Intel® Software Guard Extensions (Intel® SGX) SGX2

Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, Carlos Rozas, Mark Shanahan, Bin (Cedric) Xing

June 18, 2016

Copyright © Intel Corporation 2016

Legal Disclaimers

- The comments and statements are the presenter's and not necessarily Intel's

- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

- No computer system can be absolutely secure.

Agenda

Agenda:

- SGX Review
- SGX2 Motivation and Usage
- Software Flows
- Instruction Set and Architectural Features
- SDK Support

SGX Review

Reduced attack surface with SGX

Application gains ability to defend its own secrets

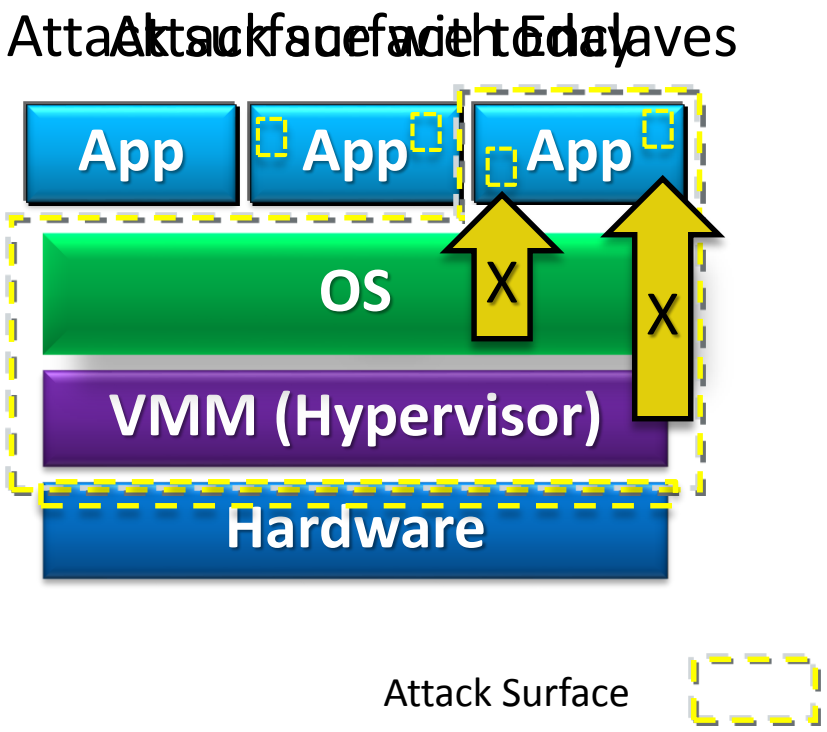
- Smallest attack surface (app + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets

Familiar development/debug

- Single application environment
- Build on existing ecosystem expertise

Familiar deployment model

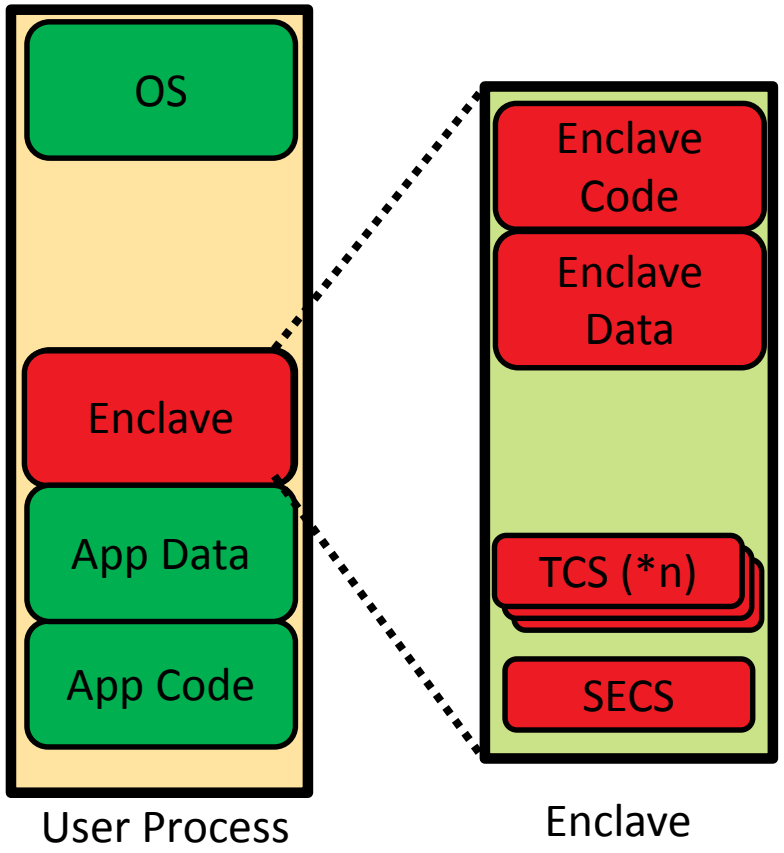
- Platform integration not a bottleneck to deployment of trusted apps



Scalable security within mainstream environment

What is an Enclave?

