Intel's SGX
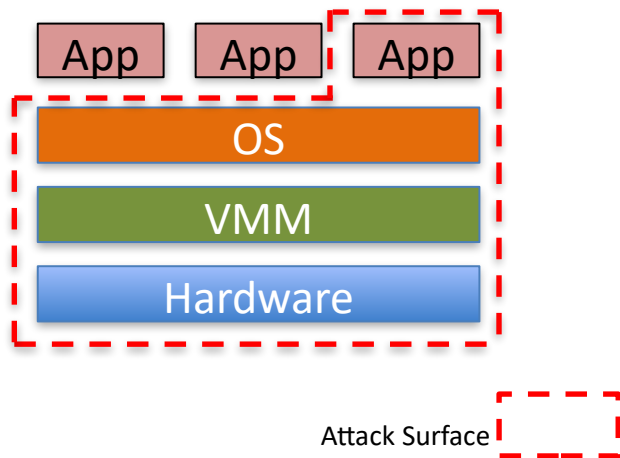
Some of the slides borrowed from Intel
Innovative Instructions and Software Model for Isolated Execution, HASP 2013 (F. McKeen et. al.)
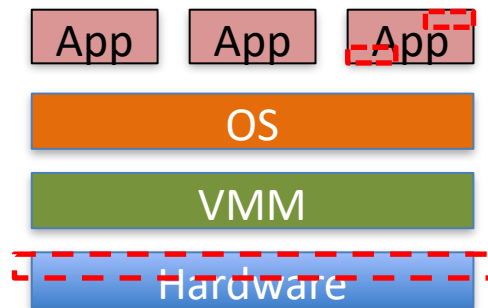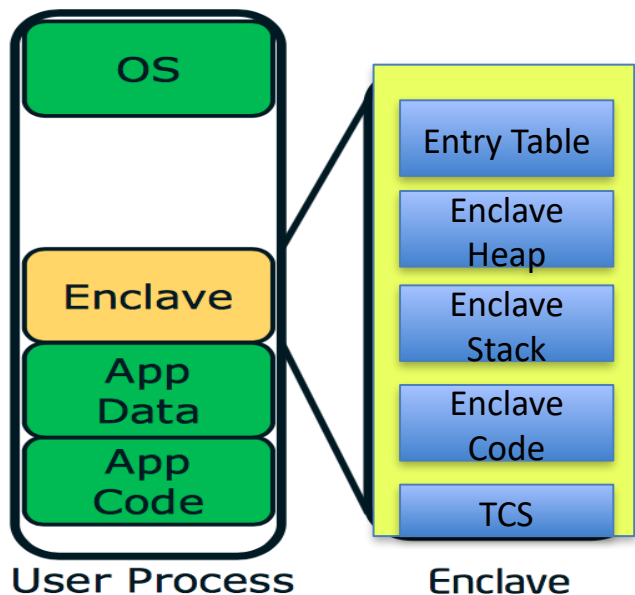
Reduced Attack Surface with SGX

