USC Viterbi
School of Engineering

SOFTWARE GUARD EXTENSIONS

(SGX)

BY ARSHIA SHAIK & KEVIN SHAH

12th October, 2022

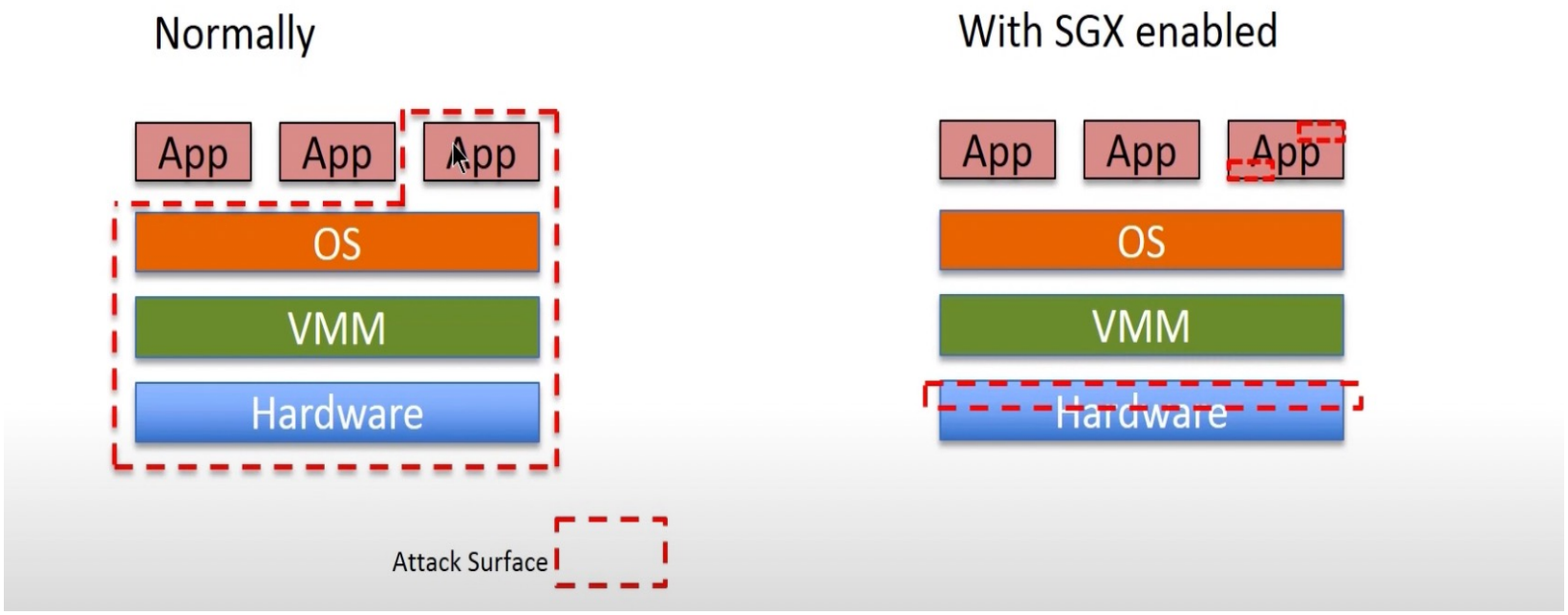
What is Intel Software Guard Extensions (SGX)?

- Intel Software Guard Extensions (SGX) is a security instruction set that are built into some Intel CPUs

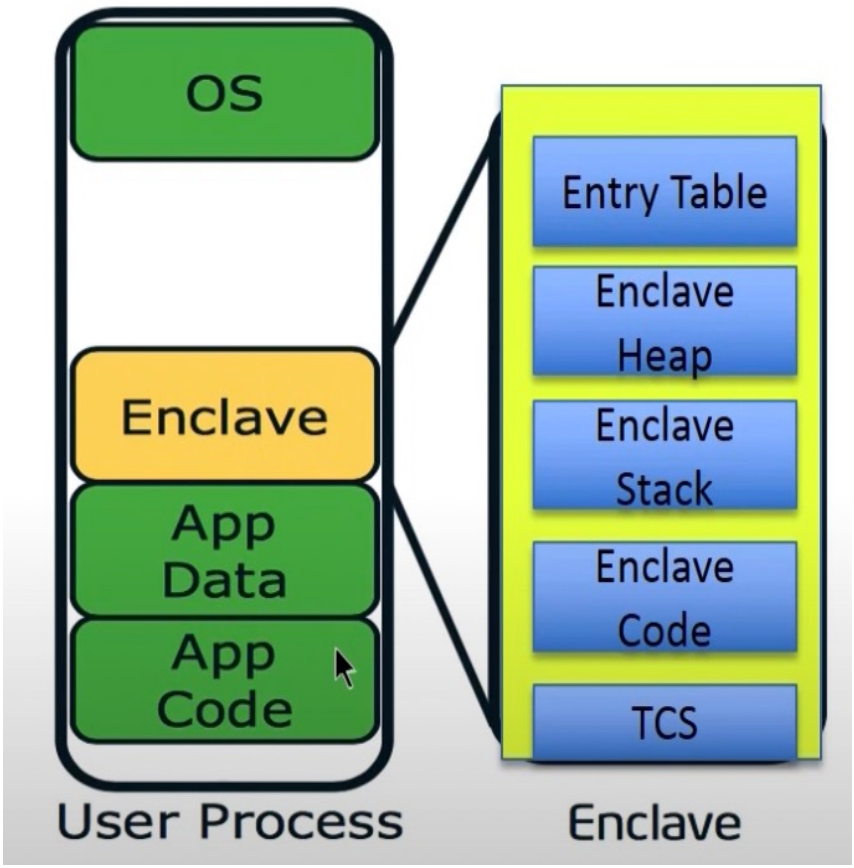
- SGX gives developers the ability to split a computer's memory into what are called enclaves, which are private, predefined areas in memory that can better protect users' sensitive information