Enclave: trusted execution environment embedded in a process



- Own code and data
- Provides Confidentiality
- Provides integrity
- Controlled entry points
- Supporting multiple threads
- Full access to app memory

SGX2: Enclave Dynamic Memory Management (EDMM)

Enclave Dynamic Memory Management (EDMM) Motivation

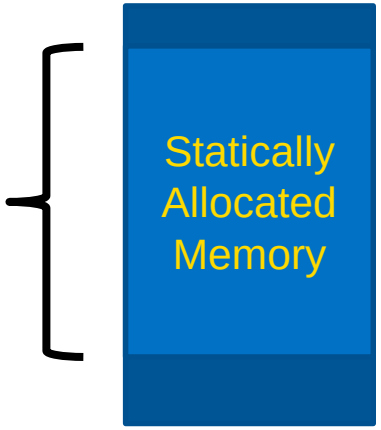
In SGX1, an enclave's set of committed pages is fixed at load time

- All memory that enclave might ever use must be added before initialization

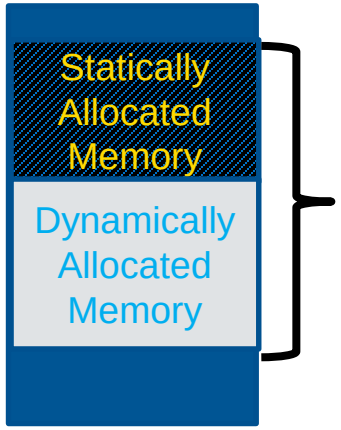
EDMM instructions allow run-time changes to enclave memory while maintaining the SE security properties

- Reduces enclave build time and enables new usages
- Build a minimal enclave with minimal memory
- Expand to meet platform requirements

OS must allocate all enclave memory prior to enclave initialization



Enclave Setup (SGX1)



Memory may be allocated, modified, and deallocated at run-time

Enclave Setup (Enclave Malloc)

Example Software Usages

Heap and thread-pool management

On-demand stack growth

Dynamic module/library loading

Concurrency management in applications such as garbage collectors

Write-protection of EPC pages after initial relocation

On-demand creation of code pages (JIT, encrypted code modules)

Application Memory Management with EDMM

Memory management operations:

- Page allocation and deallocation
- Thread (TCS) creation/destruction
- Page permissions modification

Cooperative process between the OS and the enclave memory manager

- OS performs memory management on enclave’s behalf
- Enclave confirms request before changes take effect

Enclave memory manager hides complexity from application code

Security Considerations

SGX guarantees the integrity of enclave code and data pages

Integrity of static enclave memory is provided by measurement

Dynamic memory management requires a new approach

- OS must be involved in page-table management, TLB flushes, …
- OS cannot be trusted not to violate SE security guarantees
  - Example: OS converts a read-only page to a read/write page

Solution: Enclaves oversee every dynamic memory management transaction

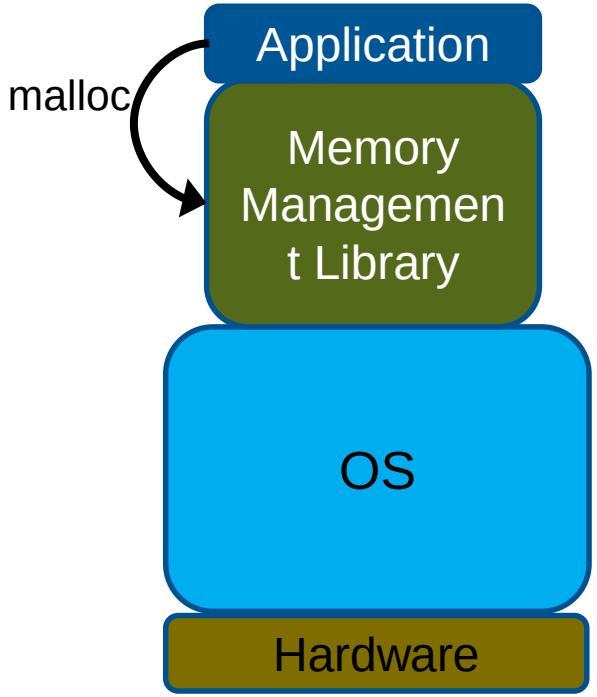
- Enclave requests OS to perform memory management operation
- OS performs operation
- Enclave accepts the result, testing for unexpected behavior

Software Flows

Heap Extension in Non-Enclave Applications

Many software applications rely on a memory management library, such as the C runtime, to manage the application's heap

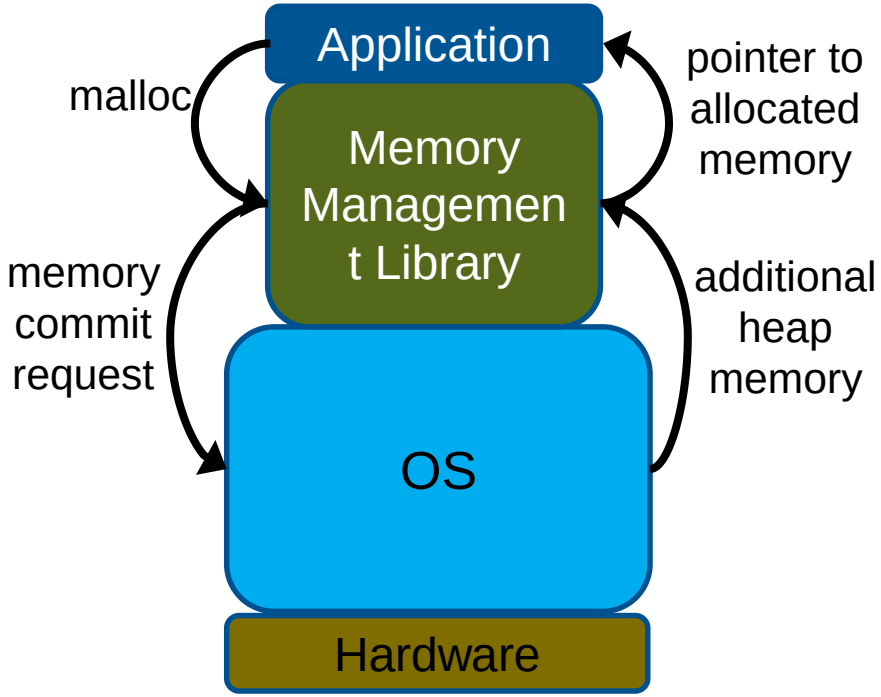
- Applications request memory through the memory manager's API
  - Example: malloc



