Intel® Software Guard Extensions (Intel® SGX) Support for Dynamic Memory Management Inside an Enclave

Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, Carlos Rozas Intel Corporation

Saeid Mofrad

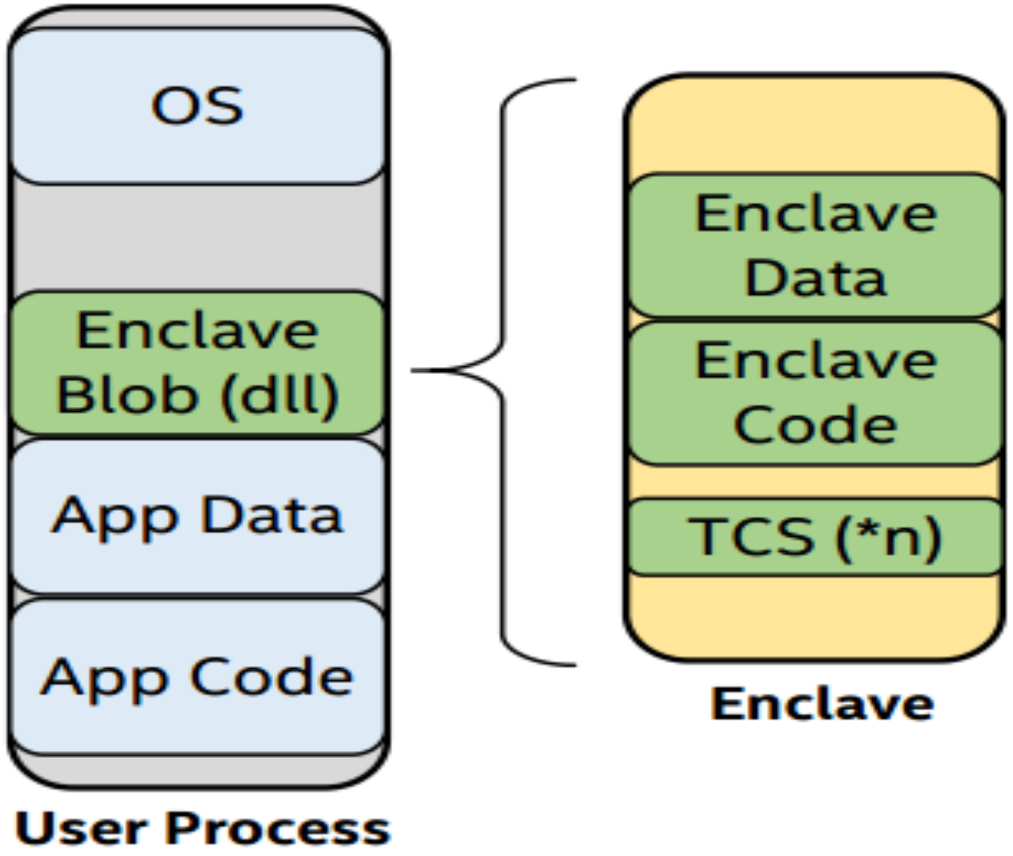
1-INTRODUCTION:

SGX:

Software Guard Extensions provides the capability to protect specified areas of an application from outside access. The area is called an **enclave** and hardware provides confidentiality and integrity for the specified area. SGX allows software developers to build trusted modules inside an application to protect secrets. A software developer specifies the contents of an enclave and a relying party can confirm that the area is instantiated correctly on a remote machine.

