Software Guard Extension From Dream to Reality

The 7th Summer School on Cyber and Computer Security

Technion – October 2nd, 2018

Ittai Anati – ittai.anati@intel.com

Disclaimers

- Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Learn more at Intel.com, or from the OEM or retailer.
- You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.
- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
- No computer system can be absolutely secure.
- Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
-

Agenda

- SGX Fundamentals (level setting)
- SGX1 deeper dive

Break

- SGX2
- Flexible Launch Control, VMM Oversubscription, Key Separation and Sharing
- Provisioning, Attestation and Recovery

Part I - SGX fundamentals

The Dream

Challenge: Keeping Secrets on an Open Platform

Software solutions are responsible for keeping sensitive data which comes in many forms

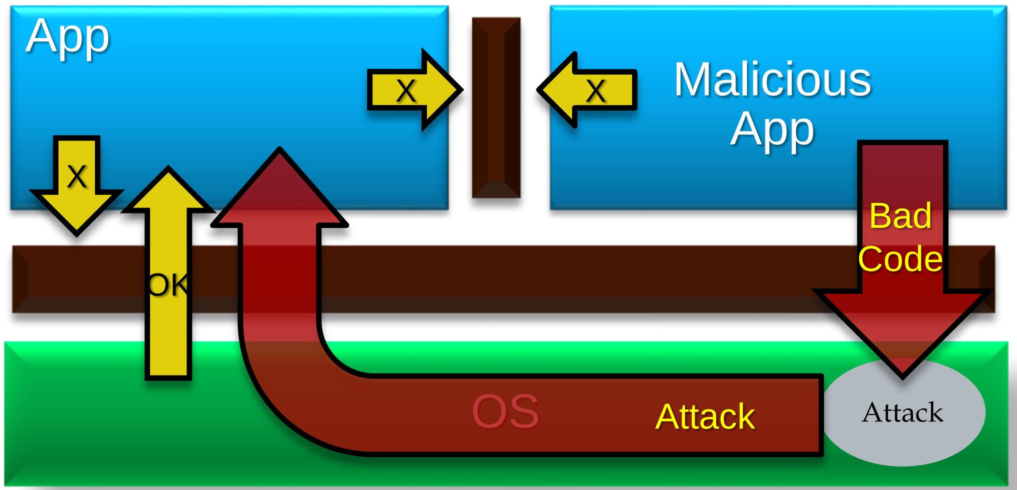
- Authentication Credentials
- Personal information
- Financial information
- Intellectual property
- Medical records
- Protected content

Who do we trust to safeguard these secrets?

- How many SW components are in our Trusted Compute Base (TCB)?

Build

Why is it so hard?



Protected Mode (rings) allows the OS to protect itself from apps …

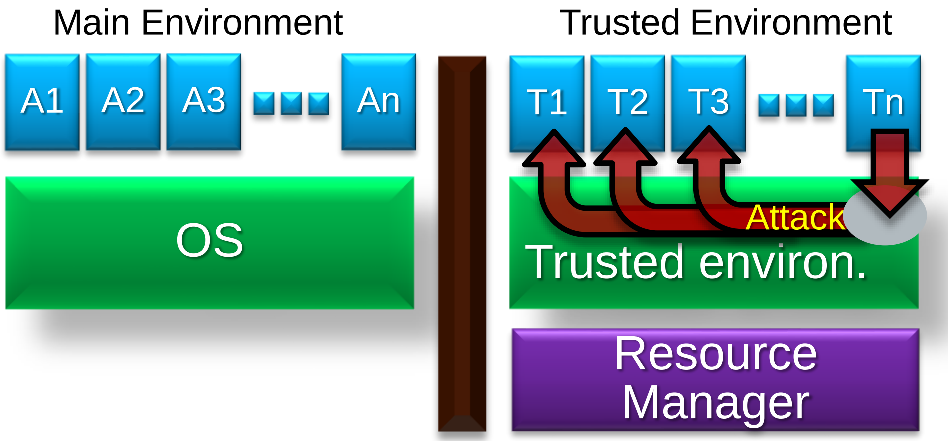
… and apps from each other …

… UNTIL a malicious attacker gains full privileges and then tampers with the OS or other apps

Apps not protected from privileged code attacks

Build

A Separate Trusted Execution Environment (TEE)