Heap Extension in Non-Enclave Applications

Memory management libraries utilize operating system facilities to obtain committed memory to back the application heap

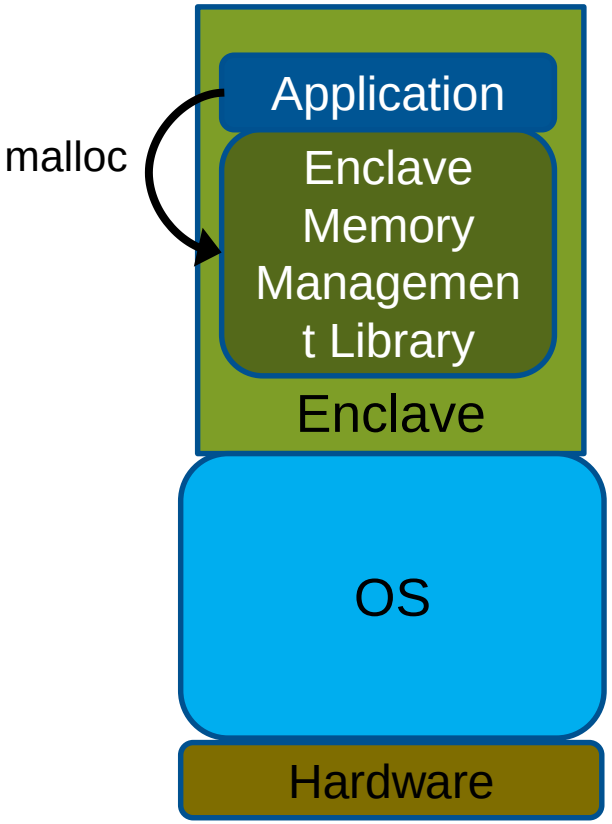
- Memory manager maintains a set of free pages that is used to fulfill heap allocation requests
- When the manager exhausts its supply of free pages, it requests additional memory through the OS API
  - Example: sbrk
  - May be committed lazily or on demand



Heap Extension in Enclave Applications

Enclaves contain their own memory management library for managing the heap in enclave applications

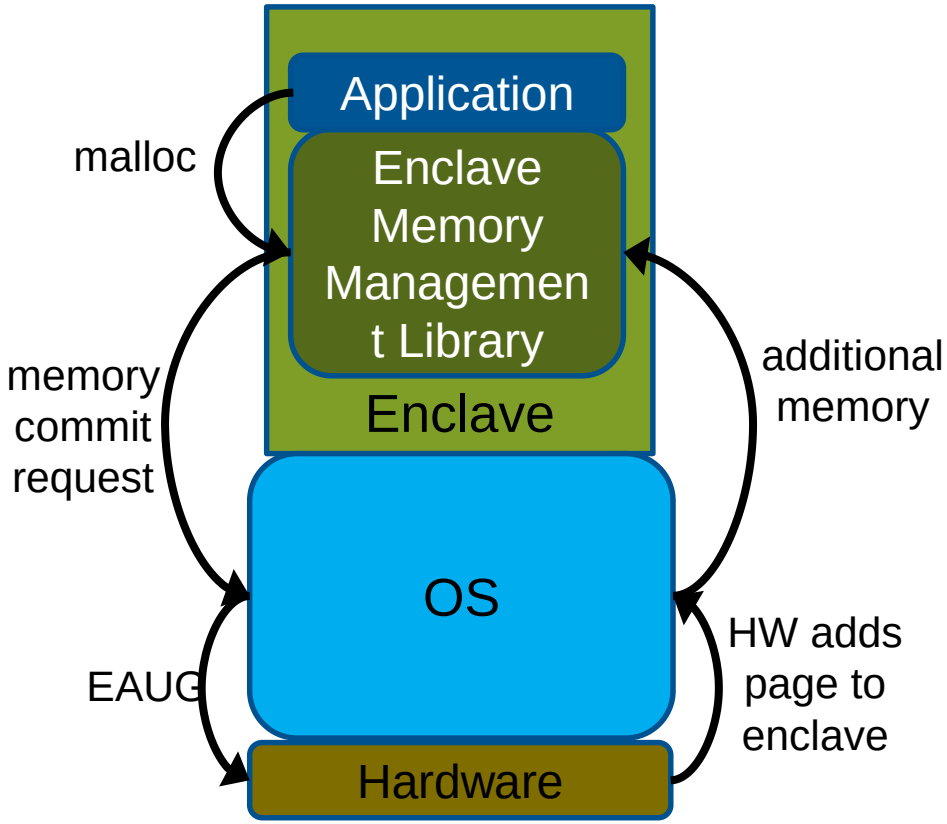
- As before, applications request memory through the memory manager's API
- Enclave memory manager API will likely be similar to existing memory management libraries
  - Example: malloc