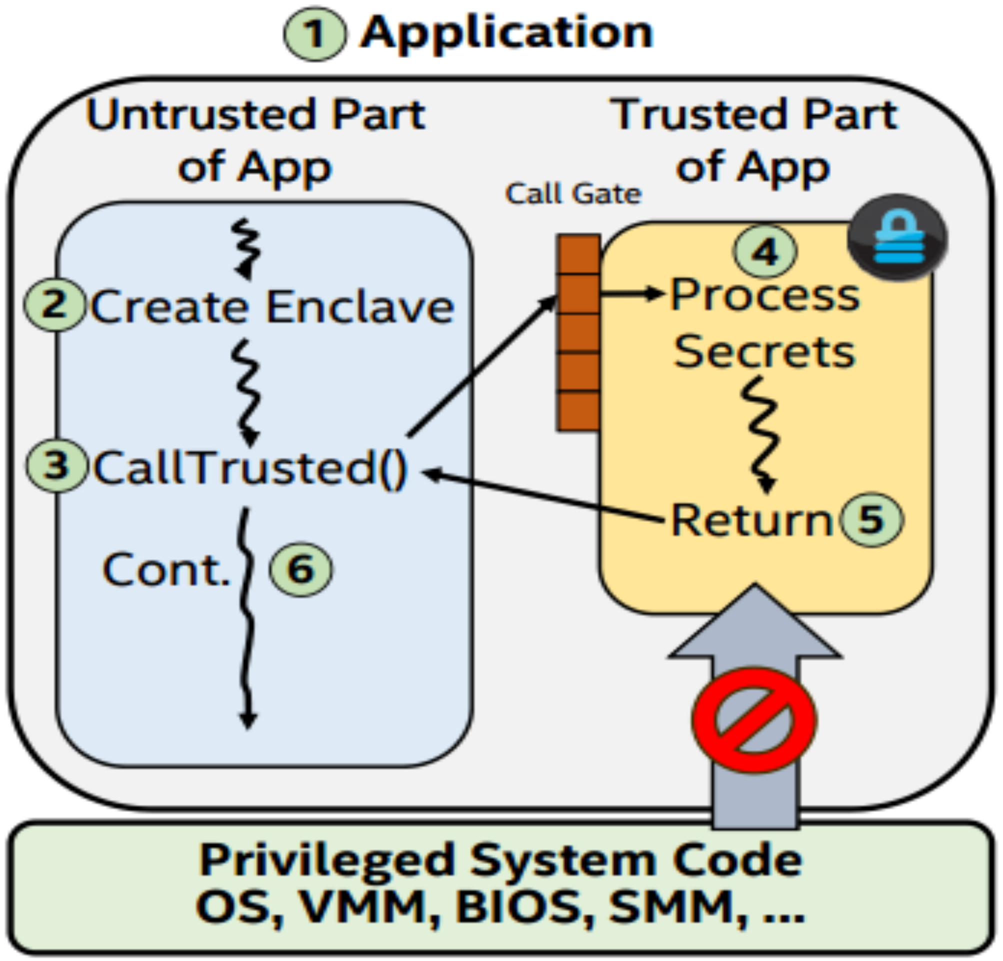
PROCESS VIEW

Protected execution environment embedded in a process



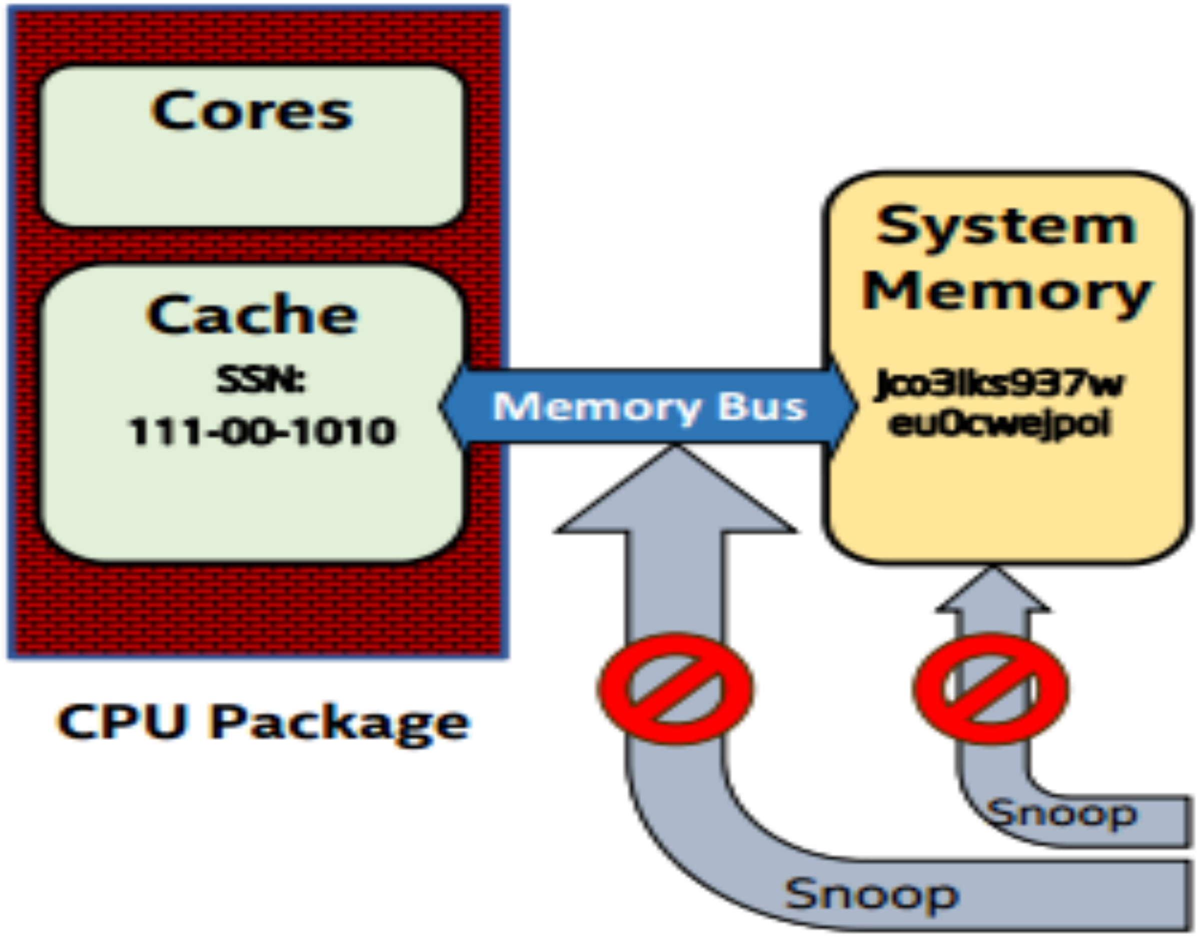
- With its own code and data
- Providing Confidentiality & Integrity
- Controlled entry points
- Multi-thread support
- Full access to app memory and processor performance

EXECUTION FLOW



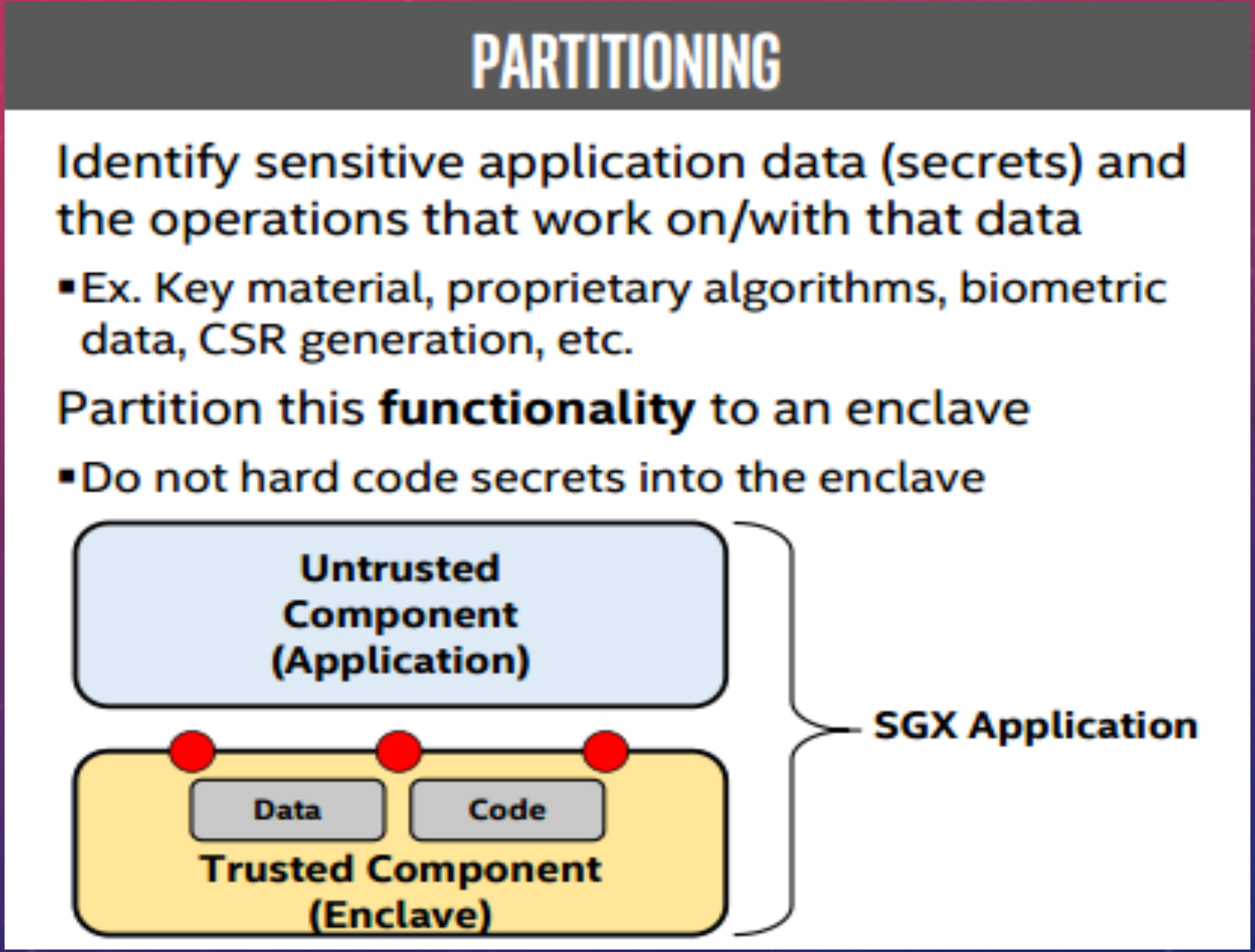
1. App built with trusted and untrusted parts
2. App runs & creates the enclave which is placed in trusted memory
3. Trusted function is called, execution transitioned to the enclave
4. Enclave sees **all process data** in clear; external access to enclave data is denied
5. Trusted function returns;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

