Intel® Software Guard Extensions(Intel® SGX)

Carlos Rozas
Intel Labs

November 6, 2013

Legal Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

No computer system can provide absolute security under all conditions. Built-in security features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

Intel®, the Intel® Logo, Intel® Inside, Intel® Core™ , Intel® Atom™, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and/or other countries. Other names and brands may be claimed as the property of others.

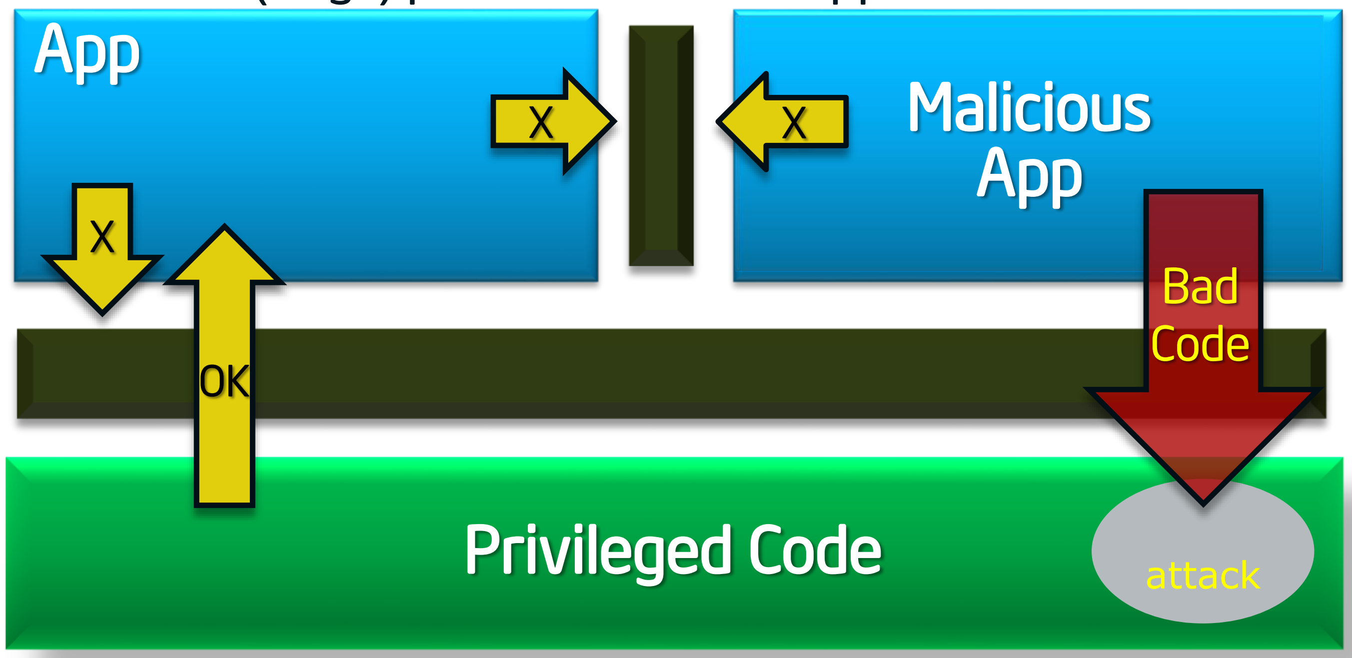
Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

Outline

- Problem Statement
- Attack Surface and Overview
- Programming environment
  - System programming view
  - Day in the life of an enclave
- SGX Access Control & Off Chip protections
- Attestation and Sealing
- Developing with SGX
- Summary

The Basic Issue: Why Aren't Compute Devices Trustworthy?

Protected Mode (rings) protects OS from apps …



… and 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 not protected from privileged code attacks

Reduced attack surface with SGX

Application gains ability to defend its own secrets

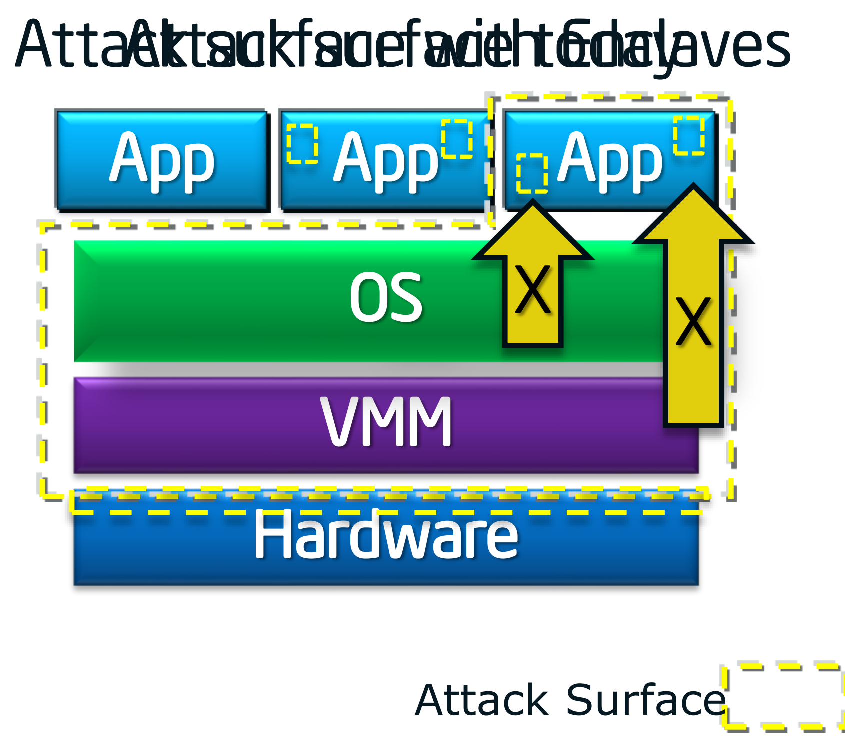
- Smallest attack surface (App + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets

Familiar development/debug

- Single application environment
- Build on existing ecosystem expertise

Familiar deployment model

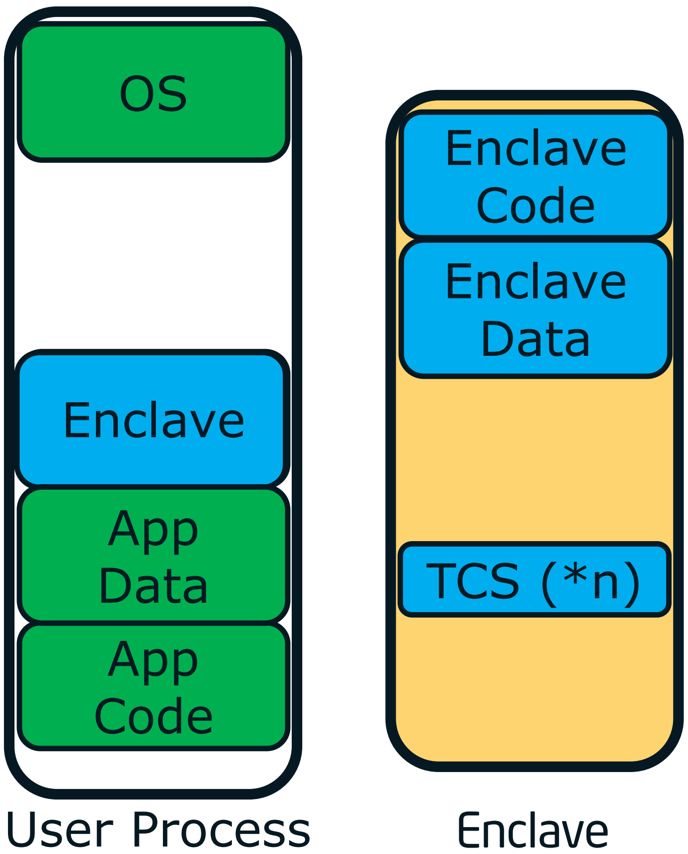
- Platform integration not a bottleneck to deployment of trusted apps



Scalable security within mainstream environment

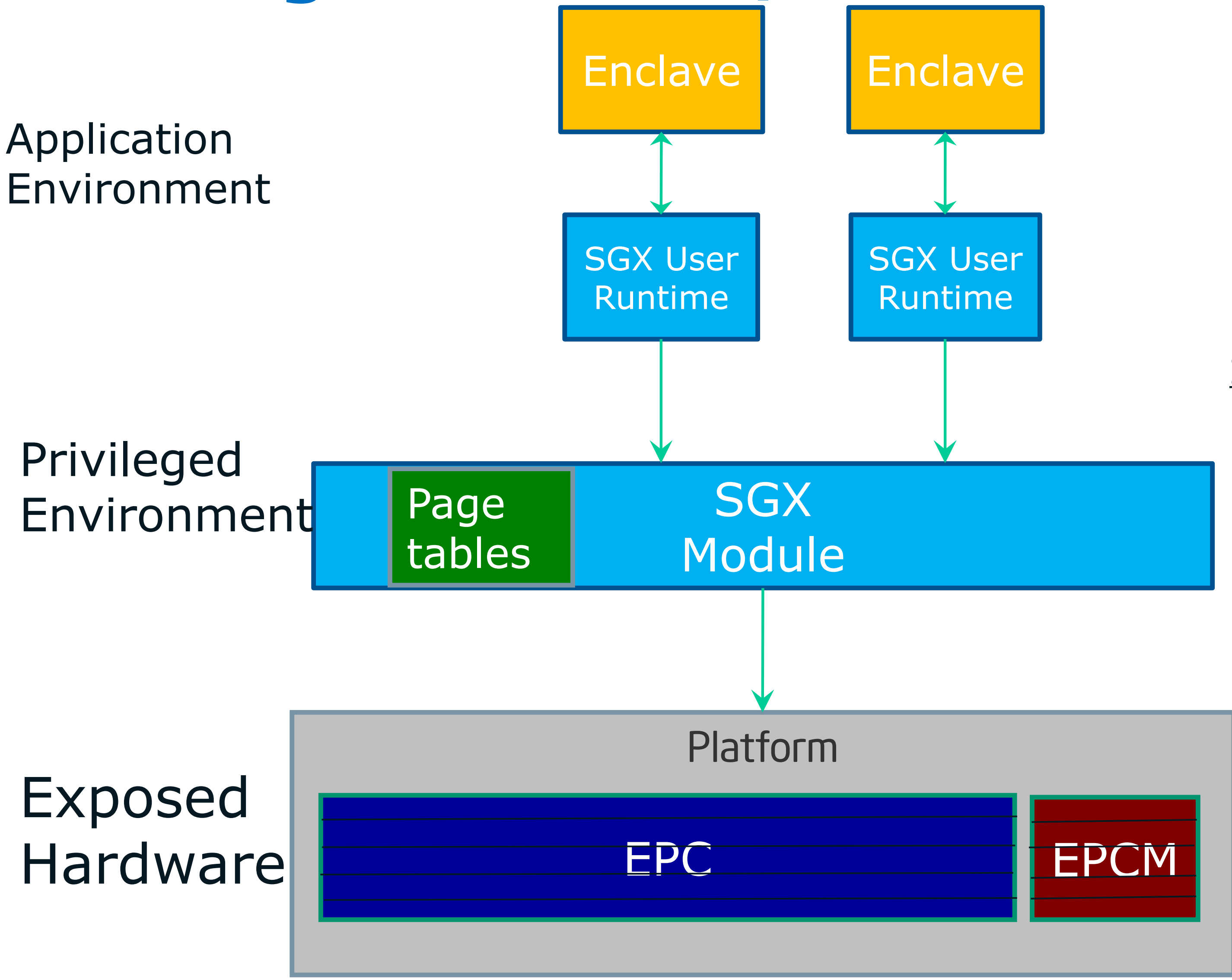
SGX Programming Environment

Trusted execution environment embedded in a process



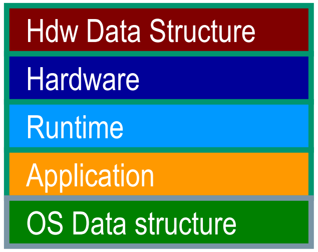
- With its own code and data
- Provide Confidentiality
- Provide integrity
- With controlled entry points
- Supporting multiple threads
- With full access to app memory