Heap Extension in Enclave Applications

Enclave memory manager utilizes enclave-specific OS APIs to obtain committed EPC memory to back the application heap

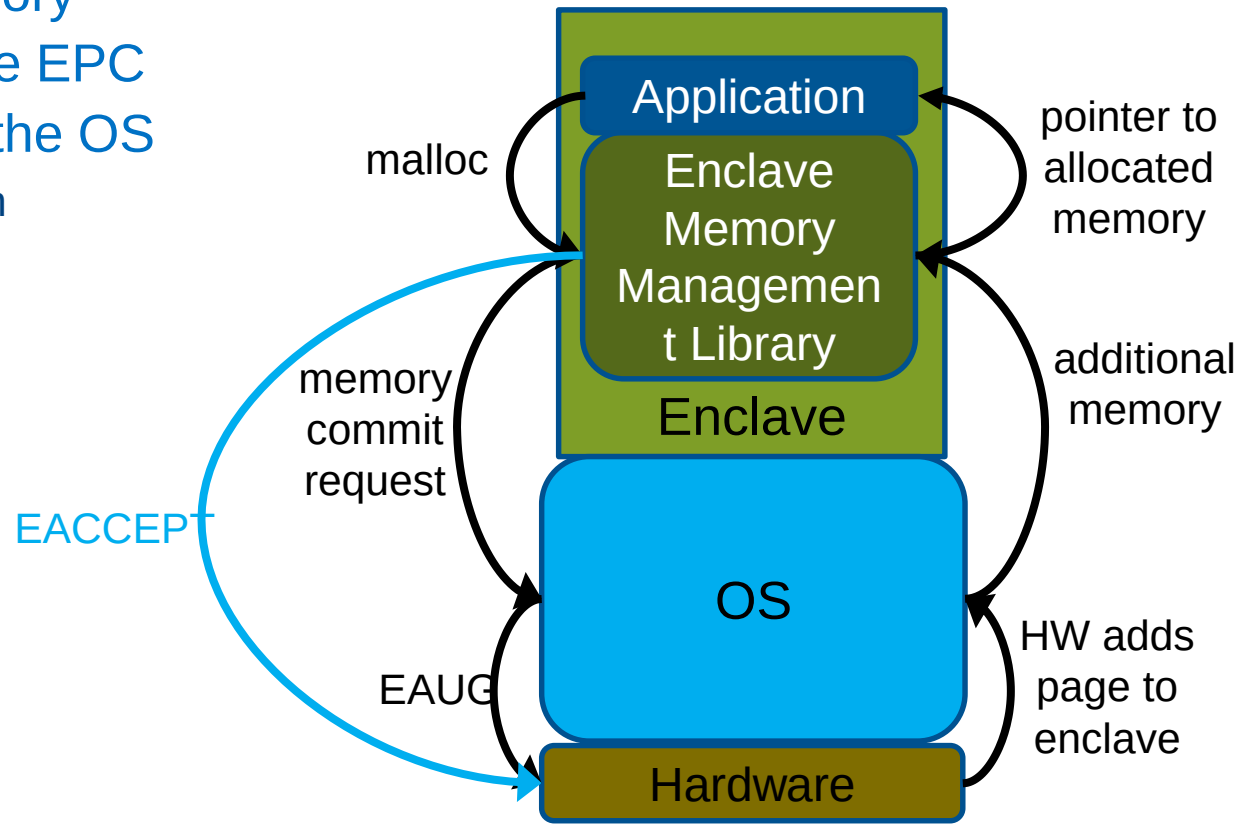
- As before, memory manager maintains a set of free pages that is used to fulfill heap allocation requests
- When the manager exhausts its supply of free pages, it requests additional memory through the OS API
  - May be committed lazily or on demand
- OS implementation uses SGX2 instruction EAUG to add EPC pages to the enclave



Heap Extension in Enclave Applications

Before accessing newly committed pages, the enclave memory manager must accept the EPC allocation performed by the OS

- Call to EACCEPT instruction



Instruction Set Architecture

Data Structures

EPCM

- Per-page meta-data information introduced in SGX1
- Structure extended to support PENDING, MODIFIED, and Permission Restriction states
  - May not be set simultaneously

SECINFO

- 64B data structure introduced in SGX1
- Structure extended to support and PENDING, MODIFIED, and Permission Restriction (PR) bits
  - Bit 3: PENDING
  - Bit 4: MODIFIED
  - Bit 5: PR

PAGE_TYPE

- Per-EPC page type information introduced in SGX1
- Datatype extended to include a new value for deallocated pages (PT_TRIM)
  - PT_TRIM encoding = 4