SECURITY PERIMETER



- Security perimeter is the CPU package boundary
- Data and code unencrypted inside CPU package
- Data and code outside CPU package is **encrypted and integrity checked**
- External memory reads and bus snoops see only encrypted data

Application development consideration:



PARTITIONING

Identify sensitive application data (secrets) and the operations that work on/with that data

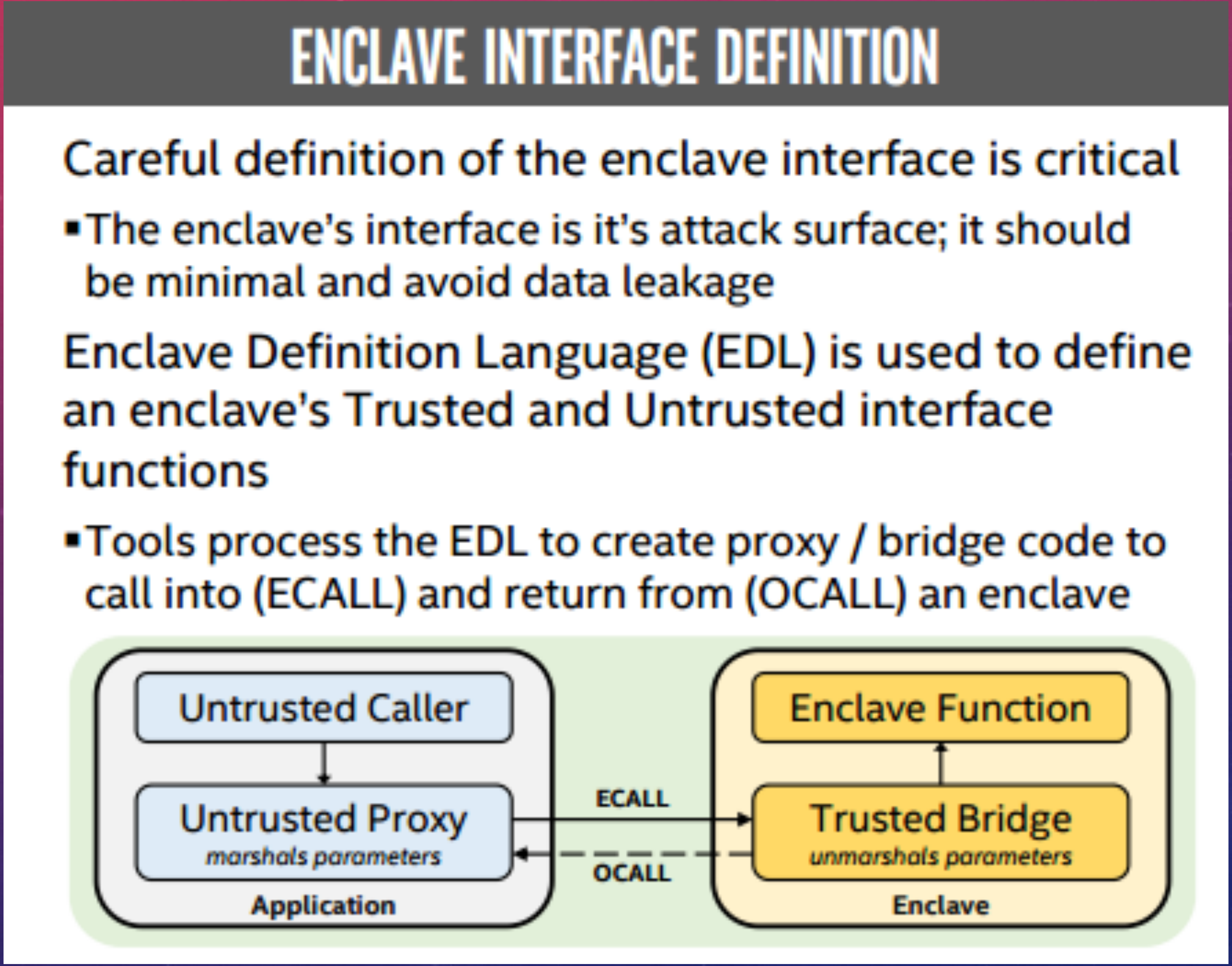
- Ex. Key material, proprietary algorithms, biometric data, CSR generation, etc.

Partition this **functionality** to an enclave

- Do not hard code secrets into the enclave

Untrusted Component (Application)
Data
Code
Trusted Component (Enclave)
SGX Application

Application development consideration[3]:



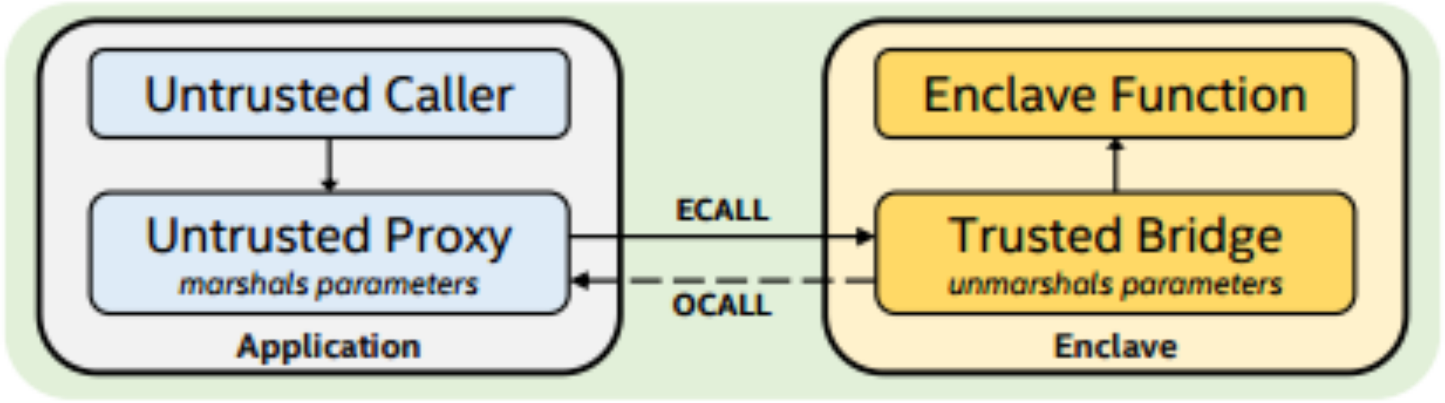
ENCLAVE INTERFACE DEFINITION

Careful definition of the enclave interface is critical

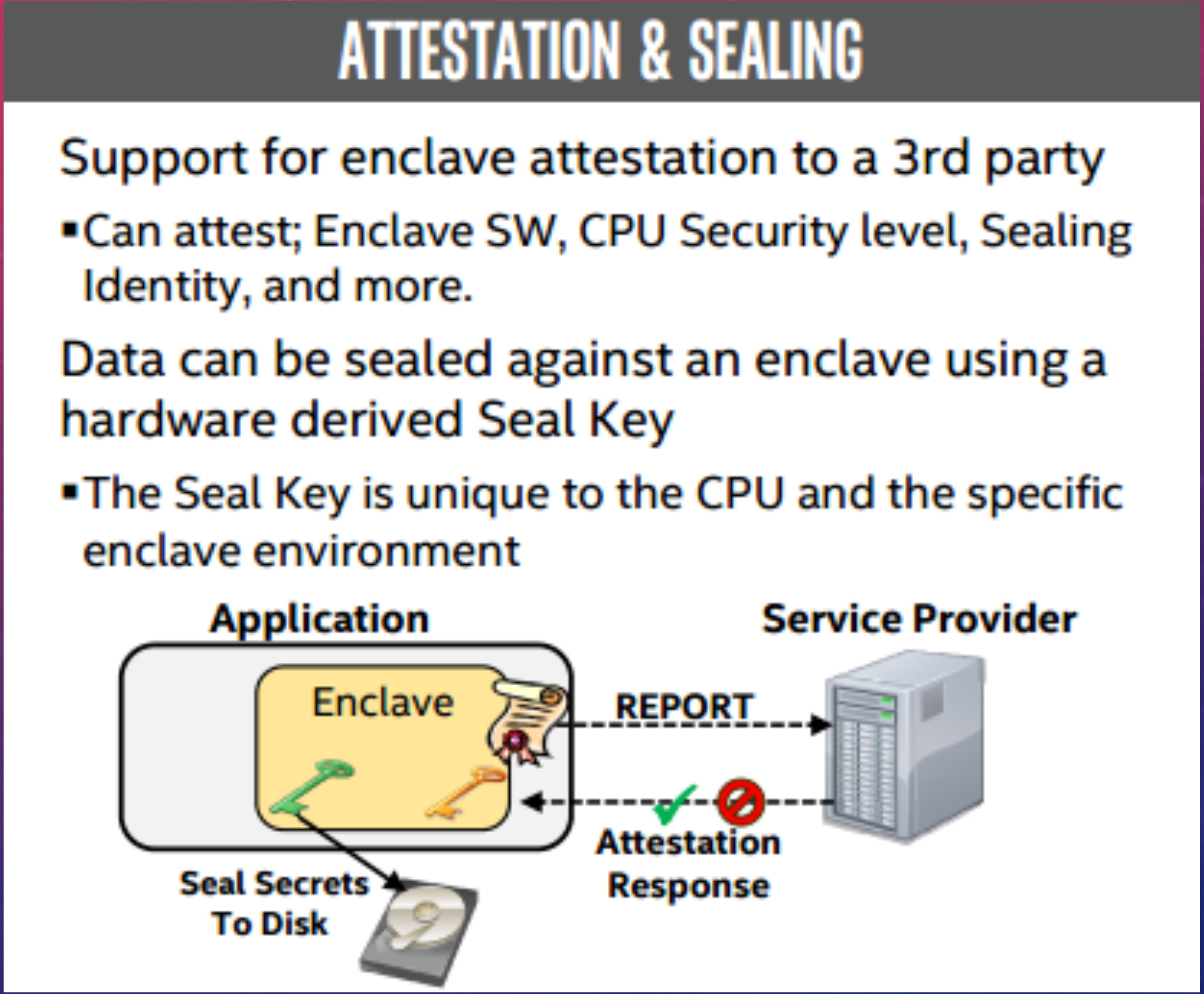
- The enclave's interface is it's attack surface; it should be minimal and avoid data leakage

Enclave Definition Language (EDL) is used to define an enclave's Trusted and Untrusted interface functions

- Tools process the EDL to create proxy / bridge code to call into (ECALL) and return from (OCALL) an enclave



Application development consideration[3]:



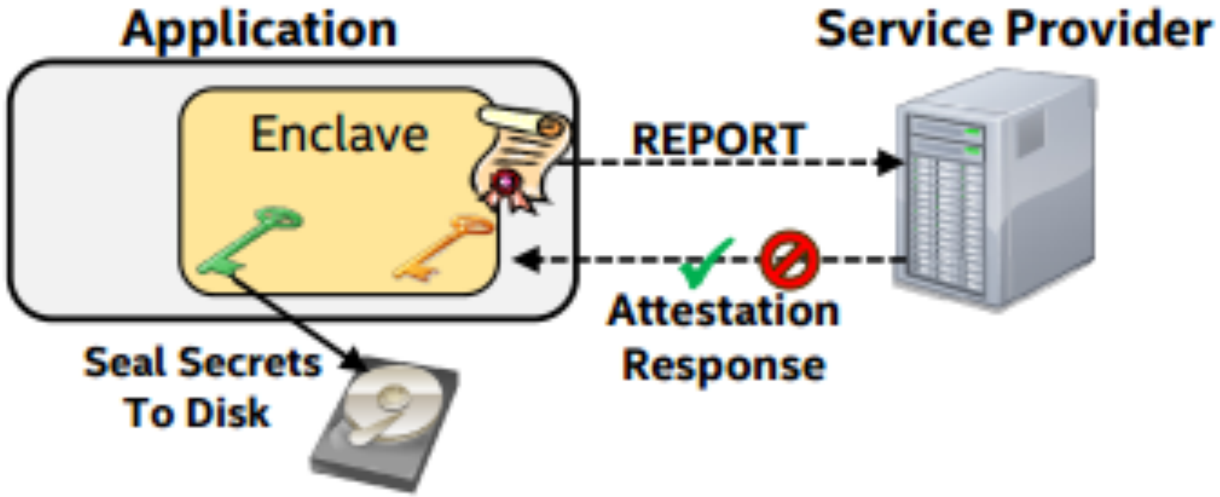
ATTESTATION & SEALING

Support for enclave attestation to a 3rd party

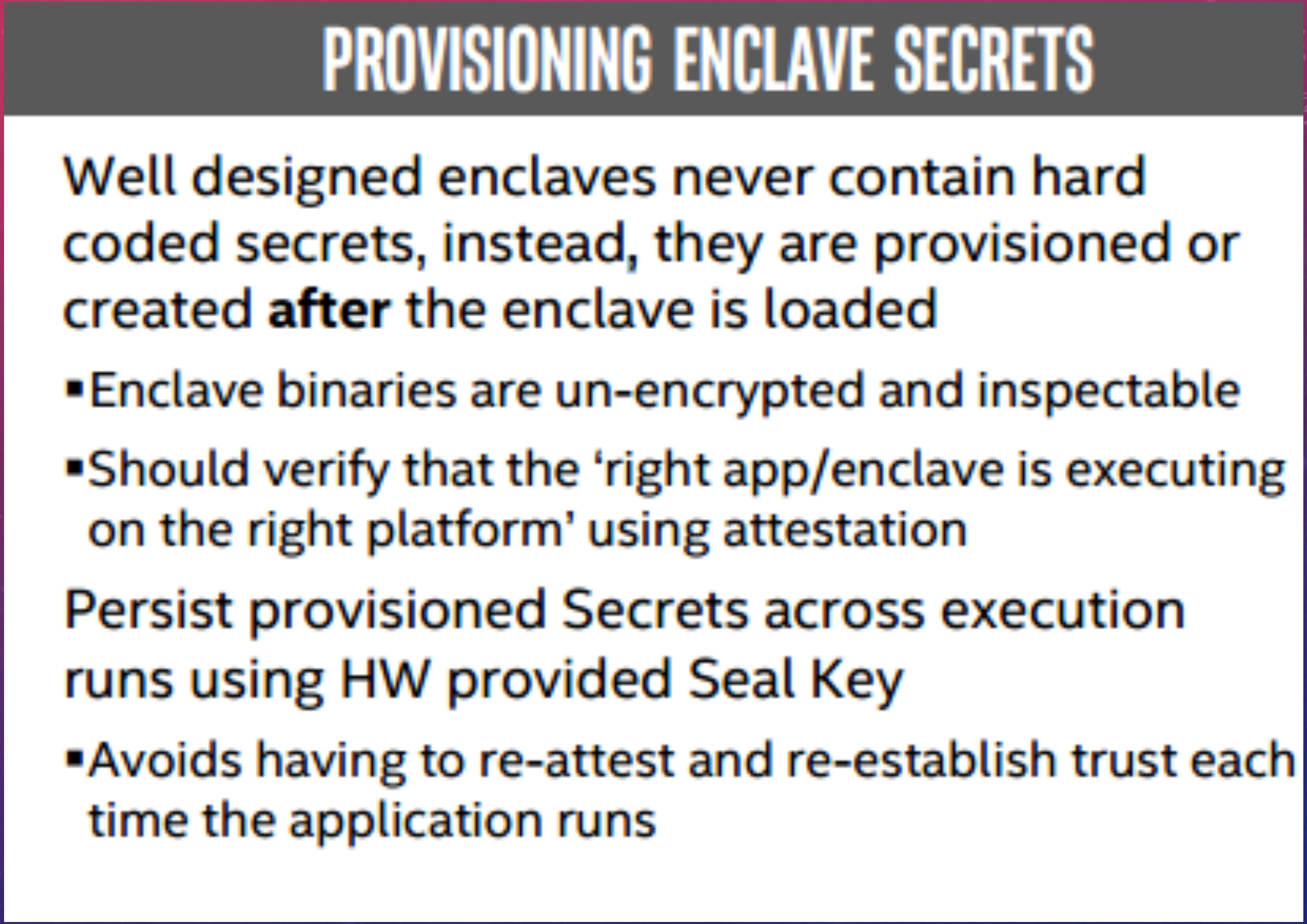
- Can attest; Enclave SW, CPU Security level, Sealing Identity, and more.

Data can be sealed against an enclave using a hardware derived Seal Key

- The Seal Key is unique to the CPU and the specific enclave environment



Application development consideration[3]:



PROVISIONING ENCLAVE SECRETS

Well designed enclaves never contain hard coded secrets, instead, they are provisioned or created **after** the enclave is loaded

- Enclave binaries are un-encrypted and inspectable
- Should verify that the 'right app/enclave is executing on the right platform' using attestation

Persist provisioned Secrets across execution runs using HW provided Seal Key

- Avoids having to re-attest and re-establish trust each time the application runs

MOTIVATION OF SGX-2: THREE SHORTCOMINGS WITH THE SGX1

- **First all enclave memory must be committed at enclave build time.** This **increases the build time**. Committing memory **places pressure on the enclave page cache (EPC)**, the enclave developer must allocate memory for worst-case memory consumption of any workload. Otherwise, the enclave developer will need to releases enclaves designed for different size workloads.
- The second shortcoming is related to the **management of access permissions associated with an enclave page.** SGX extends the access permission model by associating an additional set of access permissions with enclave page that are stored in a SGX structure called the Enclave Page Cache Map (EPCM)
- The last shortcoming is related to **library OS support** where secure exceptional handling and lazy loading code inside an enclave are important features. SGX-1 didn’t have information recorded when a general protection fault or page fault occurs inside an enclave
- **To address these problems six new instructions and new exception behavior were added to the SGX architecture known as SGX2**

2 SGX2 CONSIDERATIONS & REQUIREMENTS:

- Manipulating memory and permissions of an enclave must be done with the knowledge and consent of the enclave.
- If enclave code is changed incorrectly or without knowledge of the enclave, execution should be suspended until the condition is resolved. It enables the enclave to manage its own security
- The system resource manager (OS or VMM) must be able to manage and allocate the resources as requested using standard techniques and priorities.
- Manipulation of memory permissions involves both the system permissions and the EPCM permissions. EPCM permissions allow the enclave developer to specify the restrictions and access control for the enclave

- SGX 2 memory management -> (**system manager**) which manages the system resources
- internal enclave resource manager (**internal manager**) which manages the enclave memory from inside the enclave.
- A protocol which consists of communication between the system manager and an internal manager is this:

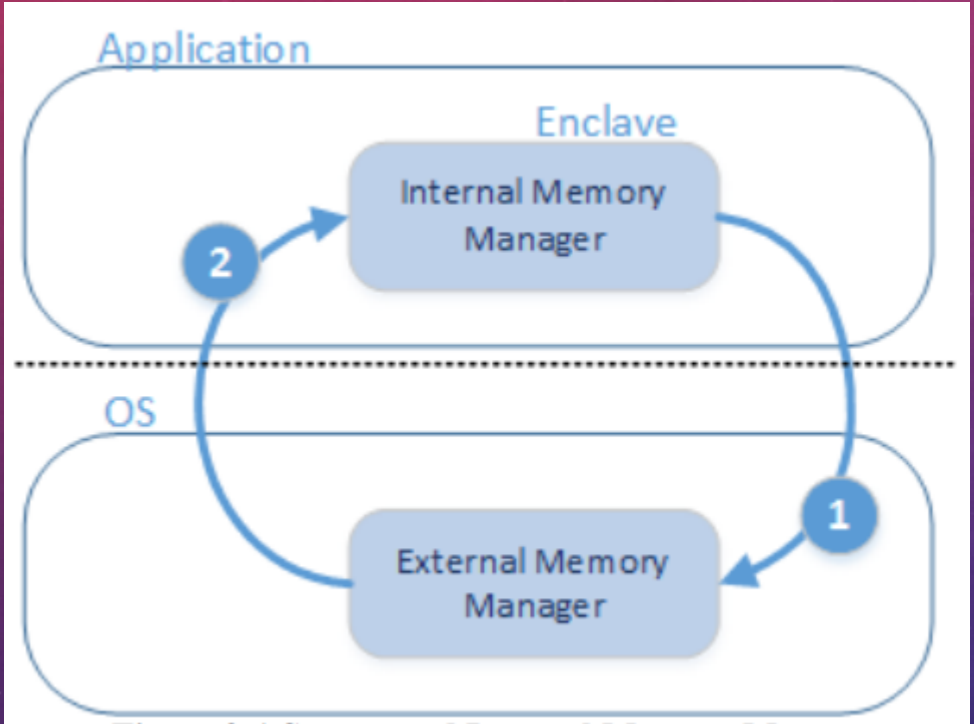


Figure 2-1 System and Internal Memory Managers

- **The system memory manager**
- allocating memory -> paging memory -> changing permissions, -> changing page types.- >managing the page table entry permissions -> initiating EPCM permissions of the enclaves (by calling instruction.)

- **The internal manager**
- **starting** memory change requests -> **verifying** that the system manager has processed the requests correctly.
- The internal manager does not have direct access to the page tables and must request the system manager to make changes in page table entry (PTE) permissions.

2.1 SECURITY CONSIDERATIONS

- Must ensure that changes in permission do not affect the security of the enclave.
- When restrict page permissions -> check **permission restrictions are complete** and the previous cached address translations **or cached permissions are removed**. SGX2 checks old permissions are removed from the TLBs
- SGX1 allows the system memory manager to remove pages from an enclave using the EREMOVE leaf function. However, since the enclave doesn't participate in this process it doesn't know if the page removed.

2.2 SOFTWARE CONSIDERATIONS

- Internal memory manager wants to **reallocate the memory resources**: add a thread; must allocated as Thread Control Structure (TCS), State Save Area (SSA) pages. Add more memory to enclave.
- **Exception Reporting Inside an Enclave**: for Library OS usage. In this case the exception condition should be reported inside the enclave. SGX2 adds several exception conditions to the SSA frame when exiting an enclave. They include page faults (#PF) and general protection violations (#GP).
- **Demand Loading of Library Pages**: The internal manager must have a mechanism to load the page without allowing access until the copy is complete. SGX2 adds a leaf function to perform the copy securely.

3.1 ENCLAVE MALLOC

- The following is protocol:
- 1. Internal manager requests memory -> enclave runtime system from its internal pool of memory. memory pool low the internal manager -> requests the system manager to allocate more memory.
- 2. The system manager allocates virtual address space but does not commit memory and -> returns a reference to the virtual address space to the internal manager
- 3. The enclave internal manager -> returns a reference to the enclave. When the enclave accesses the newly allocated memory, -> a page fault is generated as memory has not been committed.
- 4. The OS page fault handler detects that the virtual address has been allocated but memory has not been committed. -> The OS commits memory by using EAUG and maps the committed but **pending** page into the enclave address space-> The OS then sends a signal to the enclave internal manager.
- 5. The internal manager receives the -> The internal manager checks that the virtual address has been committed -> the internal manager executes EACCEPT which allows the enclave to access the pending page. -> The signal handler returns back to the application which eventually results in the enclave execution resuming.

3.2 ENCLAVE FREE

- The following is an example protocol:
- 1. The enclave releases memory **->** internal manager release address space back to the OS.
- 2. The system manager executes EMODT on all pages **->** change the page type to PT_TRIM and -> clear the EPCM access permission bits. This begins the process of decommitting memory. The system manager then executes ETRACK on the SECS of the calling enclave and then sends IPIs to logical processors which may contain TLB mappings to the pages that had been trimmed.
- 3. Once all logical processors responded to the IPI, control is returned to the internal manager.
- 4. The internal manager verifies that committed memory has been decommitted by executing EACCEPT to verify that the pages trimmed and all stale TLB mappings have been flushed. The internal manager needs to update its tracking information that the virtual address has no committed memory.
- 5. The system manager can later reclaim the committed memory by executing EREMOVE on the trimmed pages.

3.3 CHANGING PAGE PERMISSIONS

- Change is permissive then the following protocol:
- 1. internal manager runs EMODPE to extend the page permissions in the EPCM.
- 2. The internal manager requests the system manager to extend page permissions in the page tables.
- If the change in permission is restrictive then the following protocol:
- 1. The internal manager requests that the system manager to restrict permissions on a page.
- 2. The system manager executes EMODPR and updates page table permissions. After permissions have been updated, the system manager executes ETRACK on the SECS of the calling enclave and sends IPIs to all processors that may be executing inside the enclave to flush TLB mappings.
- 3. After all IPIs have been acknowledged, control is returned to the internal manager. The internal manager verifies that page permissions restricted and TLB mappings flushed by executing EACCEPT

3.4 THREAD CONTROL STRUCTURE ALLOCATION

- 1. Internal manager initializes from a regular EPC page with appropriate TCS values. If the enclave memory has not been committed then internal manager will need to perform a request to allocate memory as described in section 3.1., then internal manager requests that the system manager convert the page to a TCS.
- 2. The system manager executes EMODT to set the page type to PT_TCS and to clear the EPCM access permission bits. The page is also marked modified which prevents the page from being used as a TCS.
- 3. The system manager then executes ETRACK. The system manager sends IPIs to flush all old mappings to the page and returns control to the internal manager.
- 4. The internal manager executes EACCEPT on the modified TCS page. EACCEPT will verify that TLB mappings flushed and perform consistency checks on the TCS page then clearing the modified bit and making the page available to EENTER.

DYNAMIC LOADING OF MODULES

- SGX2 provides EACCEPTCOPY which allows the internal manager to atomically initialize the contents and permission of a page.
- 1. the internal manager indicates to the system manager that a virtual address space allocated but not committed (same as in 3.1).
- 2. When an enclave attempts to access a page in this virtual address, a page fault is generated and the system manager commits memory by executing EAUG and signals the internal manager.
- 3. The internal manager identifies the virtual address as belonging to a module page to be loaded. The system manager may load the contents of the page into regular memory or the enclave runtime system may need to request the content be loaded into regular memory.
- 4. The internal manager then copies the contents of the module into private enclave memory. The internal manager should verify the integrity of the contents and apply any required relocations. Finally, the internal manager copies the contents and initializes permissions using EACCEPTCOPY.