Normally

With SGX enabled

Attack Surface

Malware that can subvert any one of app, OS, VMM, or hardware can steal secrets

Small attack surface (App + Hardware) Malware cannot steel secrets inspite of subverting OS, BIOS, VMM, most parts of the App, etc.
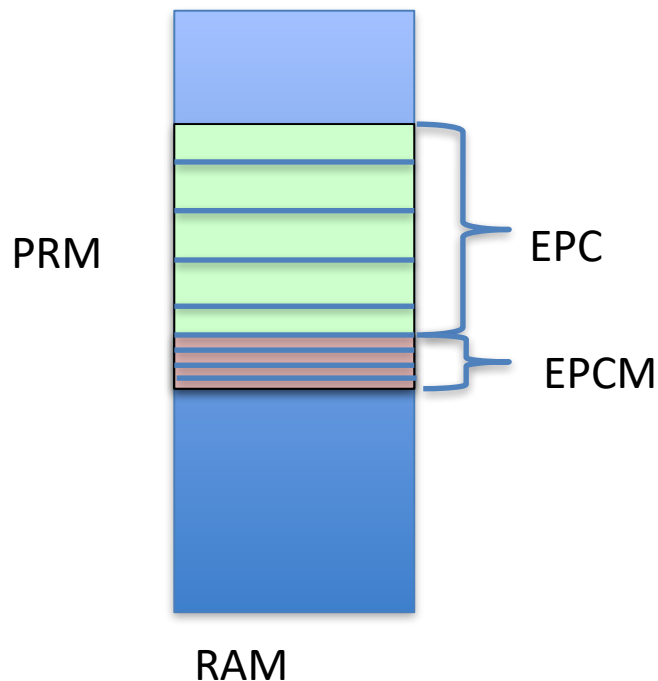
Enclaves
(reverse sandbox)

- Enclave has its own code and data areas
  Provides confidentiality and integrity
  With controlled entry points

- However, 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 access to tampered 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 (eg. Registers / caches), but unauthorized access is prevented
  - Enclave code and data is automatically encrypted it leaves the CPU package

Physical Memory

PRM

EPC

EPCM

RAM

- **PRM – processor related memory** 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
  - All SGX enclaves mapped into the PRM
- **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