- SGX is designed to be useful for implementing secure remote computation, secure web browsing, and digital rights management (DRM)

Reduced Attack Surface with SGX



Enclaves

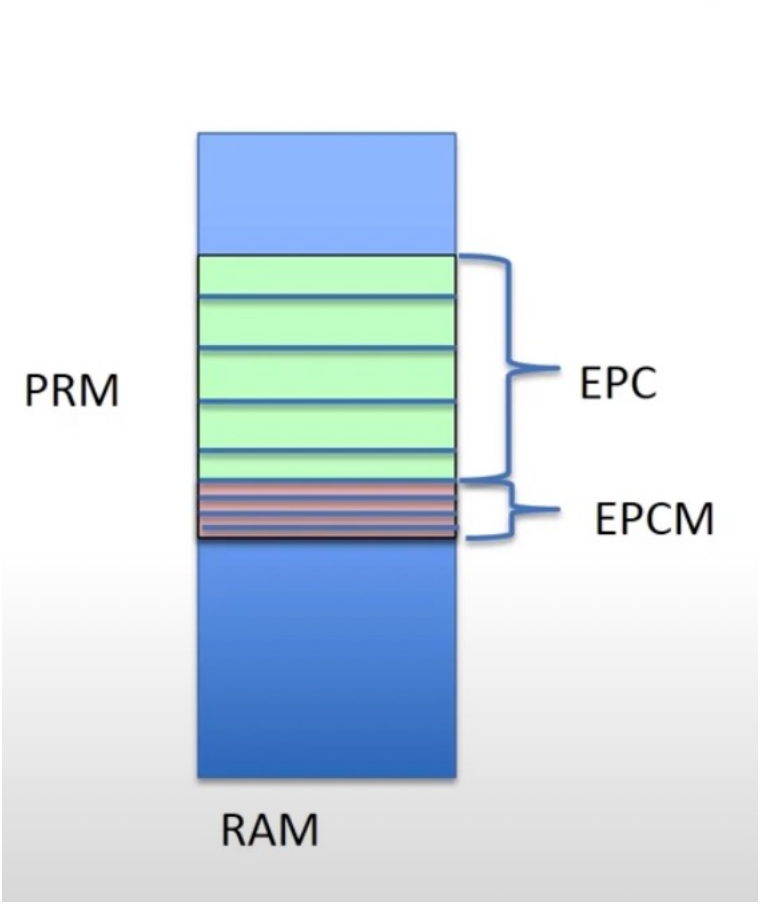


- Enclave has its own code and data area that provides confidentiality and integrity with controlled entry points
- Enclave code and data cannot be accessed from outside the enclave, not even by the operating system
- TCS: Thread control Structure SGX supports multi-threading; one TCS for each thread supported)

Enclave Properties

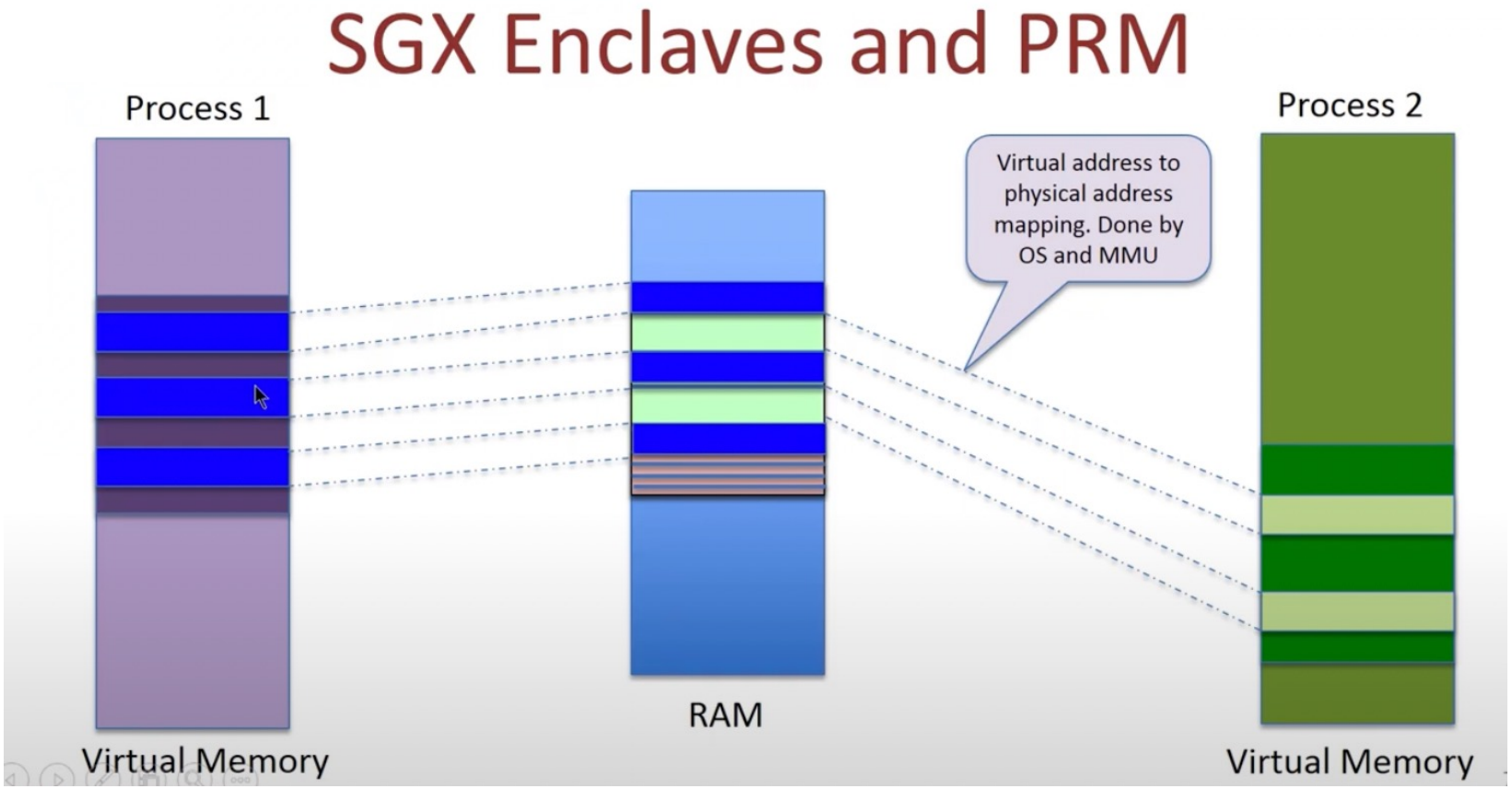
- Achieves confidentiality and integrity: Tampering of code/data is detected and unauthorized access to protected code/data is prevented
- Code outside enclave cannot access code/data inside the enclave
- Even though OS is untrusted, it should still be able to manage page translation and page tables of the enclave
- Enclave code and data
  - Enclave code and data is in the clear when in the CPU package (e.g., Registers/caches), but unauthorized access is prevented
  - Enclave code and data is automatically encrypted when it leaves the CPU package