SGX High-level HW/SW Picture



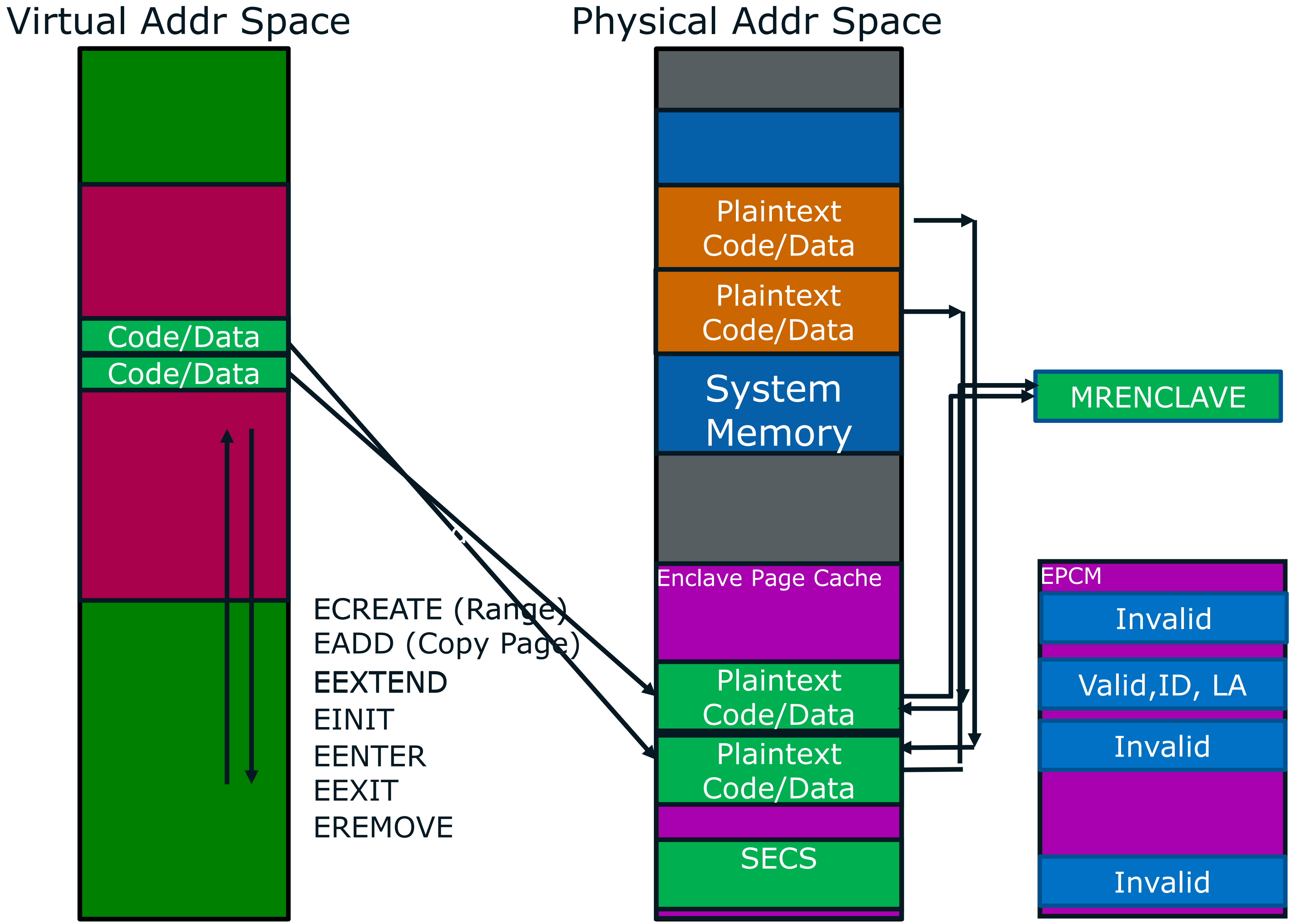
Instructions
EEXIT
EGETKEY
EREPORT
EENTER
ERESUME