PRM

EPC

EPCM

RAM
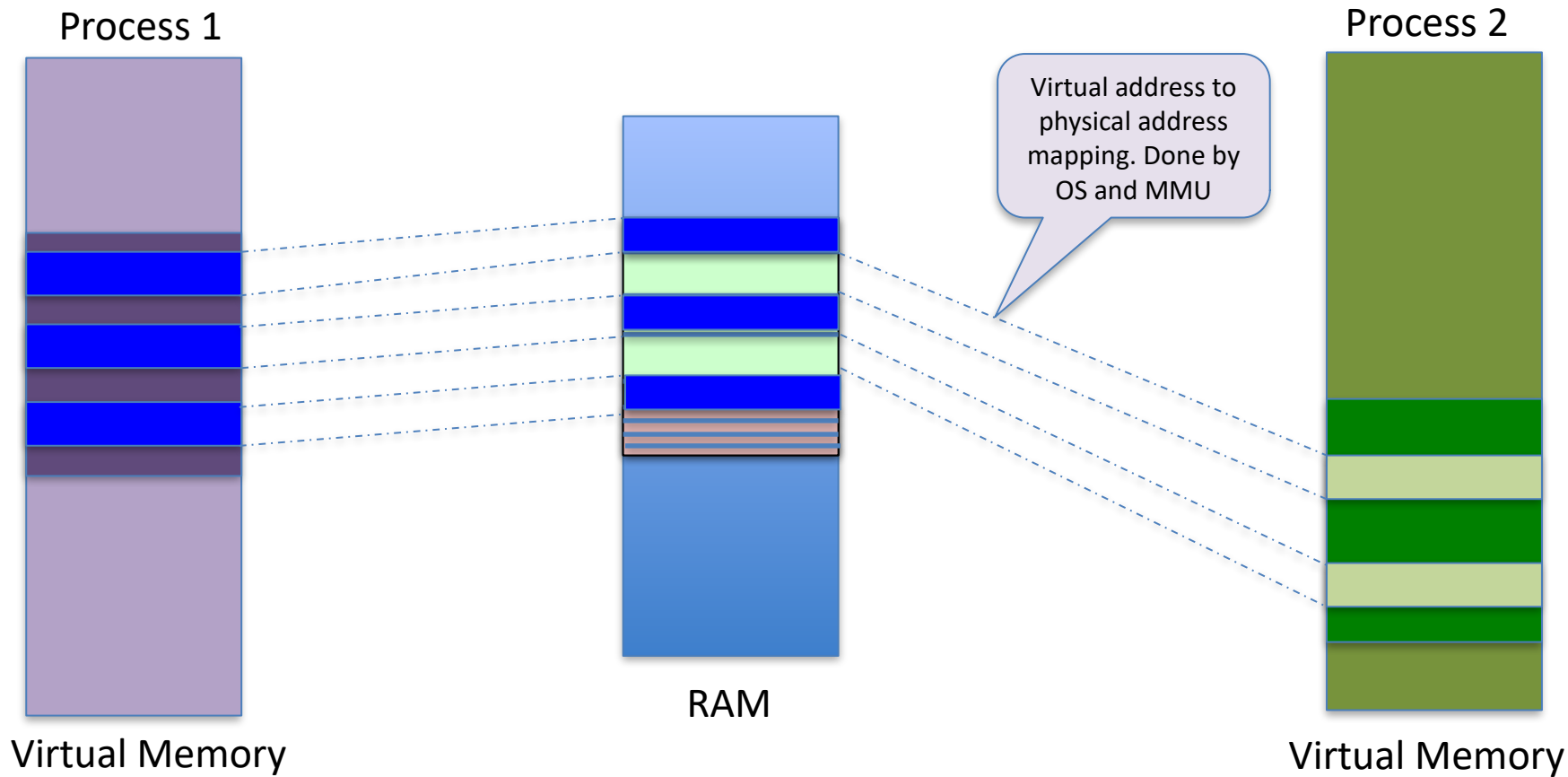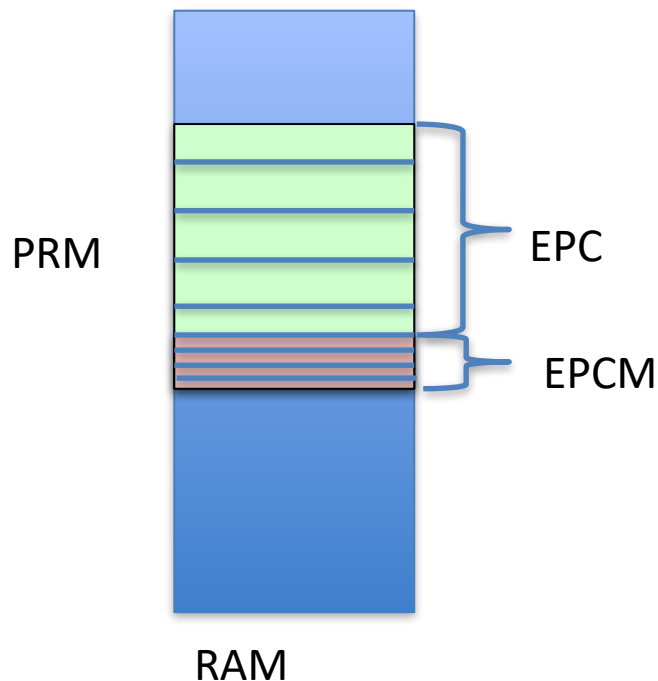
- **EPCM: Enclave page cache map**
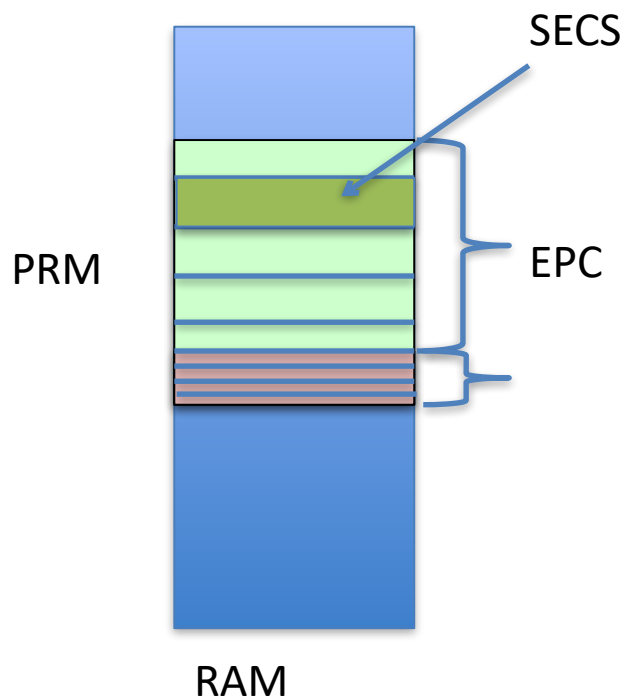 - one for each EPC
  - Used by hardware for access control
  - It stores management related aspects for the corresponding EPC
    - Aspects such as valid / invalid; r/w/x permissions
    - Type of page
    - Virtual address range through which, the EPC can be accessed
    - It is an additional layer of security compared to legacy paging and segmentation since we do not trust the OS