Physical Memory

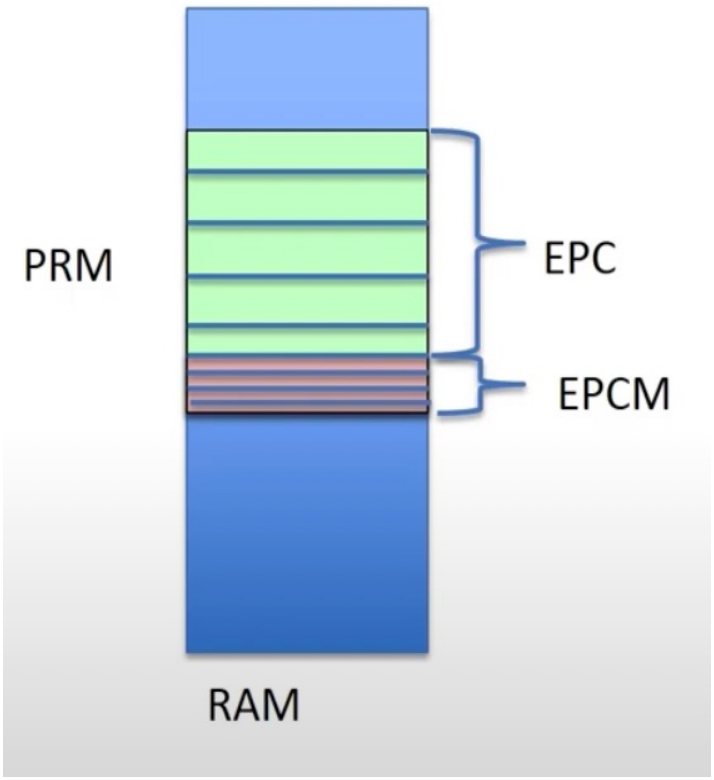


- PRM-processor related memory is allocated by the BIOS. Access to PRM is blocked by external agents such as DMA, graphics engine, etc.
- To the other devices, this range is treated as non existent memory
- EPC Pages: Enclave page cache holds enclaves from any application.
  - Divided into 4KB pages
  - If an EPC page is valid, it either contains an SGX enclave page or EPCM (EPC micro-architecture structure)