Examples

1. *Virtualization*
2. *Security Co-Processors*
3. *Separate Environment*

Works for <u>one</u> app … but not many

Works for <u>one</u> vendor … but not many

Privilege isolation doesn’t scale across the ecosystem

Build

Intel® SGX – The Philosophy

Enclaves

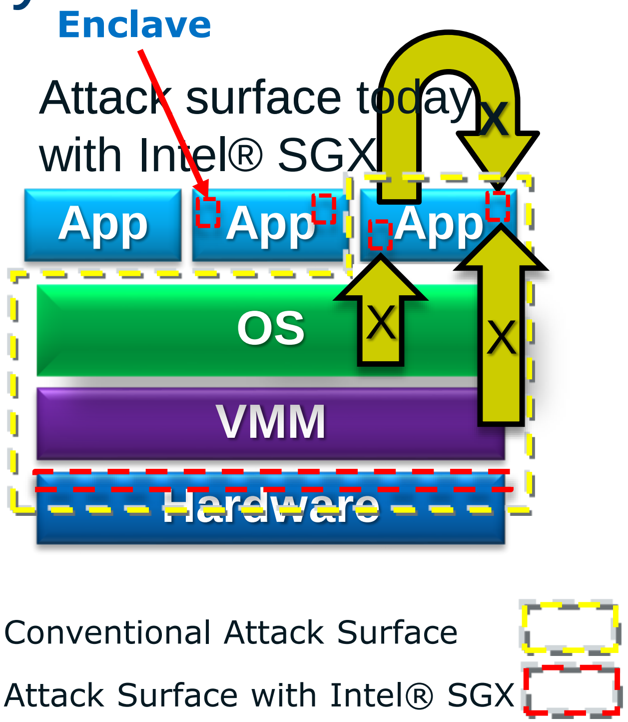
- Confidentiality and Integrity-protected data & code
- Controlled access to secrets
- Smaller attack surface

Familiar development/debug environment

- Standard OS environment and programming model
- Single application environment
- Builds on existing ecosystem expertise

Familiar deployment model

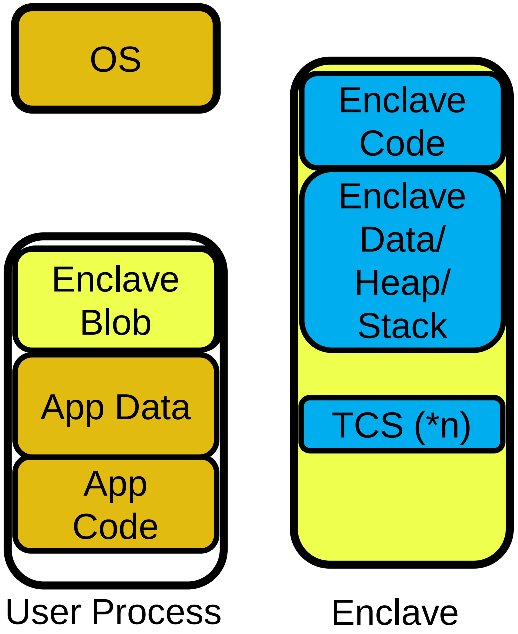
- Platform integration not a bottleneck to deployment of trusted apps