Instructions
ECREATE
EADD
EEXTEND
EINIT
EBLOCK
ETRACK
EWB
ELD
EPA
EREMOVE

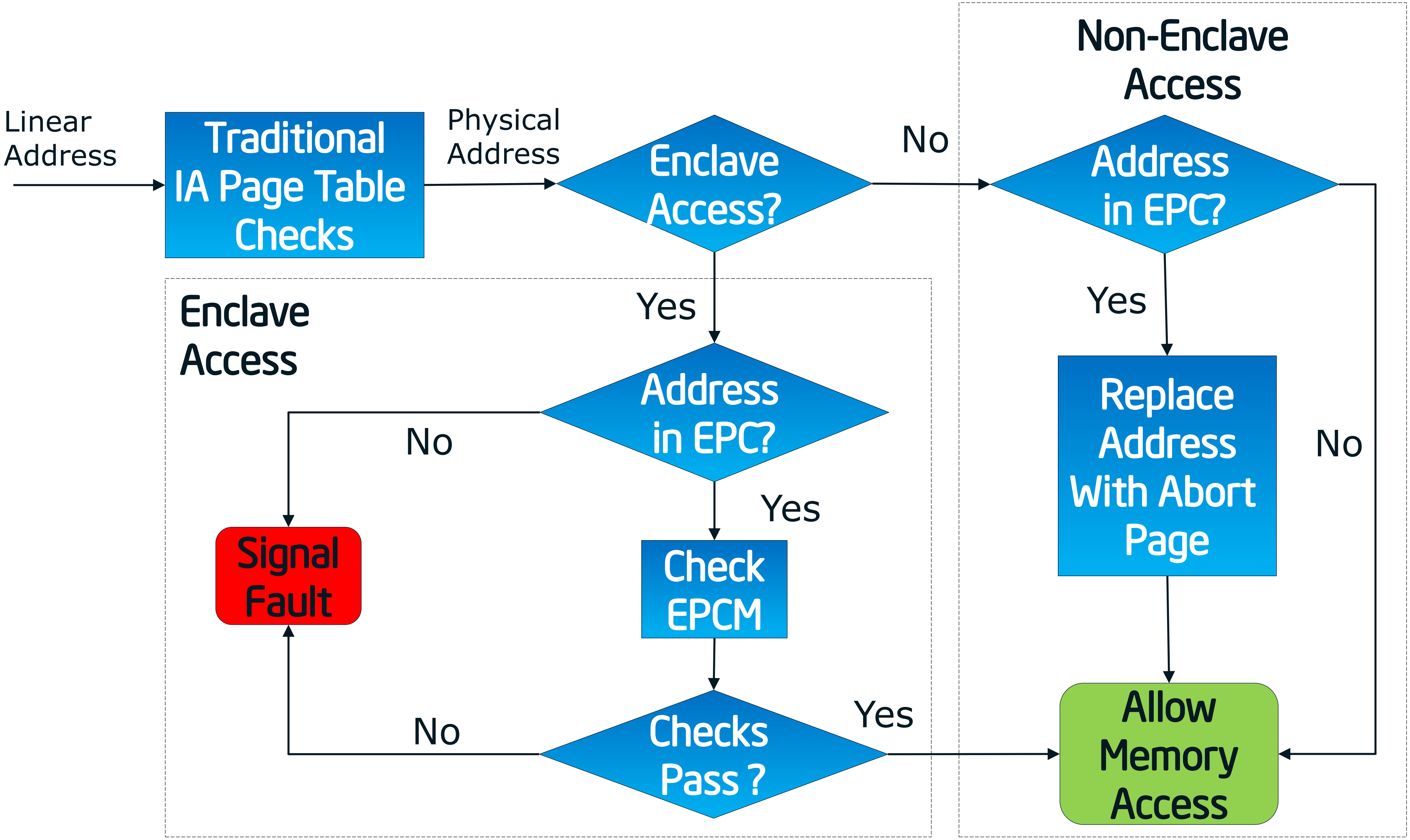


Life Cycle of An Enclave

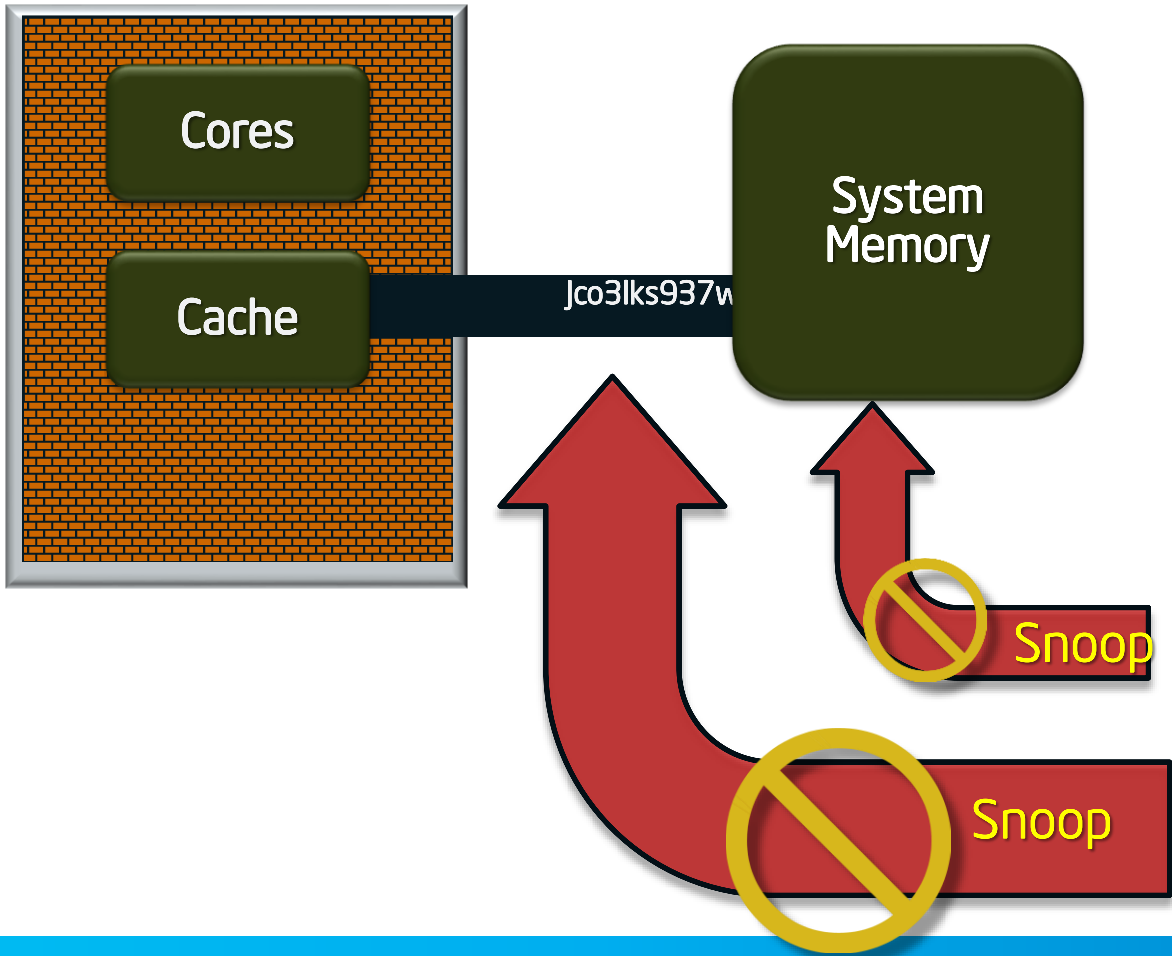
Build



SGX Access Control



Protection vs. Memory Snooping Attacks



Non-Enclave Access

1. Security perimeter is the CPU package boundary
2. Data and code unencrypted inside CPU package
3. Data and code outside CPU package is encrypted and/or integrity checked
4. External memory reads and bus snoops see only encrypted data

Outline

- Problem Statement
- Attack Surface and Overview
- Programming environment
  - System programming view
  - Day in the life of an enclave
- SGX Access Control & Off Chip protections
- **Attestation and Sealing**
- **Developing with SGX**
- **Summary**

The Challenge – Provisioning Secrets to the Enclave

- An enclave is in the clear before instantiation
  - Sections of code and data could be encrypted, but their decryption key can't be pre-installed
- Secrets come from outside the enclave
  - Keys
  - Passwords
  - Sensitive data
- The enclave must be able to convince a 3rd party that it's trustworthy and can be provisioned with the secrets
- Subsequent runs should be able to use the secrets that have already been provisioned

Trustworthiness

- A service provider should vet the enclave's Trusted Computing Base (TCB) before it should trust it and provide secrets to it
  - The enclave's software
  - The CPU's hardware & firmware