SGX Enclaves and PRM



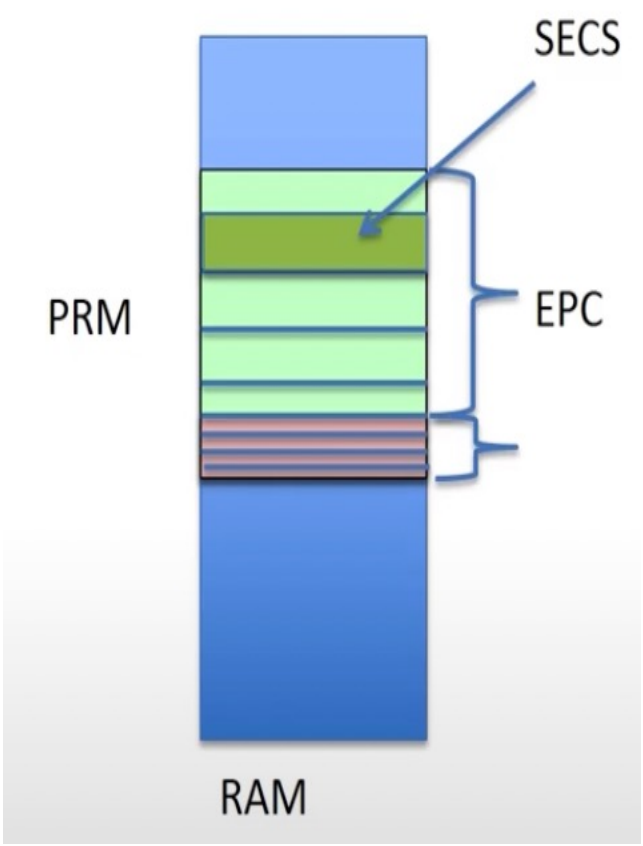
Physical Memory



EPCM: Enclave page cache map

- One for each EPC, used by hardware for access control
- Stores management related aspects for the corresponding EPC
  - Aspects such as valid/invalid; r/w/x permissions
  - Type of page
  - Virtual address range through which, the EPC can be accessed
  - Additional layer of security compared to legacy paging and segmentation since we do not trust the OS

