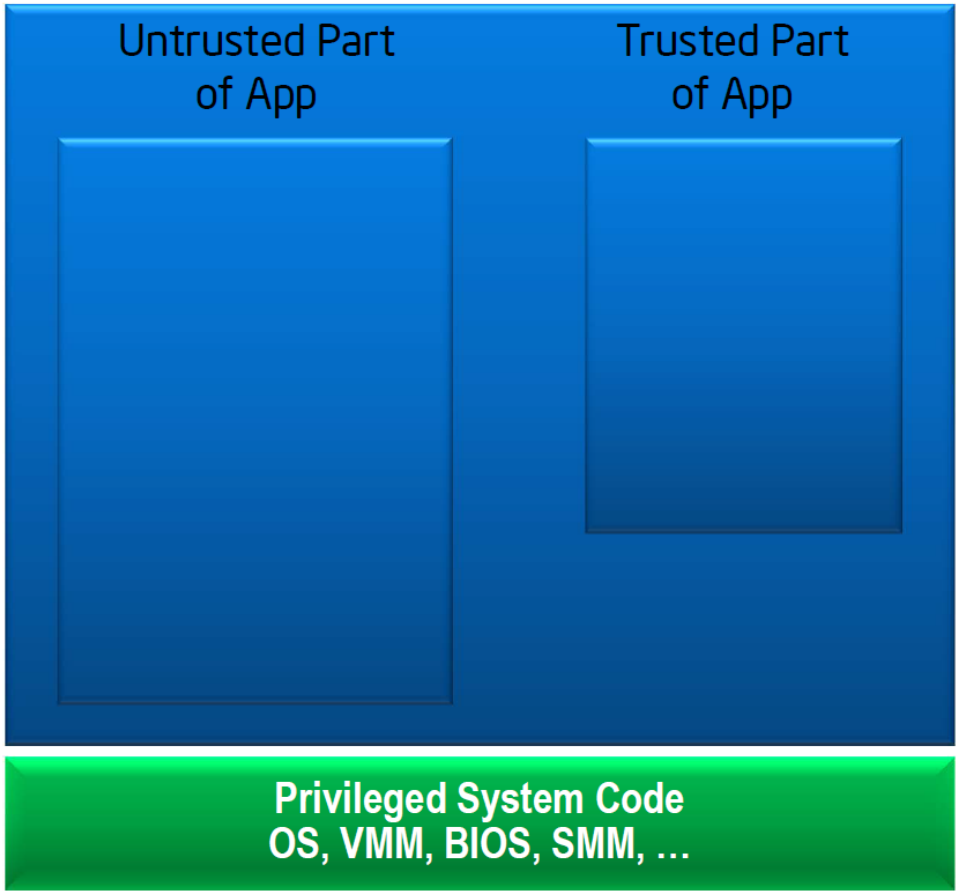
Attestation

A software attestation scheme that allows a remote party to authenticate the software running inside an enclave.

Protects the privacy and integrity of the computation!

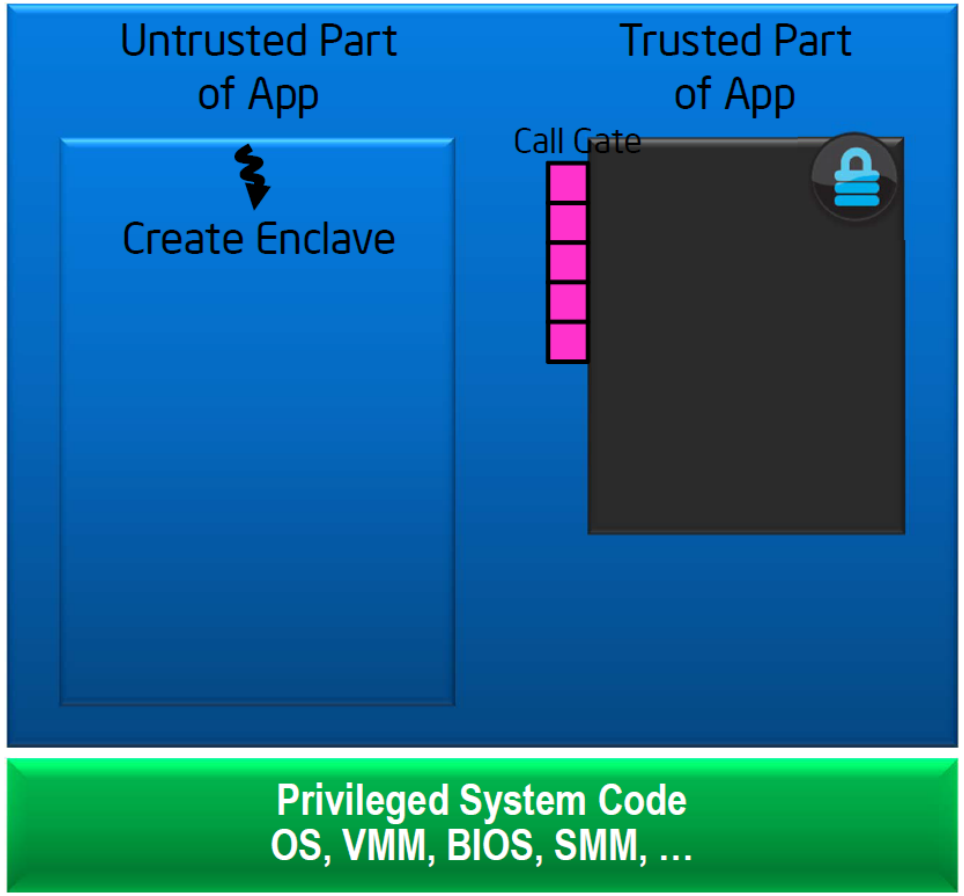
How SGX Secure Enclaves Work

- App is built with trusted and untrusted parts



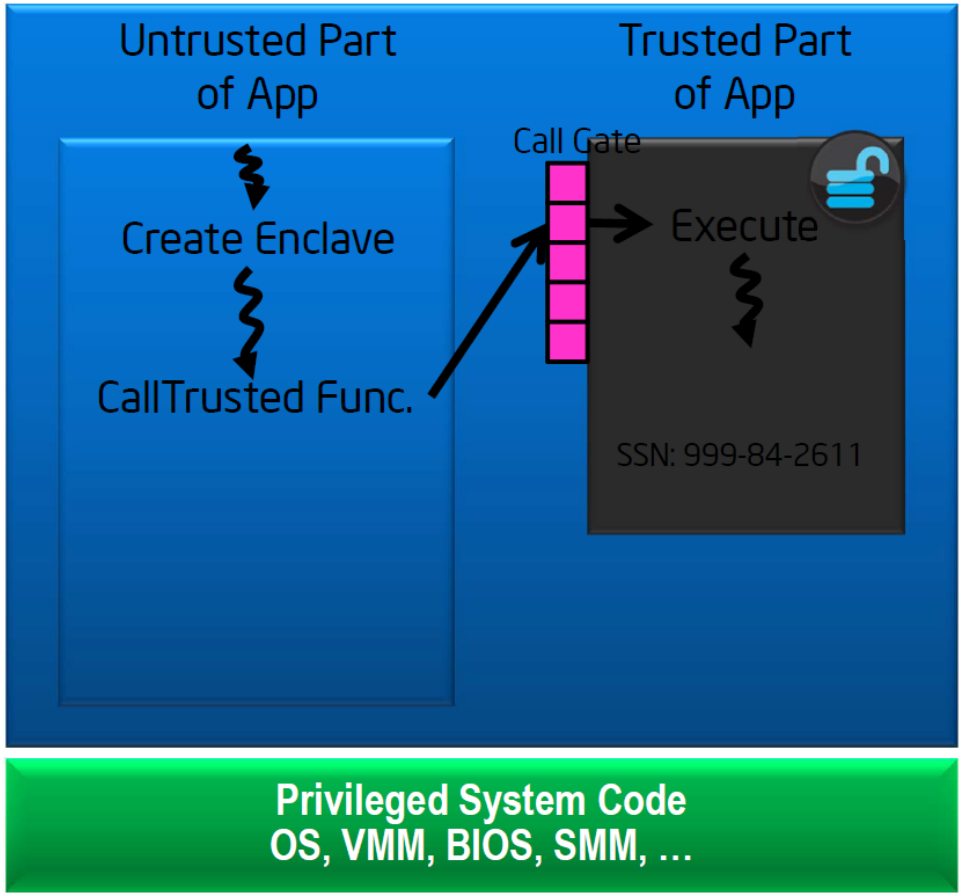
How SGX Secure Enclaves Work

- App is built with trusted and untrusted parts
- App runs & creates enclave which is placed in trusted memory



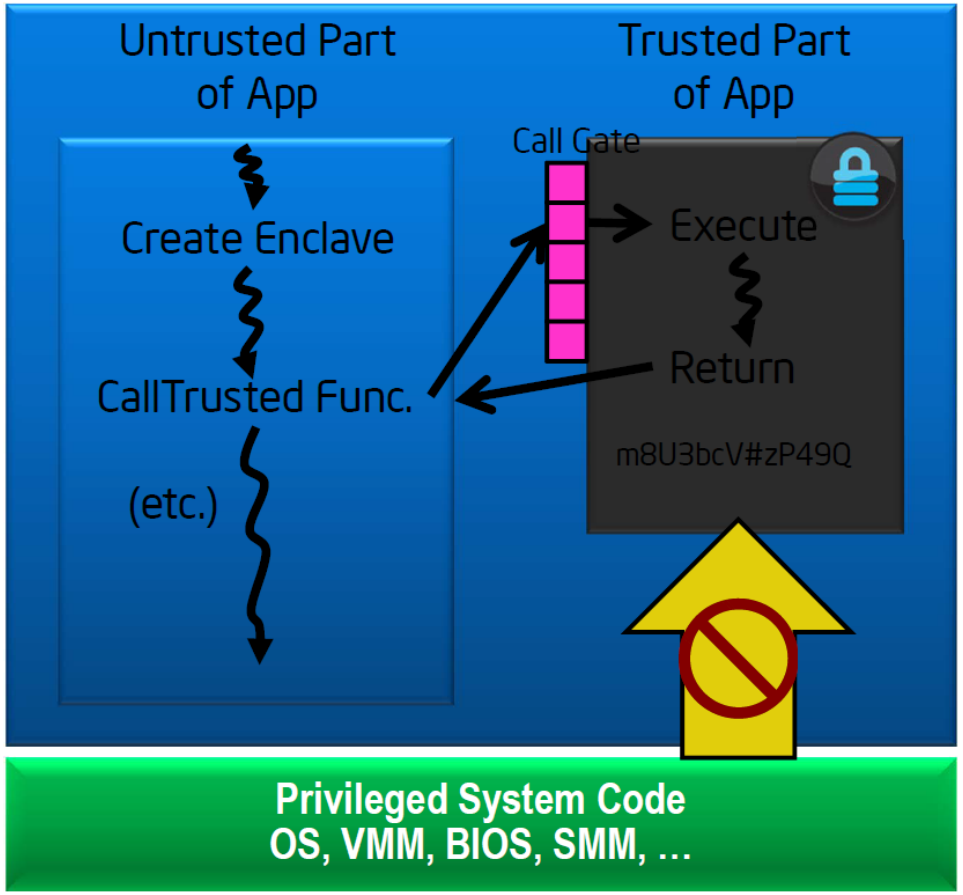
How SGX Secure Enclaves Work

- App is built with trusted and untrusted parts
- App runs & creates enclave which is placed in trusted memory
- Trusted function is called; code running inside enclave sees data in clear



How SGX Secure Enclaves Work

- App is built with trusted and untrusted parts
- App runs & creates enclave which is placed in trusted memory
- Trusted function is called; code running inside enclave sees data in clear;
- Function returns; enclave data remains in trusted memory
- External access to enclave data is denied

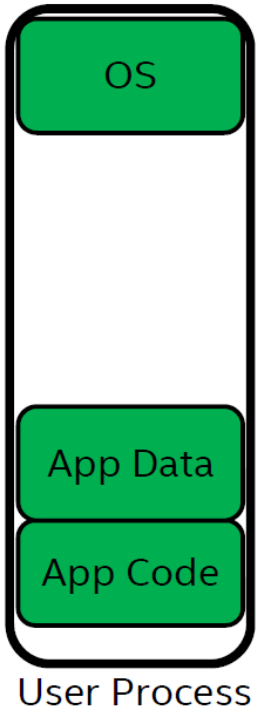


SGX High Level Overview

- Programming Environment
- Access Control (Isolation)
- Attestation & Sealing
- Memory Snooping Protection

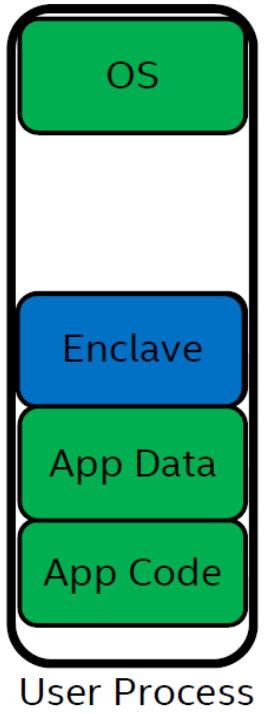
SGX Programming Environment

- Trusted execution environment embedded in a process



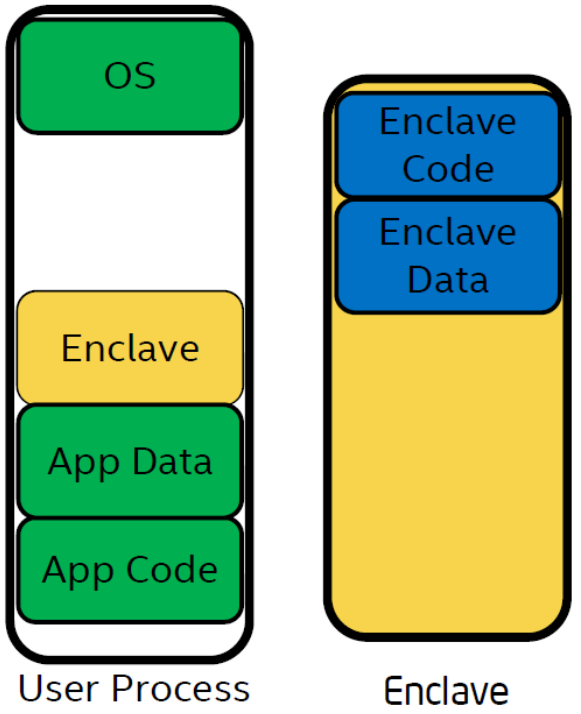
SGX Programming Environment

- Trusted execution environment embedded in a process
- The process creates a secure Enclave



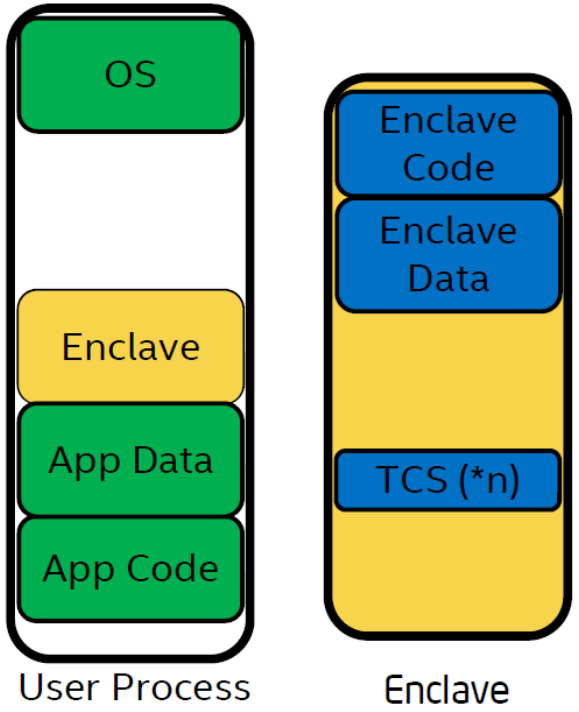
SGX Programming Environment

- Trusted execution environment embedded in a process
- The process creates a secure Enclave
  - Contains Code & Data
  - Provides Confidentiality
  - Provides Integrity
  - Controlled Entry Points

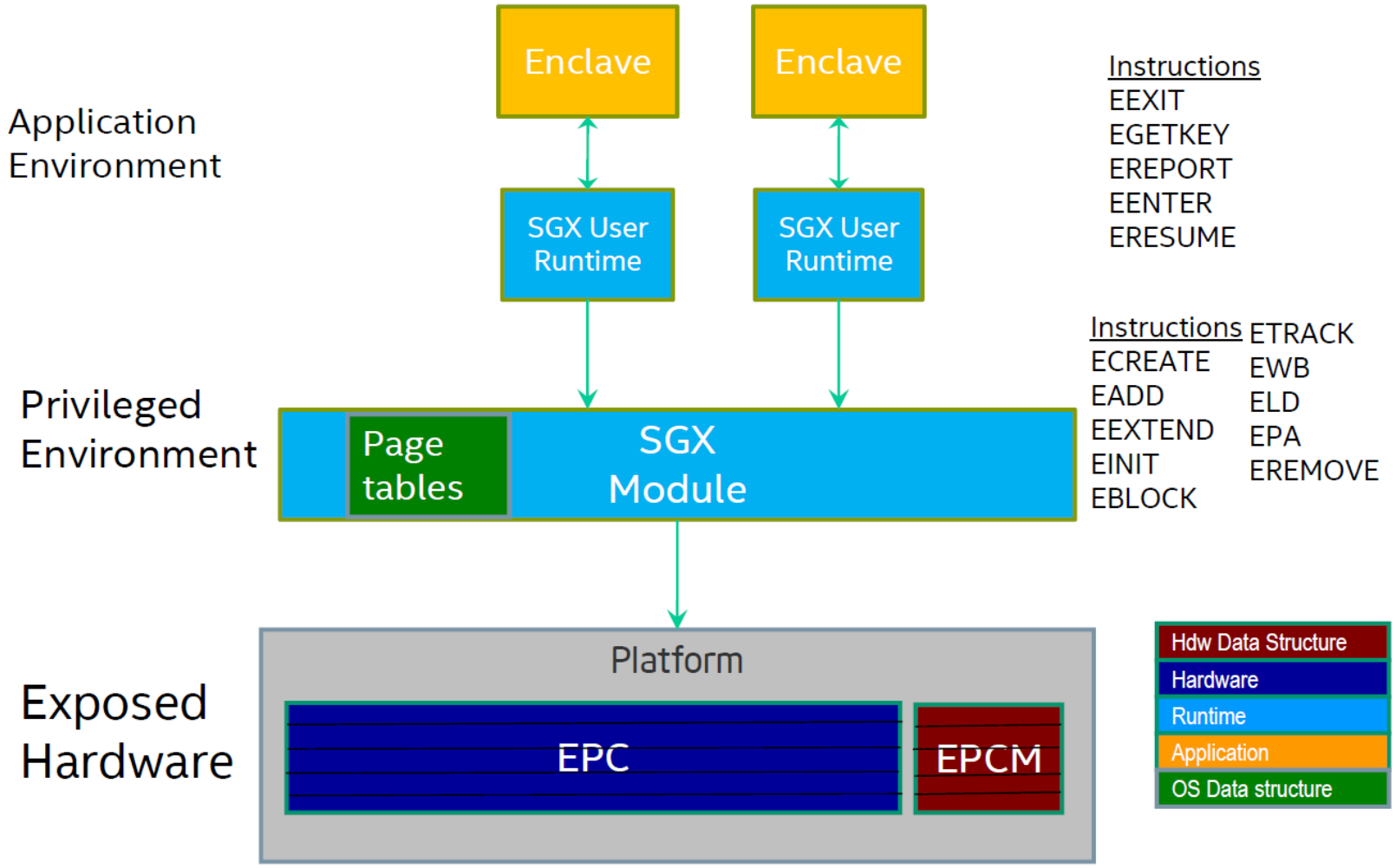


SGX Programming Environment

- Trusted execution environment embedded in a process
- The process creates a secure Enclave
  - Contains Code & Data
  - Provides Confidentiality
  - Provides Integrity
  - Controlled Entry Points
  - Supporting multiple threads
  - Full access to application's memory