Scalable security within mainstream environment

Enclave Programming Environment

Protected execution environment embedded in a process



Provides Confidentiality Integrity and Anti-replay

With its own code and data

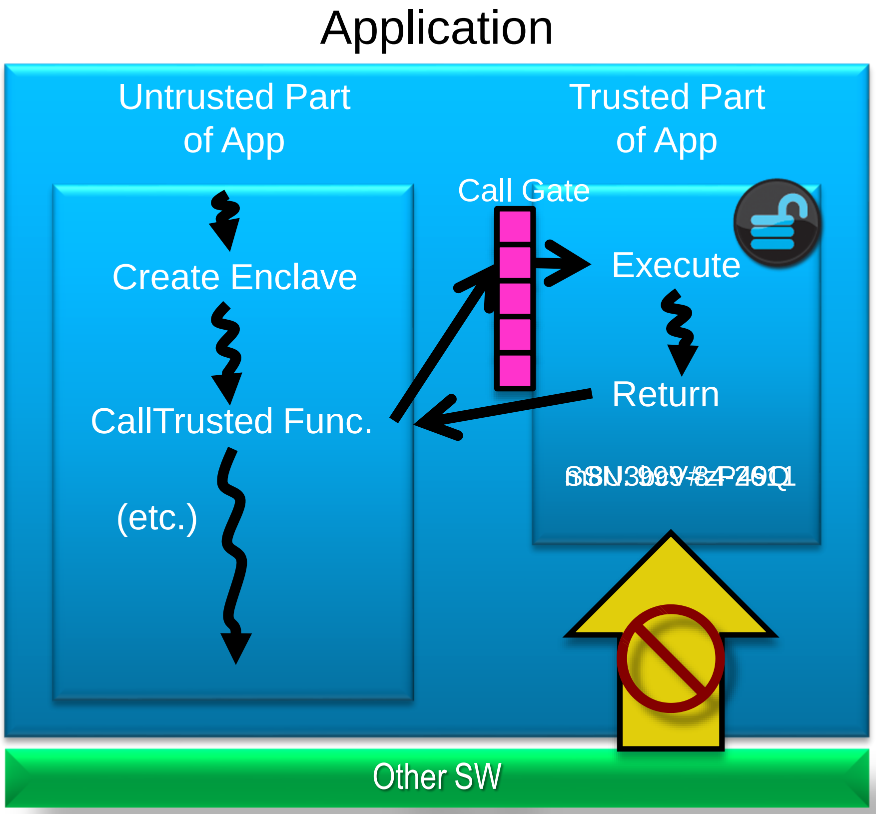
With controlled entry points for multiple threads

With access to app memory

Build

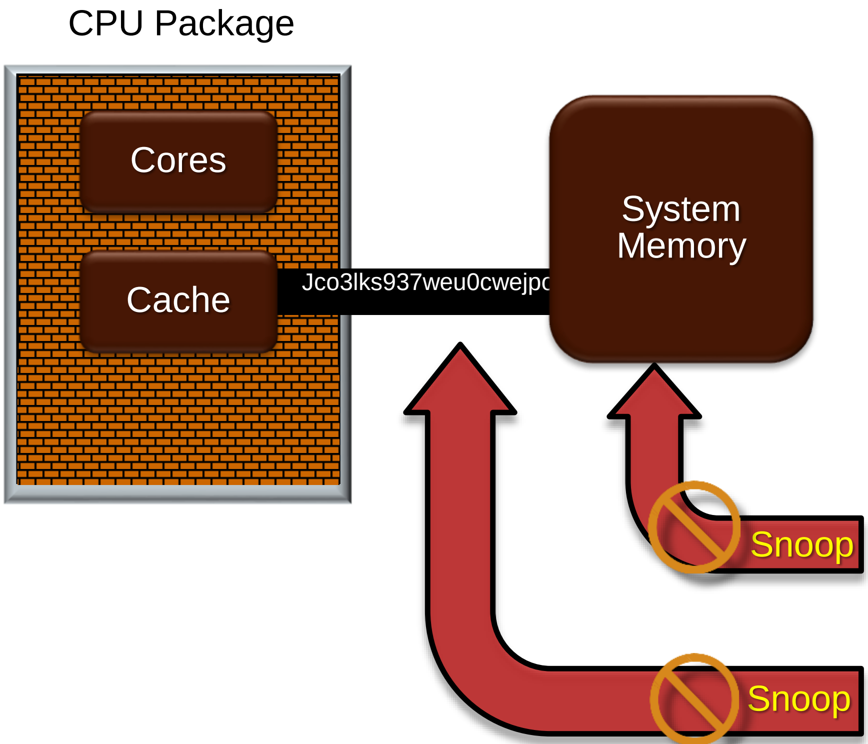
Intel® SGX: Greater protection against SW Attacks

- App created with trusted and untrusted sections
- App runs & calls OS to place enclave in trusted memory
- App calls trusted function (EENTER)
  - code running inside enclave can see and access data in clear
  - external access to code/data is denied
- Function ends and returns (EEXIT)
  - enclave data remains protected in trusted memory



Build

Greater protection against Memory Snooping Attacks



- 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 with replay protection
- External memory reads and bus snoops can only see encrypted data
- Attempts to modify memory will be detected

Intel® SGX Architecture

Intel® SGX - Overview

A HW assisted paradigm introduced on 6th generation Core™ for applications to locate and protect critical secrets in a structure called an “enclave”

- Against SW attacks originated at any privilege level
- Against many hardware based attacks

Remote anonymous attestation to verify code and data signatures at the application level