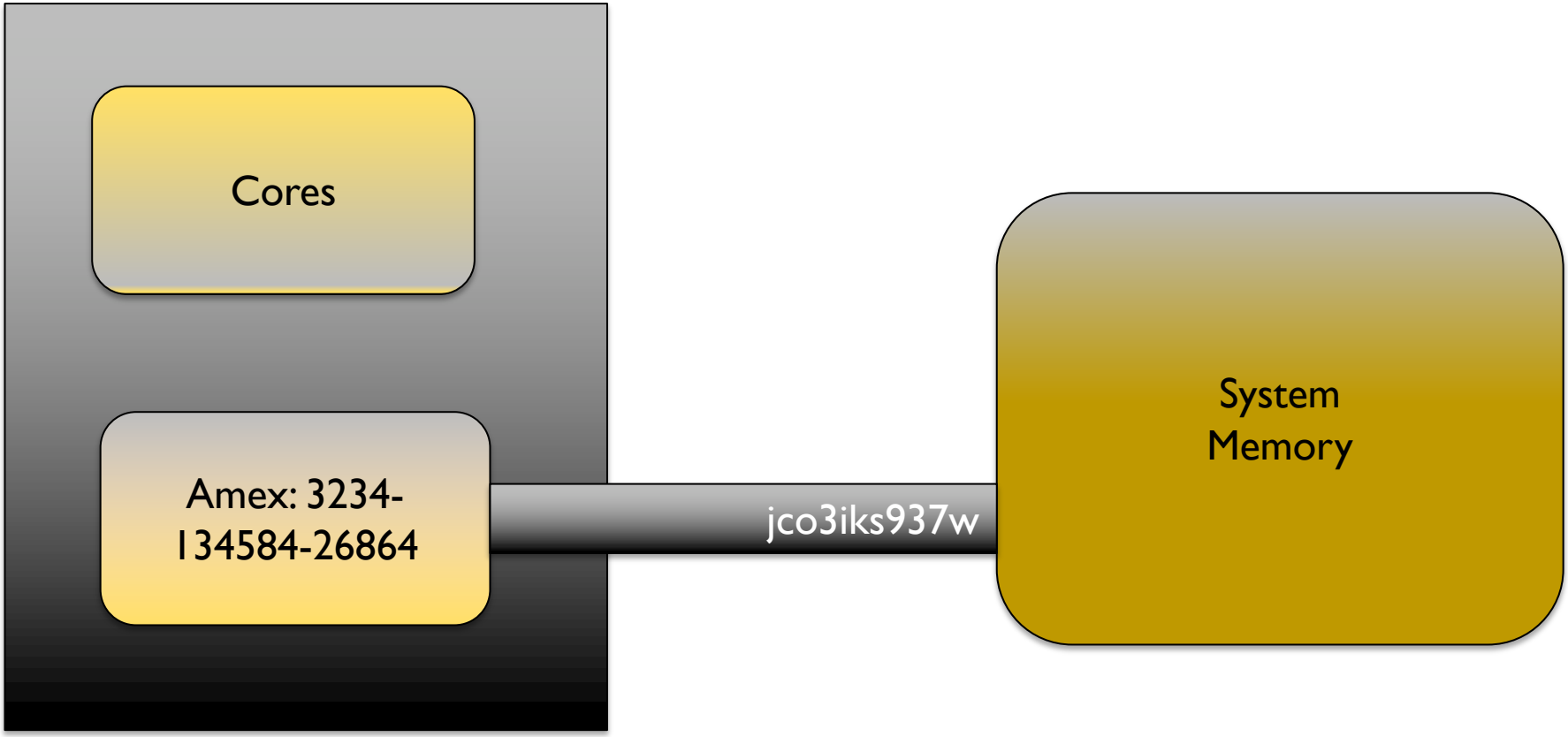
Physical Memory



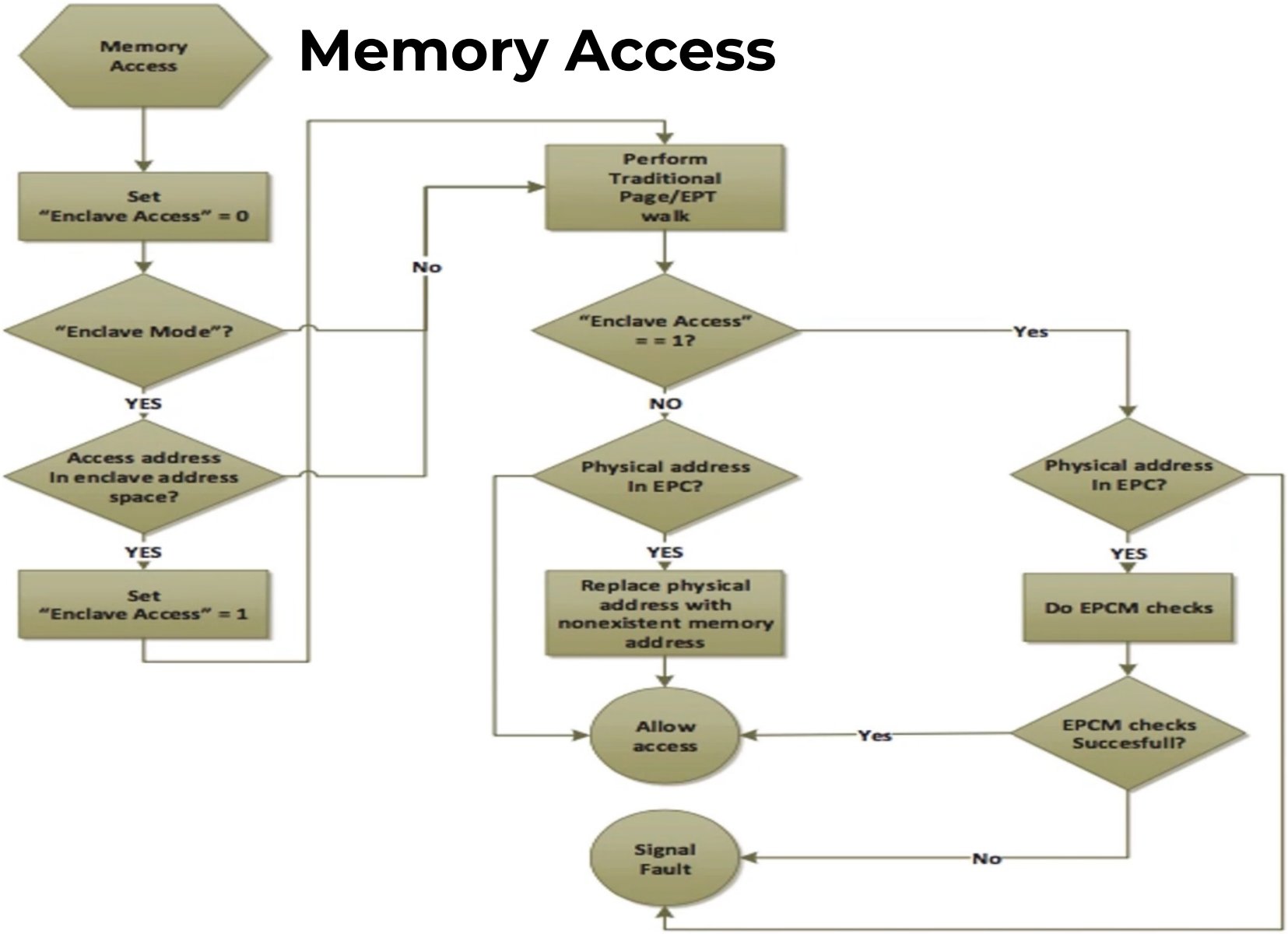
SECS: SGX Enclave Control Store

- One for each enclave
- 4KB(present in an EPC)
- Contains global metadata about the enclave
  - EPC pages that are used
    - Mapping information
    - Crypto log of each used EPC page
- Range of protected addresses used by the enclave
  - 32/64-bit operating mode
  - Debug access