Page Type: PT_TRIM

Trimmed pages

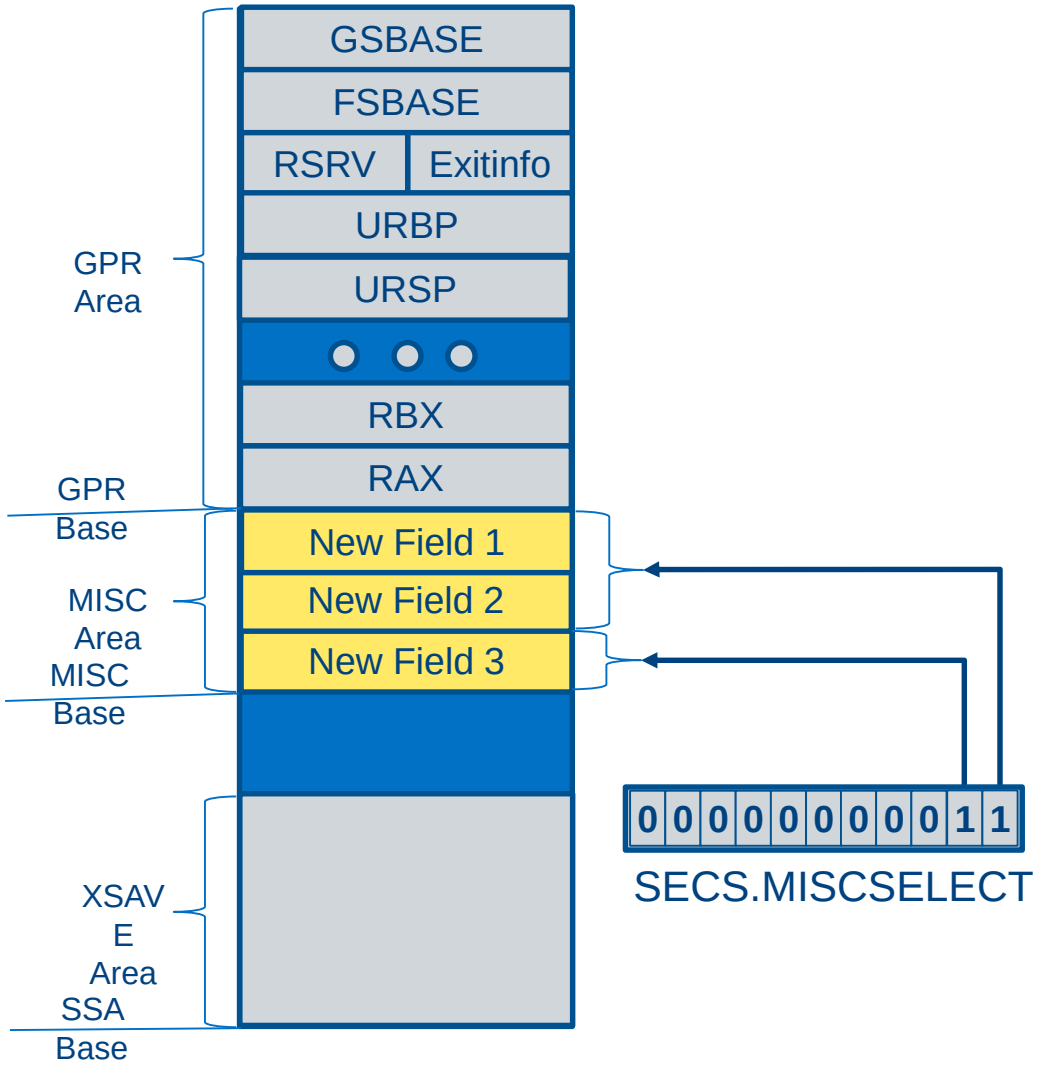
- Trimmed state indicate that a page is in the process of being deallocated
- Trimmed pages are not accessible to software

Trimming

- System Manager executes EMODT which sets type to PT_TRIM and set modified state
- Enclave Manager executes EACCEPT which clears modified bit
- System Manager executes EREMOVE which marks the page invalid.

MISC Area Review

- MISC Area part of SSA Frame
- MISC Area fields supported by CPU are enumerated by CPUID.SE_LEAF.0.EBX
- Enclave writer opt-in to use of supported MISC Area fields using new dedicated MISCSELECT field in SECS
  - Similarly to opting-in for use of XSAVE features
  - Frame size calculation includes new MISC area
- Effective value of MISCSELECT becomes attestable property of enclave
  - It stirred into enclave keys
  - It becomes part of enclave REPORT



SGX2 Information and Error Codes

MISC Components	Offset (Bytes)	Size (bytes)	Description
EXINFO	base(GPRSGX)-16	16	if CPUID.(EAX=12H, ECX=0):EBX[0] = 1, exception information on #GP or #PF that occurred inside an enclave can be written to the EXINFO structure if specified by SECS.MISCSELECT[0] = 1.
Future Extension	Below EXINFO	TBD	Reserved. (Zero size if CPUID.(EAX=12H, ECX =0:EBX[31:1] = 0)

SGX2 MISC Field

PFINFO Field		
Name	Offset	Size
MADDR	0	8
ERRCD	8	4
RESERVED	12	4

ERRCD Field

PFINFO.ERRCD Field									
Bit Layout									
31	**16**	15	14	6	5	4	3	2	1	0
Zero		SGX	Zero		PK	I/D	RSVD	1*	WR	P

ENCLS leafs

ENCLS[EAUG]

- Allocates and zeroes a PT_REG page with read/write permissions
- Put page in Pending stage to prevent application access to page
- Requires call to EACCEPT to make page available to enclave

ENCLS[EMODT]

- Changes page type from PT_REG to PT_TCS or PT_TRIM
- Puts page into Modified state to prevent access to the page
- Requires call to EACCEPT to make page available to enclave

ENCLS[EMODPR]

- Restricts the access rights of the page
- Puts page into Permission Restricted State
- Page remains accessible inside enclave
- Requires call to EACCEPT to guarantee the change across all processors

ENCLU Leafs

ENCLU[EACCEPT]

- Checks that page performed correctly (allocation, deallocation, permissions change, etc.)
- Verifies that all TLB mappings to target page have been flushed
- Uses the epoch-tracking mechanism introduced by SGX1
- Clears PENDING, MODIFIED, or Permission Restricted states