Physical Memory

SECS

EPC

PRM

RAM

- 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
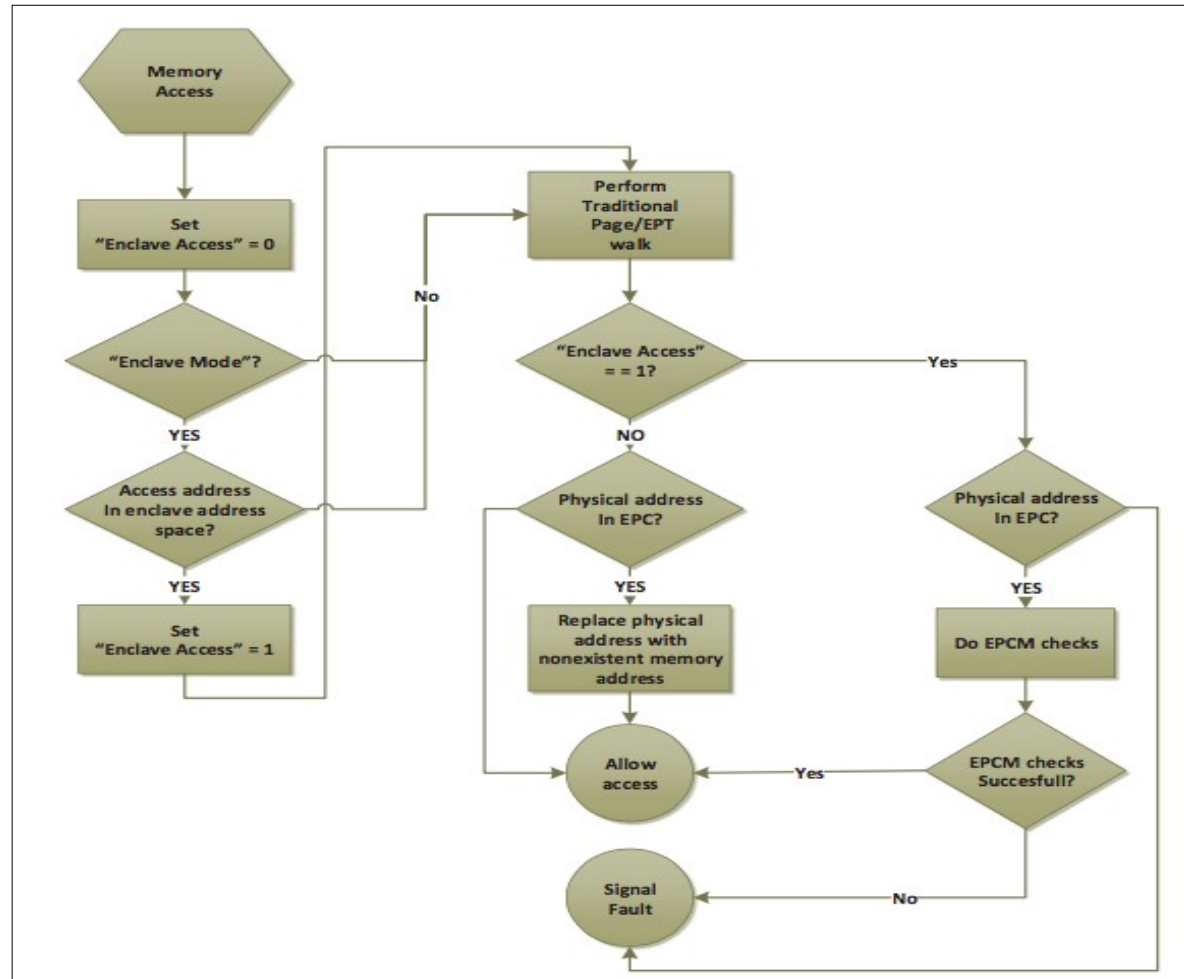   - 32 / 64 bit operating mode
    - Debug access