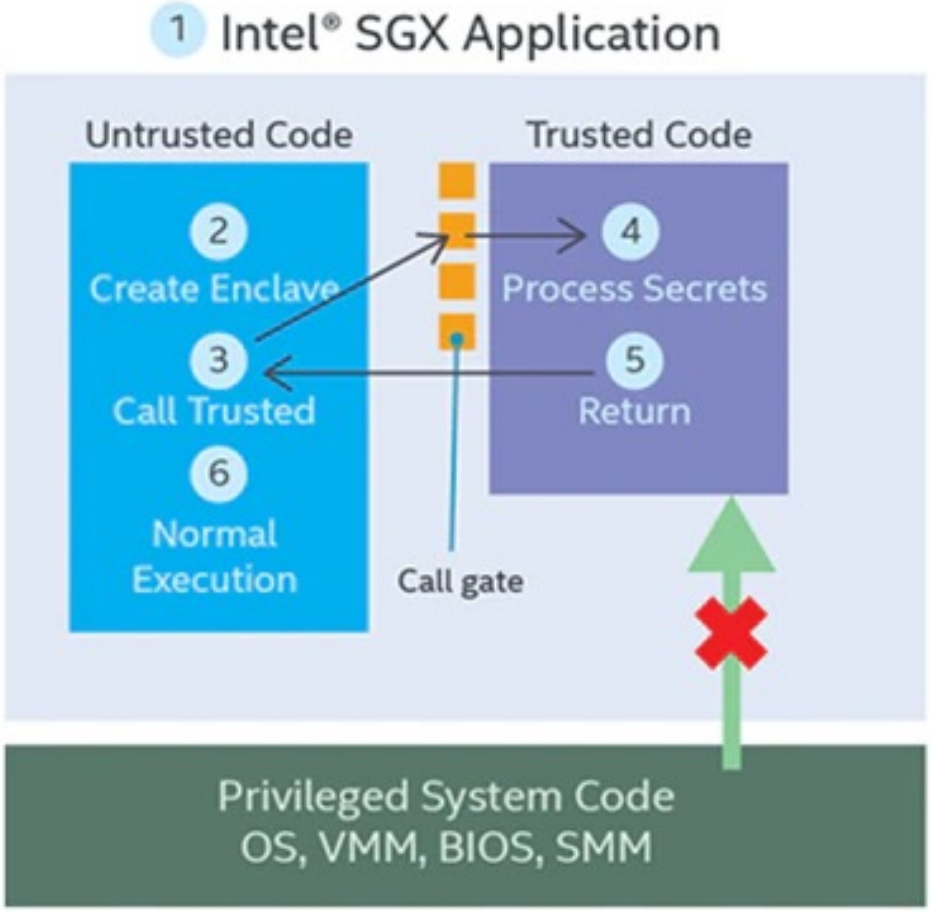
EPC Encryption



Memory Access



Application Execution Flow



1. App is built with trusted and untrusted parts
2. App runs and creates the enclave, which is placed in trusted memory
3. Trusted function is called, and execution is transitioned to the enclave
4. Enclave sees all process data in the clear; external access to the enclave data is denied
5. Function returns; enclave data remains in trusted memory
6. Normal execution resumes

SGX Instruction Set

- SGX architecture includes 17 new instructions, new processor structures (that you saw earlier), and a new mode of execution

Enclave Build Instructions	
Instruction	Description
ECREATE	Declare base and range
EADD	Add 4k page
EEXTEND	Measure 256 bytes
EINIT	Declare enclave built
EREMOVE	Remove page

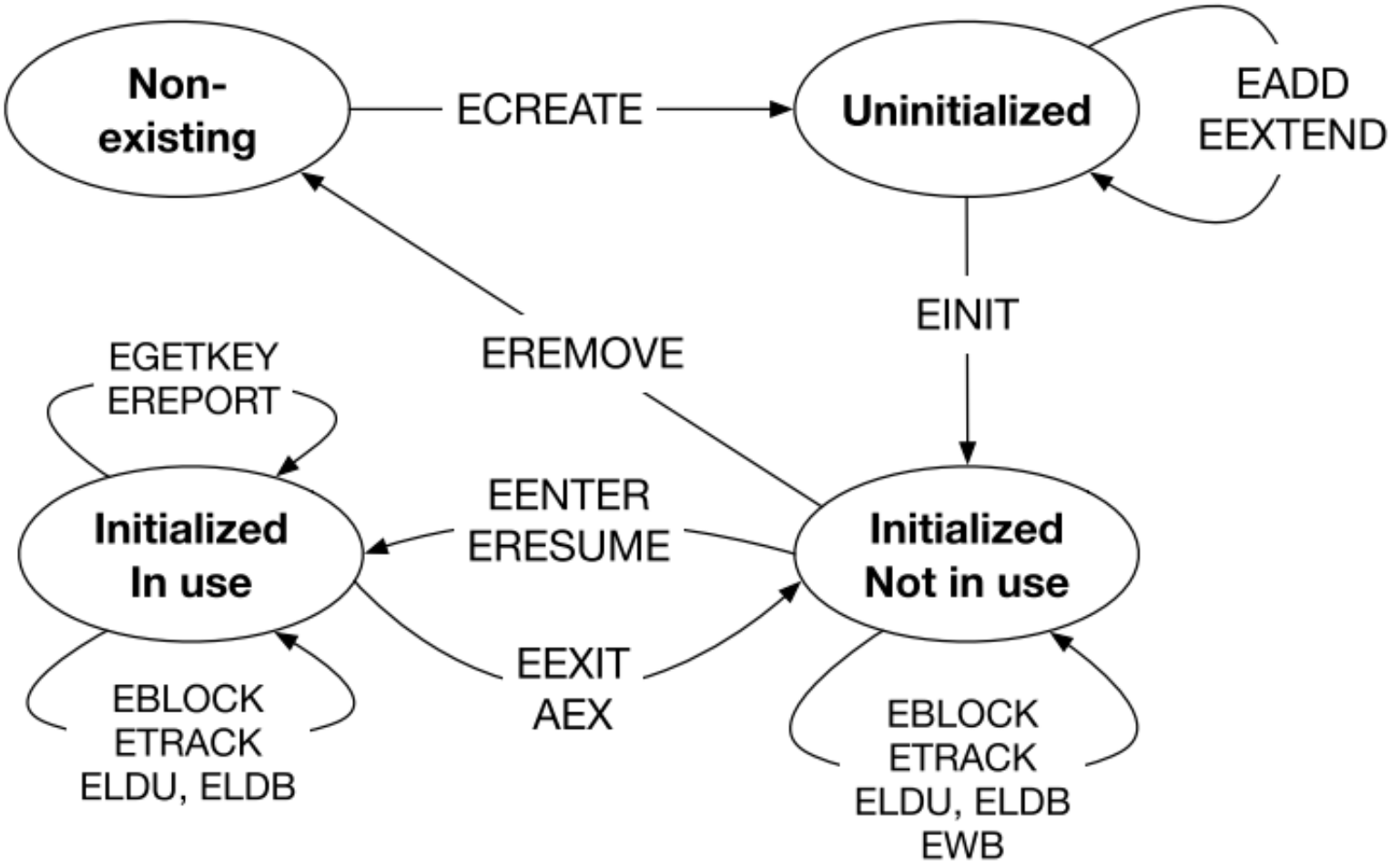
Enclave Paging Instructions	
Instruction	Description
EPA	Create version array page
ELDB/U	Load an evicted page into protected memory
EWB	Evict a protected page
EBLOCK	Prepare for eviction
ETRACK	Prepare for eviction