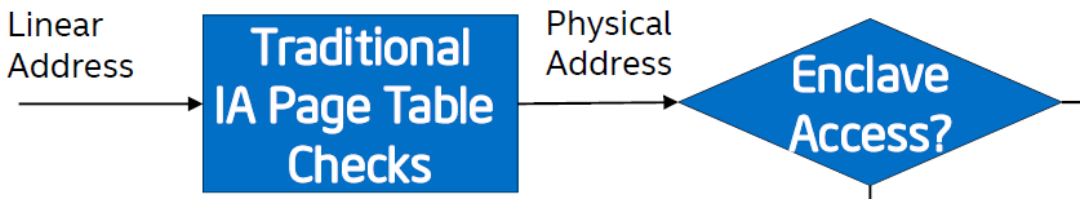


SGX High-level HW/SW Picture

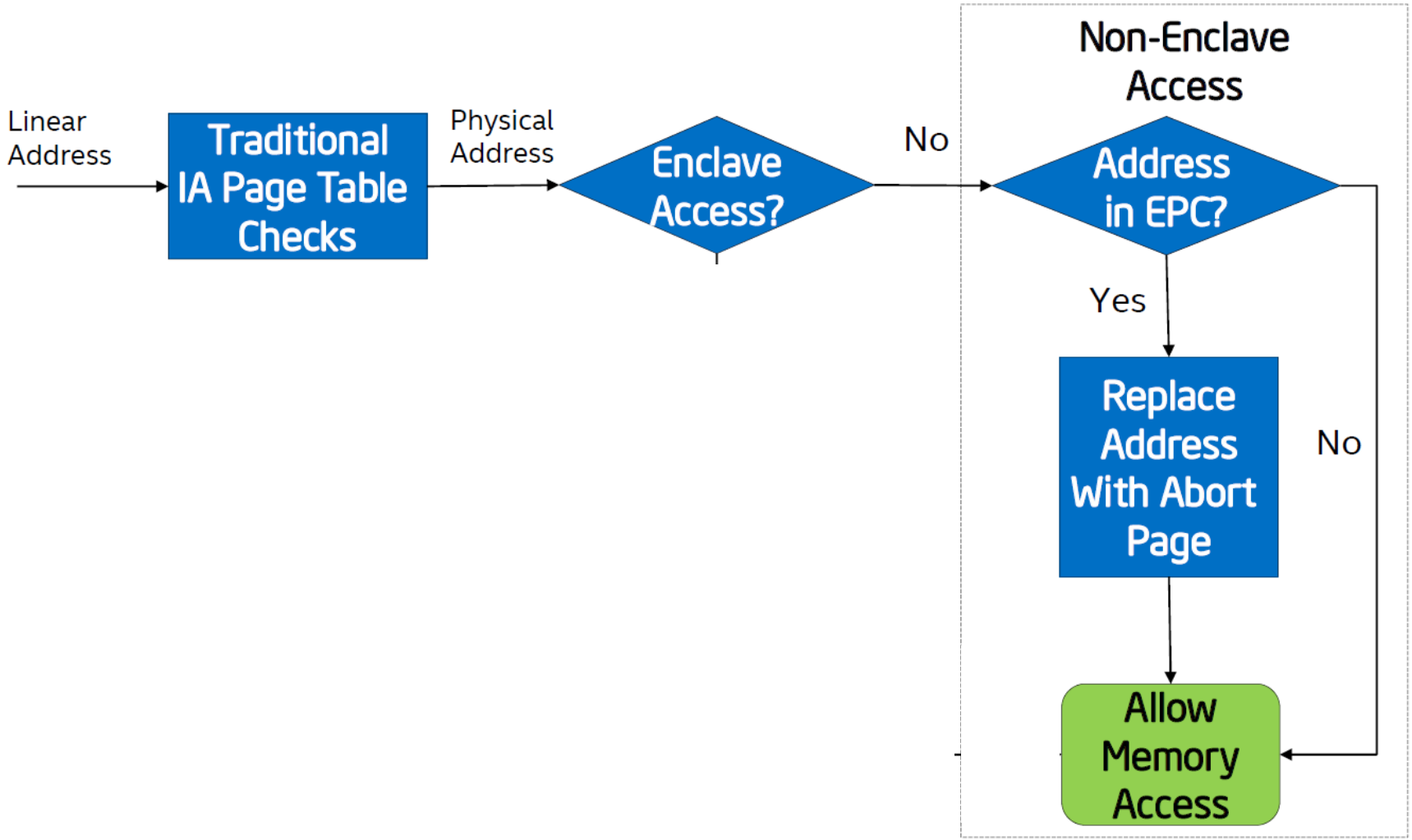


SGX Access Control

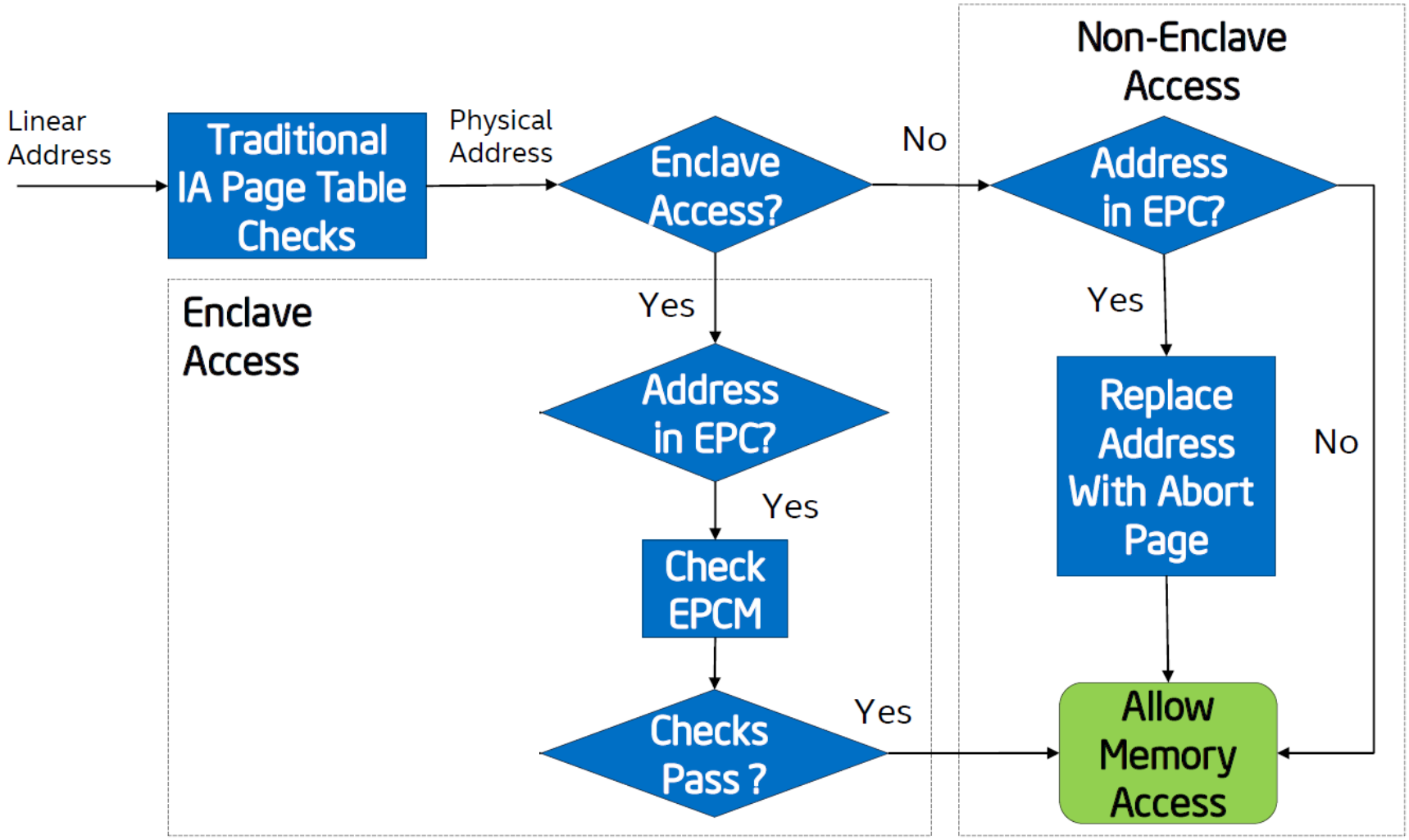
SGX Access Control



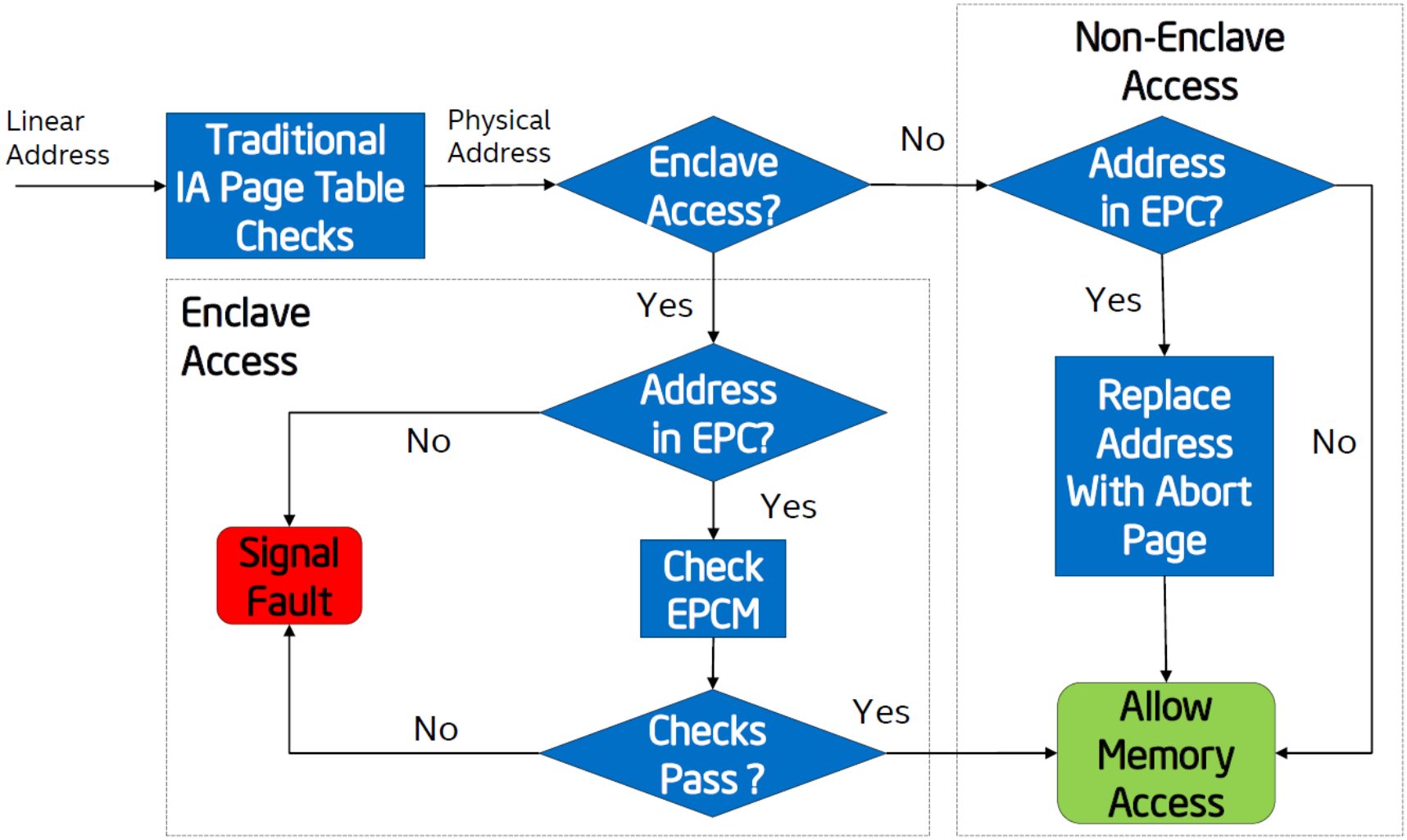
SGX Access Control



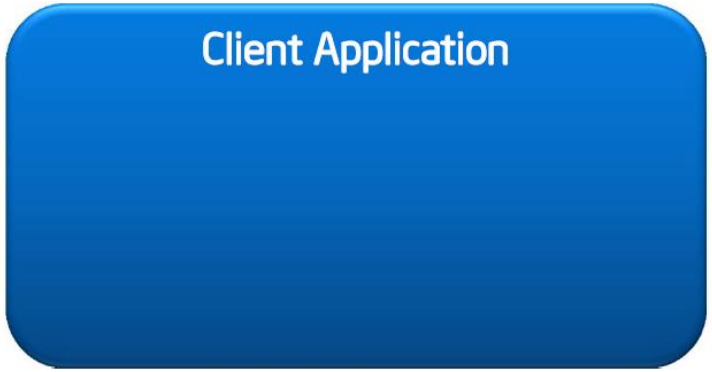
SGX Access Control



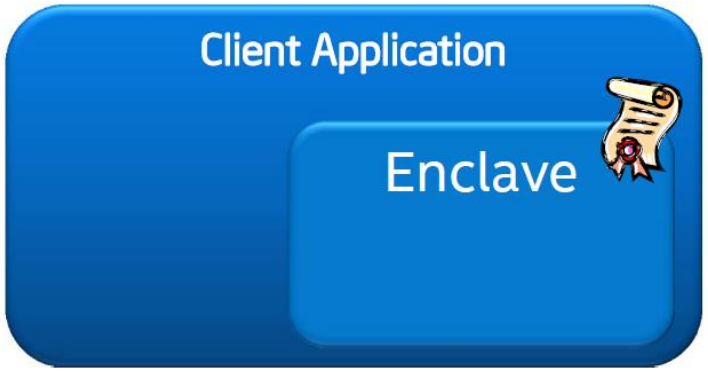
SGX Access Control



Critical Feature: Attestation and Sealing

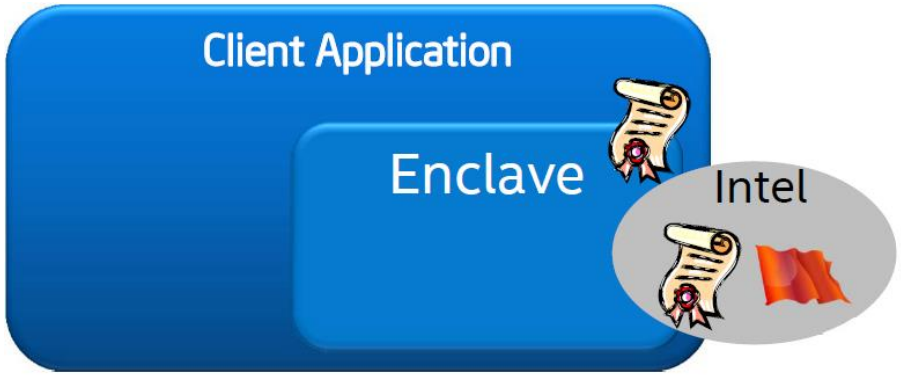


Critical Feature: Attestation and Sealing



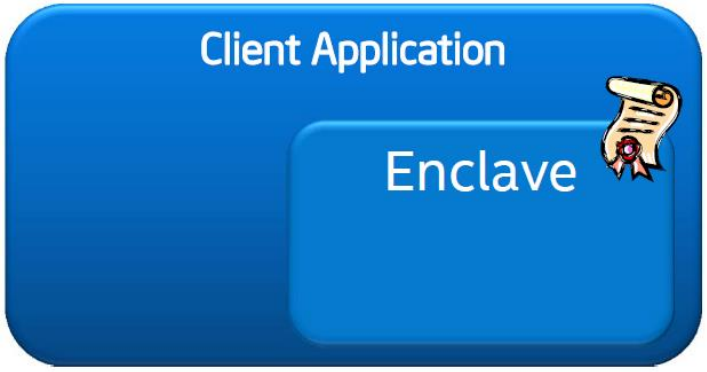
- Enclave built & measured

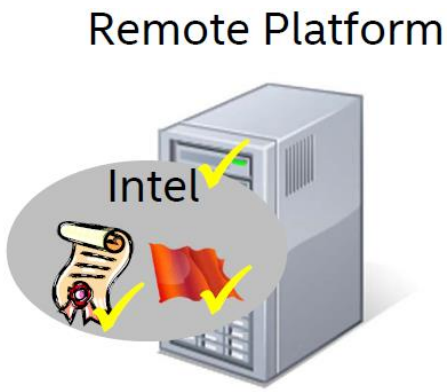
Critical Feature: Attestation and Sealing



- Enclave built & measured
- Enclave requests REPORT (HW-signed blob that includes enclave identity information)

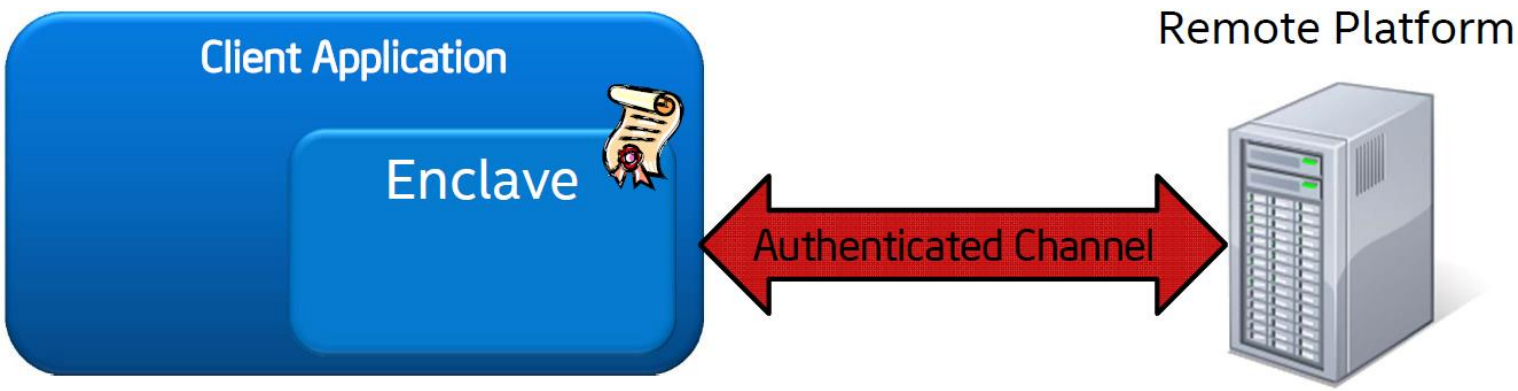
Critical Feature: Attestation and Sealing





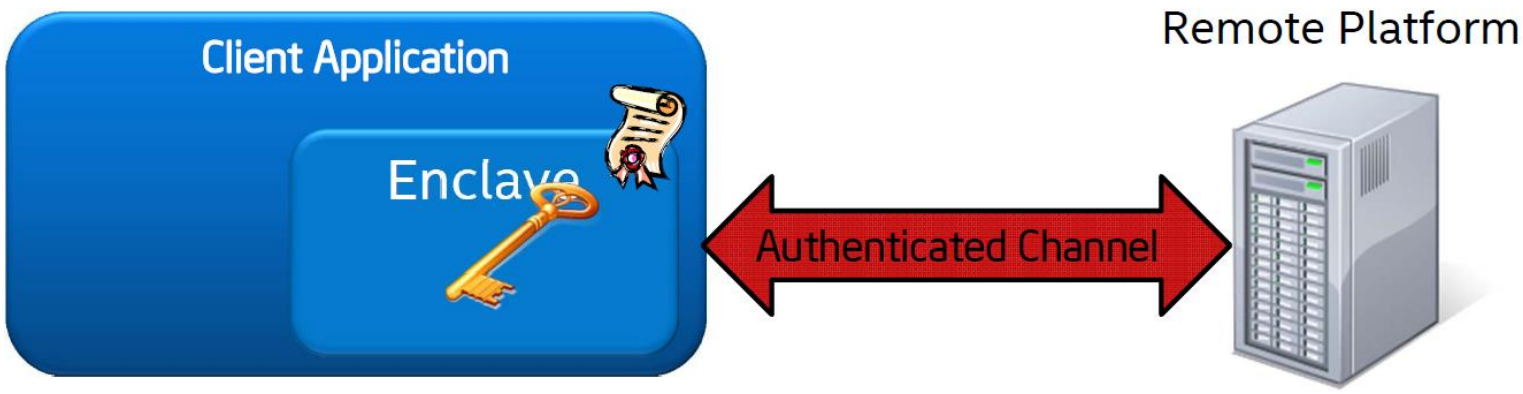
- Enclave built & measured
- Enclave requests REPORT (HW-signed blob that includes enclave identity information)
- REPORT sent to server & verified by the server

Critical Feature: Attestation and Sealing



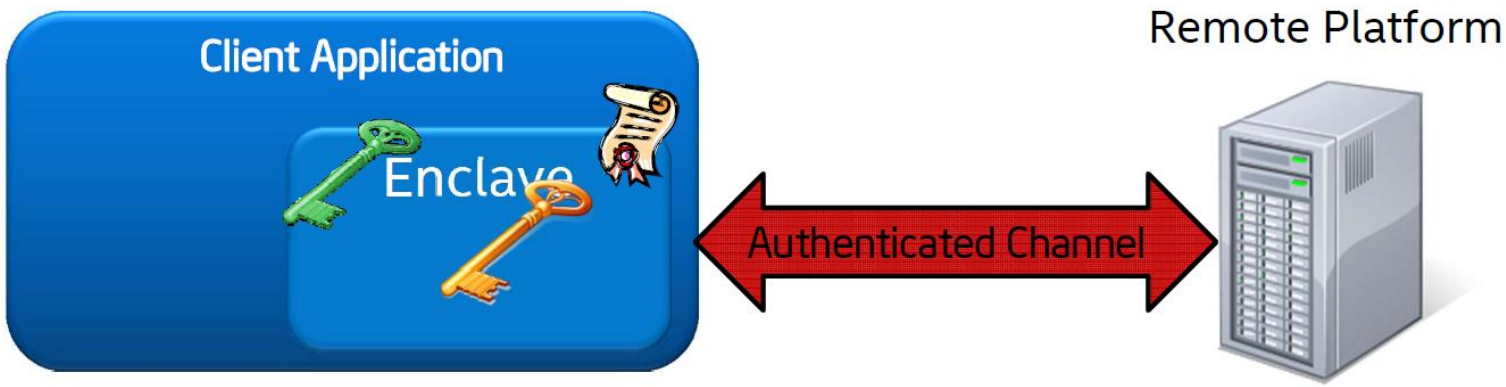
- Enclave built & measured
- Enclave requests REPORT (HW-signed blob that includes enclave identity information)
- REPORT sent to server & verified by the server

Critical Feature: Attestation and Sealing



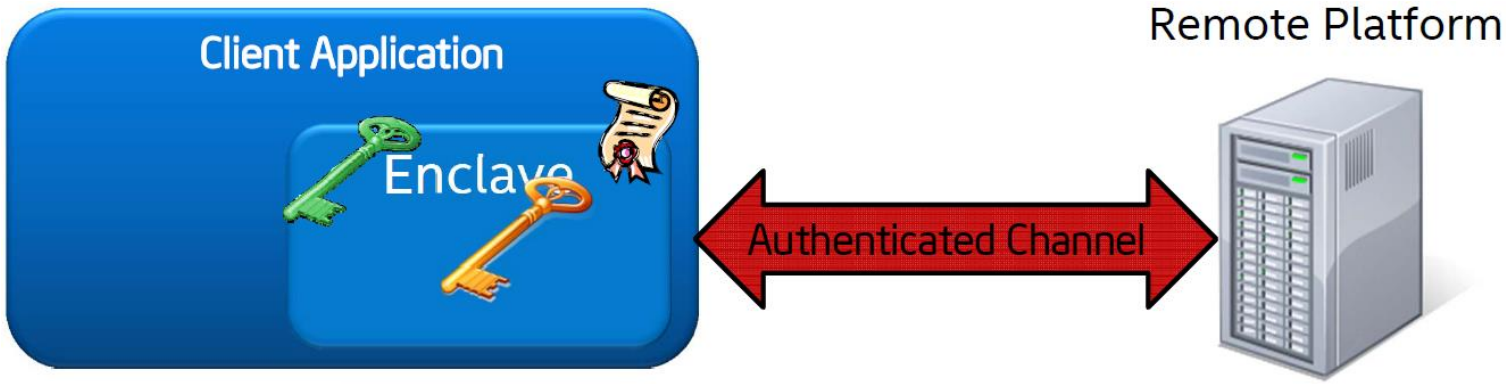
- Enclave built & measured
- Enclave requests REPORT (HW-signed blob that includes enclave identity information)
- REPORT sent to server & verified by the server
- Attestation Key sent to enclave, first secret provisioned

Critical Feature: Attestation and Sealing



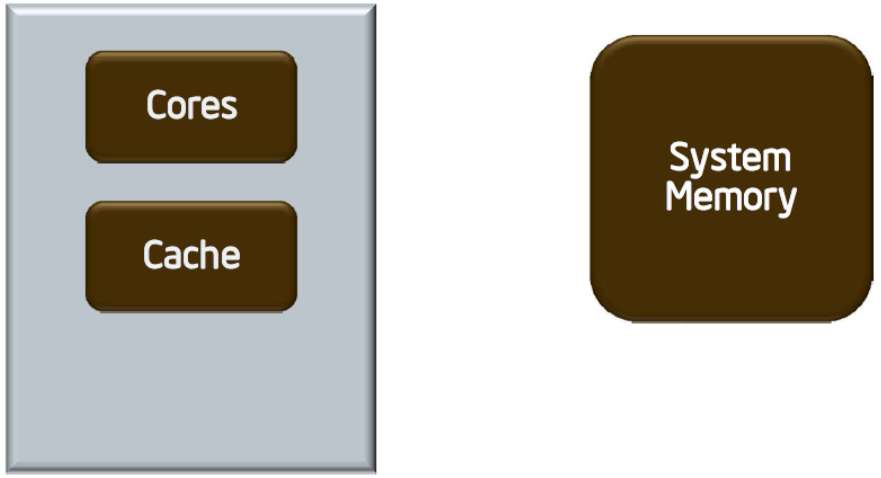
- Enclave built & measured
- Enclave requests REPORT (HW-signed blob that includes enclave identity information)
- REPORT sent to server & verified by the server
- Attestation Key sent to enclave, first secret provisioned
- Enclave-platform-specific Sealing Key generated (EGETKEY)

Critical Feature: Attestation and Sealing



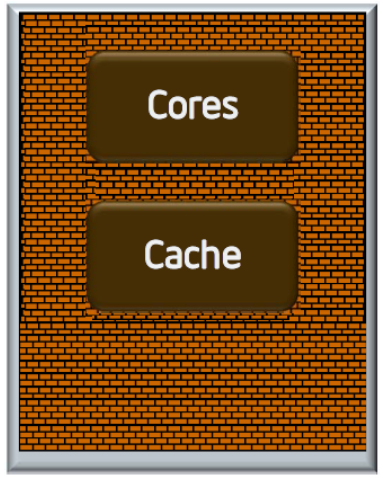
- Enclave built & measured
- Enclave requests REPORT (HW-signed blob that includes enclave identity information)
- REPORT sent to server & verified by the server
- Attestation Key sent to enclave, first secret provisioned
- Enclave-platform-specific Sealing Key generated (EGETKEY)
- Attestation Key encrypted via Sealing Key & stored for later use.