EPC Encryption

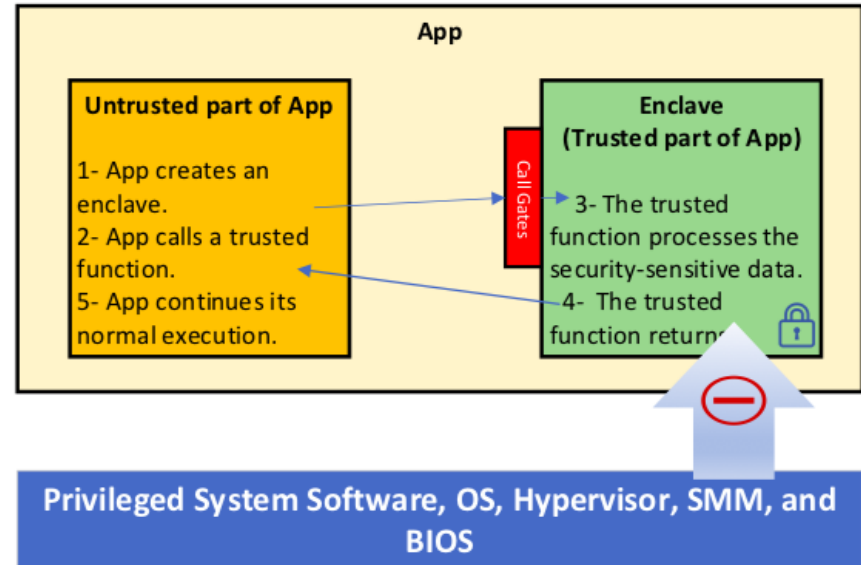- Hardware unit that encrypts and protects integrity of each EPC
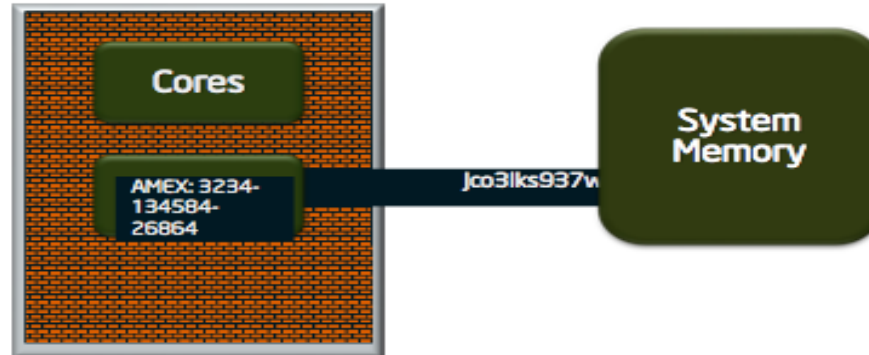
Memory Access

x

Application Execution Flow

App built with trusted and untrusted part

1. Untrusted part creates and executes the enclave
   1. Enclave is placed in the EPC. It is encrypted and trusted
2. Trusted function is called and execution is transferred into the enclave
3. Trusted function executes
4. Trusted function returns
5. Application continues execution

App

Untrusted part of App

1- App creates an enclave.
2- App calls a trusted function.
5- App continues its normal execution.