Enclave Security Instructions	
Instruction	Description
EREPORT	Enclave report
EGETEKY	Generate unique key

Enclave Debug Instructions	
Instruction	Description
EDBGRD	Read inside debug enclave
EDBGWR	Write inside debug enclave

Enclave Entry and Exit Operations	
Instruction	Description
EENTER	Enter enclave
ERESUME	Resume enclave
EEXIT	Leave enclave
AEX	Asynchronous enclave exit

Enclave Life Cycle



Memory Encryption Engine (MEE)

- Sits between CPU and Memory Controller Unit (Memory Management Unit)
- Encrypts traffic between processor and DRAM
- When cache line (EPC) is evicted, MEE encrypts and generates a MAC of the cache line and stores the encryption key and MAC
- During cache miss, MEE decrypts the line and verifies it integrity using the stored MAC before transferring it back to CPU
- MEE stores the encryption keys and MACs in a hash tree

Attestation

- Mechanism to prove that a given program is running on a genuine enclave

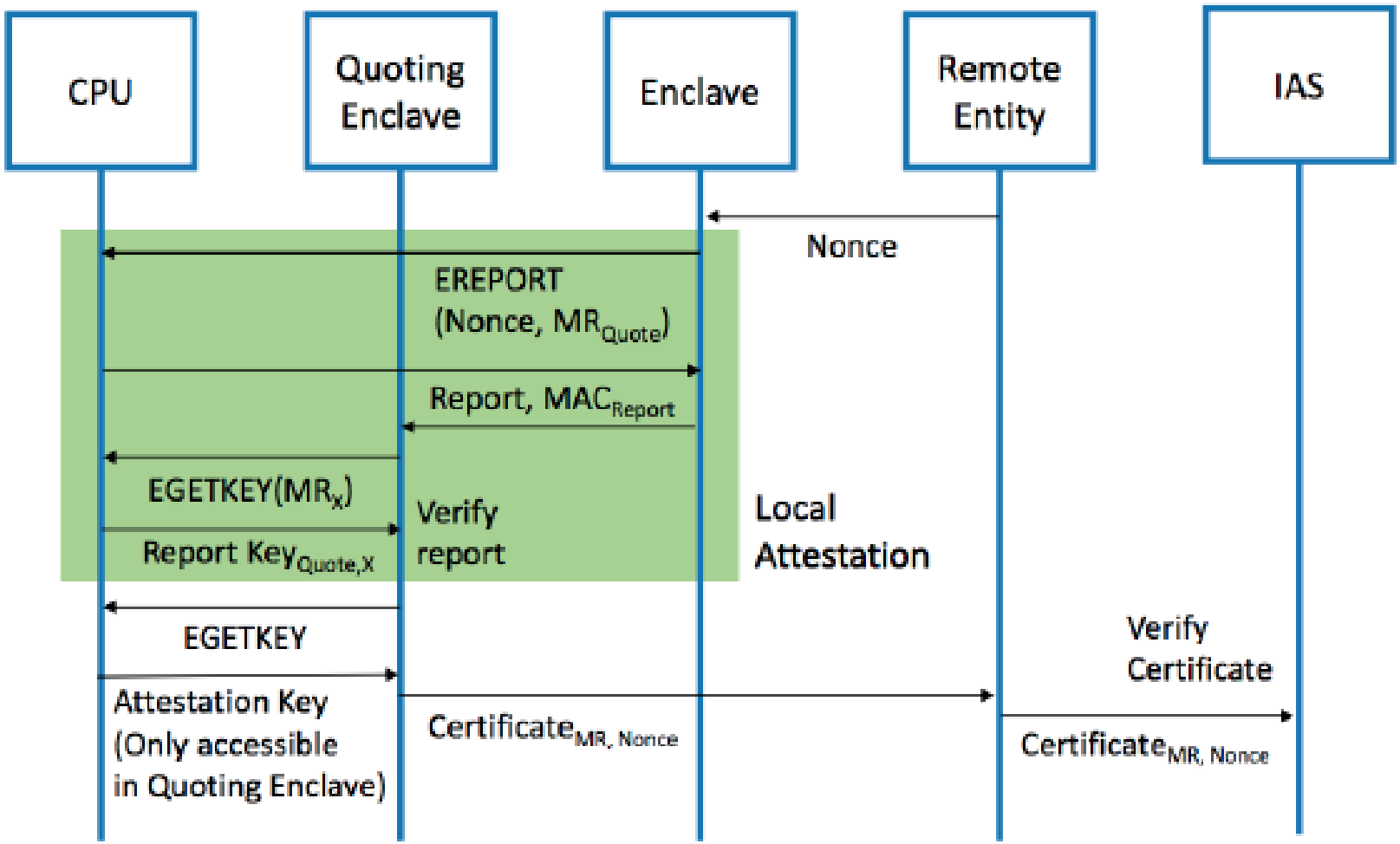
- Two attestation techniques:
  - Intra-machine (proving to another enclave on the same machine)
  - Inter-machine (proving to a third party)

- Uses register MRENCLAVE to generate a sha-256 hash of an internal log that measures enclave's activity stored in SECS