Protection against Memory Snooping Attacks



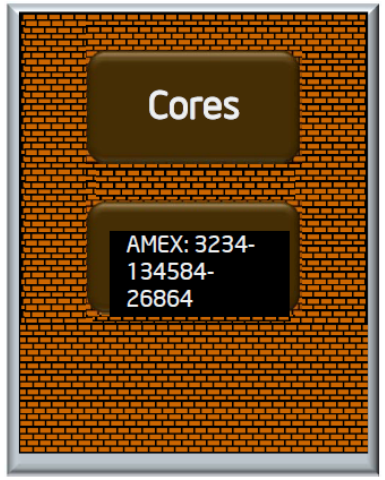
Protection against Memory Snooping Attacks

- Security perimeter is the CPU package boundary



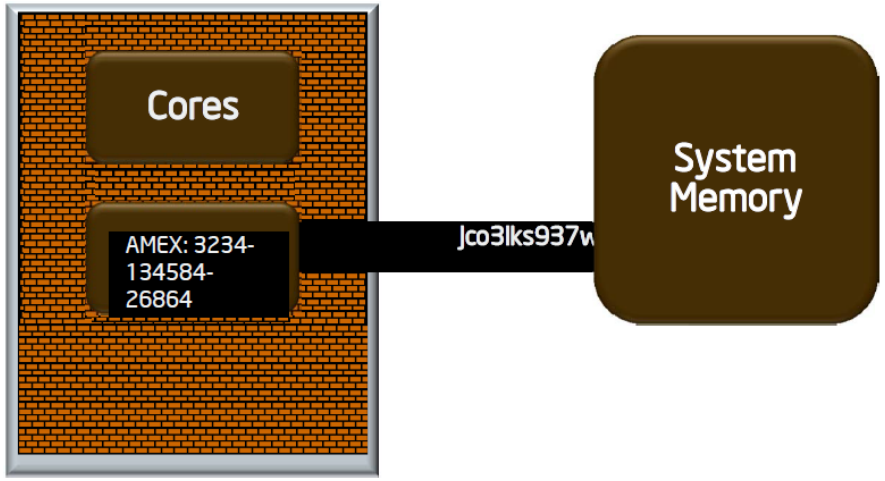
Protection against Memory Snooping Attacks

- Security perimeter is the CPU package boundary
- Data and code is unencrypted inside CPU package



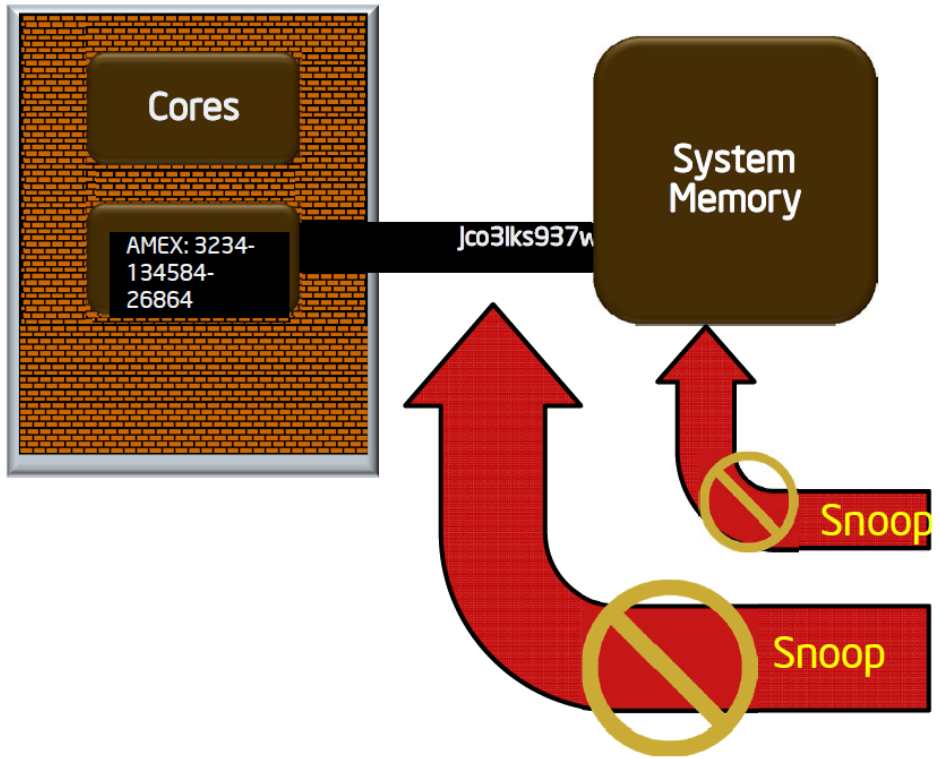
Protection against Memory Snooping Attacks

- Security perimeter is the CPU package boundary
- Data and code is unencrypted inside CPU package
- Data and code outside CPU package is encrypted and/or integrity checked



Protection against Memory Snooping Attacks

- Security perimeter is the CPU package boundary
- Data and code is unencrypted inside CPU package
- Data and code outside CPU package is encrypted and/or integrity checked
- External memory reads and bus snoops see only encrypted data
  - SGX does NOT protect against leakage via access patterns to the external memory!

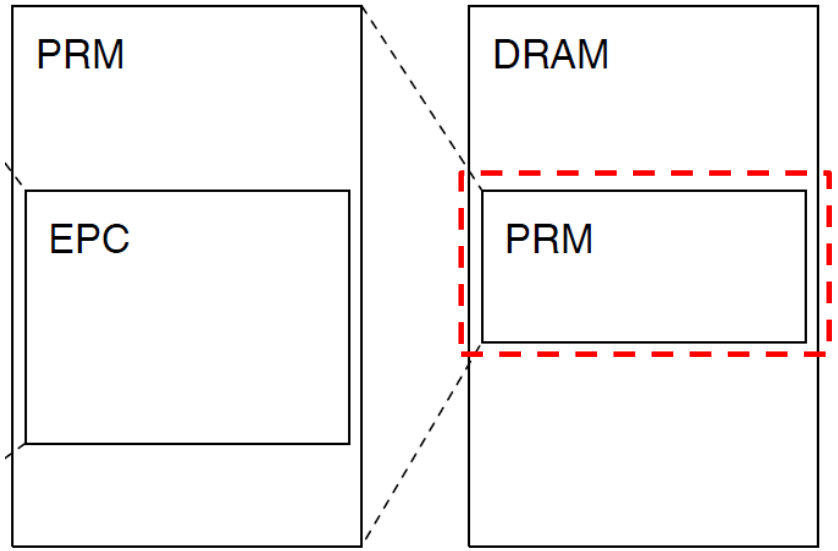


SGX Memory Organization

- Physical Memory Organization
- Memory Layout of SGX Enclave

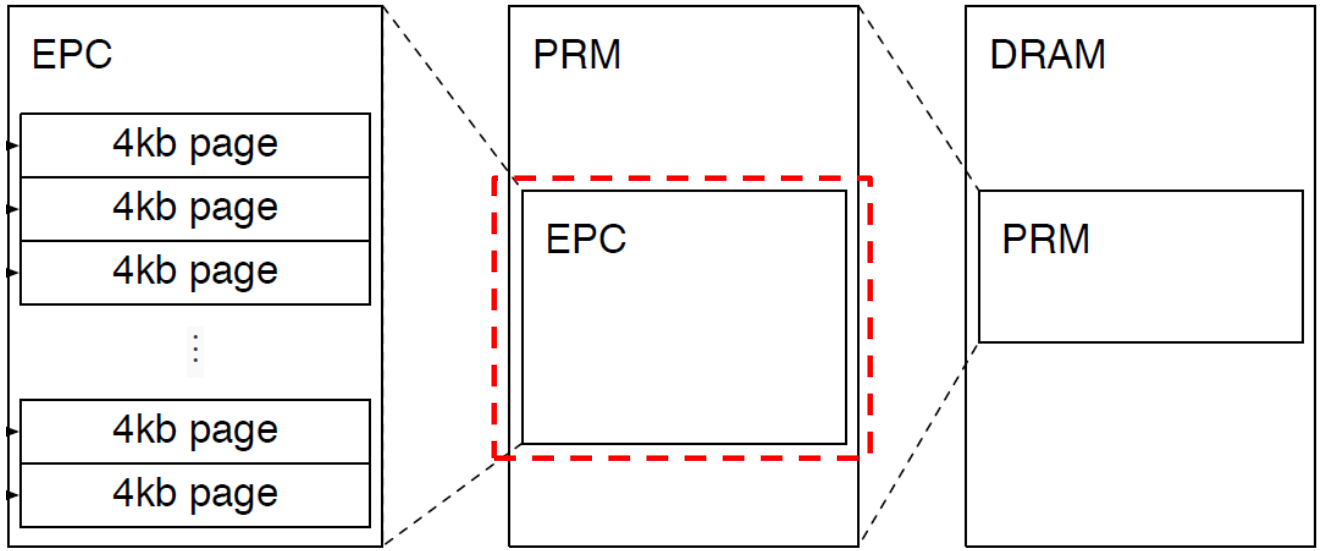
SGX Physical Memory Organization

- The Processor's Reserved Memory (PRM) is a reserved region in DRAM.



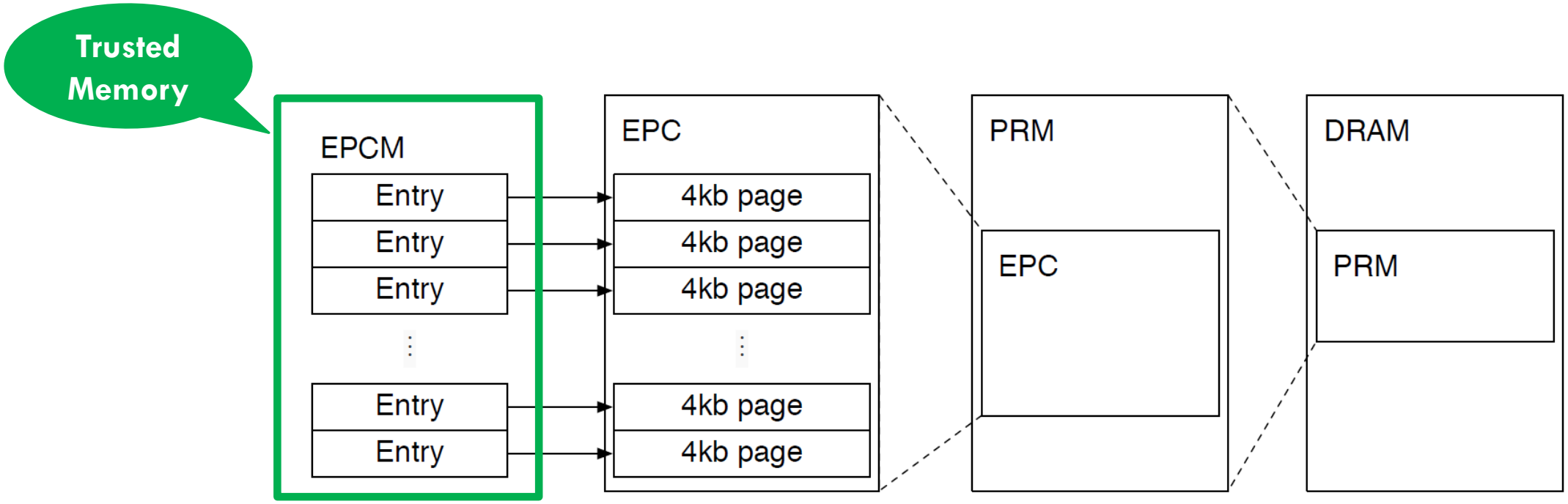
SGX Physical Memory Organization

- The Processor's Reserved Memory (PRM) is a reserved region in DRAM.
- The Enclave Page Cache (EPC) contains enclave's code & data.



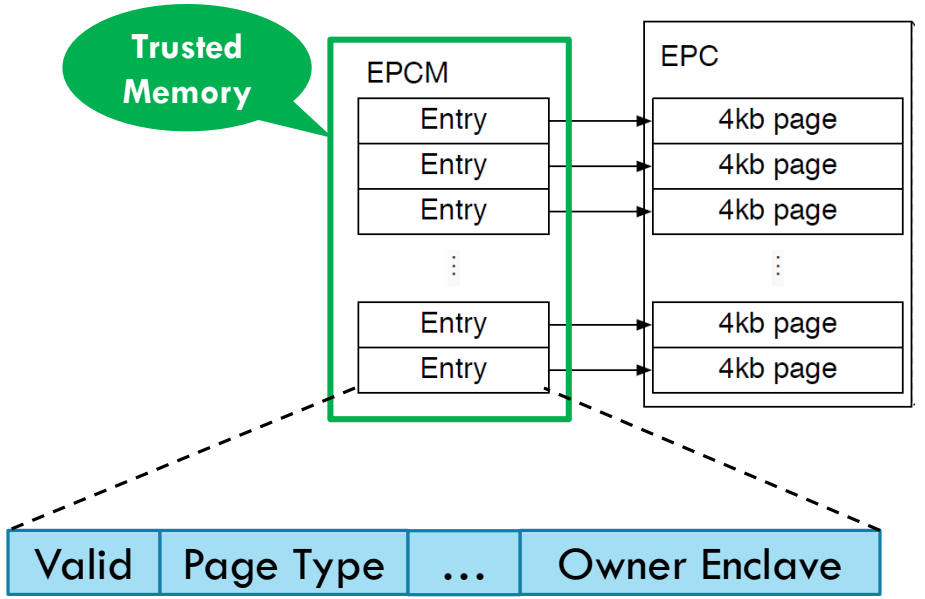
SGX Physical Memory Organization

- The Processor's Reserved Memory (PRM) is a reserved region in DRAM.
- The Enclave Page Cache (EPC) contains enclave's code & data.
- The Enclave Page Cache Map (EPCM) contains an entry to point to each EPC page.



Enclave Page Cache Map (EPCM)

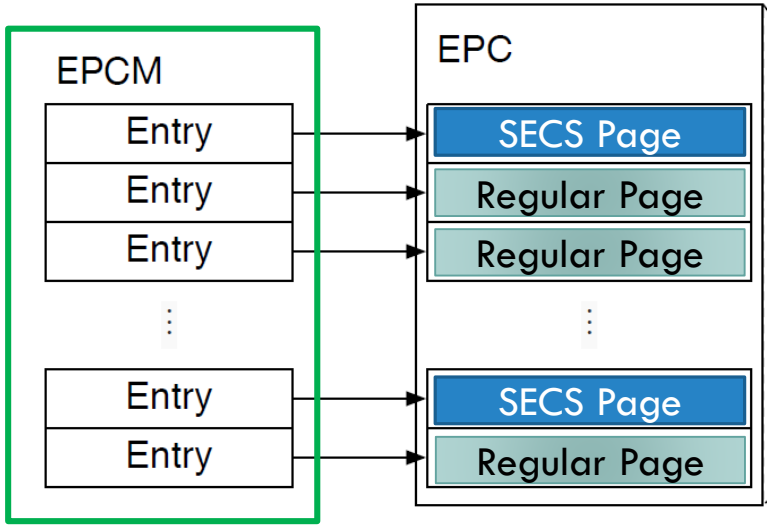
- EPCM entries are used by SGX hardware to perform checks that ensure that the (untrusted) OS is behaving as expected
  - E.g., the same EPC page cannot be allocated to two enclaves
- The EPCM's content is only used by SGX's security checks
  - The application and OS programmer can ignore it.
- EPCM Entry Fields
  - VALID → EPC page is available or allocated
  - PT → Page type, e.g. Regular (PT_REG), or SECS Page
  - ENCLAVESECS → Points to the SECS Page of owner enclave



Field	Bits	Description
VALID	1	0 for un-allocated EPC pages
PT	8	page type
ENCLAVESECS		identifies the enclave owning the page

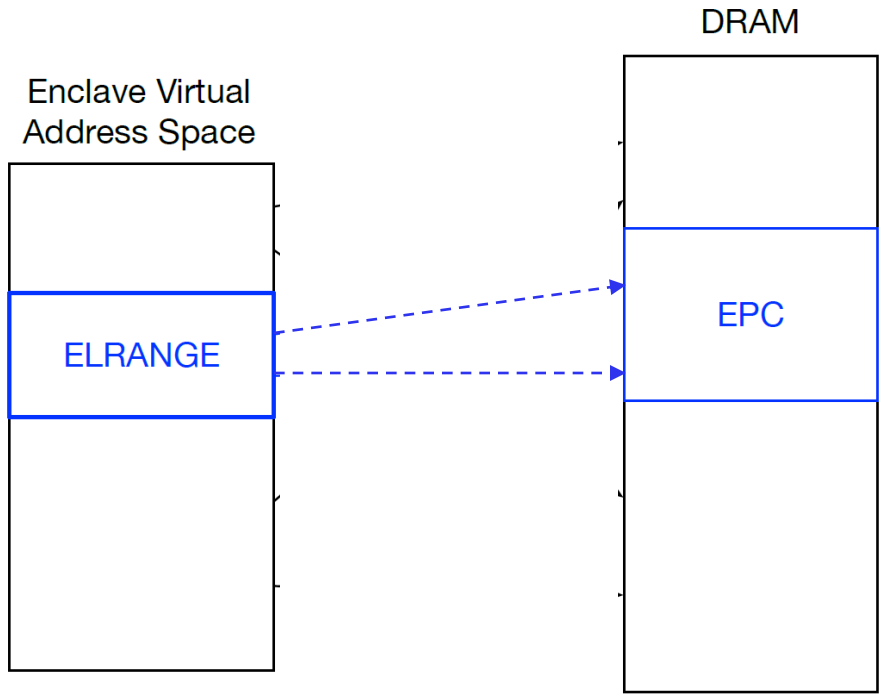
SGX Enclave Control Structure (SECS)

- The SGX Enclave Control Structure (SECS) stores critical metadata of each SGX enclave
  - E.g., enclave's measurement for software attestation
  - Enclave Attributes
- Each SECS is stored in a dedicated EPC page with the page type PT_SECS.
- SECS Pages cannot be accessed by:
  - System Software (OS/Hypervisor etc.)
  - Even the enclave's code itself.



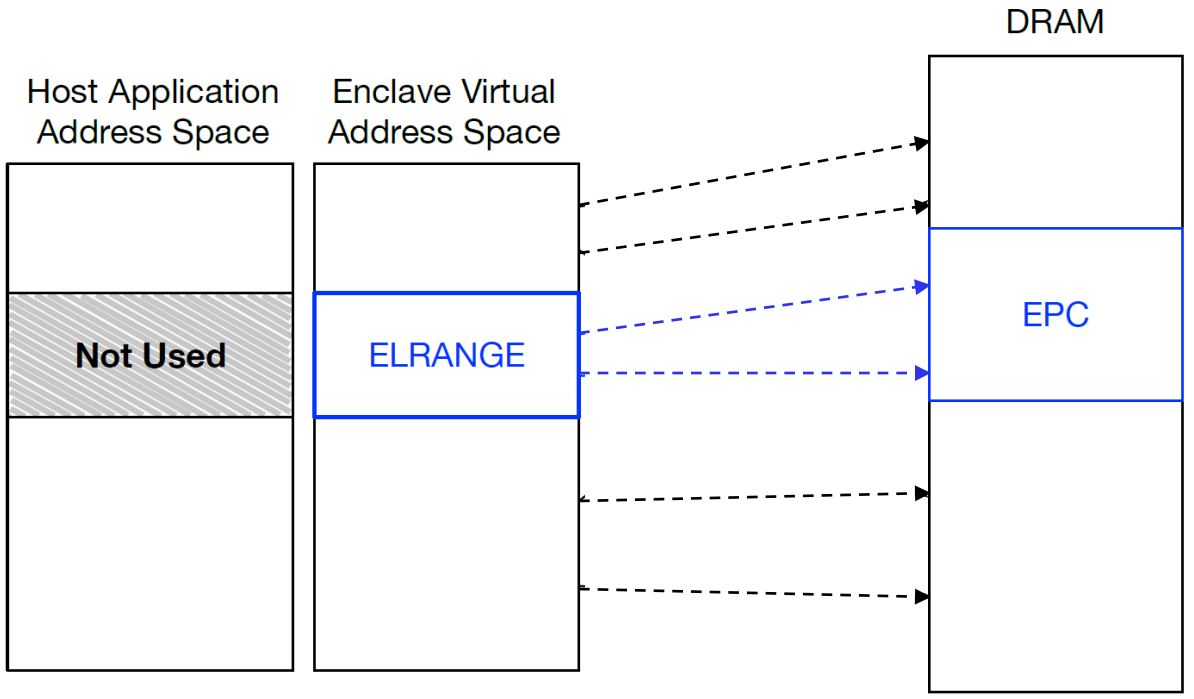
The Enclave Linear Address Range (ELRANGE)

- EPC pages are accessed using a dedicated region in the enclave's virtual address space, called ELRANGE.