Call Gates

Enclave (Trusted part of App)

3- The trusted function processes the security-sensitive data.
4- The trusted function returns

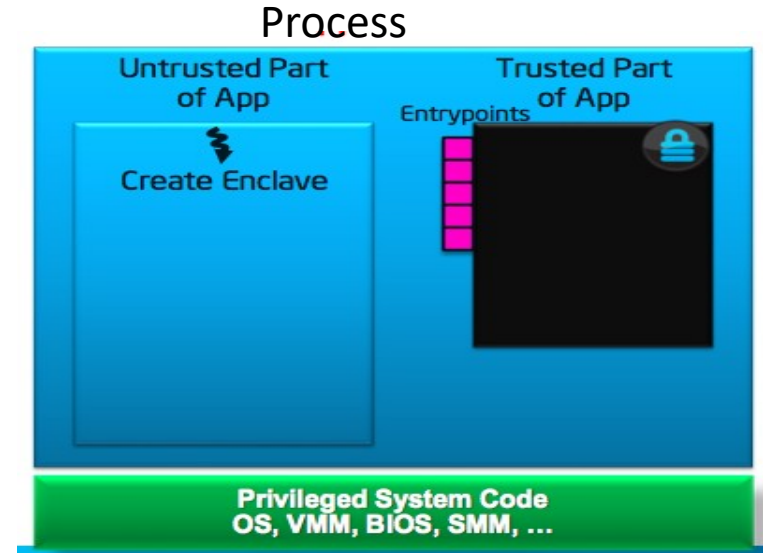**Privileged System Software, OS, Hypervisor, SMM, and BIOS**

Enclave Life Cycle
(creation)

ECREATE Instruction

- Creates a SECS (SGX enclave control structure)
  - Contains global information about the enclave
- System software can choose where (in the process virtual space) the enclave should be present
- Also specifies
  - Operating mode (32/64 bit)
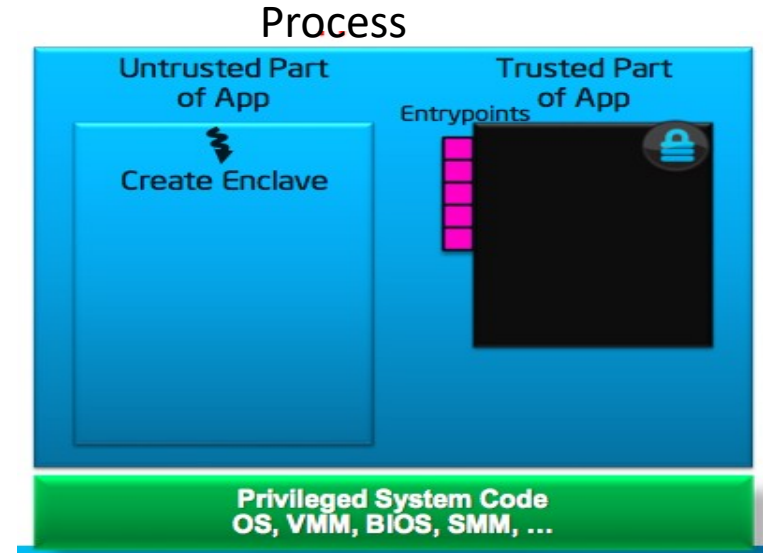 - Processor features that is supported
  - Debug allowed

Process

Untrusted Part of App

Create Enclave

Trusted Part of App

Entrypoints

Privileged System Code
OS, VMM, BIOS, SMM, ...