3.6 LIBRARY OS SUPPORT

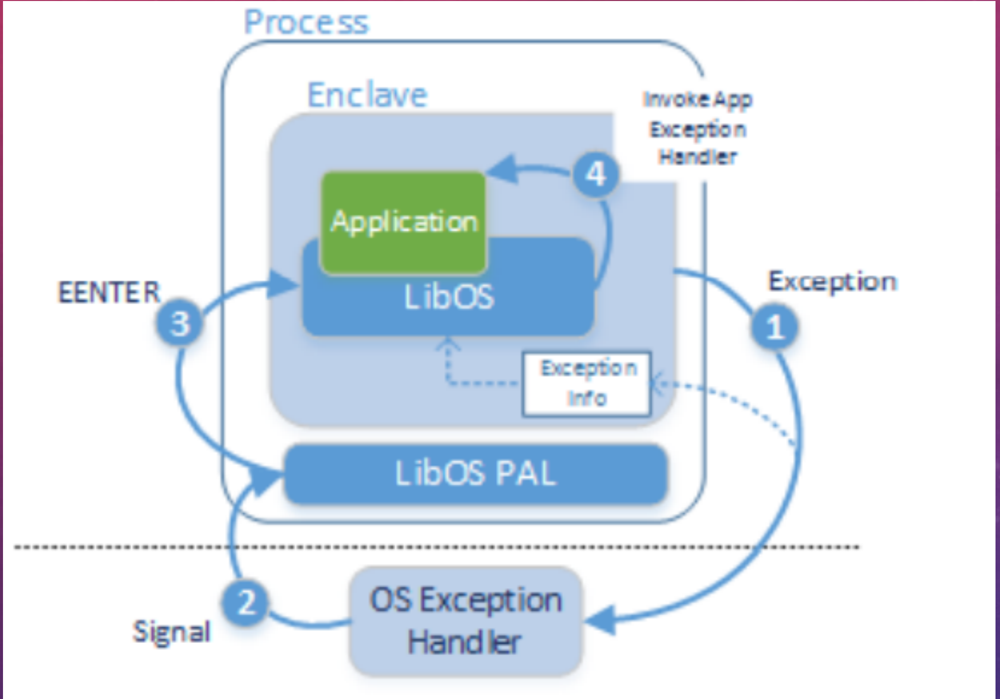


Figure 3-2 SGX enabled LibOS exception handling

1. The process begins with an exception generated inside an enclave. The processor records exception information in the SSA and delivers the exception to the OS exception handler.
2. If the OS cannot handle the exception, the OS signals the LibOS PAL (Platform Adaptation Layer) exception handler.
3. 3. The LibOS PAL executes EENTER to invoke the LibOS exception handler inside the enclave.
4. 4. The LibOS exception handler reads the exception information then generates an OS specific exception context, and invokes the application exception handler inside the SGX enabled LibOS.

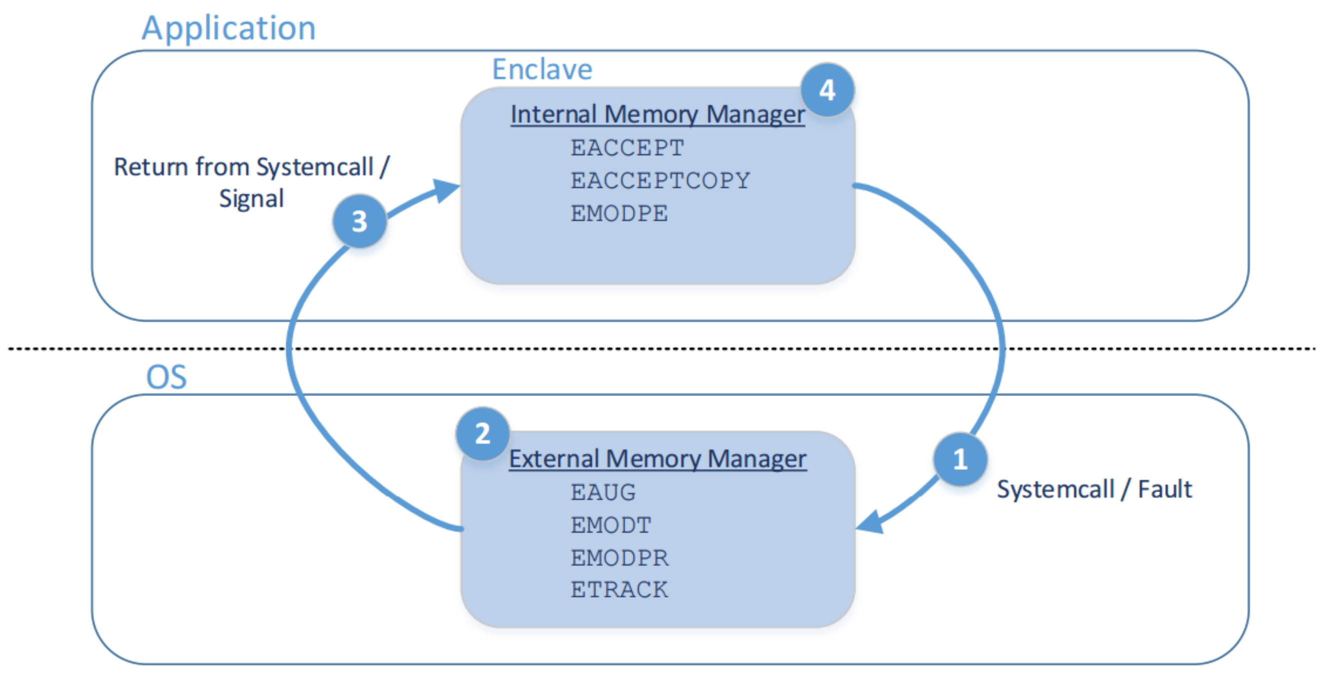


Figure 3-1 SGX2 Software Flow

4.1 SGX2 ISA, ENCLS LEAF FUNCTIONS , EAUG

- EAUG augments the enclave with a page of EPC memory -> associates that page with an SECS page, and updating the linear address and security attributes in the page's EPCM-> puts the page in "Pending" state.
- two input parameters, a pointer to the destination page in EPC, and a pointer to the enclave's SECS page.
- While in "Pending" state, the page cannot be accessed by anyone, including the enclave. Only after the enclave approves the page by using the ENCLU[EACCEPT] the page be accessible to the enclave.

ENCLS LEAF FUNCTIONS , EMODT

- EMODT modifies the type of an EPC page and puts the page in "Modified" state. Allowed page types are PT_TCS and PT_TRIM. The operation receives two input parameters, a pointer to the target page in EPC, and a pointer to the page's new security attributes. While in "Modified" state, the page cannot be accessed by anyone, including the enclave. Only after the enclave approves the page by using the ENCLU[EACCEPT] leaf function, will the page be accessible to the enclave.

ENCLS LEAF FUNCTIONS , EMODPR

- EMODPR This leaf function restricts the access rights associated with an EPC page of an initialized enclave and puts the page in "Permission Restriction" state. The operation receives two input parameters, a pointer to the target page in EPC, and a pointer to the page's new security attributes. The operation will fail if it attempts to extend the permissions of the page. While in "Permission Restriction" state, the page cannot be accessed by anyone, including the enclave. Only after the enclave approves the page by using the ENCLU[EACCEPT] leaf function, will the page be accessible to the enclave.