ENCLU[EACCEPTCOPY]

- Copies contents of an enclave page into a page allocated by EAUG.
- Uses SECINFO field inside enclave to specify the destination type and permissions
- Clears PENDING State

ENCLU[EMODPE]

- Extends the EPCM permissions associated with a regular page
- Page remains accessible to enclave with original permissions
- TLB shoot-down not necessary to gain access to extended permissions
- Requires call to OS to extend the page-table permissions

SDK Support

A Little Background on the SGX Run-Time System

Kernel Module (SGX Driver)

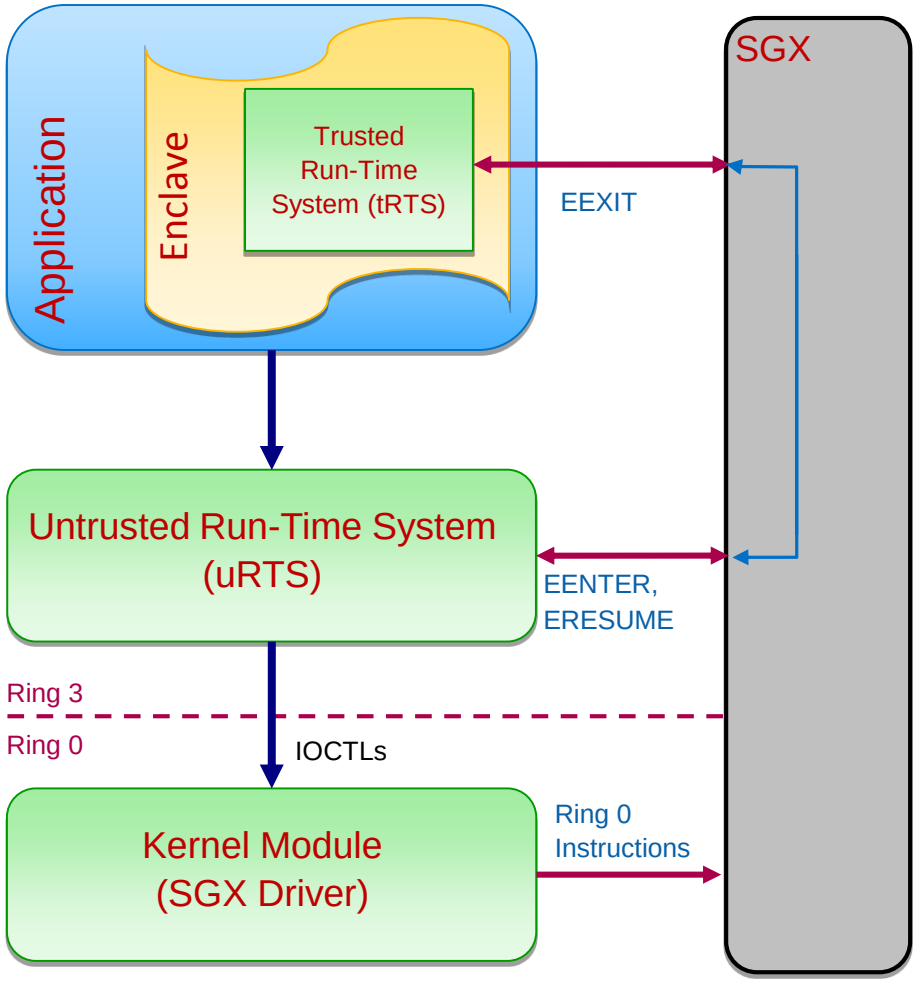
- Manages EPC Physical Memory
  - Enclave Loading
  - Memory Allocation/Deallocation
  - Page Swapping / Page Faults

Untrusted Run-Time System (uRTS)