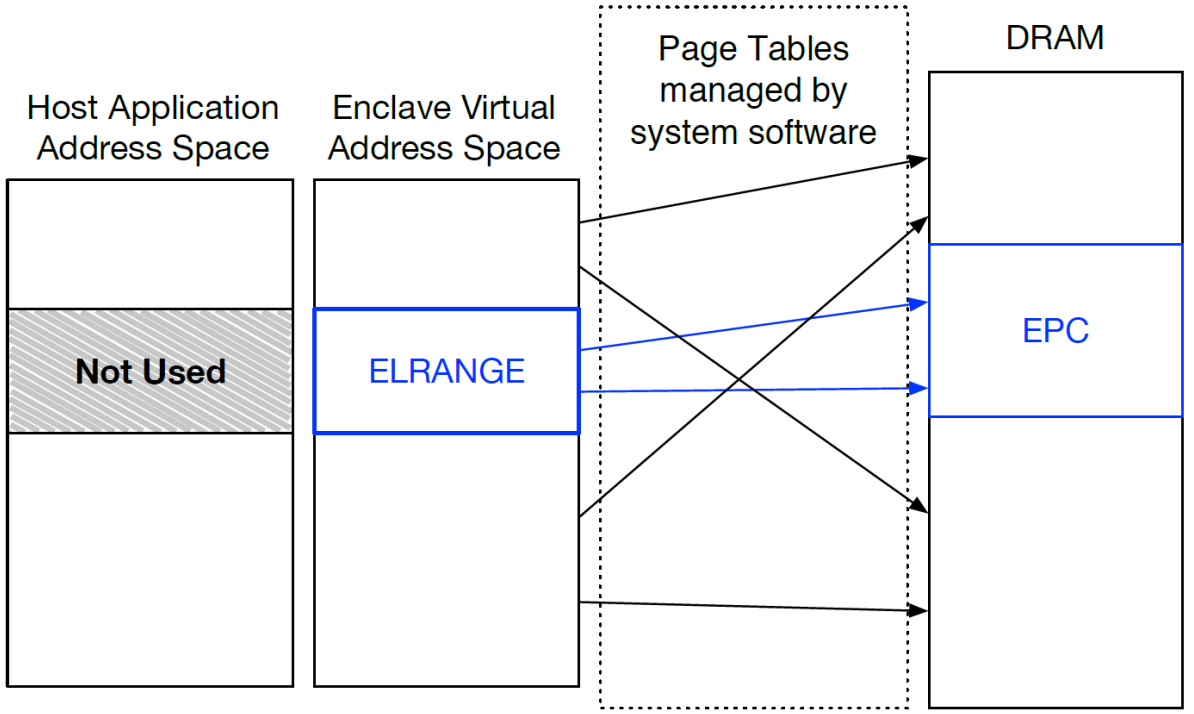
The Enclave Linear Address Range (ELRANGE)

- EPC pages are accessed using a dedicated region in the enclave's virtual address space, called ELRANGE.
- The rest of the virtual address space is used to access the memory of the host process.



The Enclave Linear Address Range (ELRANGE)

- EPC pages are accessed using a dedicated region in the enclave's virtual address space, called ELRANGE.
- The rest of the virtual address space is used to access the memory of the host process.
- The memory mappings are established using the page tables managed by system software.



SGX Enclave Attributes

An enclave's attributes are the sub-fields in the ATTRIBUTES field of the enclave's SECS.

- DEBUG → Enables *Read/Write* enclave's memory in Debug mode.
- XFRM → Defines *Extended Features Request Mask* to specify architectural extensions.
- MODE64BIT → Set to true for enclaves that use the 64-bit Intel architecture.

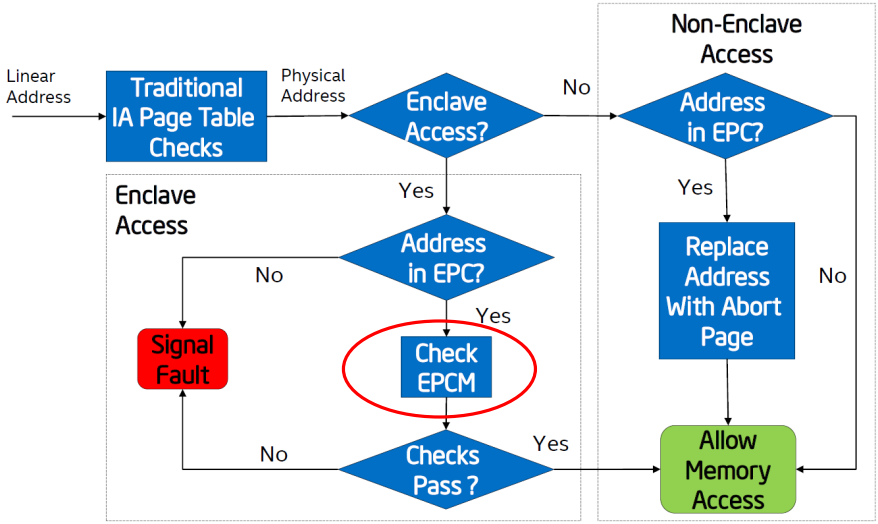
Field	Bits	Description
DEBUG	1	Opts into enclave debugging features.
XFRM	64	The value of XCR0 (§ 2.6) while this enclave's code is executed.
MODE64BIT	1	Set for 64-bit enclaves.

Address Translation for SGX Enclaves

- The OS and hypervisor are in full control of the page tables and EPTs.
- Each enclave's code uses this address translation → Possible Security Problems
- When an EPC page is allocated, its intended virtual address is recorded in the EPCM entry for the page, in the ADDRESS field.
- Upon address translation, given virtual address is verified against the stored one!
- Also, R/W/X attributes from EPCM entry override the permissions specified in page tables.

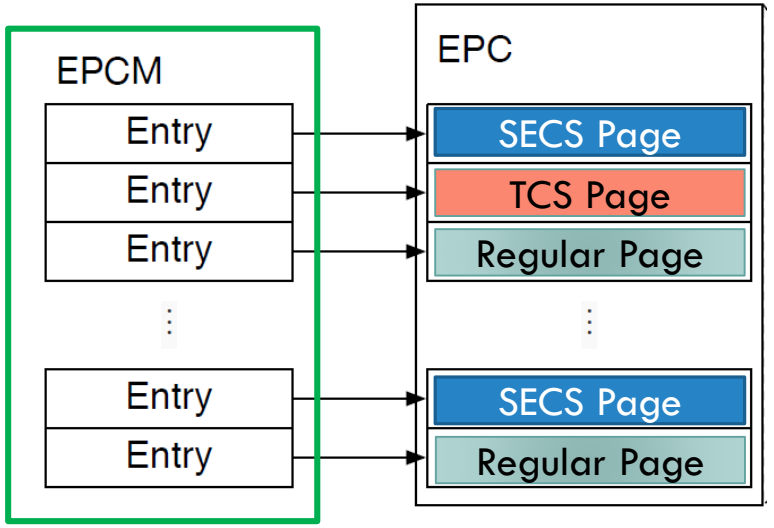
EPCM Entry:

Field	Bits	Description
ADDRESS	48	the virtual address used to access this page
R	1	allow reads by enclave code
W	1	allow writes by enclave code
X	1	allow execution of code inside the page, inside enclave



The Thread Control Structure (TCS)

- It is possible for multiple logical processors (Threads) to concurrently execute the same enclave's code at the same time, via different threads.
- SGX implementation uses a Thread Control Structure (TCS) for each thread that executes an enclave's code.
- Each TCS is stored in a dedicated EPC Page.
- The contents of an EPC page that holds a TCS cannot be directly accessed, even by the code of the enclave that owns the TCS
  - Similar restriction as EPC pages holding SECS instances.



The Life Cycle of an SGX Enclave

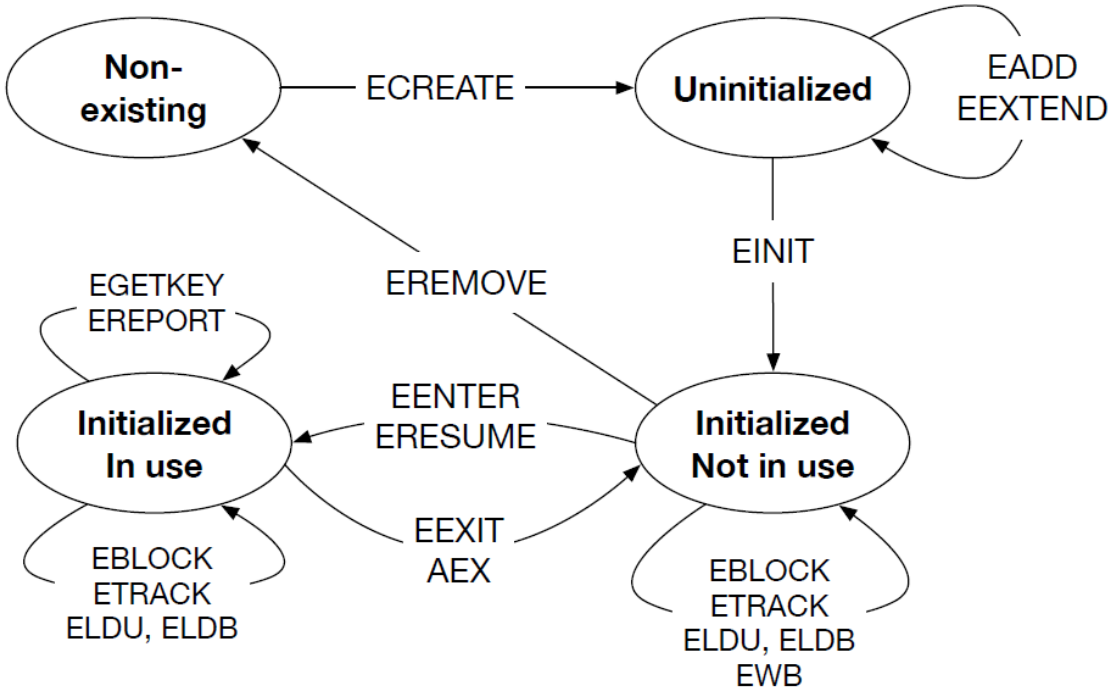
- Overview
- Relevant SGX Instructions
- Example

The Life Cycle of an SGX Enclave

An enclave's life cycle is all about the allocation of EPC pages.

Following are the major transitions during an SGX enclave's life cycle:

- Creation (ECREATE)
- Loading (EADD, EEXTEND)
- Initialization (EINIT)
- Enter/Exit the Enclave (EENTER/EEXIT)
- Teardown (EREMOVE)



Enclave Creation (ECREATE)

- Creates a unique instance of an enclave, establishes the linear address range, and serves as the enclave's root of trust
  - Enclave mode of operation (32/64)

- This information is stored within a Secure Enclaves Control Structure (SECS) generated by ECREATE.

Loading (EADD, EEXTEND)

EADD

- Add Regular (REG) or Thread Control Structure (TCS) pages into the enclave
  - System software responsible for selecting free EPC page, type, and attributes, content of the page and the enclave to which the page added to.
- Initial EPCM entry to indicate type of page (REG, TCS)
  - Linear address, RWX, associate the page to enclave SECS

EEXTEND

- Generates a cryptographic hash of the content of the enclave in 256Byte chunks
  - EEXTEND 16 times for measuring a 4K page

Initialization (EINIT)

- Verifies the enclave's content against the ISV's signed SIGSTRUCT and initializes the enclave – Mark it ready to be used
  - Validate SIGSTRUCT is signed using SIGSTRUCT public key
  - Enclave measurement matches the measurement specified in SIGSTRUCT.
- Enclave attributes compatible with SIGSTRUCT
- Record sealing identity (sealing authority, product id, SVN) in the SECS

Synchronous Enclave Entry (EENTER)

- Check that Thread Control Structure (TCS) is not busy and flush TLB entries for enclave addresses.
- Transfer control from outside enclave to pre-determined location inside the enclave
- Change the mode of operation to be in enclave mode
- Save RSP/RBP for later restore on enclave asynchronous exit
- Save XCR0 and replace it with enclave XFRM value

Synchronous Enclave Exit (EEXIT)

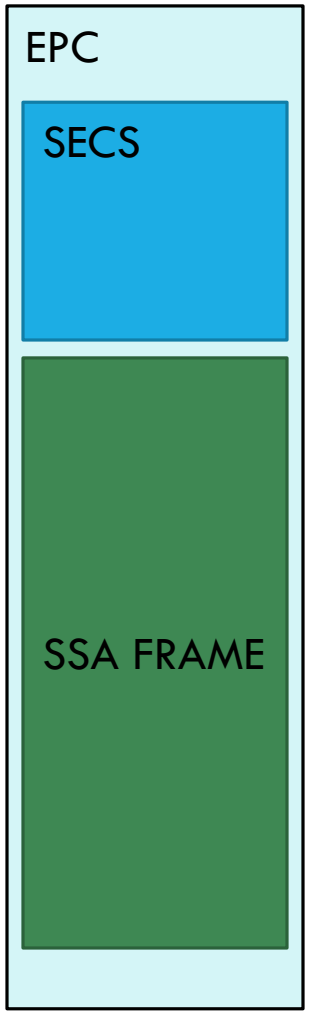
- Clear enclave mode and TLB entries for enclave addresses.
- Transfer control from inside enclave to a location outside the enclave
  - Mark TCS as not busy
- Responsibility to clear register state is on enclave writer!

Teardown (EREMOVE)

- EREMOVE deallocates/removes a 4KByte page permanently from the EPC
- A page cannot be removed until there is no thread executing code inside this enclave.
- A SECS page cannot be removed until all the regular pages of this enclave are removed.
- The SECS page is removed at the very last, and this also destroys the Enclave.

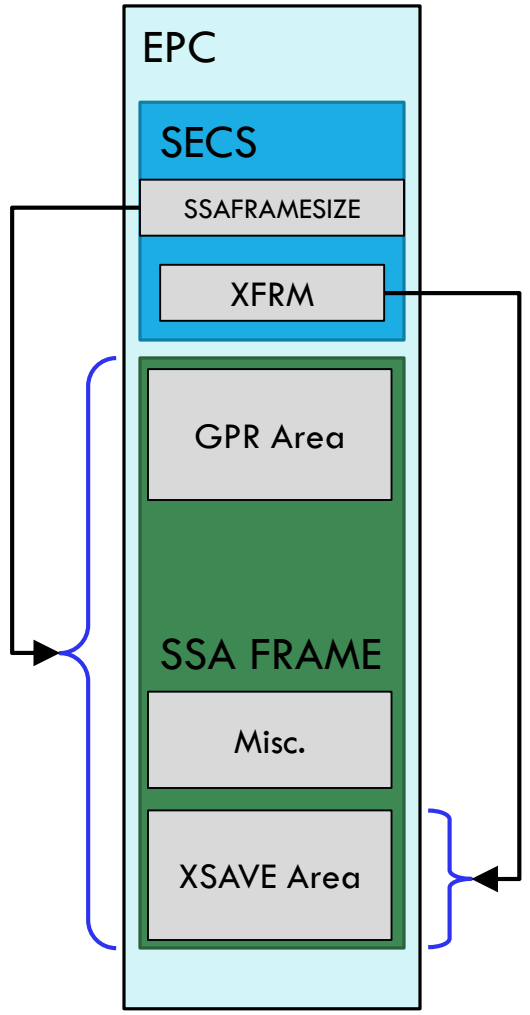
Context Switching/Exception Handling

- A context switch or exception (e.g. Interrupt) may occur during Enclave's code execution.
- Enclave's execution context must be stored
  - General Purpose Registers (GPRs)
  - Special Registers indicated by Requested-Feature BitMap (RFBM) register
- The area used to store an enclave thread's execution context while a hardware exception is handled is called **State Save Area (SSA)**
  - SSA is implemented by special EPC Page(s)
  - Notice that the saved context (i.e. SSA) is protected being inside the EPC

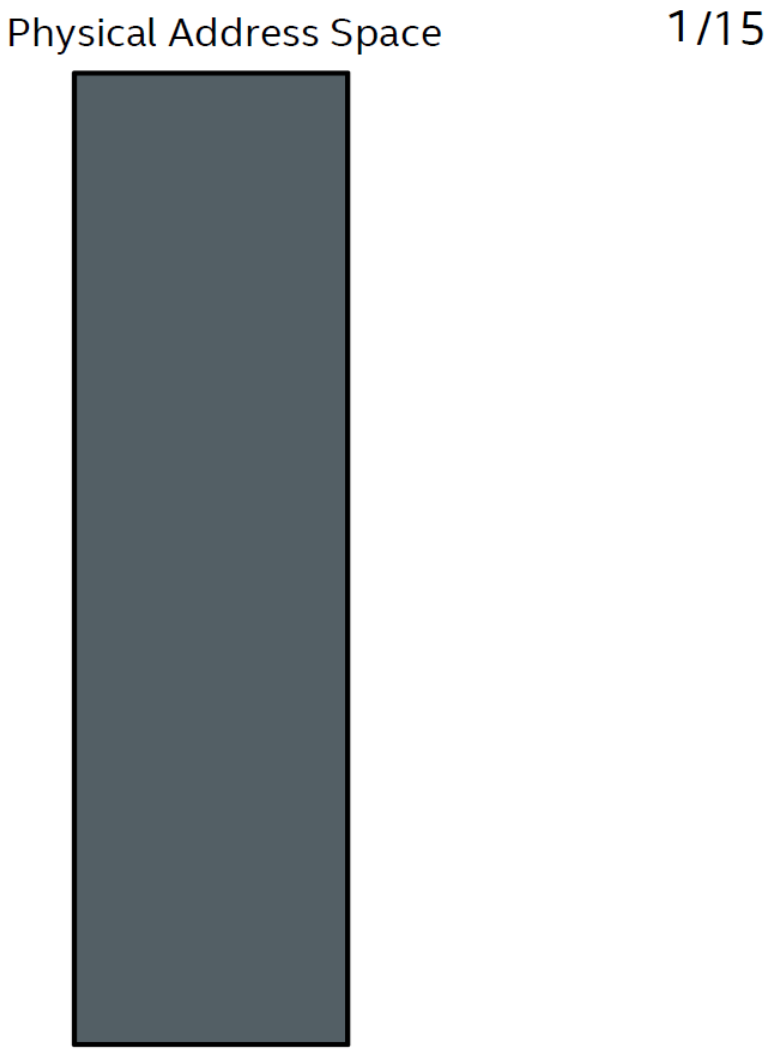


The State Save Area (SSA)

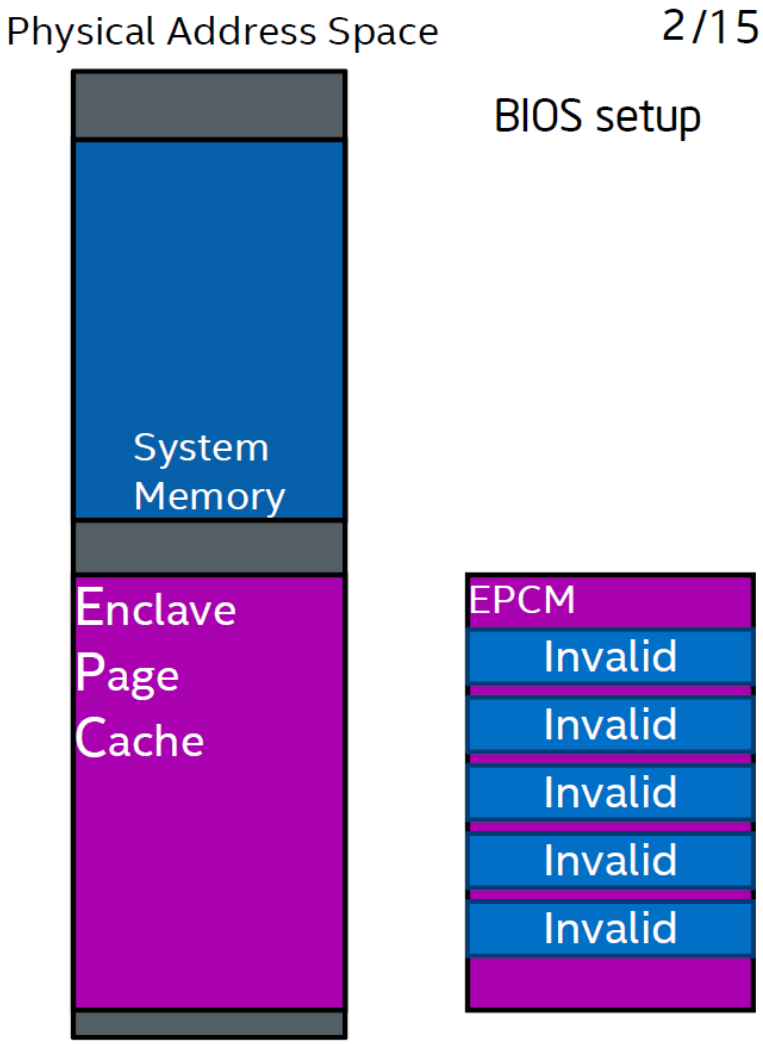
- **State Save Area (SSA)** stores the Enclave's execution context
- SSAFRAMESIZE field in SECS defines size of the SSA frame (in pages)
- GPRs are saved to GPR area on top of SSA frame
- Special Feature Registers are saved into XSAVE area at bottom of SSA frame
  - XFRM field in SECS controls the size of XSAVE area
- ECREATE instruction checks that all areas fit within an SSA frame