Local attestation between applications

HW based unique keys for sealing

Synergistic and scalable with other CPU and platform features

- Multi-core, DRNG, AES-NI, SHA-NI, Trusted time, Monotonic counter

Protected Memory - Enclave Page Cache (EPC)

Fixed area of physical memory for holding secure pages

- Setup and locked by BIOS until next reset and verified by the CPU
- Divided into 4KB pages with their own protected access rights assignments

Restricted access

- Access restricted to enclaves and SGX instructions
- Enclaves can only access their own content according to the assigned access rights

Instructions to swap EPC pages through main memory without compromising security

- Encryption, integrity, anti-replay

Managed by the OS as a system resource, but opaque to the OS

SGX Instruction Breakdown (supervisor/user)

Management

- ECREATE
  EADD
  EEXTEND
  EREMOVE } Build
- EINIT } Verify

Execution

- EENTER
  EEXIT } Synchronous
- AEX - Async EXIT
  ERESUME } Interrupts

Sealing & Attestation

- EGETKEY
  EREPORT } Keys

Paging

- EPA
  EBLOCK
  ETRACK } Protection Management
- EWB
  ELDB
  ELDU } Swap

Debug

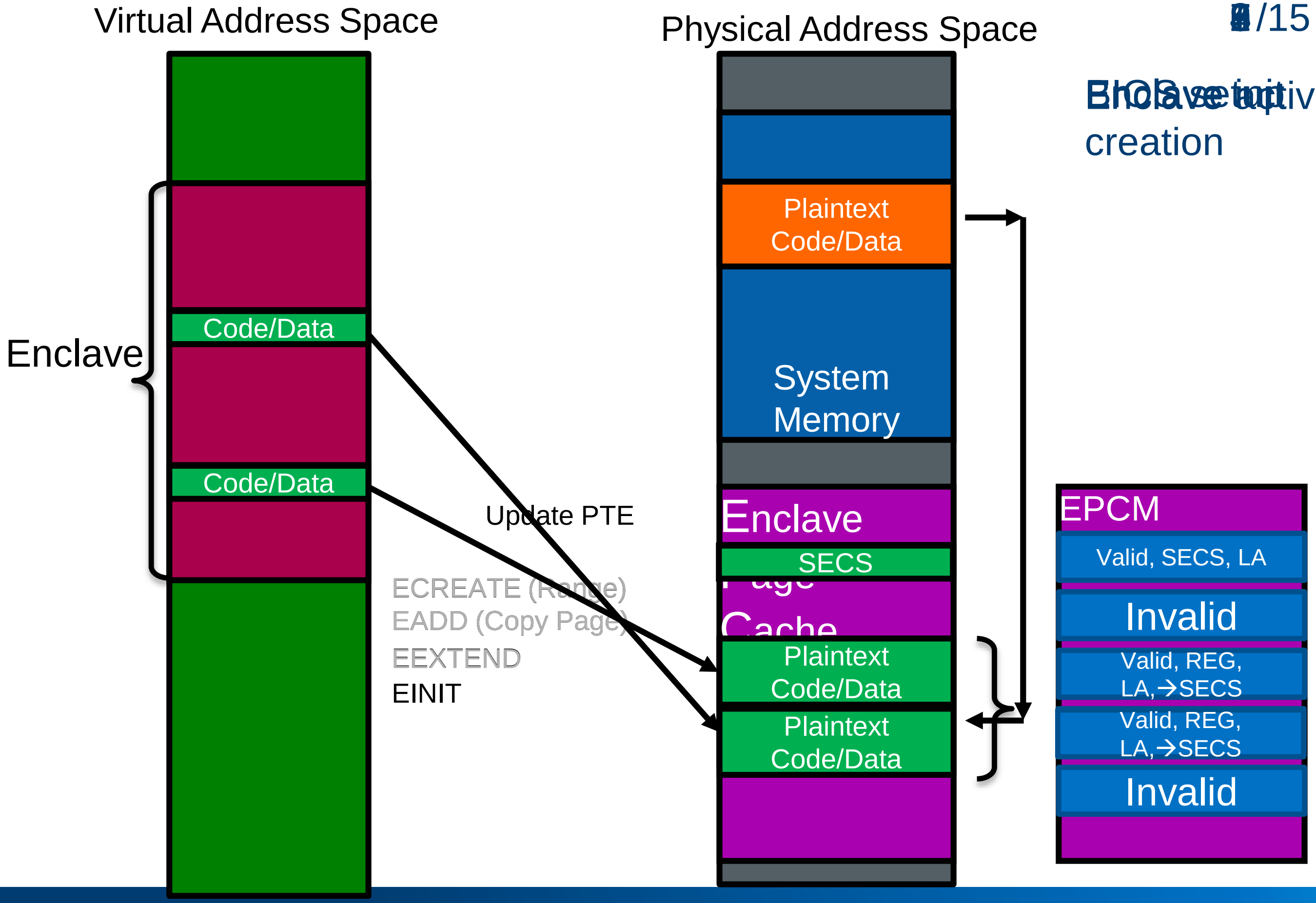
- EDBGRD
  EDBGWR

Enclave Life Cycle

Build

/15

Enclave creation

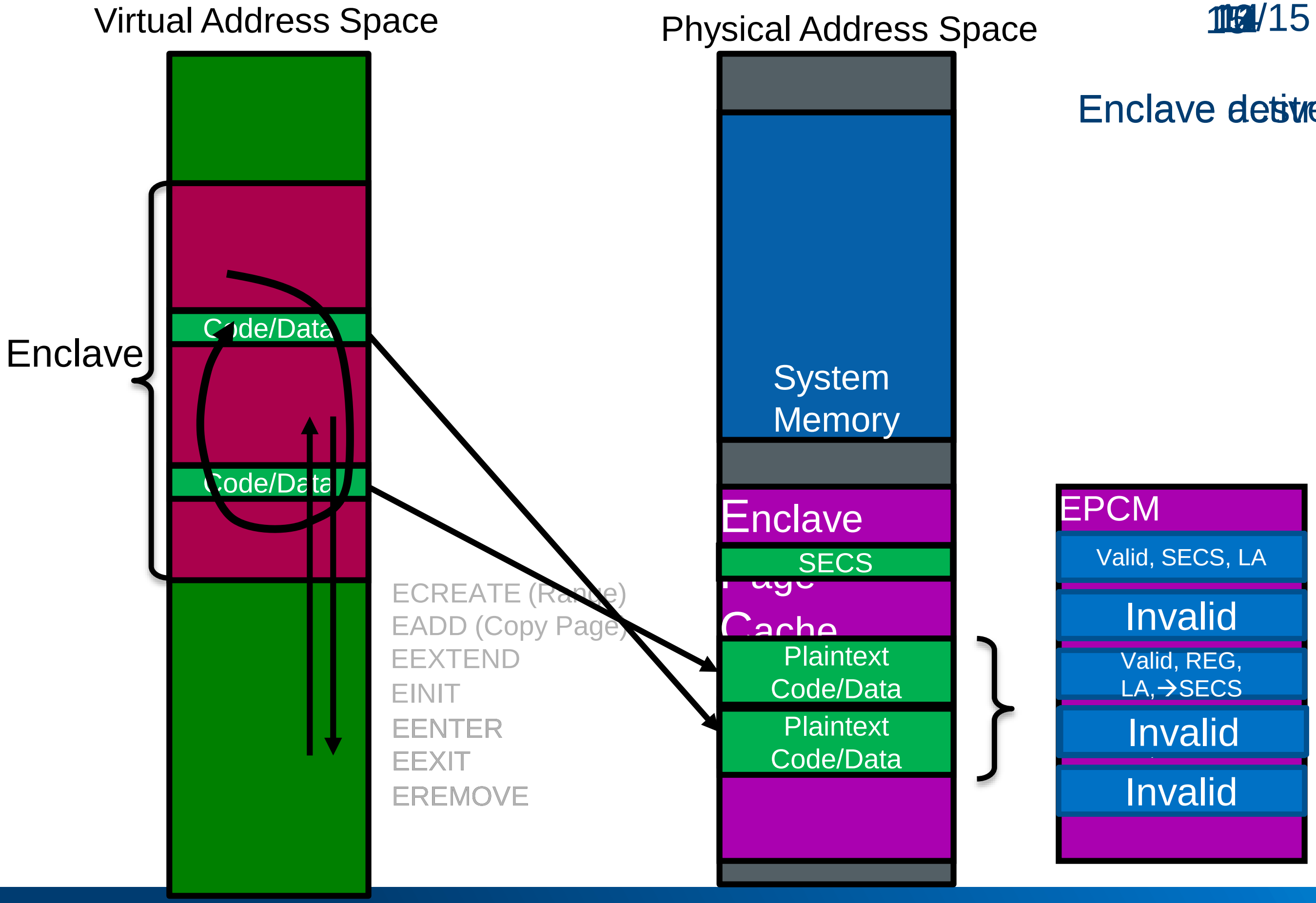


Enclave Life Cycle

Build

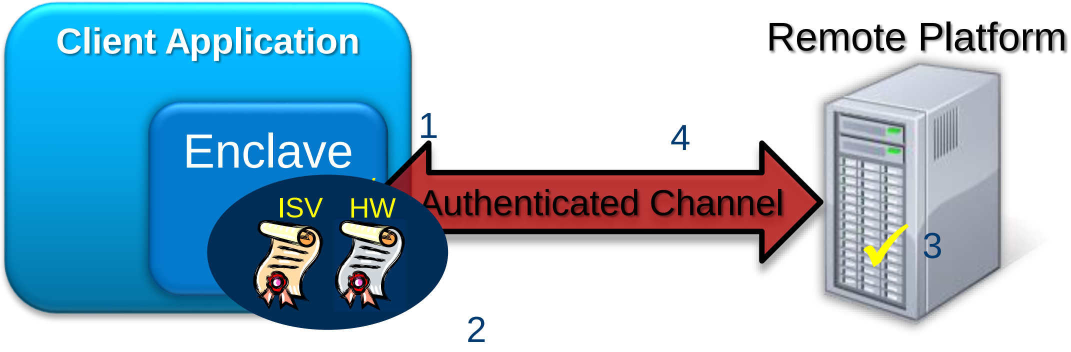
15/15

Enclave destruction



Example

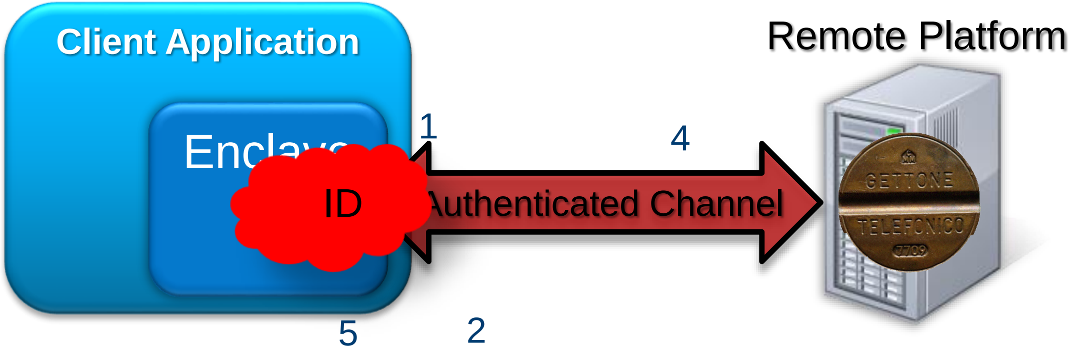
Example: Secure Transaction Stage 1



1. Enclave built & measured against ISV's signed manifest
2. Enclave uses HW to obtain REPORT
3. REPORT & ephemeral key sent to server & verified
4. A trusted channel is established with remote server