- Intel® SGX provides the means for an enclave to securely prove to a 3rd party:
  - What software is running inside the enclave
  - Which execution environment the enclave is running at
  - Which Sealing Identity will be used by the enclave
  - What's the CPU's security level

Attestation – Software TCB

- When building an enclave, Intel® 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
- A software TCB verifier should:
  - Securely obtain the enclave’s software TCB
  - Securely obtain the expected enclave’s software TCB
  - Compare the two values

Local Attestation

- "Local attestation": The process by which one enclave attests its TCB to another enclave on the same platform
- Using Intel® SGX's *EREPORT* and *EGETKEY* instructions
  - EREPORT generates a cryptographic REPORT that binds MRENCLAVE to the target enclave's REPORT KEY
  - EGETKEY provides the REPORT KEY to verify the REPORT

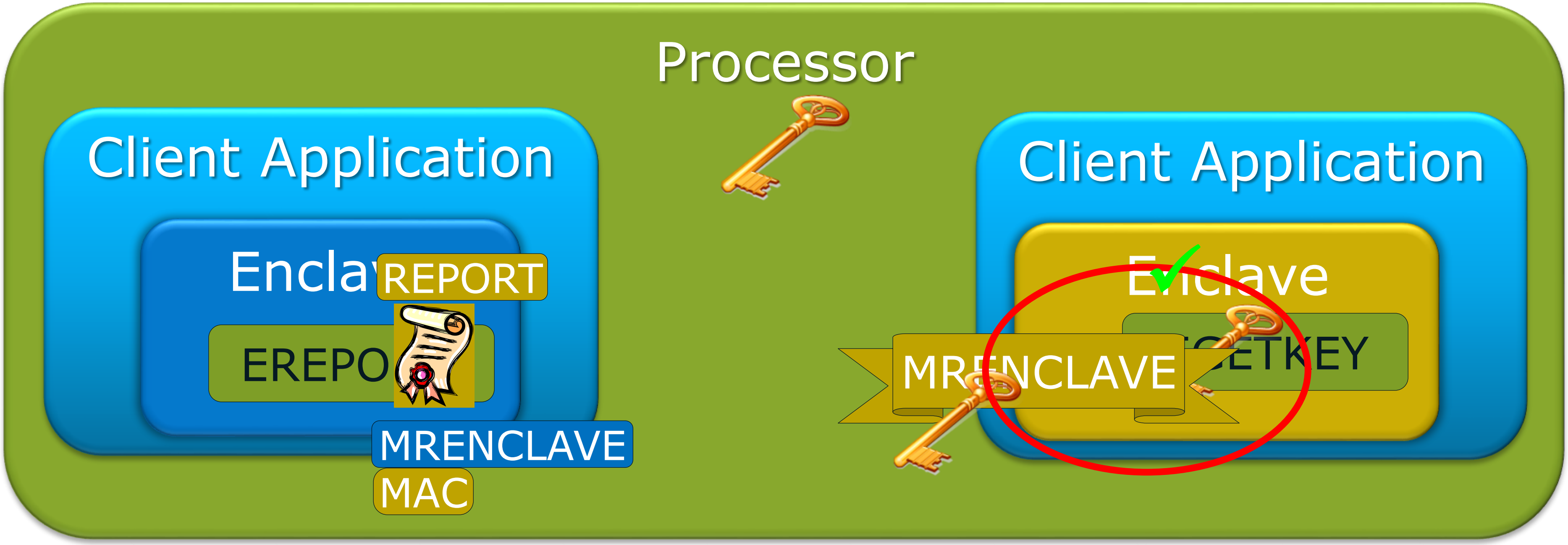
TCB component	Attestation
CPU Firmware & hardware	Symmetric - CPU REPORT KEY
Software	MRENCLAVE

Remote Attestation

- “Remote attestation”: The process by which one enclave attests its TCB to another entity outside of the platform
- Intel® SGX Extends Local attestation by allowing a Quoting Enclave (QE) to use Intel® EPID to create a QUOTE out of a REPORT
  - Intel® EPID is a group signature scheme

TCB component	Attestation
CPU Firmware & hardware	Asymmetric - Intel® EPID
Software	MRENCLAVE

Local Attestation - Flow



1. Verifying enclave sends its MRENCLAVE to reporting enclave
2. Reporting enclave creates a cryptographic REPORT that includes its MRENCLAVE
3. Verifying enclave obtains its REPORT key and verifies the authenticity of the REPORT

Remote Attestation - Flow

1. Verifying enclave becomes the Quoting Enclave.
2. After verifying the REPORT the, QE signs the REPORT with the EPID private key and converts it into a QUOTE
3. Remote platform verifies the QUOTE with the EPID public key and verifies MRENCLAVE against the expected value

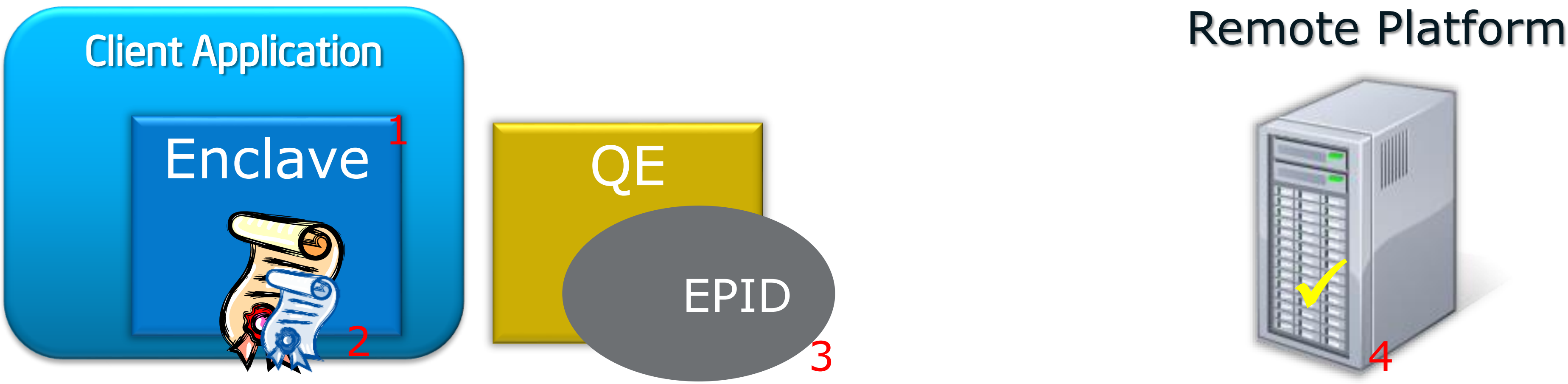
Sealing Authority

- Every enclave has an Enclave Certificate (SIGSTRUCT) which is signed by a Sealing Authority
  - Typically the enclave writer
  - SIGSTRUCT includes:
    - Enclave's Identity (represented by MRENCLAVE)
    - Sealing Authority's public key (represented by MRSIGNER)
- *EINIT* verifies the signature over SIGSTRUCT prior to enclave initialization