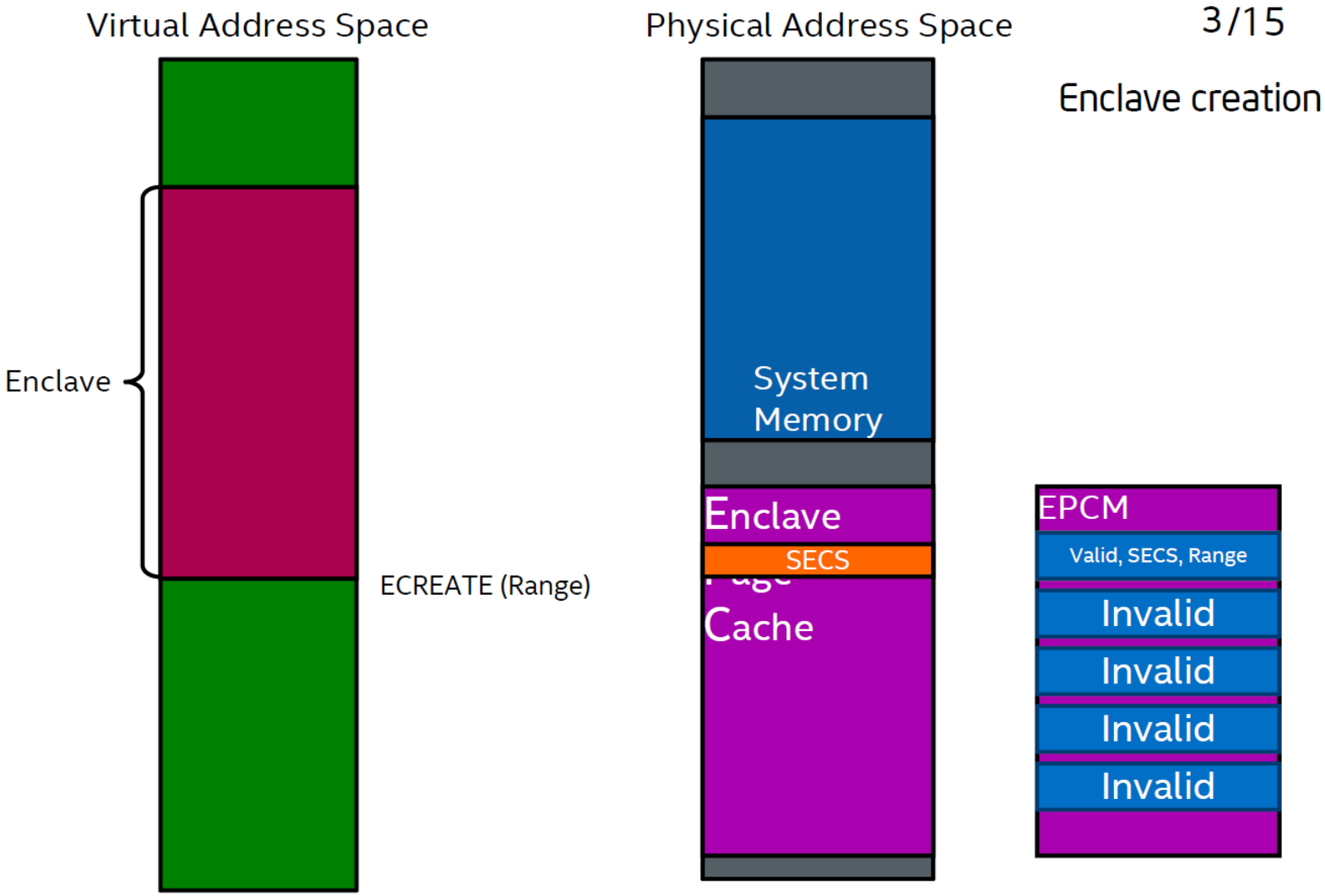
SGX Enclave Life Cycle Example



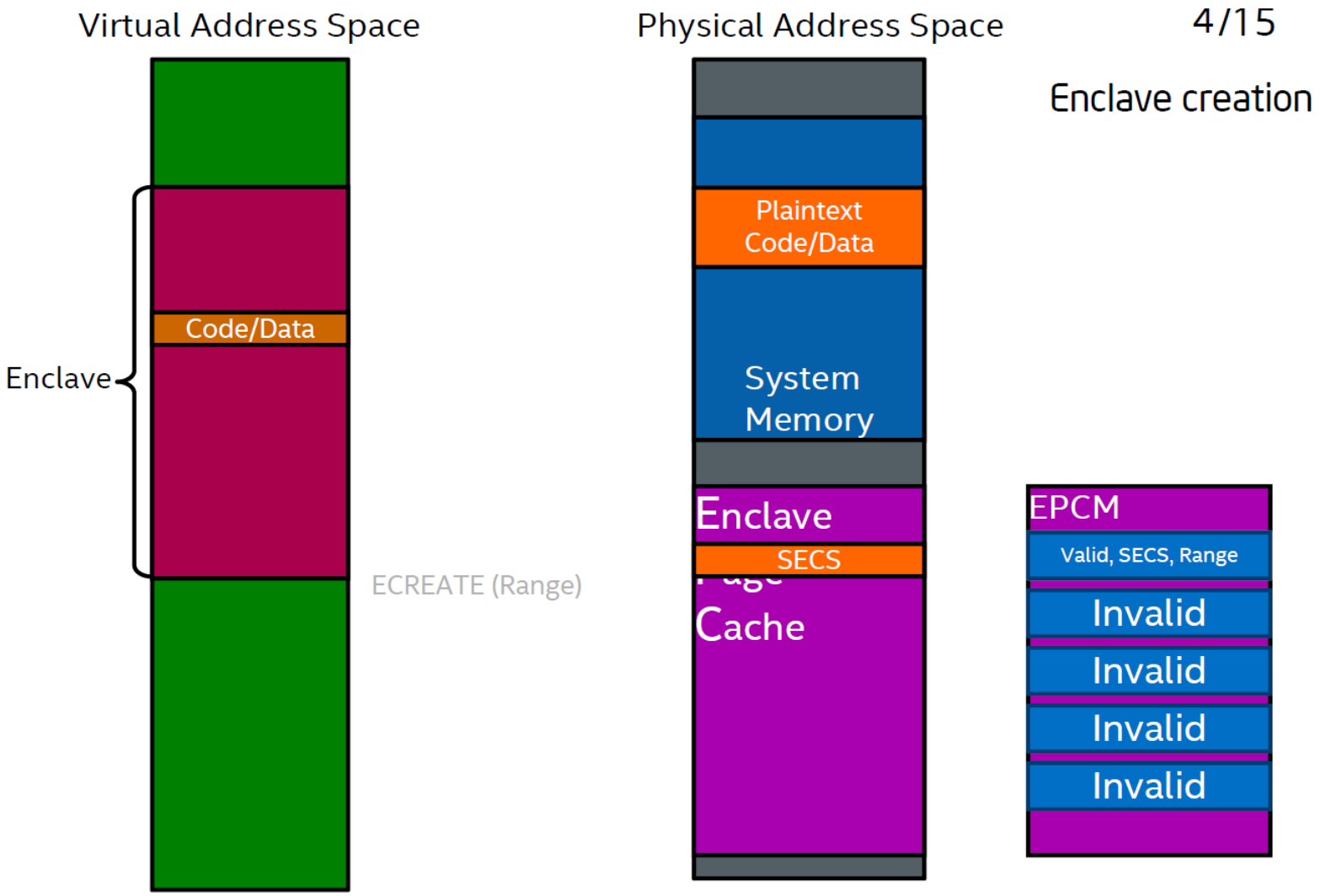
SGX Enclave Life Cycle Example



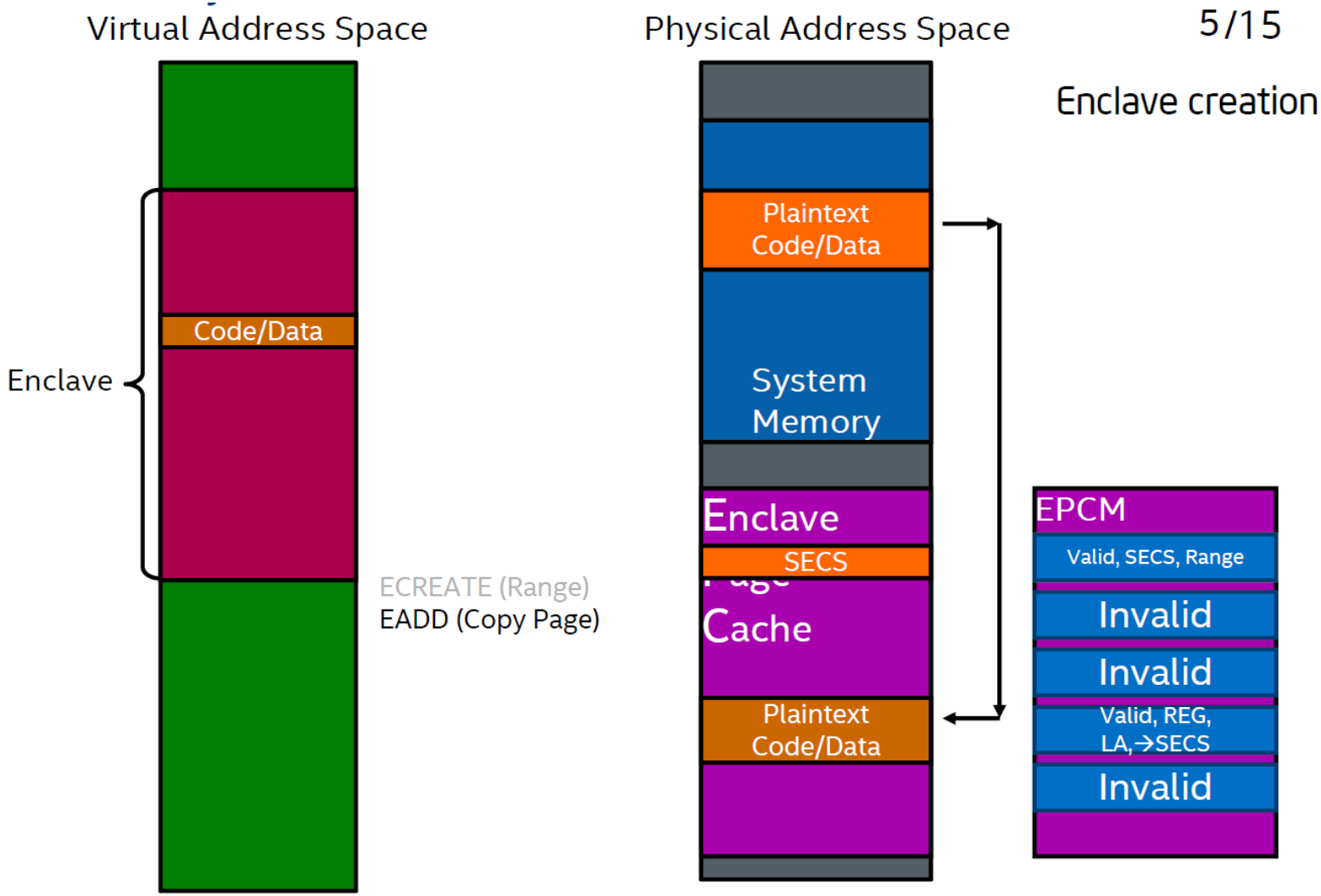
SGX Enclave Life Cycle Example



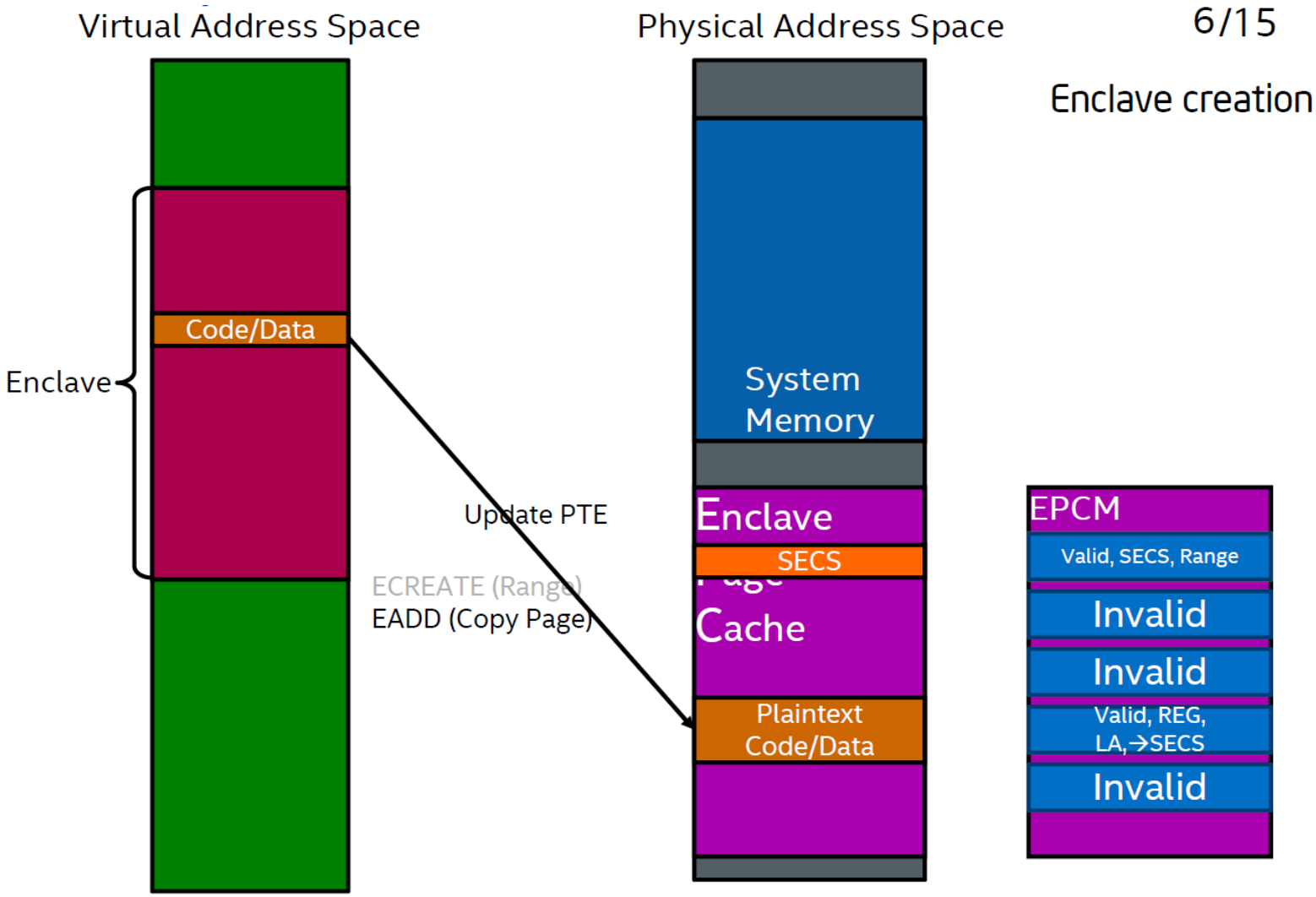
SGX Enclave Life Cycle Example



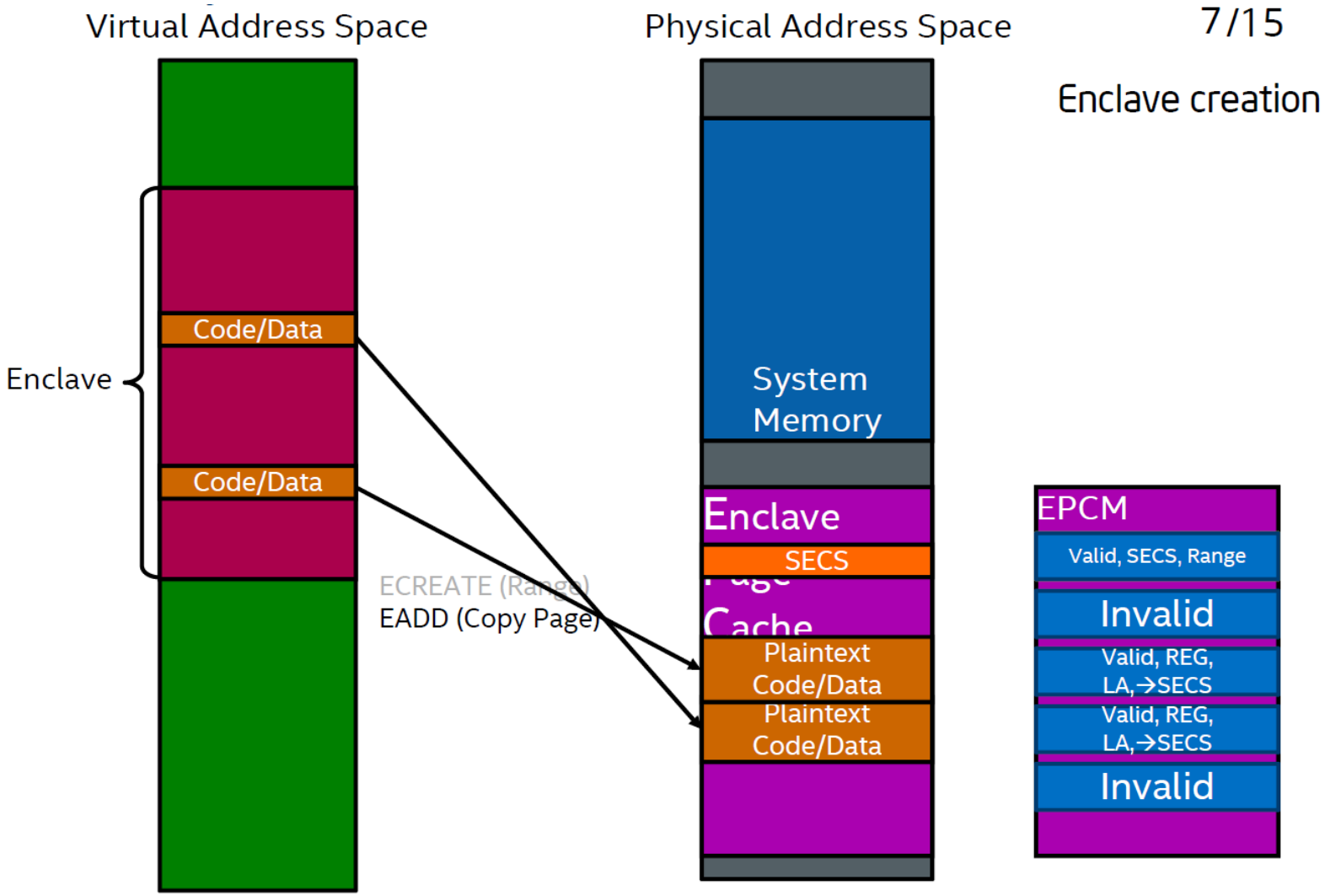
SGX Enclave Life Cycle Example



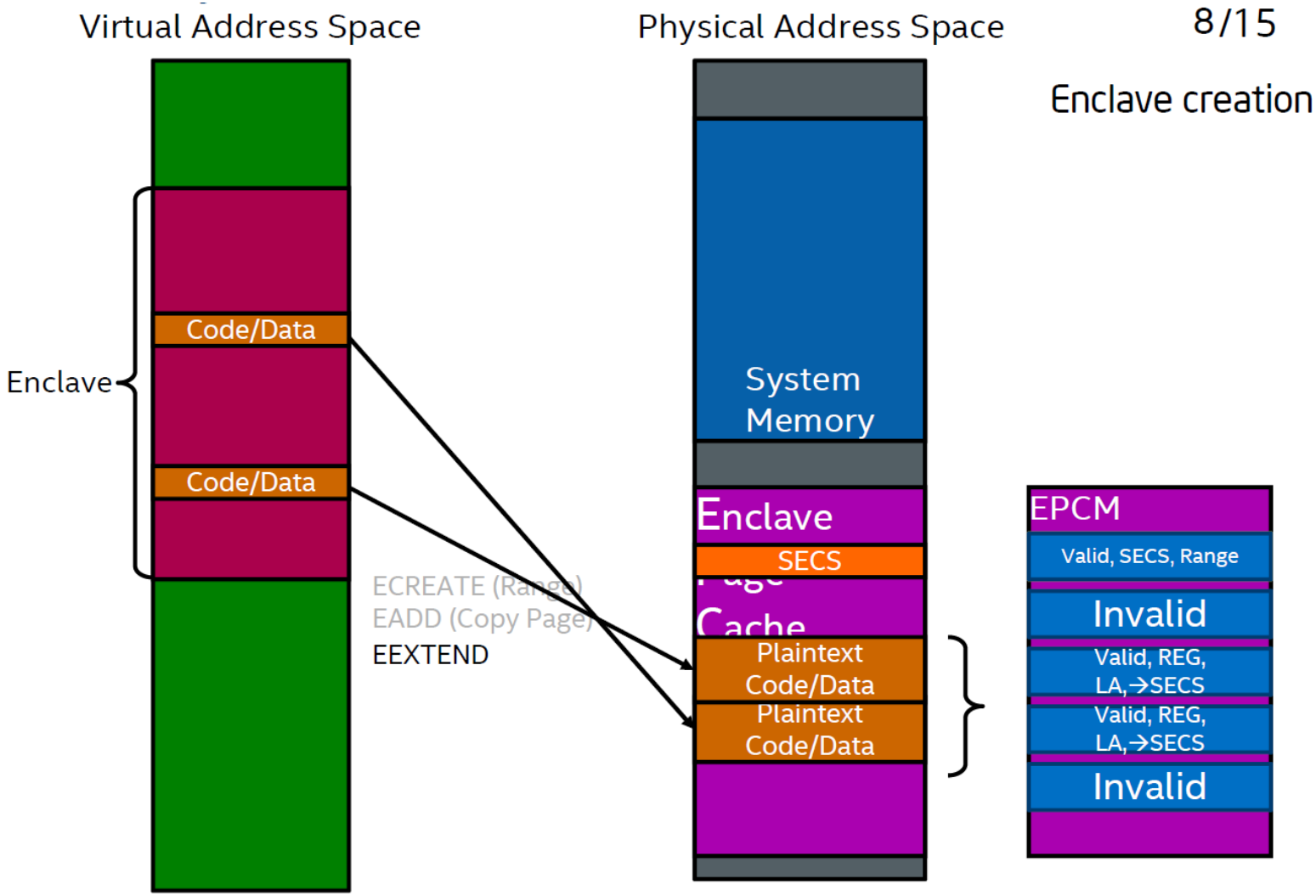
SGX Enclave Life Cycle Example



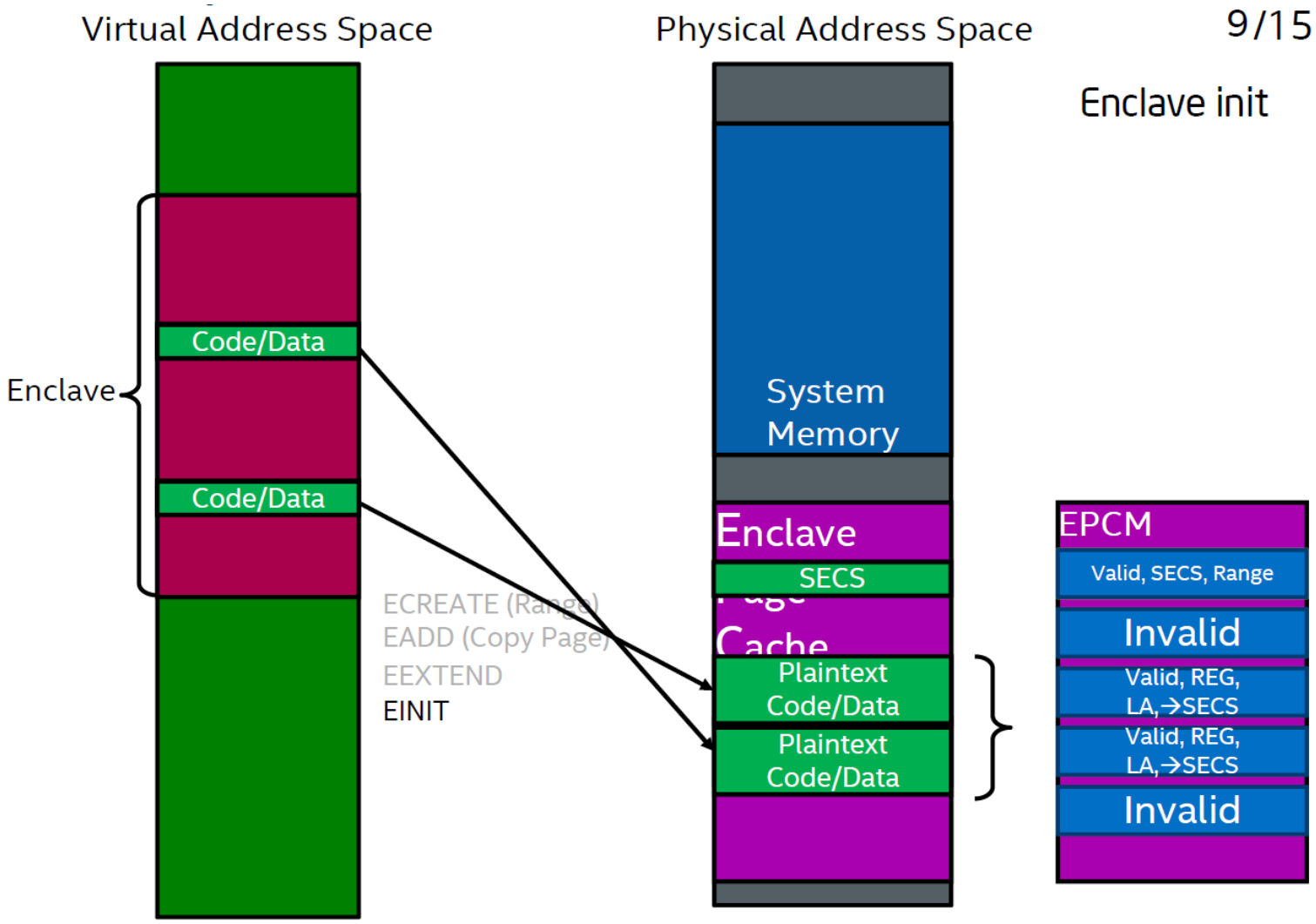
SGX Enclave Life Cycle Example



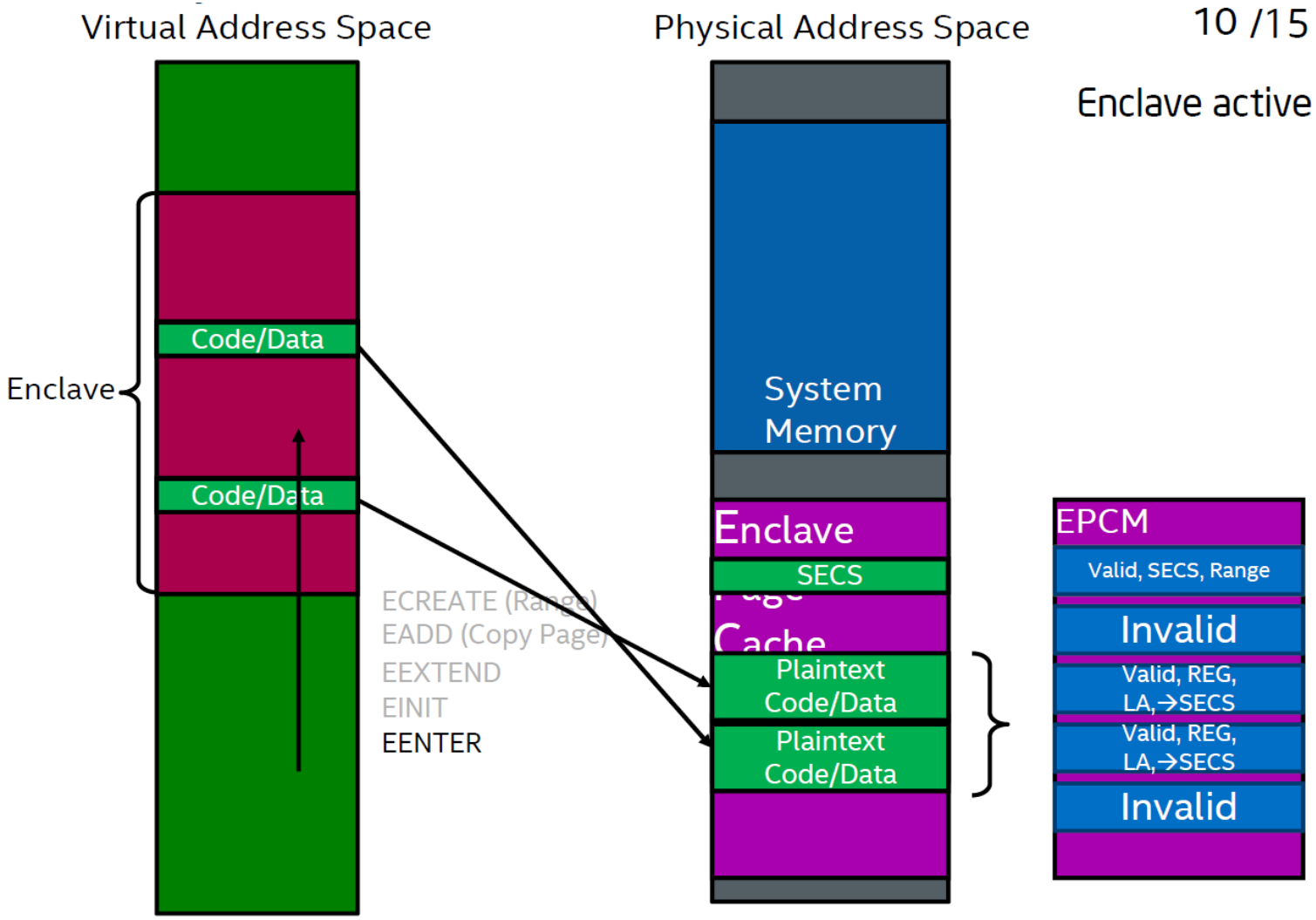
SGX Enclave Life Cycle Example



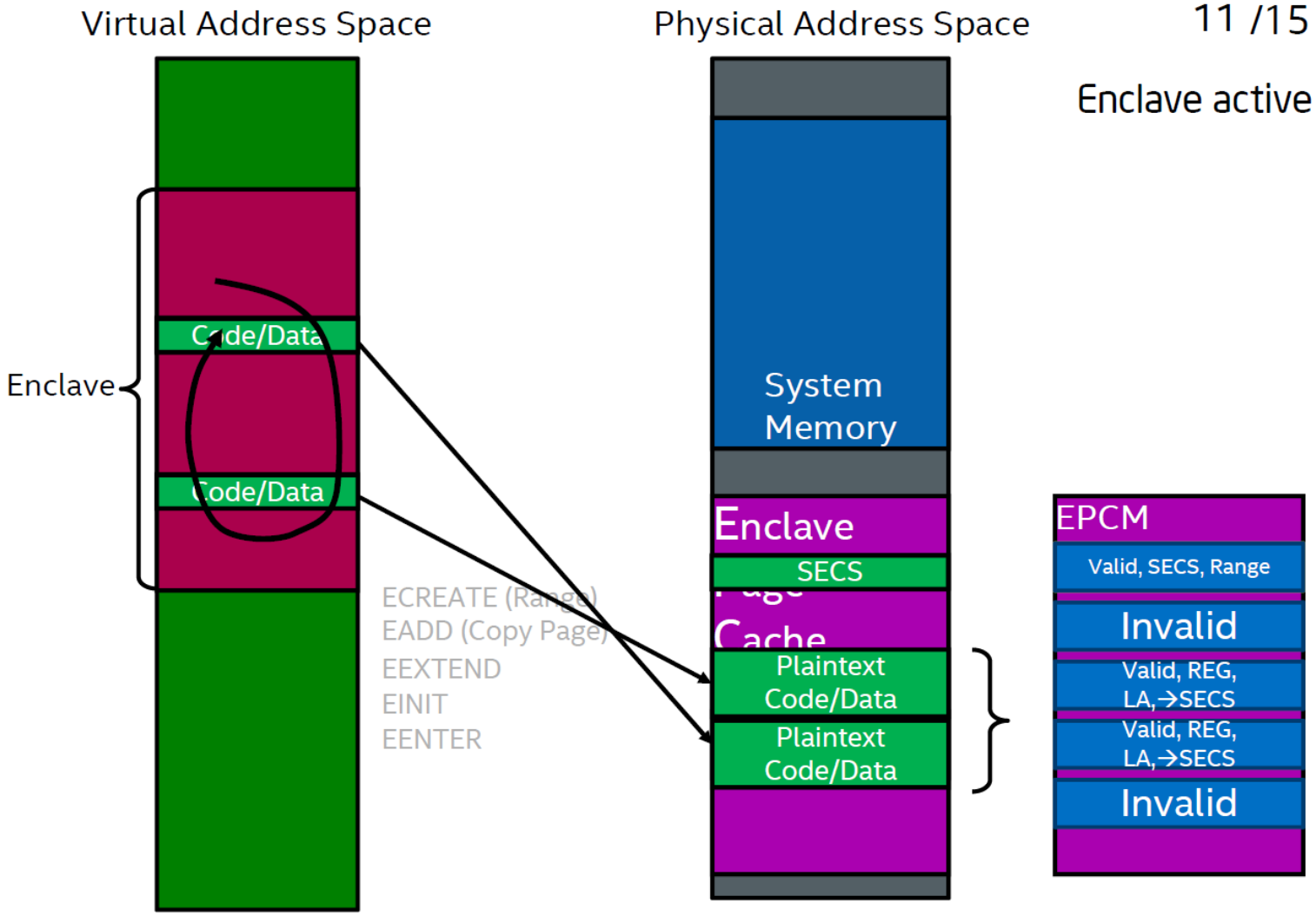
SGX Enclave Life Cycle Example



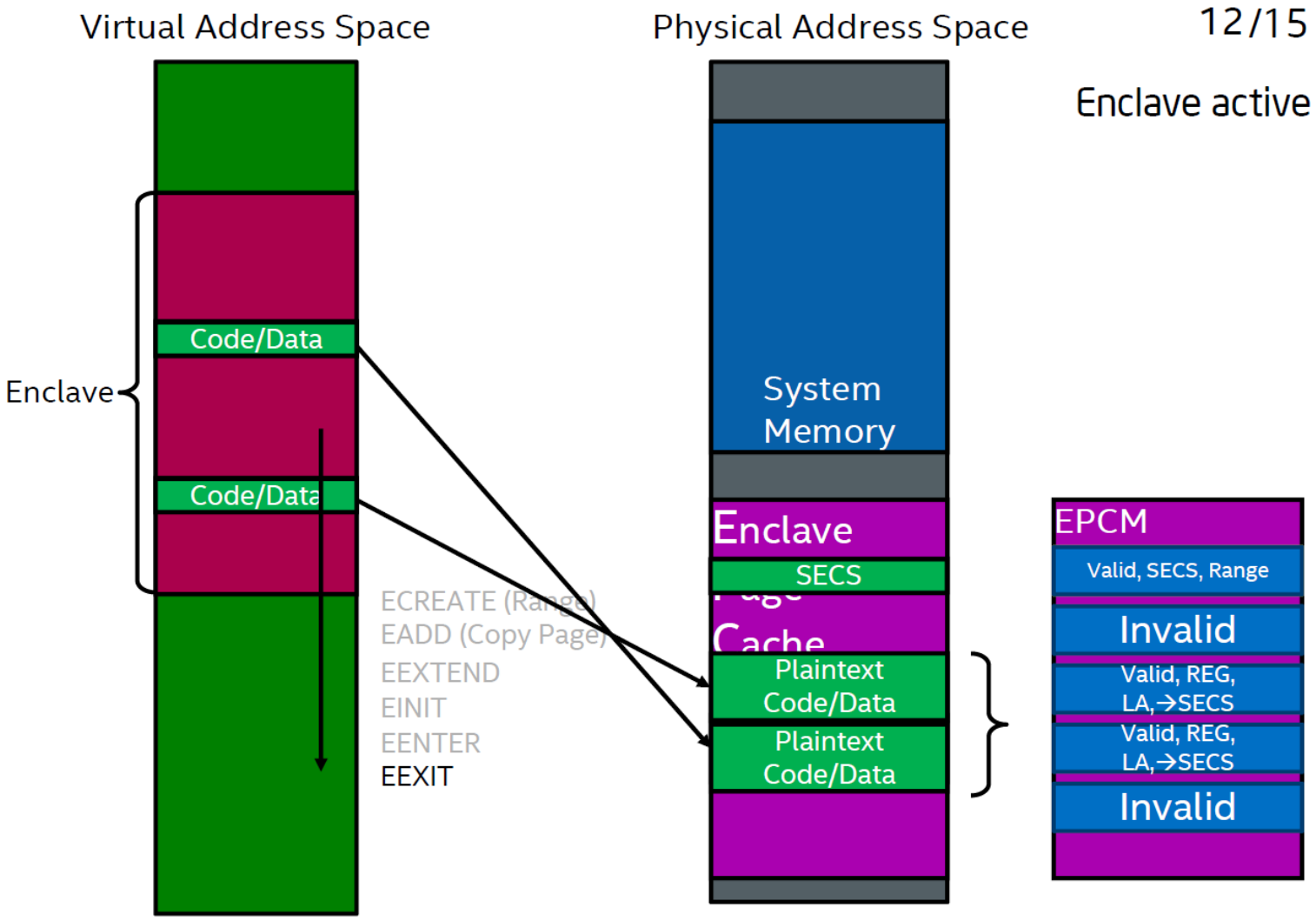
SGX Enclave Life Cycle Example