Enclave Life Cycle
(adding pages)

EADD Instruction

- System software should select free ECS page
- EADD will initialize EPCM with
  - Page type (TCS / REG)
  - Linear address that will access the page
  - RWX permissions
  - Associate the page in SECS structure
- EADD will then record EPCM information in a crypto log stored in the SECS
  - This is the measurement of the enclave
  - Used for gaining assurance
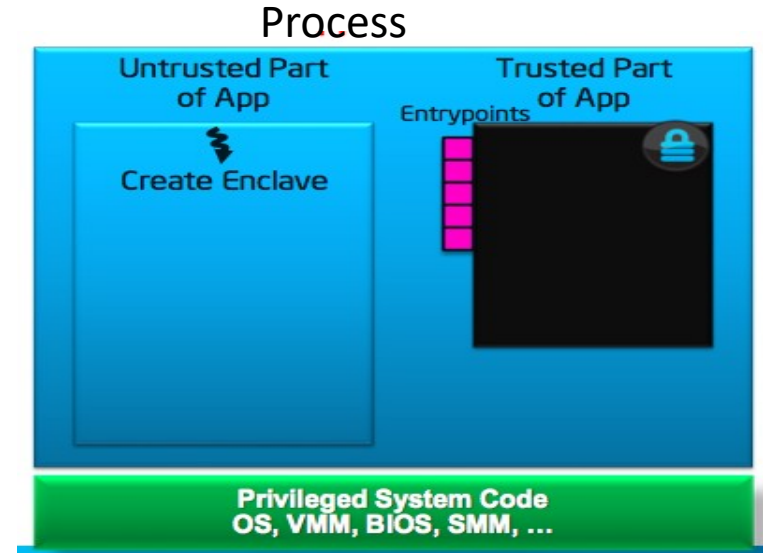- Copy 4K bytes of data from unprotected memory into the enclave

Process

Untrusted Part of App

Create Enclave

Trusted Part of App

Entrypoints

Privileged System Code
OS, VMM, BIOS, SMM, …

Enclave Life Cycle
(measuring pages)

EEXTEND

- Measure a 256 byte region in an EPC page
  - This region is specified by the developer
  - The measurement comprising of a 64 bit address and a 256 byte information in the SECS
  - 16 invocations EEXTEND needed to measure the whole page
- Correct construction of the enclave would result in a matching with the enclave owner
  - The enclave owner's signature is stored in a SIGSTRUCT structure
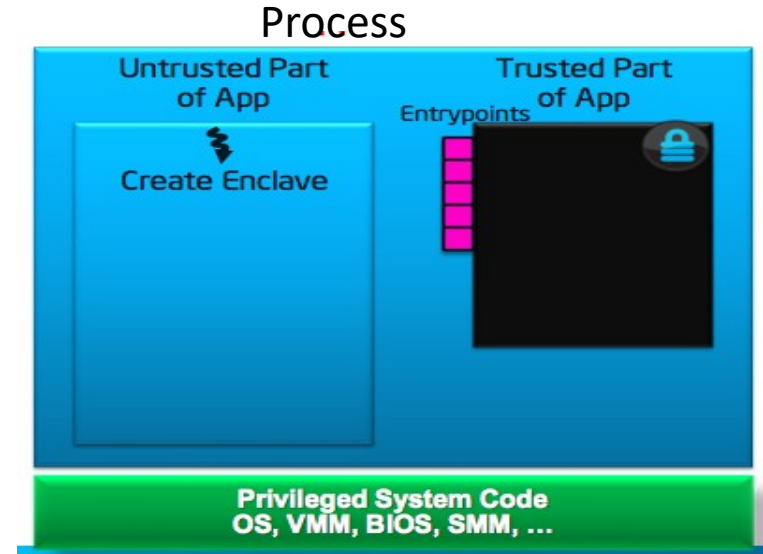 - This can also be remotely verified