Sealing

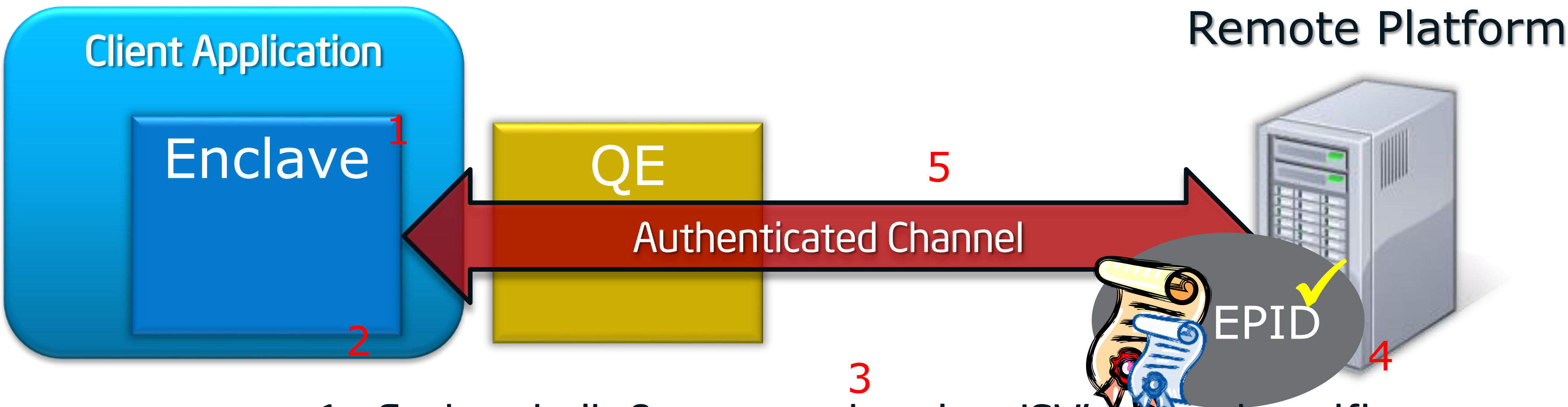
- “Sealing”: Cryptographically protecting data when it leaves the enclave.
- Enclaves use EGETKEY to retrieve an enclave, platform persistent key and encrypts the data
- EGETKEY uses a combination of enclave attributes and platform unique key to generate keys
  - Enclave Sealing Authority
  - Enclave Product ID
  - Enclave Product Security Version Number (SVN)

Example: Secure Transaction



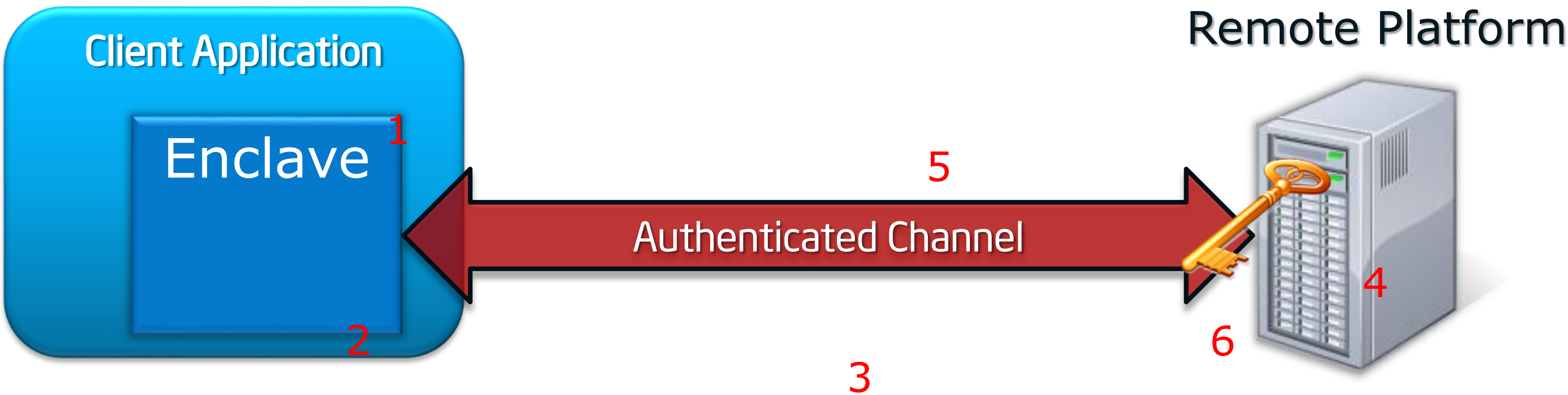
1. Enclave built & measured against ISV's signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls *EGETKEY* to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use

Example: Secure Transaction



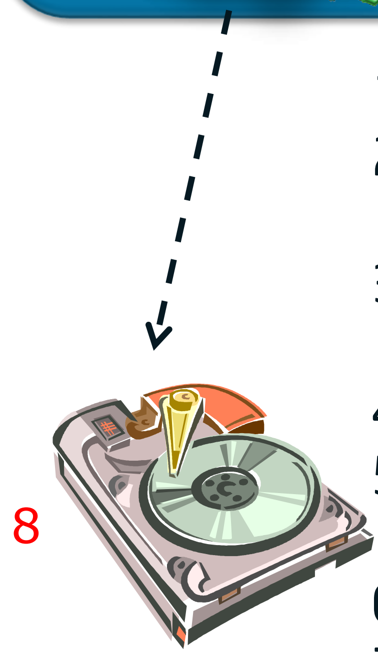
1. Enclave built & measured against ISV's signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls *EGETKEY* to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use