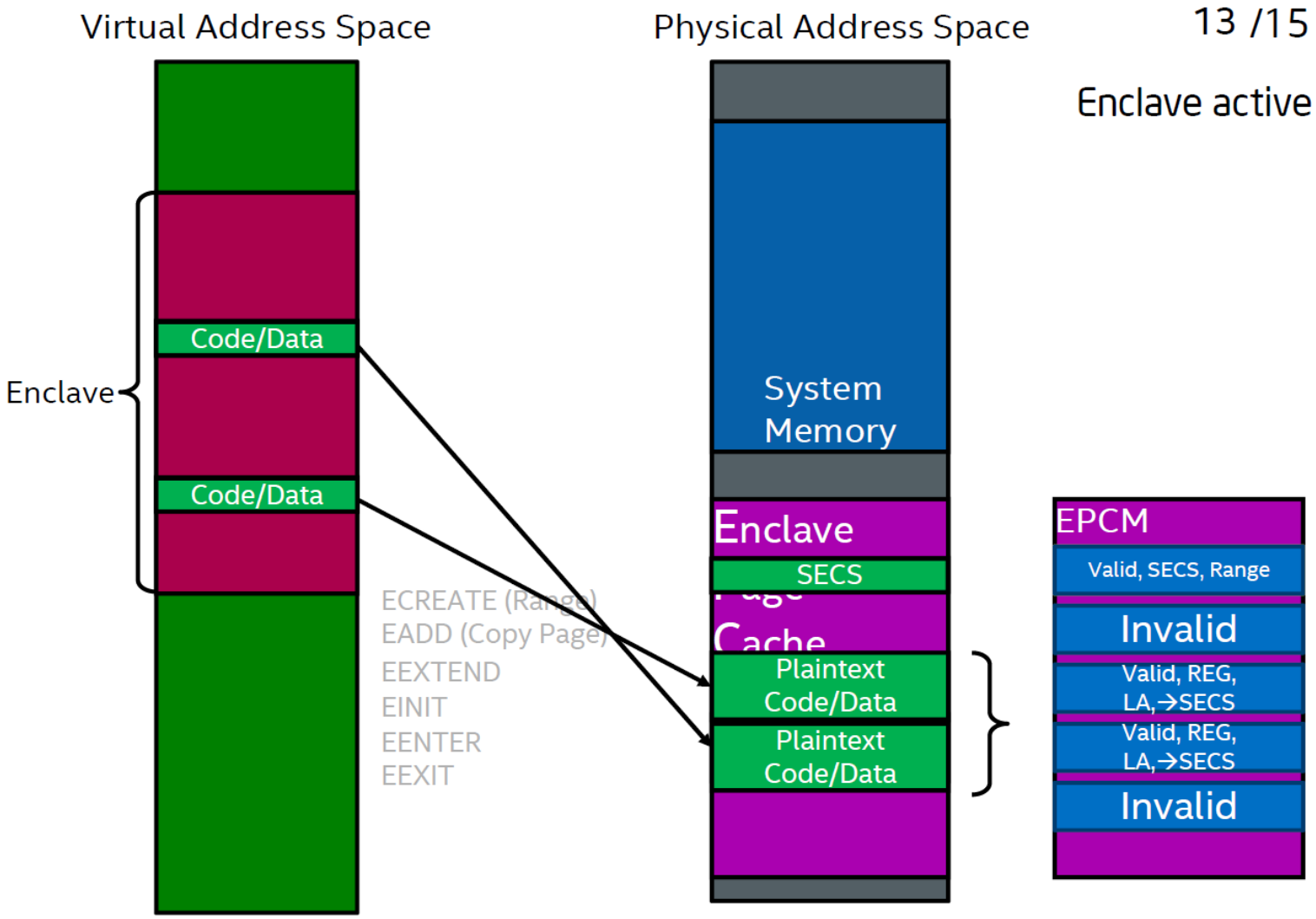
SGX Enclave Life Cycle Example



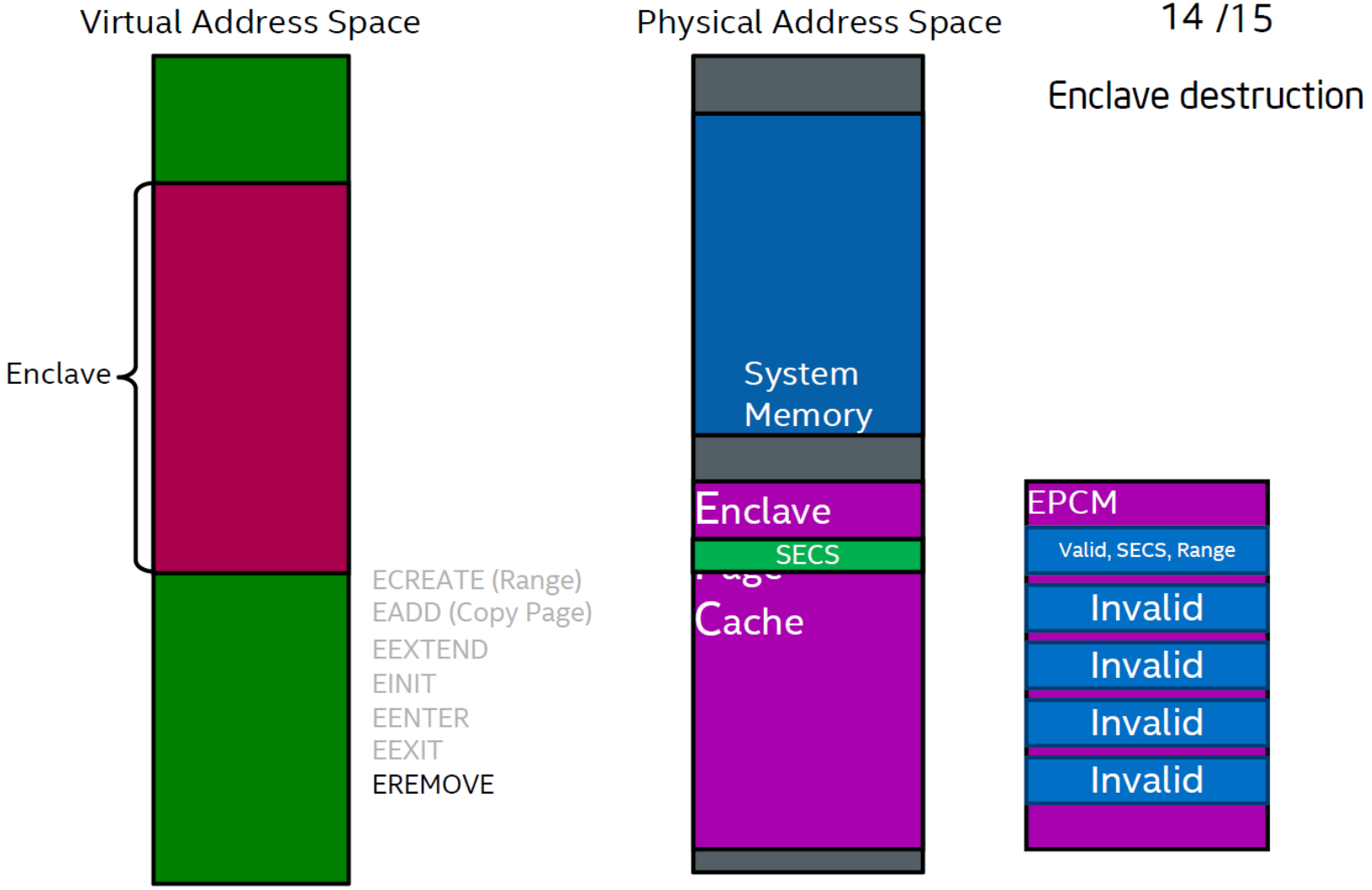
SGX Enclave Life Cycle Example



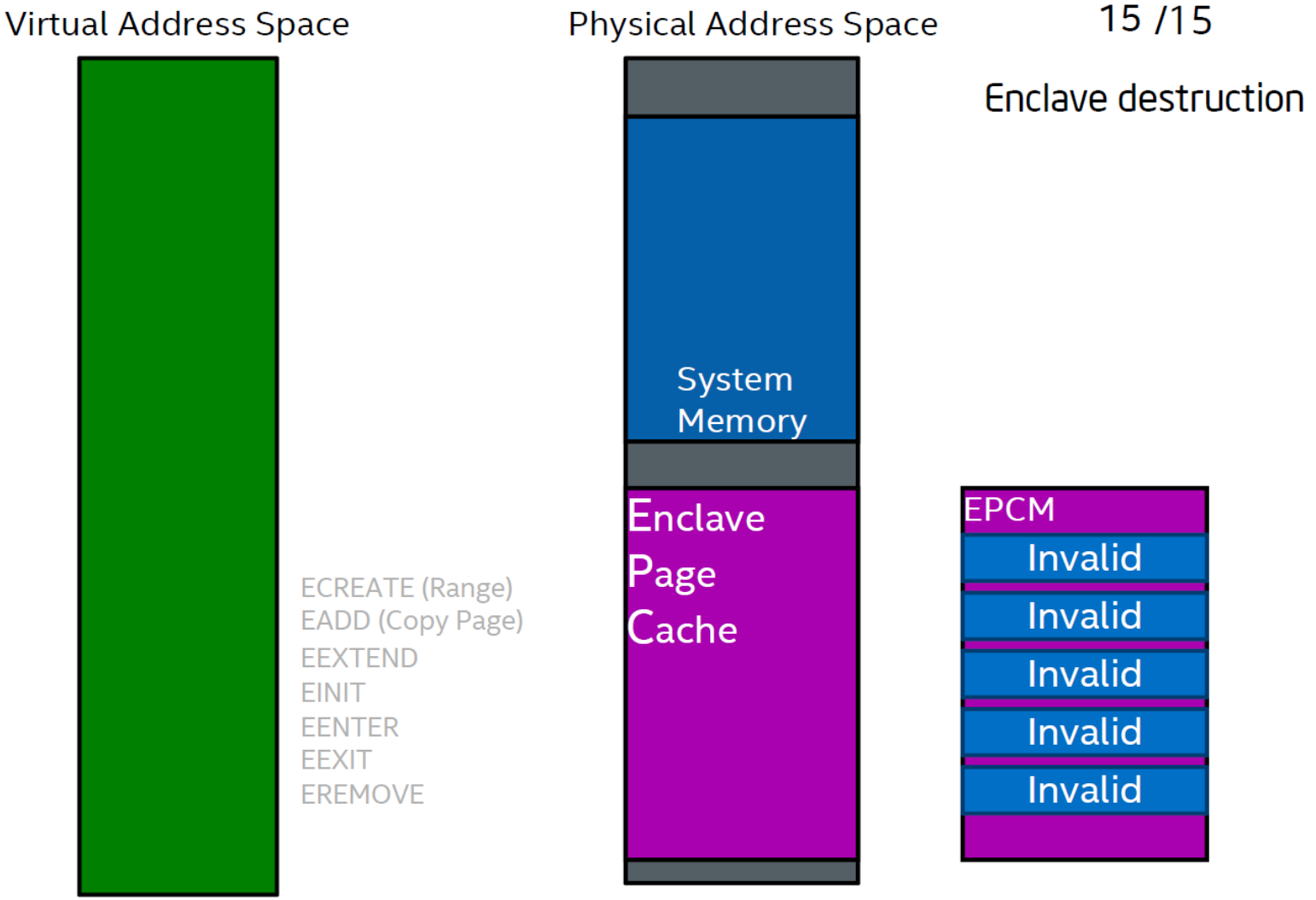
SGX Enclave Life Cycle Example



SGX Enclave Life Cycle Example



SGX Enclave Life Cycle Example



EPC Page Swapping

- High Level Overview
- Paging Instructions
- Examples

EPC Page Swapping

- EPC memory is set by BIOS and limited from size perspective
- We need a way to remove an EPC page, place into unprotected memory, and restore it later.
- Page must maintain same security properties (confidentiality, anti-replay, and integrity) when restored
- EPC paging instructions provide ability to encrypt page and produce meta data needed to meet requirements