Process

Untrusted Part of App

Create Enclave

Trusted Part of App

Entrypoints

Privileged System Code
OS, VMM, BIOS, SMM, …

Enclave Life Cycle
(initializing pages)

EINIT

- Should be invoked after all pages have been added
- Verify that the signature matches that of the owner's signature
- If EINIT is successful, it allows the enclave to be entered

Process

Untrusted Part of App
Create Enclave

Trusted Part of App
Entrypoints

Privileged System Code
OS, VMM, BIOS, SMM, …

Enclave Life Cycle
(enter/exit)

Process invokes the enclave through pre-defined entry points using EENTER instruction

EENTER
- Changes made to enclave mode
- Need to know the location to transfer control and location where to save state in case of an interrupt
- Defines an Asynch. Exit pointer, which where IRET returns to after servicing an interrupt
  - It is outside the enclave
  - And typically will have an instruction **ERESUME**
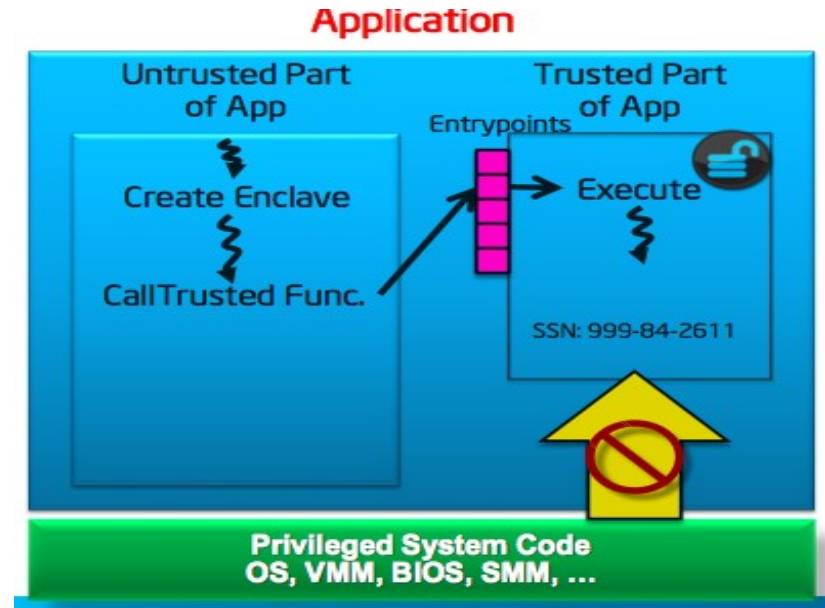
Entry into the Enclave

- Set TCS to busy
- Change mode to enclave mode
- Save state of SP, BP, etc. for return in case of async. Exit
- Save AEP
- Transfer control from outside the enclave to inside

Exit from Enclave

- **EEXIT**
  - Clear enclave mode and flush TLB entries
  - Mark TCS as free.
  - Transfer control outside the enclave

Asynchronous Exit (AEX)

- Occurs when an interrupt / exit occurs
- Processor state is securely saved inside the enclave and replaced with synthetic states
- AEP pushed onto the stack
  (AEP is a location outside the enclave where execution goes to after IRET)
- After AEX completes, the logical processor is no longer in enclave mode

- Resuming after an interrupt
  - EERESUME instruction is invoked, which restores all registers
  - Typically EERESUME is present at the AEP location
- Resuming after a fault that occurred in the enclave?
  - Eg. A divide by zero

Instruction set Extensions for SGX

- Privileged Instructions
  - Creation related: to create, add pages, extend, initialize, remove enclave
  - Paging related: evict page, load an evicted page
- User level instructions
  - Enter enclave, leave enclave
  - Interrupt related: asynchronous exit, resume

Points to Ponder

Compare Intel SGX with ARM Trustzone

- What is the attacker assumption in each case?
- Why does SGX encrypt data in memory while Trustzone does not?