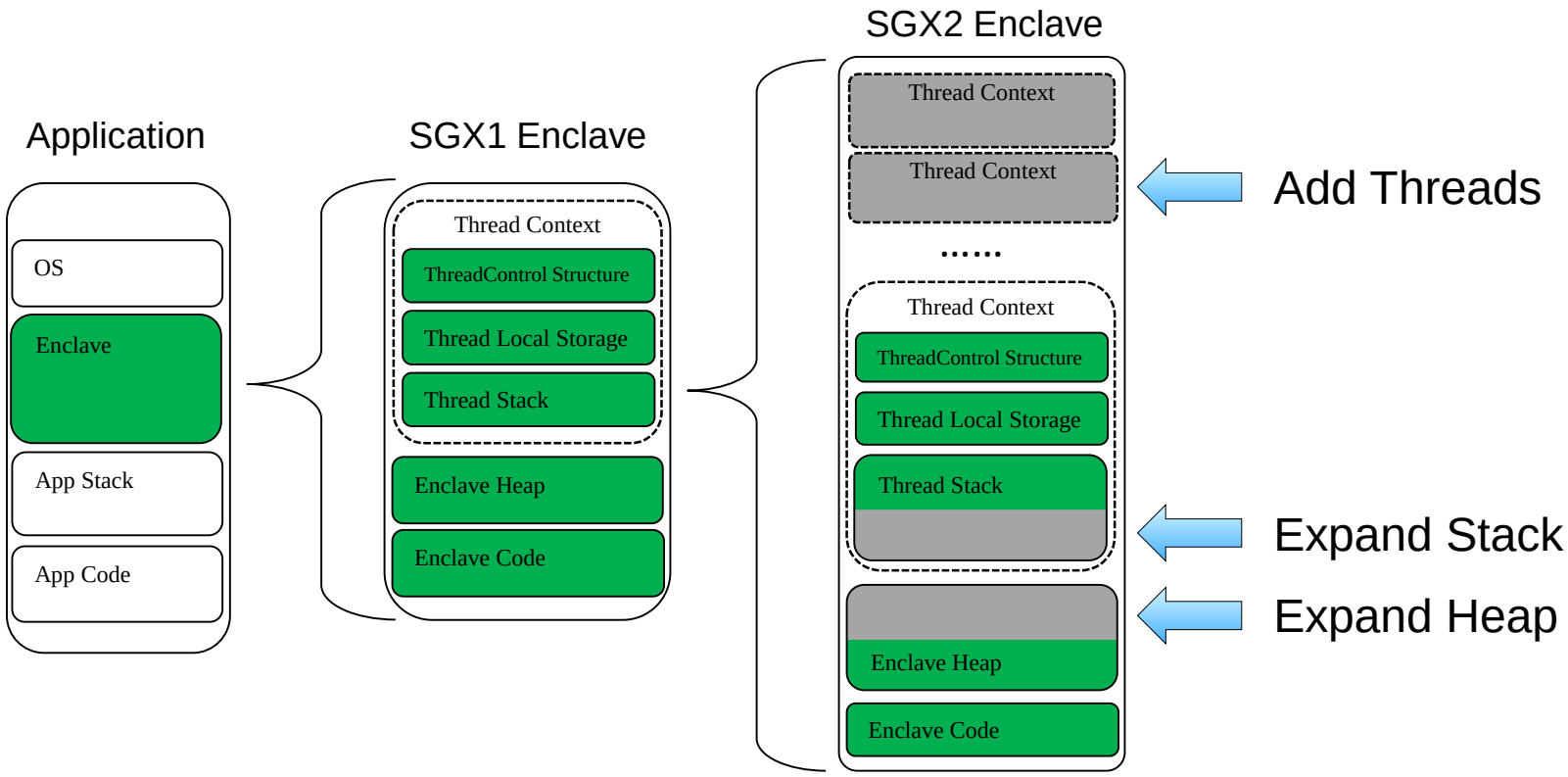
- Loading and Unloading of Enclaves
  - Sends IOCTLs to SGX Driver
- Call Management: Handle all calls to/from the enclave
  - ECall : Call from Application to the Enclave
  - OCall : Call from Enclave to the Application
- Exception Handling: Call back into the enclave

Trusted Run-Time System (tRTS)

- Enclave Loading: Complete the trusted enclave load
- Call Management: Configure ECalls/OCalls
- Exception Handling : Enclave generated exceptions



Putting SGX2 Instructions to Use



Adapt to varying workloads with Dynamic EPC Allocation:

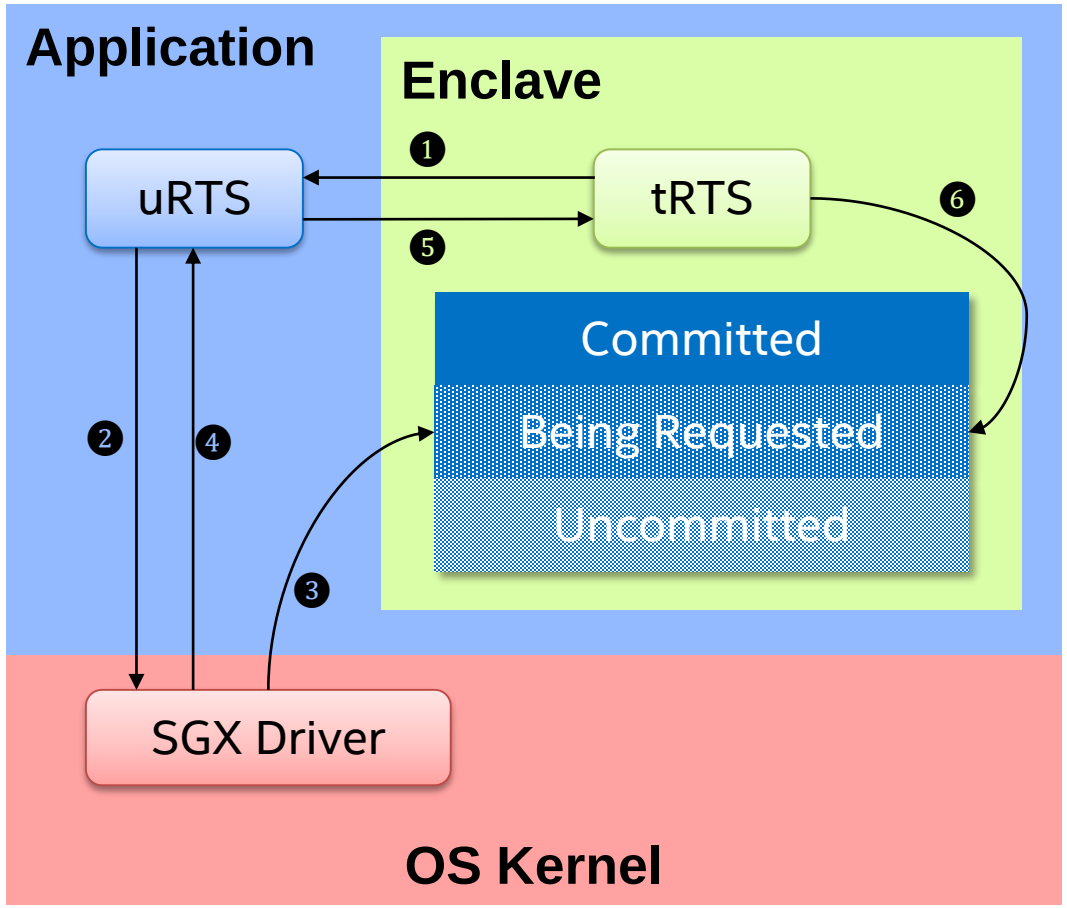
- Basic Method: OCall Based
- More Efficient Method: Page Fault Based