EPC Page Swapping

- An enclave page must be evicted only after all cached translations to that page have been evicted from all logical processors.
- Content is swapped on 4KByte page basis
- Each 4KByte EPC page produces
  - 4KByte of encrypted content
  - 128Byte of meta-data (PCMD).

Paging Operations at a High Level

When a page is evicted from EPC

- It is assigned a unique version number which is recorded in a new type of EPC page called Version Array (VA)
- Encrypted page, metadata, and EPCM information are written out to system memory

When page is reloaded

- The processor decrypts, and integrity checks the page, using crypto metadata
- The processor verifies that version is the same version that was last written out

EPC Paging Instructions

EPA

- Allocates a 4KByte page in EPC for holding an array of page versions (VA) for anti-replay protection
  - VA contains versions of paged out enclave pages, size of each version slot is 64 bits.

EBLOCK

- Blocks a page from being accessed in preparation for swapping it out
  - Any future accesses by owner enclave to BLOCKED page result in #PF
  - Returns indication that page previously blocked

ETRACK

- Sets a tracking mechanism to verify that all TLB entries for the blocked page has been flushed

EPC Paging Instructions

EWB

- Securely evicts a 4KByte page from the EPC along with it's page information
  - Assigning a unique version value for the page and storing it in the VA page.
  - Encrypt EPC page, create MAC over the encrypted page, version counter, and meta data. And write it out to external memory
- Enclave page must be first prepared for eviction:
  - I.e. Blocked and no TLB entry refer to that page.

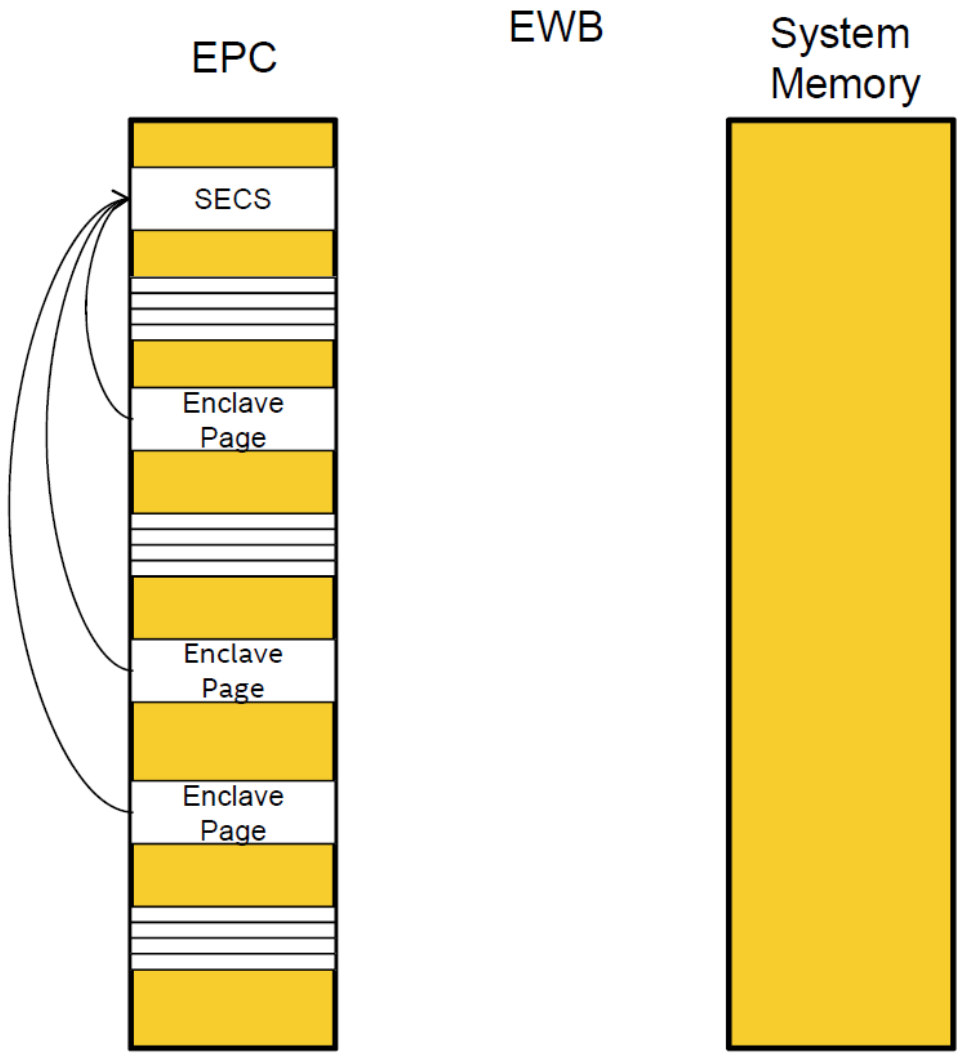
EPC Paging Instructions

ELDU/B

- Securely loads a page back from memory into the EPC into an unblocked or blocked state
  - Verify the MAC on the meta data, version counter from specific VA entry, and encrypted enclave page content
  - If verification succeed, decrypt the enclave page content into EPC page allocated by system memory and clear the VA entry.

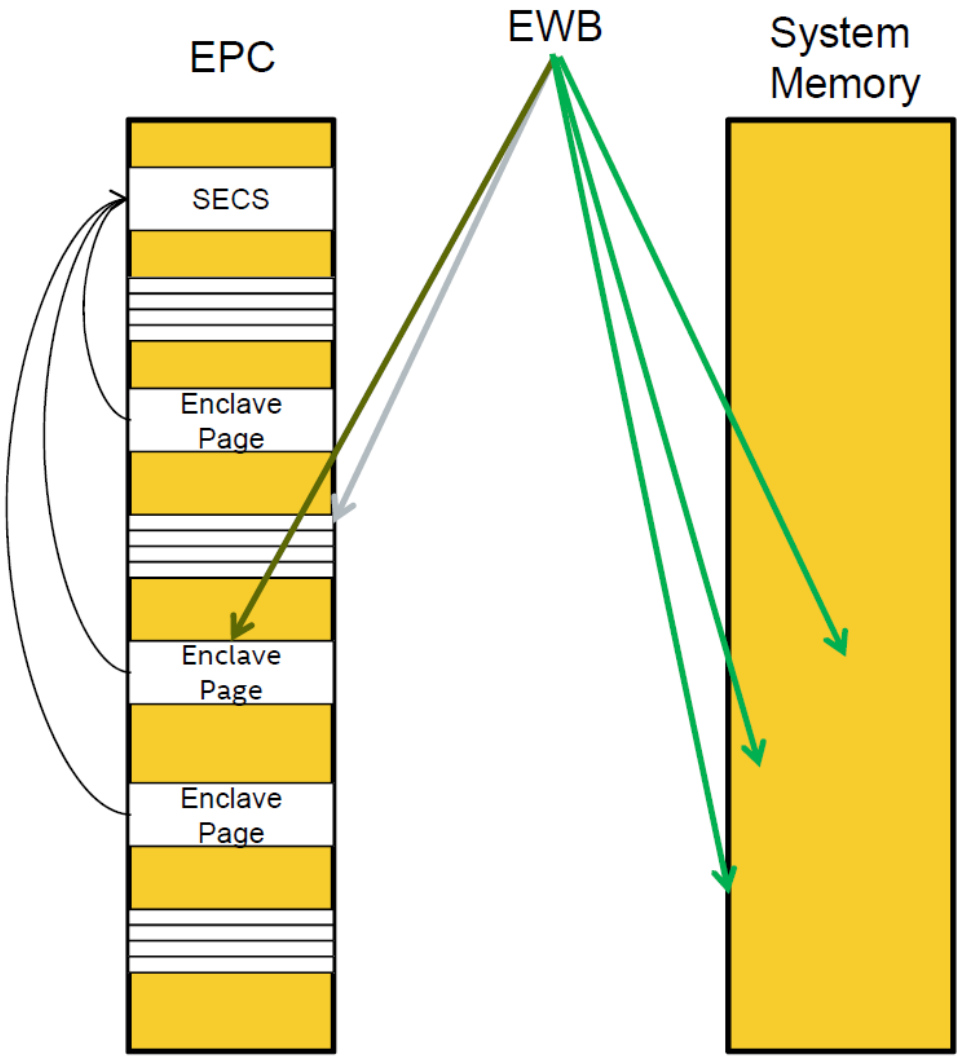
Page-out Example

- Instruction EWB writes back a page from EPC to system memory
  - Assume the page is ready (Blocked, no TLB entries)



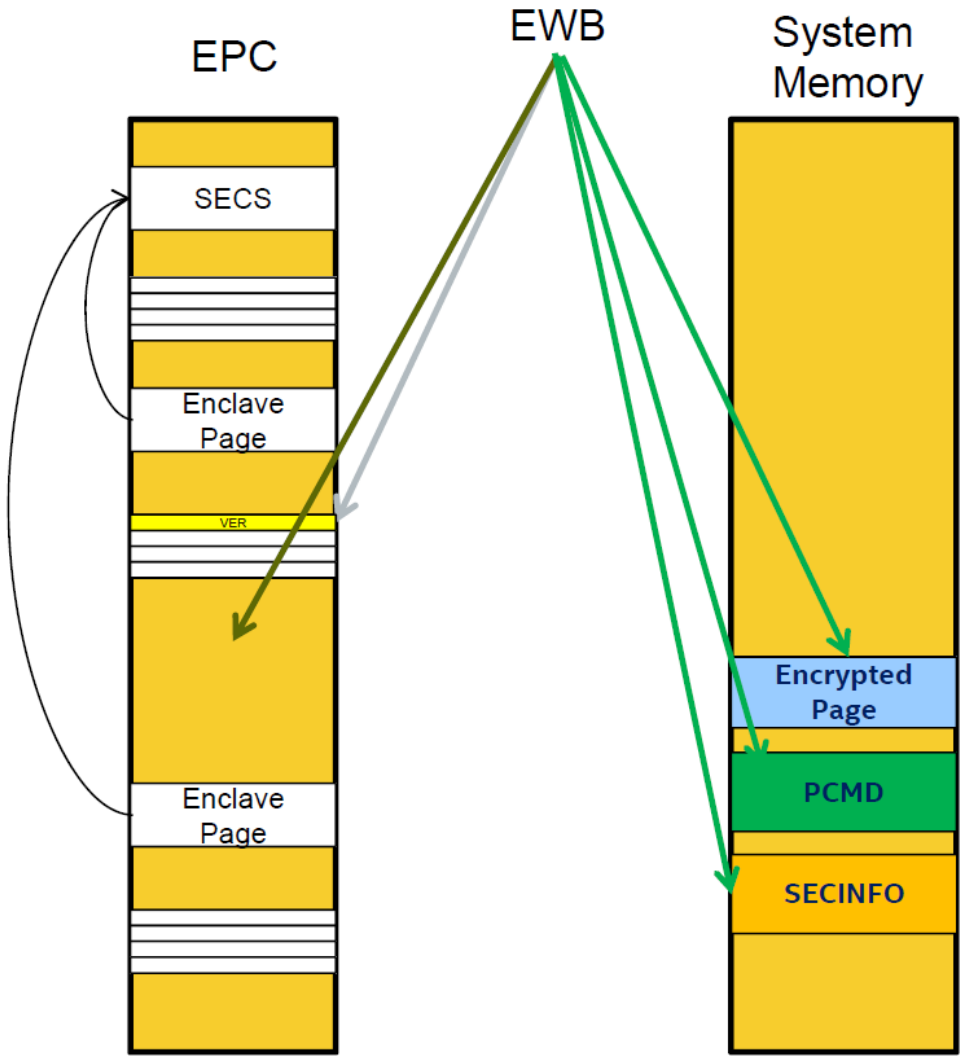
Page-out Example

- Instruction EWB writes back a page from EPC to system memory
  - Assume the page is ready (Blocked, no TLB entries)
- EWB Parameters:
  - Pointer to EPC page that needs to be paged out
  - Pointer to empty version slot
  - Pointers outside EPC location



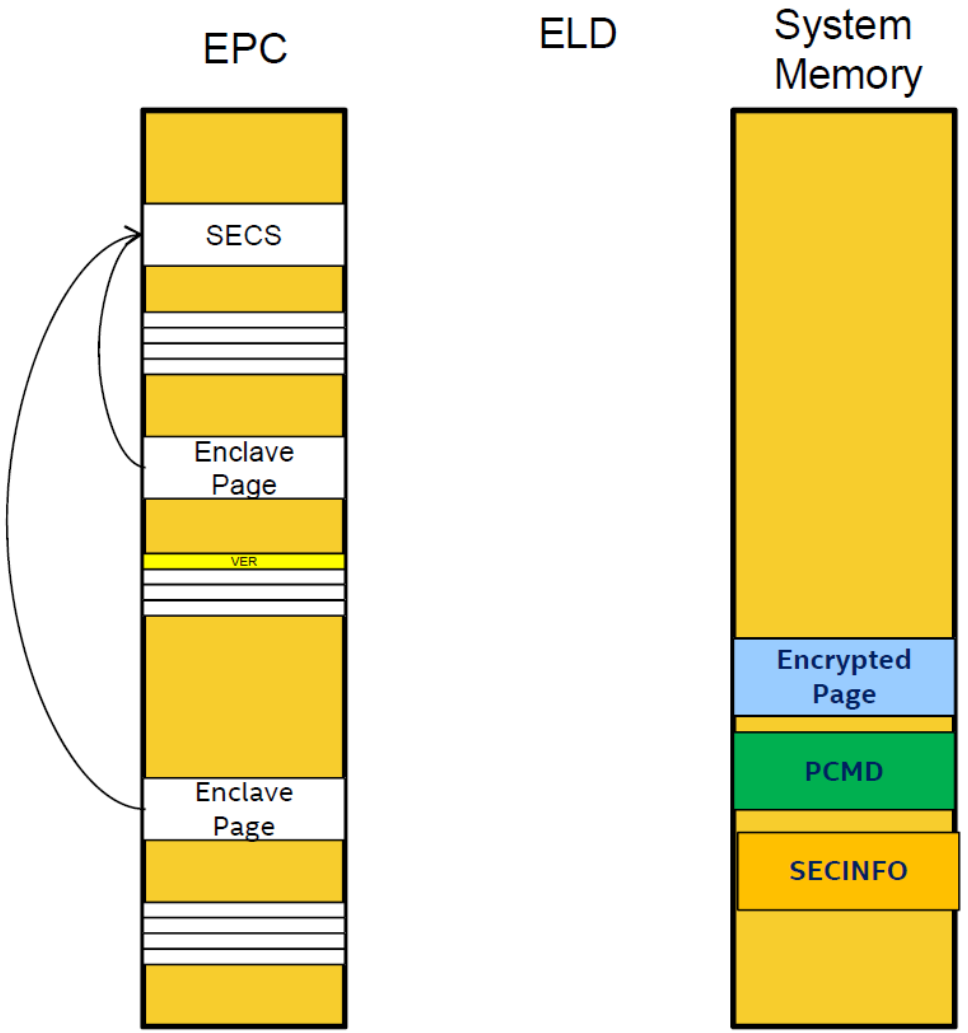
Page-out Example

- Instruction EWB writes back a page from EPC to system memory
  - Assume the page is ready (Blocked, no TLB entries)
- EWB Parameters:
  - Pointer to EPC page that needs to be paged out
  - Pointer to empty version slot
  - Pointers outside EPC location
- EWB Operation
  - Remove page from the EPC
  - Populate version slot
  - Write encrypted version to outside
- All pages, including SECS and Version Array can be paged out



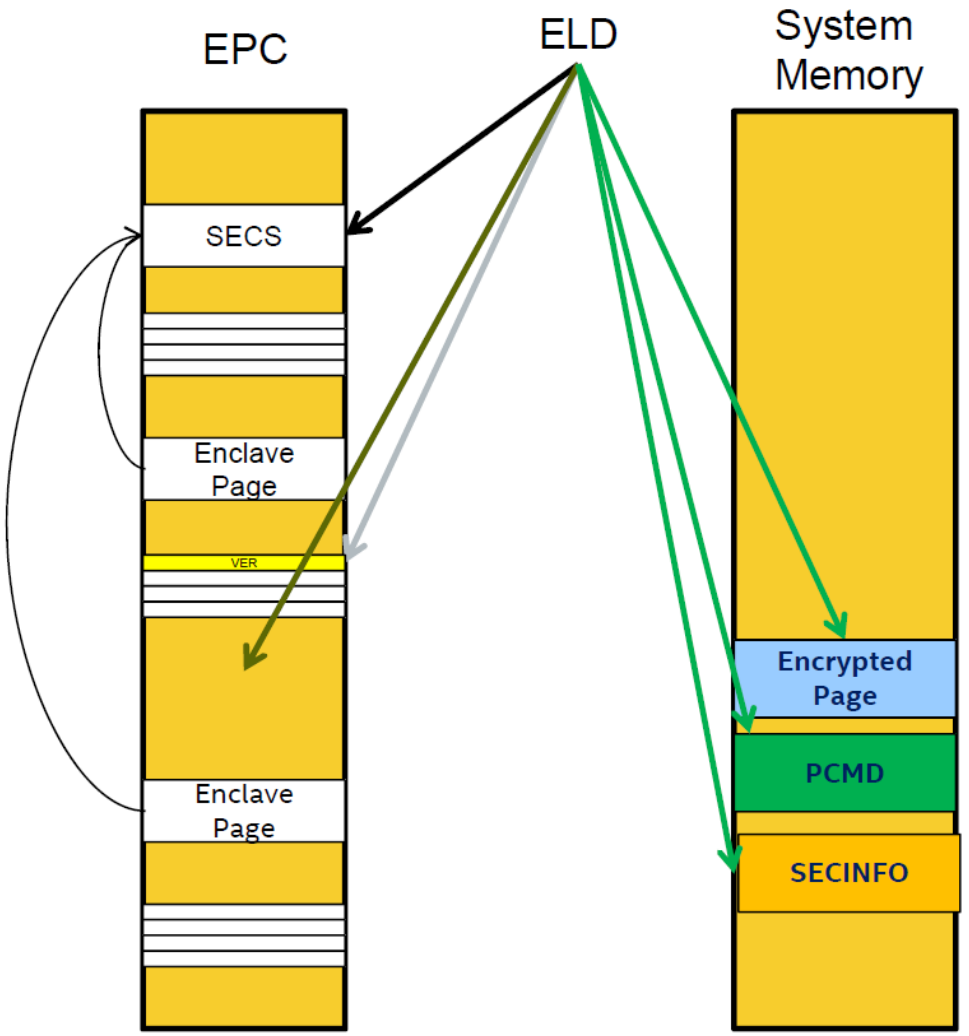
Page-in Example

- Instruction ELD loads a page from system memory into EPC



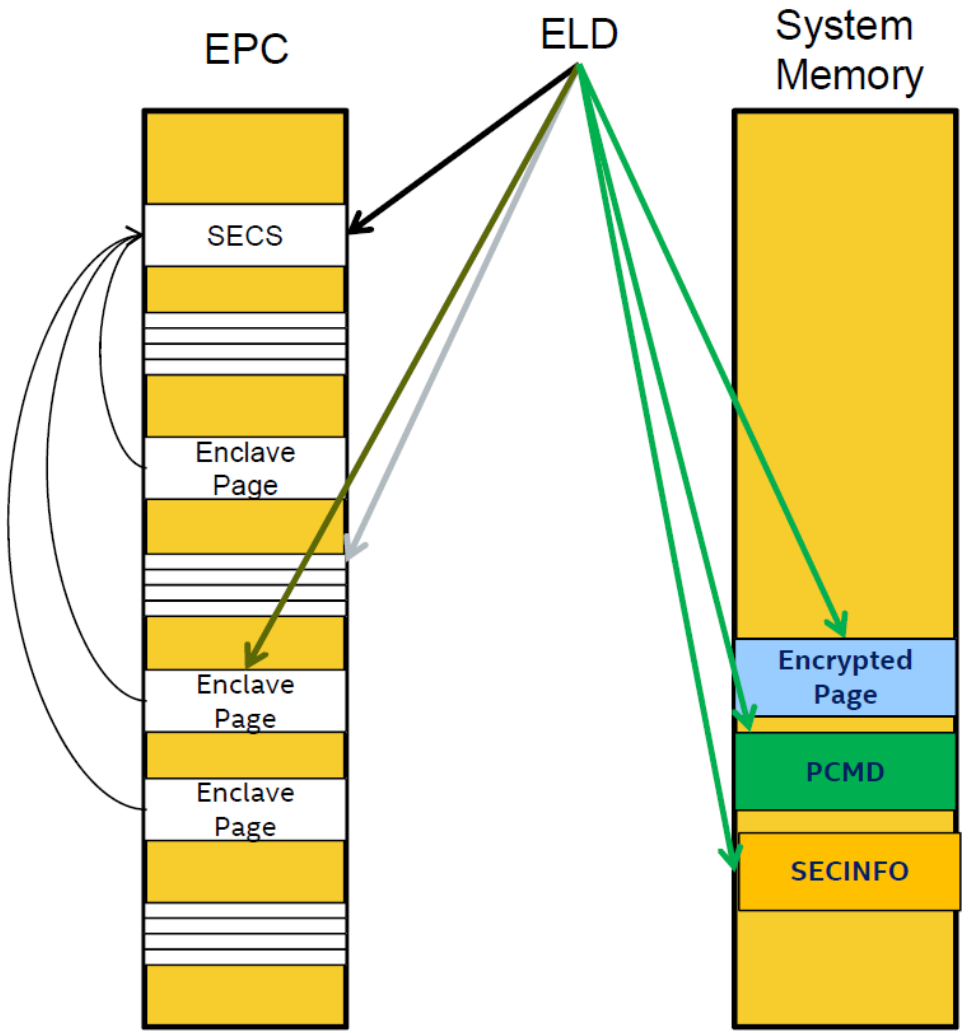
Page-in Example

- Instruction ELD loads a page from system memory into EPC
- ELD Parameters:
  - Encrypted page
  - Free EPC page
  - SECS (for an enclave page)
  - Populated version slot