Example: Secure Transaction



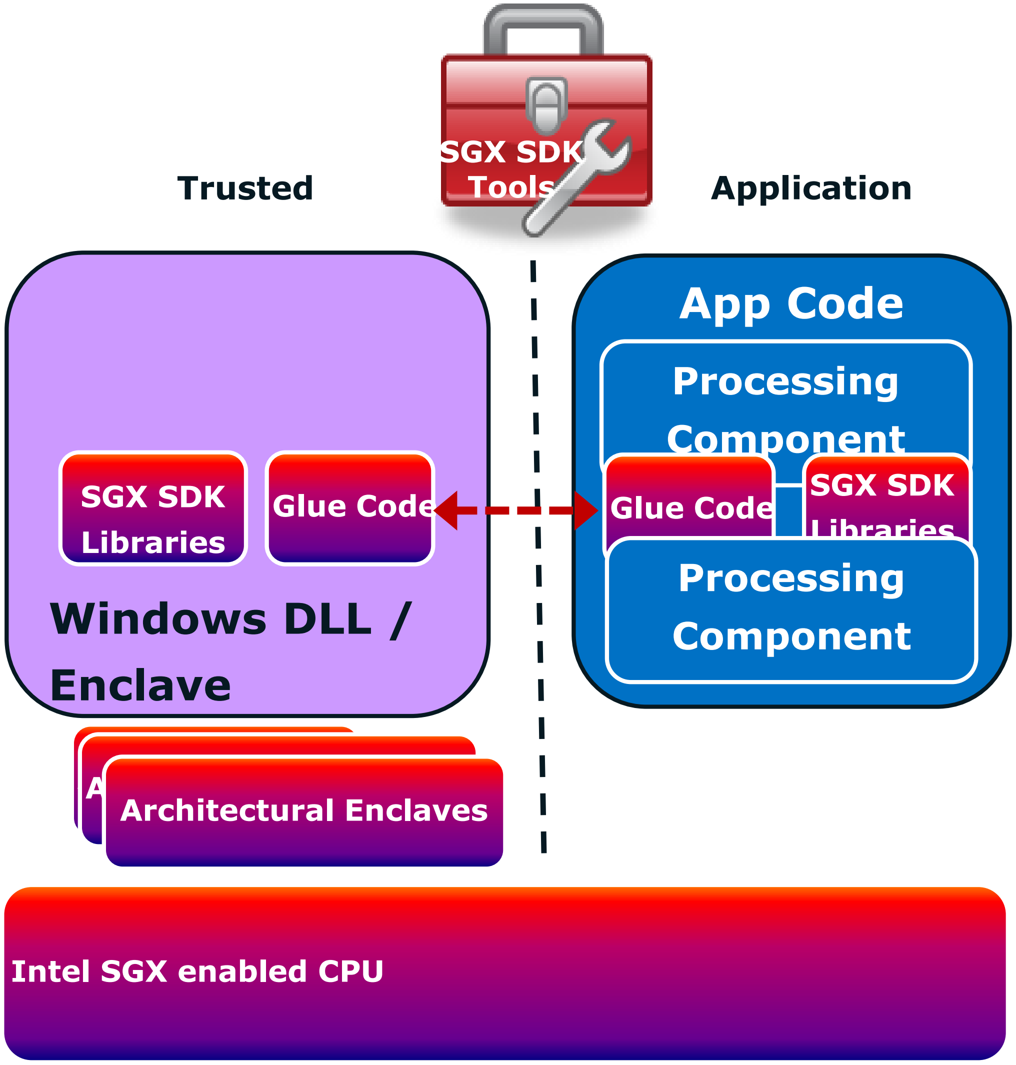
1. Enclave built & measured against ISV's signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls *EGETKEY* to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use

Example: Secure Transaction



1. Enclave built & measured against ISV's signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls *EGETKEY* to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use

Intel® SGX Software Development



- Software Developer decides which components should execute within an enclave
- Development Environment allows the Developer to quickly develop enclave enabled binaries
- Including support for common software libraries, exporting interfaces, and support for provisioning

SGX Technical Summary

- Provides any application the ability to keep a secret
  - Provide capability using new processor instructions
  - Application can support multiple enclaves
- Provides integrity and confidentiality
  - Resists hardware attacks
  - Prevent software access, including privileged software and SMM
- Applications run within OS environment
  - Low learning curve for application developers
  - Open to all developers
- Resources managed by system software

Links

Joint research poster session:
http://sigops.org/sosp/sosp13/

Programming Reference:
http://www.intel.com/software/isa

HASP Workshop:
https://sites.google.com/site/haspworkshop2013/workshop-program

Intel Labs: Security and Privacy Research

Looking for researchers with the following skills:

Cloud Computing

Operating Systems

Virtualization

BIOS and Firmware

Mobile Security

Machine Learning Algorithms

Contact Jennifer.M.Muir@intel.com

Thank You

Backup

SGX Paging Introduction

Requirement:

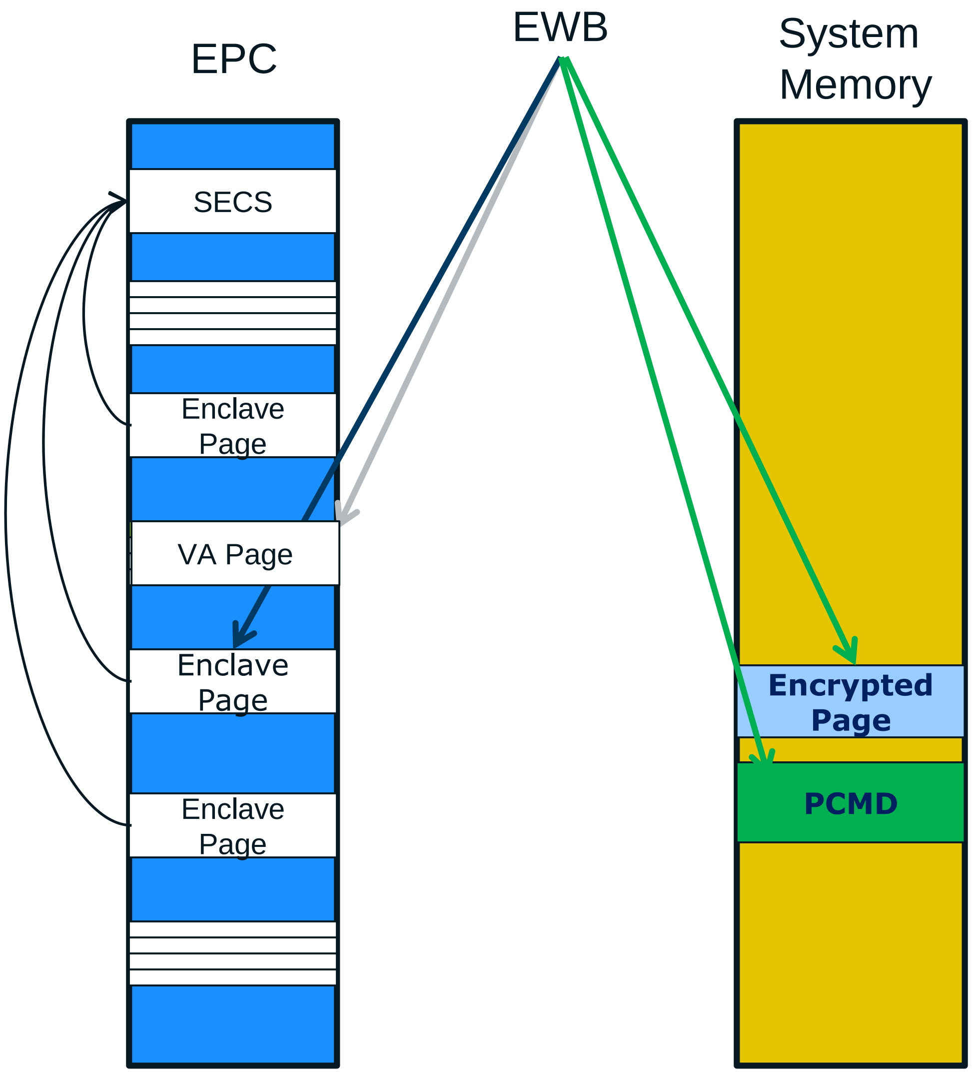
- Remove an EPC page and place into unprotected memory. Later restore it.
- Page must maintain same security properties (confidentiality, anti-replay, and integrity) when restored

Instructions:

- EWB: Evict EPC page to main memory with cryptographic protections
- ELDB/ELDU: Load page from main memory to EPC with cryptographic protections
- EPA: Allocate an EPC page for holding versions
- EBLOCK: Declare an EPC page ready for eviction
- ETRACK: Ensure address translations have been cleared

BUILD

Page-out Example



EWB Parameters:

- Pointer to EPC page that needs to be paged out
- Pointer to empty version slot
- Pointers outside EPC location

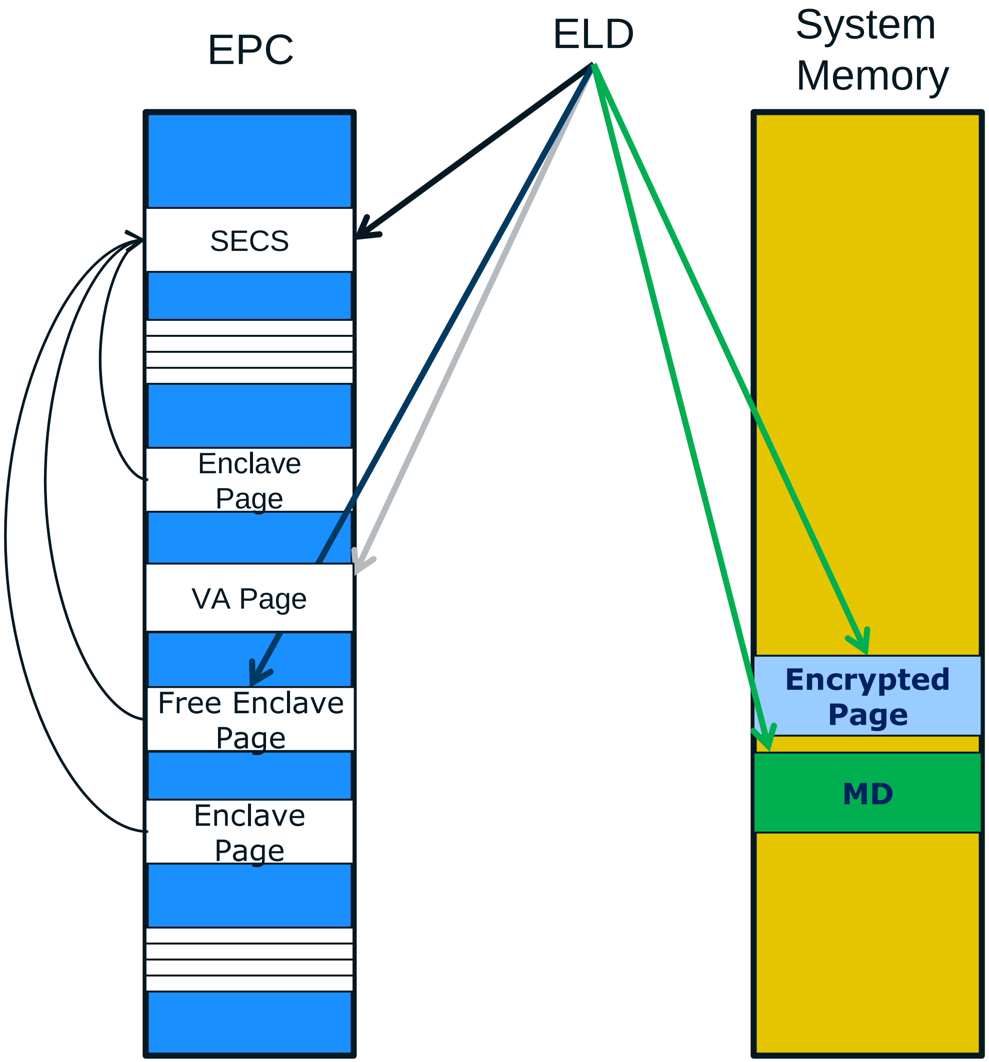
EWB Operation

- Remove page from the EPC
- Populate version slot
- Write encrypted version to outside
- Write meta-data, PCMD

All pages, including SECS and Version Array can be paged out

BUILD

Page-in Example



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
- Make back-pointer connection (if applicable)
- Free-up version slot