ENCLU LEAF FUNCTIONS, EACCEPT

- This leaf function must be executed from within an enclave. It accepts changes to a page in the running enclave by verifying that the security attributes specified in SECINFO match the page's security attributes in EPCM. The operation receives two input parameters, a pointer to the target page in EPC, and a pointer to the page's approved new security attributes. After a successful execution of EACCEPT the page's "Pending", "Modified", or "Permission Restriction" state is cleared and the page becomes accessible to the enclave.

ENCLU LEAF FUNCTIONS ,EACCEPTCOPY

- This leaf function must be executed from within an enclave. It copies the contents of an existing EPC page into an uninitialized EPC page that was created by EAUG. The operation receives three input parameters, a pointer to the target page in EPC, a pointer to the page's new security attributes, and a pointer to the page's new content. After a successful execution of EACCEPTCOPY the page's "Pending" state is cleared and the page becomes accessible for the enclave

ENCLU LEAF FUNCTIONS, EMODPE

- This leaf function must be executed from within an enclave. It extends the access rights associated with an existing EPC page in the running enclave. The operation receives two input parameters, a pointer to the target page in EPC, and a pointer to the page's new security attributes. The operation will fail if it attempts to restrict permissions of the page. Since the execution happens from within the enclave, it's trusted and takes effect immediately.

MANAGING PAGE TABLE TRANSLATIONS

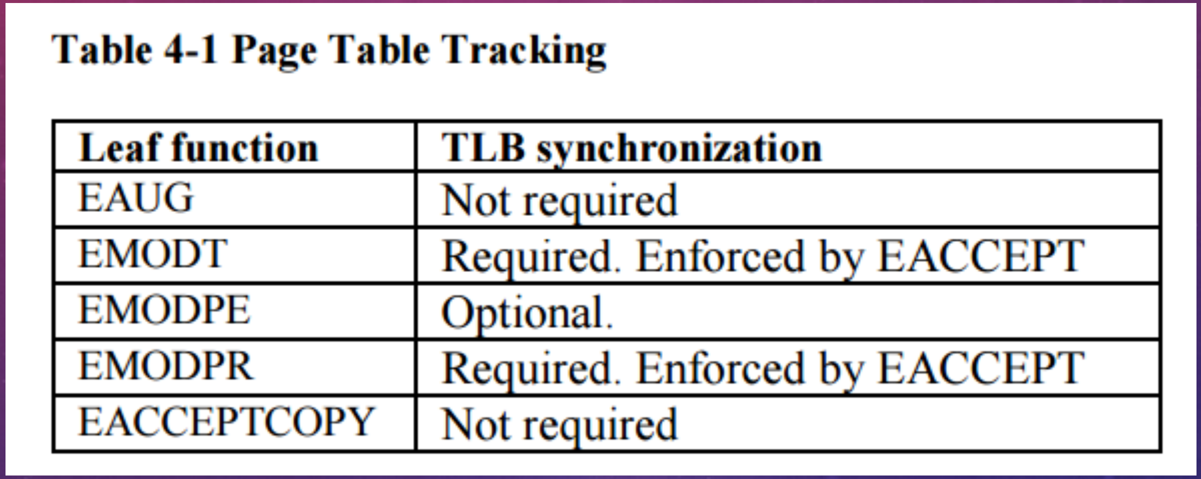


Table 4-1 Page Table Tracking

Leaf function	**TLB synchronization**
EAUG	Not required
EMODT	Required. Enforced by EACCEPT
EMODPE	Optional.
EMODPR	Required. Enforced by EACCEPT
EACCEPTCOPY	Not required

ENCLAVE EXCEPTION HANDLING ENHANCEMENTS

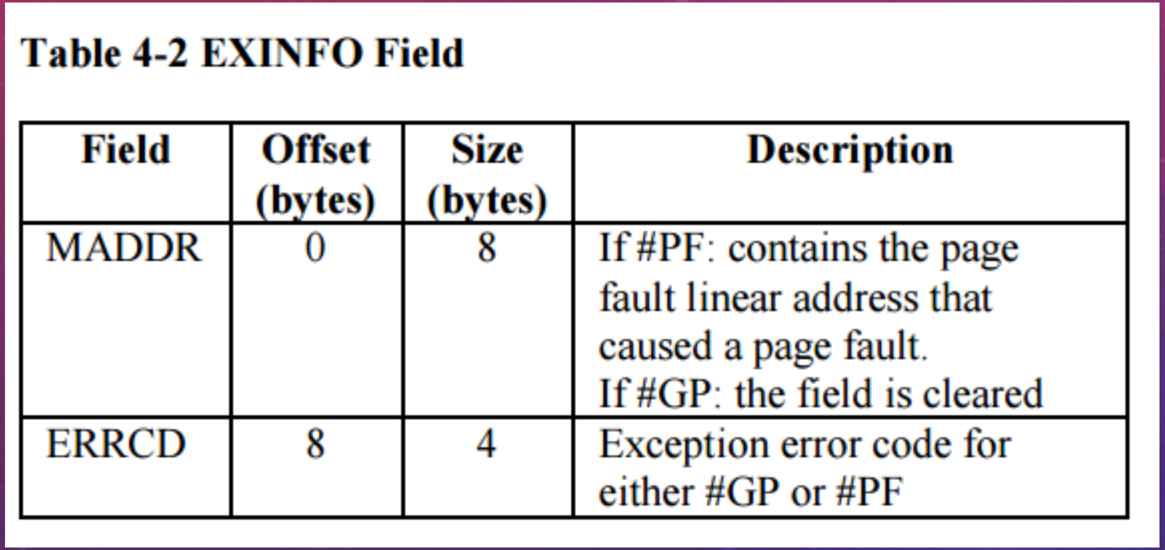


Table 4-2 EXINFO Field

Field	Offset (bytes)	Size (bytes)	Description
MADDR	0	8	If #PF: contains the page fault linear address that caused a page fault. If #GP: the field is cleared
ERRCD	8	4	Exception error code for either #GP or #PF

- the cause of the AEX is stored in the EXITINFO field in the SSA.
- If SECS.MISCSELECT.EXINFO bit is set by enclave writer, the processor saves #PF and #GP information into the EXINFO structure

4.5 EPCM-INDUCED MEMORY FAULT REPORTING

- A #PF exception is generated
- A bit in the Page Fault Error Code (PFEC) indicates that the page fault was due to EPCM access checks. This bit is located at bit position 15 and called “SGX”

SUMMARY AND RELATED WORK

- New instructions to the SGX1 provide better software development environment while maintaining the security of the enclave. The SGX2 instructions enable better protection of proprietary code which can be loaded and then protected using the EPCM.
- Allow for dynamic memory and threading support
- Support dynamic allocation of library pages in the library OS environment.

End of Presentation