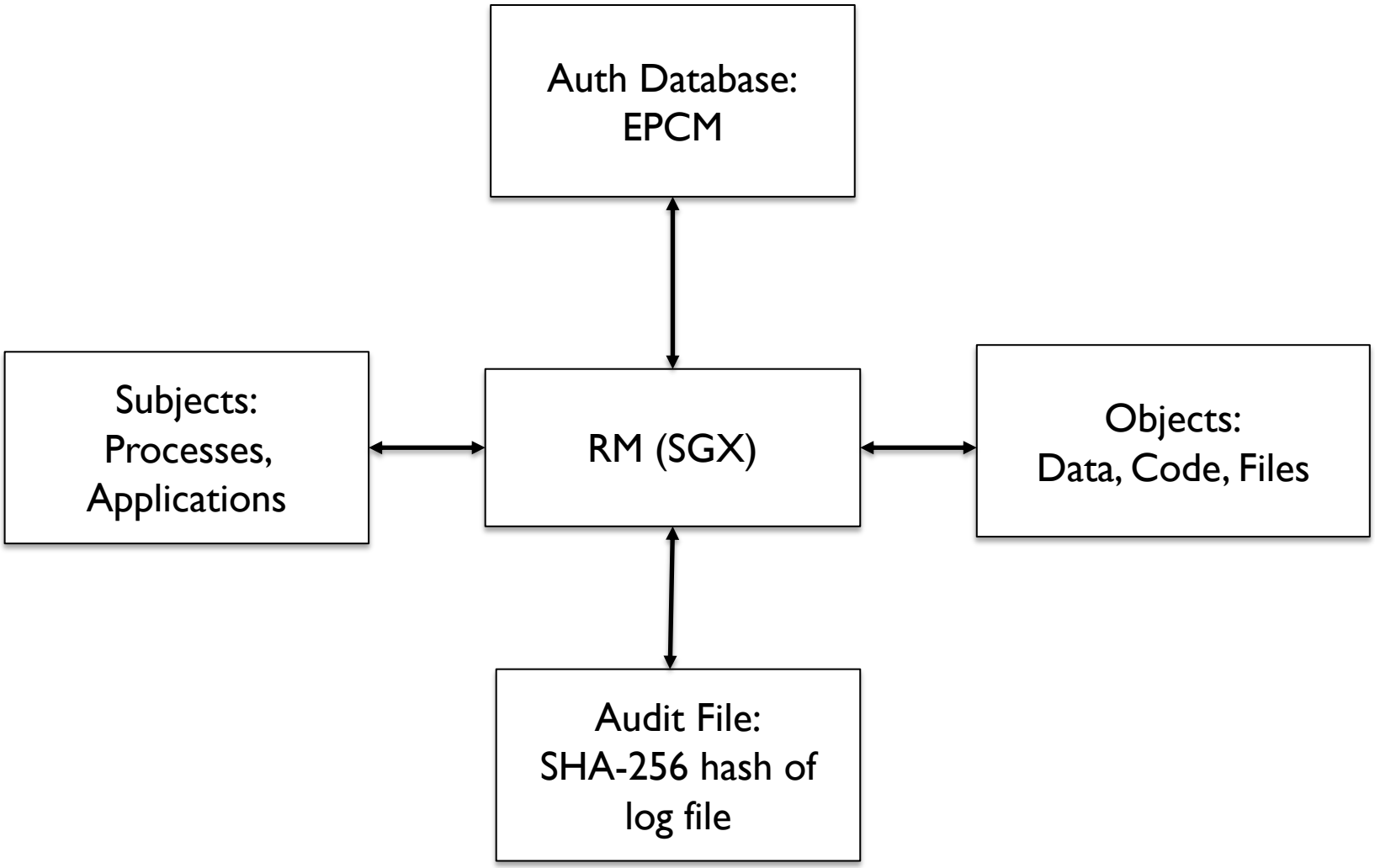
- Uses EPID ~~(processor's secret) to generate cryptographic~~ keys

Key	Purpose
Report key	Intra-machine attestation
Attestation key	Inter-machine attestation
Seal key	Sealing enclave secrets

Attestation Process



RM Components



Does it satisfy RM principles?

Tamper proof ☒

Non-Bypassable ☒

Verifiability ☒

Advantages

- Provides confidentiality and integrity (detection mechanism) of protected data (to a certain extent)
- Data is protected even from privileged system software and OS
- Provides attestation

Limitations

- Difficult to port existing applications to Intel SGX architecture
- Very few BIOS support SGX
- Cryptographic keys stored in MME are vulnerable to various attacks (has been exploited successfully by security researchers)

Vulnerabilities

1. Enclave attack: On 8 February 2019, researchers at Austria's Graz University of Technology published findings, which showed that in some cases it is possible to run malicious code from within the enclave itself. The exploit involves scanning through process memory, in order to reconstruct a payload, which can then run code on the system.
2. Plundervolt: Security researchers were able to inject timing specific faults into execution within the enclave, resulting in leakage of information. The attack can be executed remotely, but requires access to the privileged control of the processor's voltage and frequency.
3. LVI: Load Value Injection injects data into a program aiming to replace the value loaded from memory which is then used for a short time before the mistake is spotted and rolled back, during which LVI controls data and control flow.
4. ÆPIC leak: It allows for an attacker with root/admin privileges to gain access to encryption keys via the APIC by inspecting data transfers from L1 and L2 cache.

References

1. https://www.trentonsystems.com/blog/what-is-intel-sgx
2. https://nptel.ac.in/courses/106106199
3. https://insujang.github.io/2017-04-03/intel-sgx-protection-mechanism/
4. https://www.intel.com/content/dam/develop/external/us/en/images/intel-software-guard-extensions-tutorial-intel-sgx-foundation-fig03-658687.png
5. Innovative Instructions and Software Model for Isolated Executions, HASP 2013, F McKeen
6. Hardware Security: Intel SGX and VC3, Andrew Low