OCall Based EPC Allocation

Trusted Run Time System (tRTS) makes a call out of the Enclave to the Untrusted Run-Time System (uRTS) in the Application to request a range of EPC pages:

Begin with pre-defined regions in enclave memory.

1. tRTS makes OCall to uRTS with base/size of requested range
2. uRTS sends IOCTL to SGX Driver with base/size
3. SGX Driver uses ENCLS[EAUG] to commit requested pages
4. IOCTL returns to uRTS
5. OCall returns to tRTS
6. tRTS uses ENCLU[EACCEPT] to accept newly added EPC pages

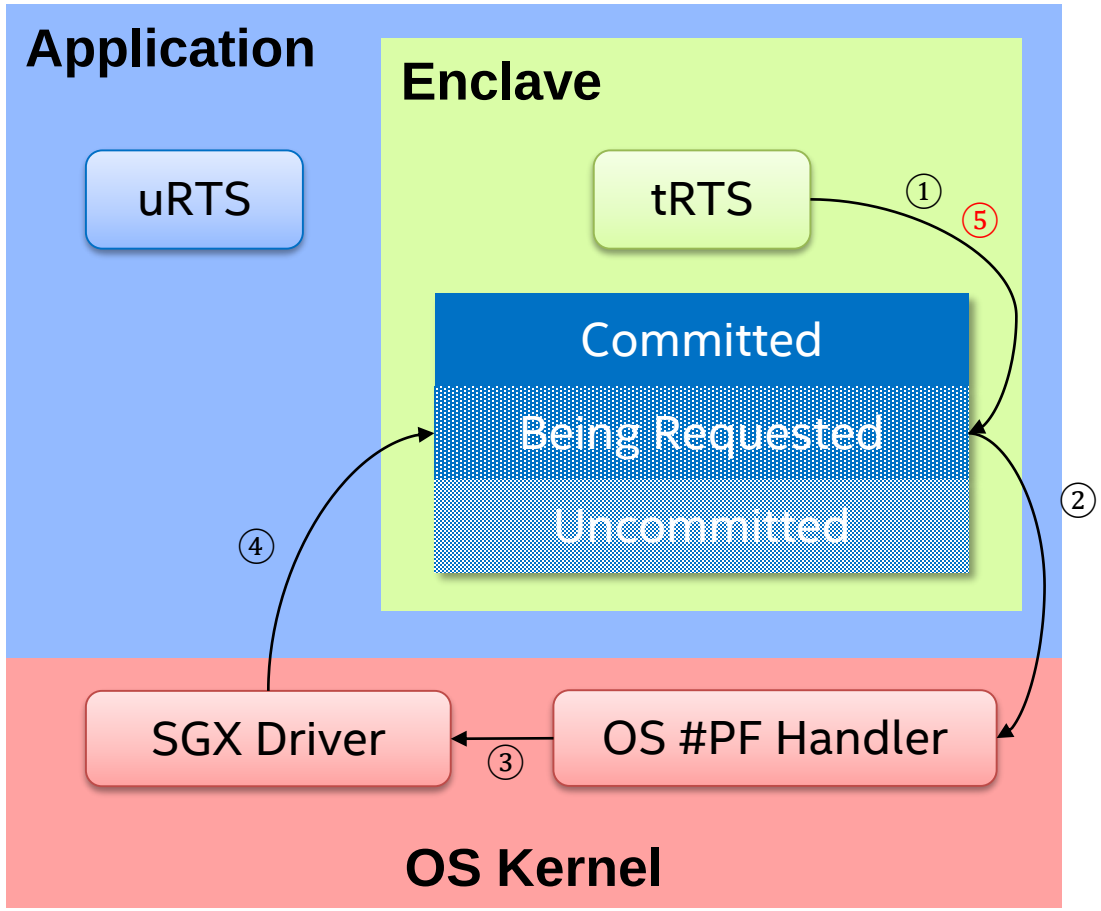


Page Fault Based EPC Allocation

Trusted Run-Time System issues ENCLU[EACCEPT] to initiate EPC page allocation.

Begin with pre-defined regions in memory which the tRTS and SGX Driver know about.

1. tRTS invokes ENCLU[EACCEPT] where a new page is requested
2. Page Fault (#PF) results
3. OS #PF handler invokes SGX Driver to handle the exception
4. Driver issues ENCLS[EAUG] on requested pages
5. ENCLU[EACCEPT] is retried and succeeds



Look in The Paper* for Details on …

A new construct (i.e. Dynamic Region) to improve performance of #PF based EPC allocation

#PF based EPC allocation in action

- Dynamic Heap Expansion
- Dynamic Stack Expansion
- Dynamic Thread Context Creation

Future considerations

- Dynamic Code Loading
- Page Attribute Modifications

**Intel® Software Guard Extensions (Intel® SGX) Software Support for Dynamic Memory Allocation inside an Enclave*

Backup