Page-in Example

- Instruction ELD loads a page from system memory into EPC
- ELD Parameters:
  - Encrypted page
  - Free EPC page
  - SECS (for an enclave page)
  - Populated version slot
- ELD Operation
  - Verify and decrypt the page using version
  - Populate the EPC slot
  - Free-up version slot

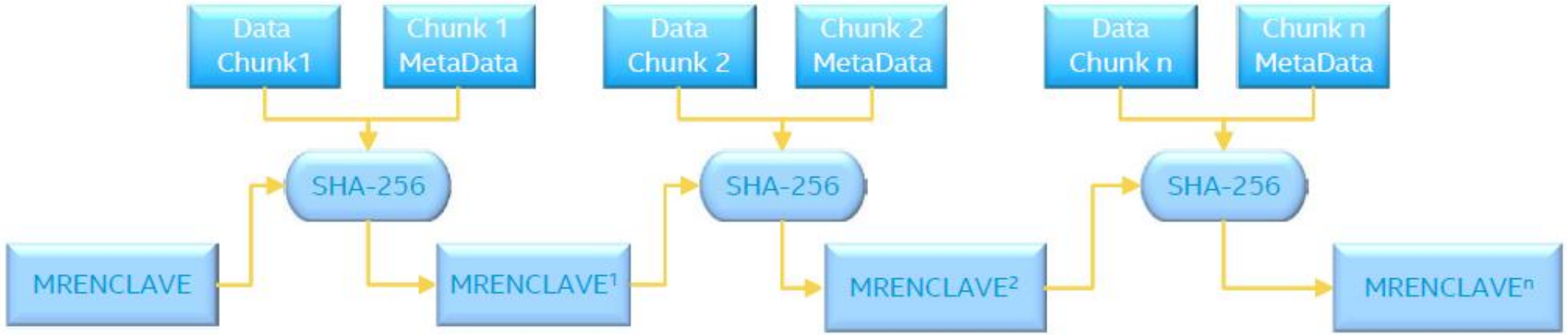


SGX Software Attestation

- Measurement
- Local Attestation
- Remote Attestation

SGX Enclave Measurement (MRENCLAVE)

- When building an enclave, SGX generates a cryptographic log of all the build activities
  - Content: Code, Data, Stack, Heap
  - Location of each page within the enclave
  - Security flags being used
- MRENCLAVE ("Enclave Identity") is a 256-bit digest of the log
  - Represents the enclave's software TCB



Attestation

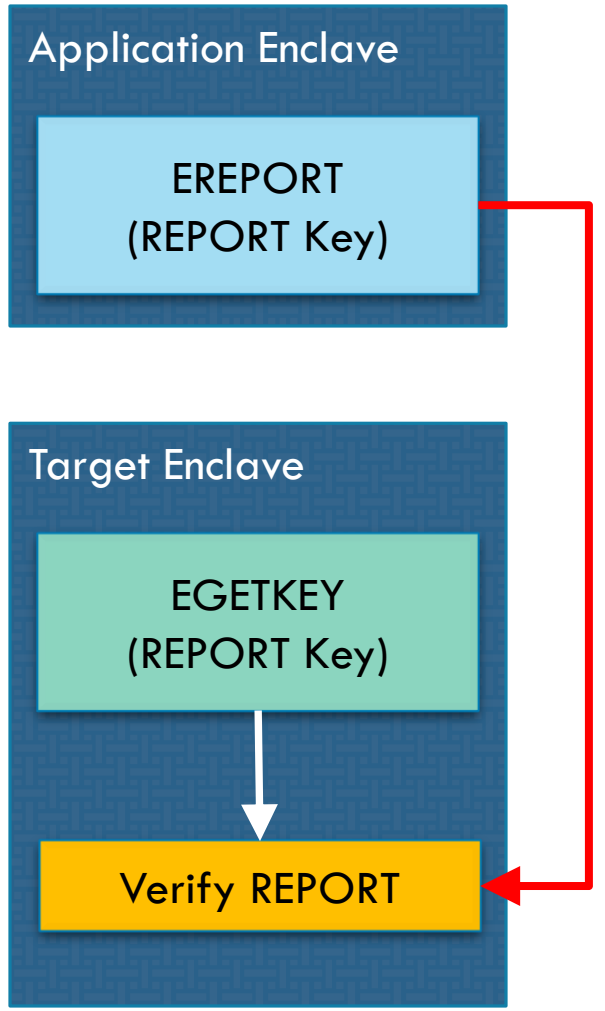
SGX provides LOCAL and REMOTE attestation capabilities

- ***Local attestation*** allows one enclave to attest its Thread Control Block (TCB) (i.e., the execution environment) to another enclave on the same platform
- ***Remote attestation*** allows one enclave to attest its TCB to another entity outside of the platform

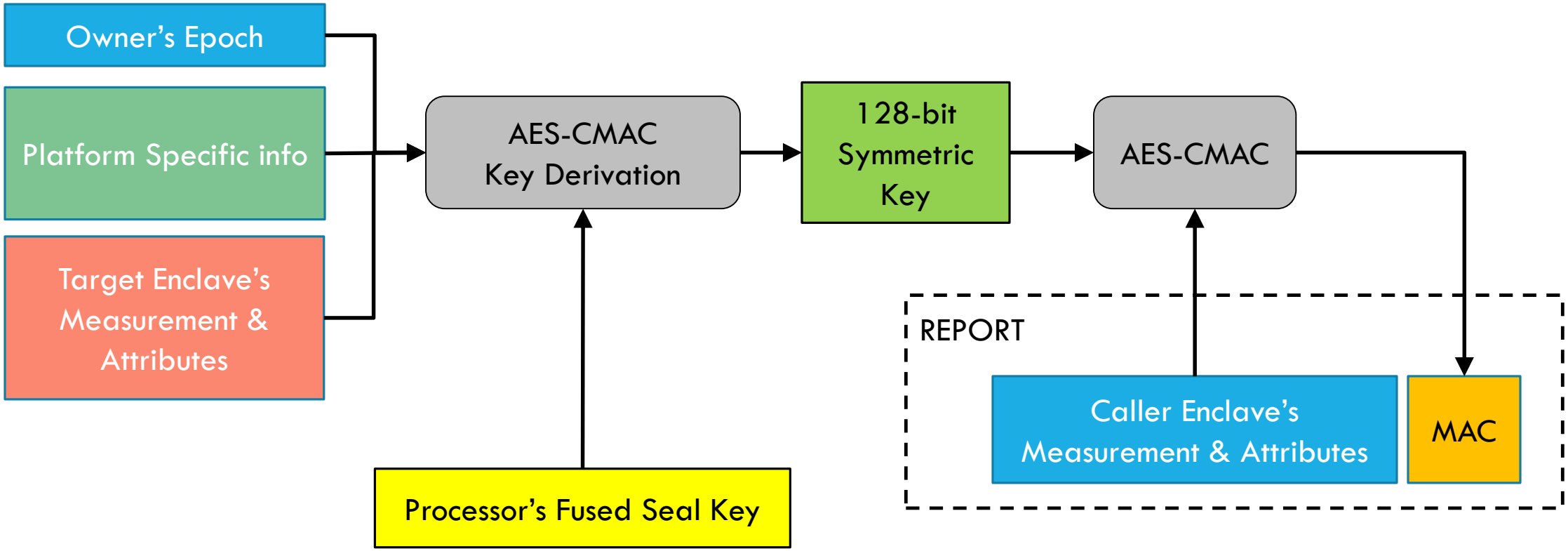
Local Attestation Overview

An application enclave proves its identity to another target enclave via the EREPORT instruction

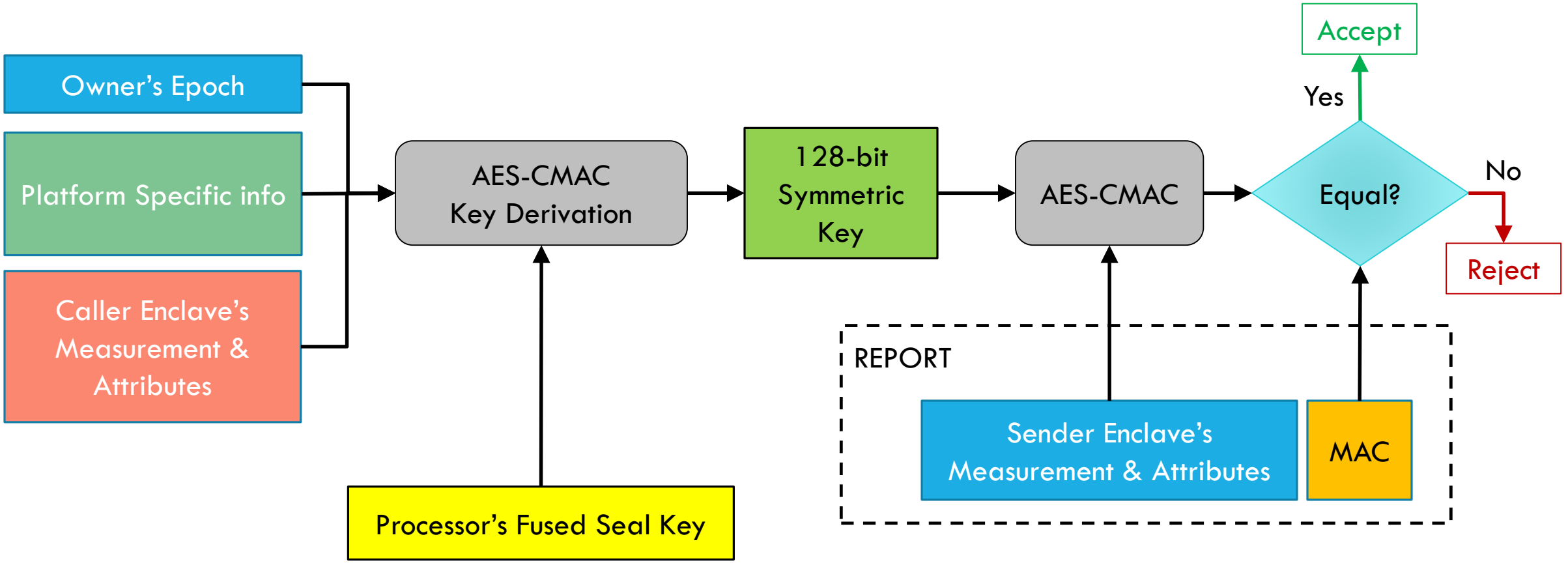
- Application Enclave calls EREPORT instruction to generate REPORT structure for a desired target enclave
  - REPORT contains calling enclave's Attributes, Measurements and User supplied data
- REPORT structure is secured using the REPORT key of the target enclave
- EGETKEY is used by the target enclave to retrieve REPORT key
- Target enclave then verifies the REPORT structure using software.



REPORT Generation (EREPORT)

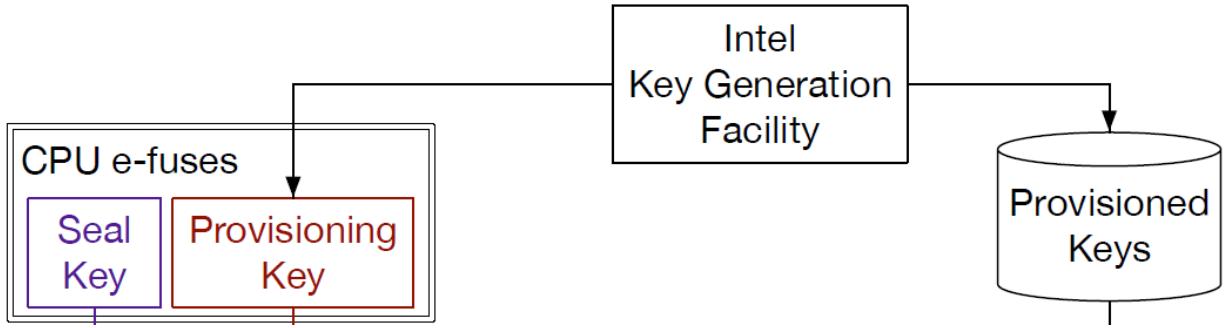


REPORT Verification (EGETKEY)



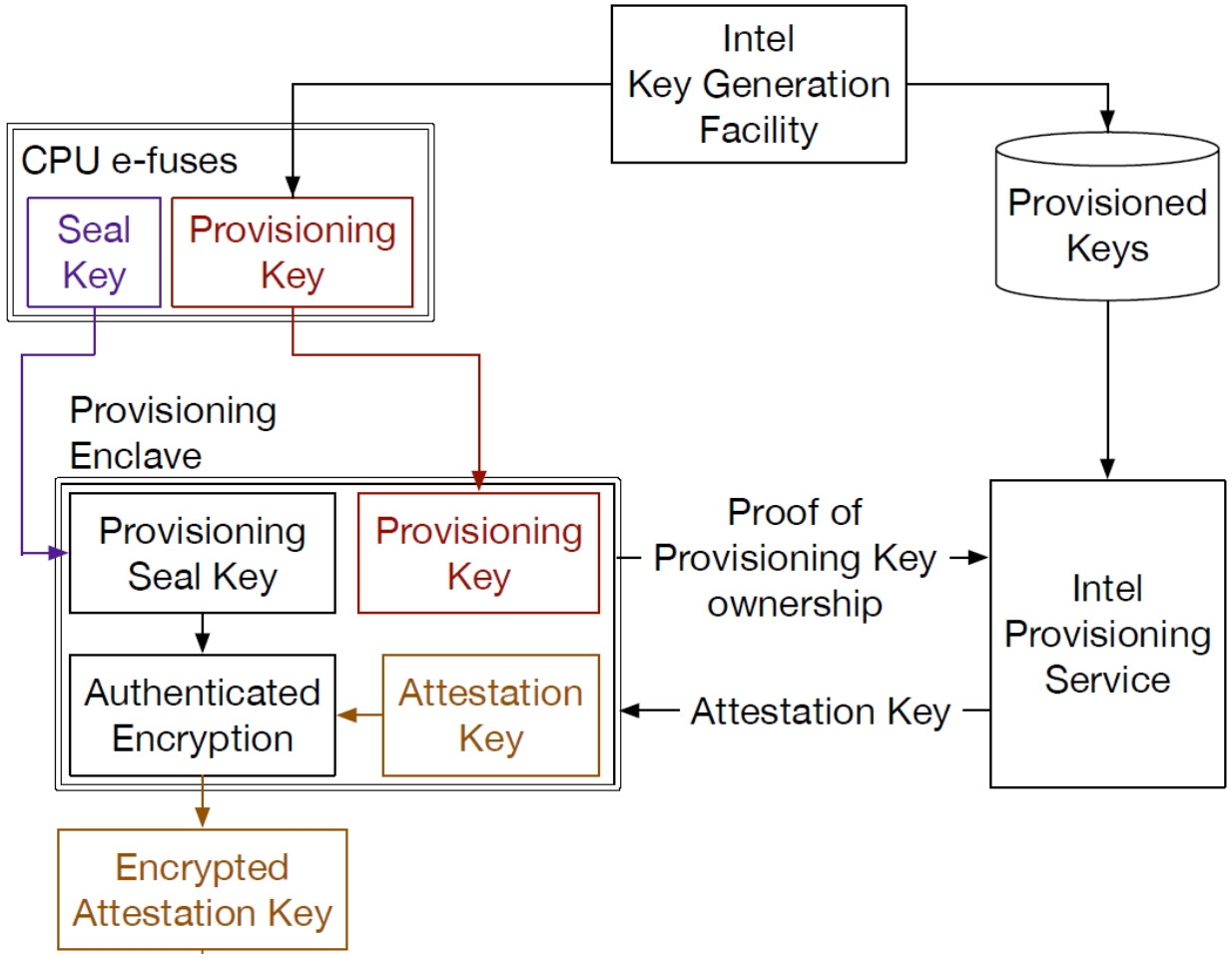
Remote Attestation

- During manufacturing, two keys are burned into the CPU
  - Fused Seal Key
  - Provisioning Key
- Fused Seal Key is used as Processor's secret
- Provisioning Key serve as a proof for remote Platform



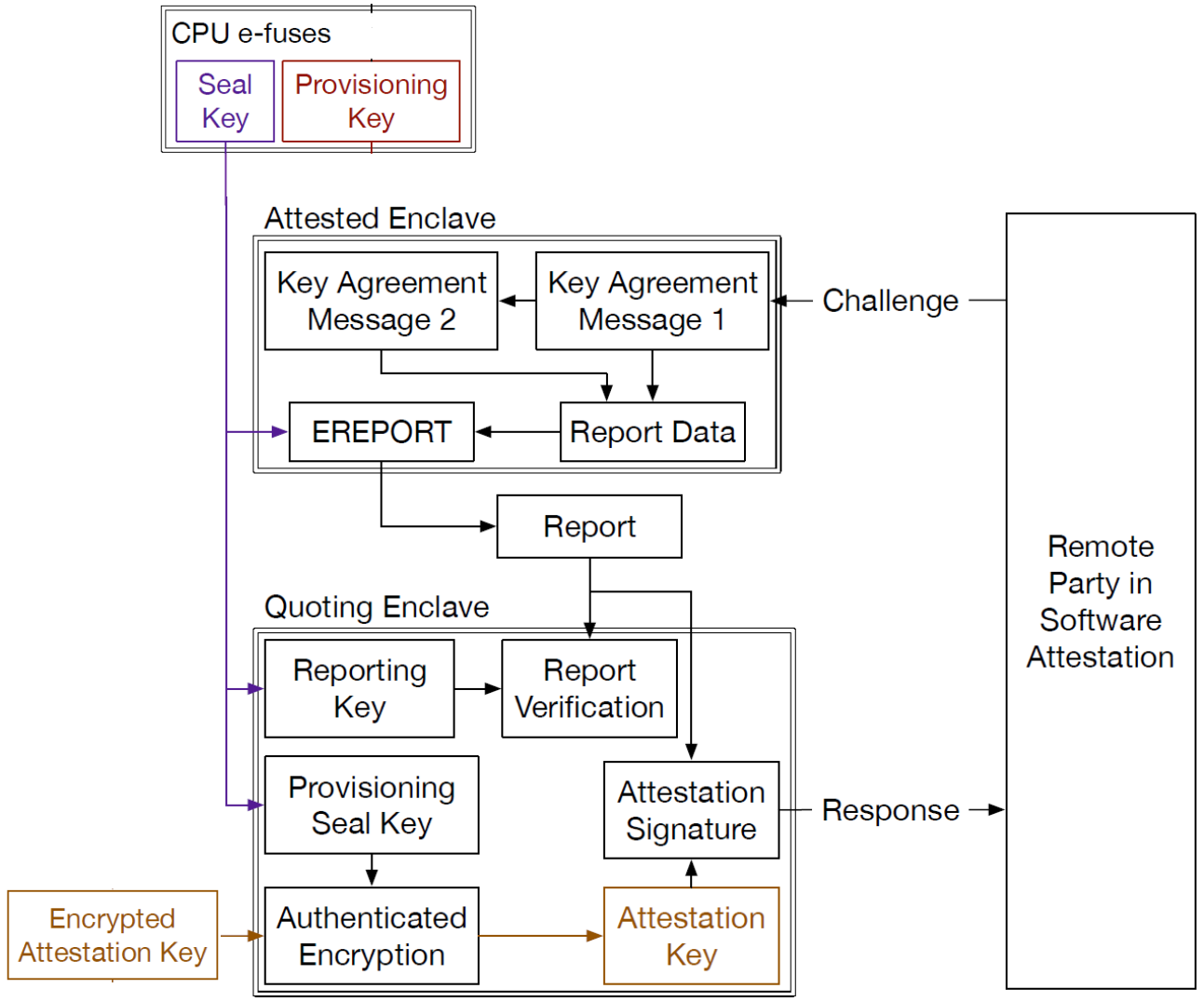
Remote Attestation

- Provisioning Key serves as a proof for remote Platform
- Remote platform issues an Attestation Key which is encrypted and stored for future use.



Remote Attestation

- A **Quoting Enclave** first performs Local Verification of application's Enclave
- Upon successful Local Attestation, Quoting Enclave decrypts the Attestation Key and signs the REPORT with this key
- The remote party verifies the signature




Thank You!