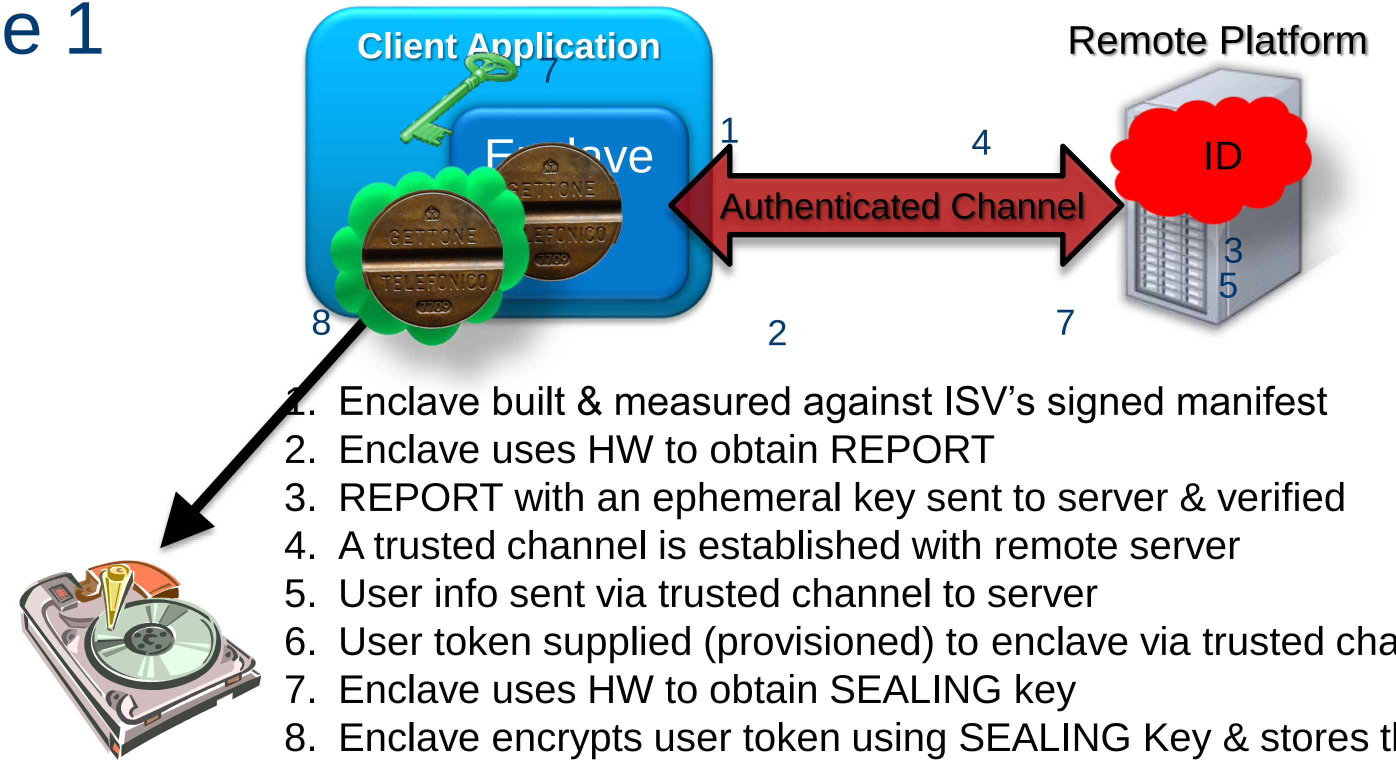
Build

Example: Secure Transaction Stage 1



1. Enclave built & measured against ISV’s signed manifest
2. Enclave uses HW to obtain REPORT
3. REPORT with an ephemeral key sent to server & verified
4. A trusted channel is established with remote server
5. User info sent via trusted channel to server
6. User Token provisioned back to enclave via trusted channel

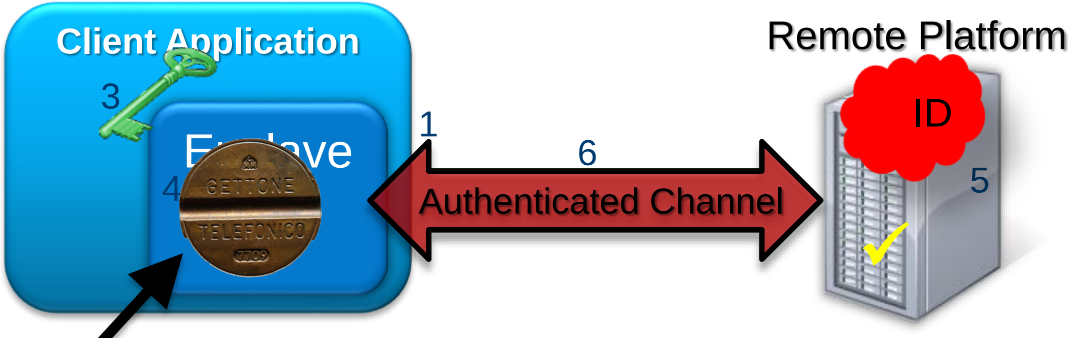
Example: Secure Transaction Stage 1



1. Enclave built & measured against ISV's signed manifest
2. Enclave uses HW to obtain REPORT
3. REPORT with an ephemeral key sent to server & verified
4. A trusted channel is established with remote server
5. User info sent via trusted channel to server
6. User token supplied (provisioned) to enclave via trusted channel
7. Enclave uses HW to obtain SEALING key
8. Enclave encrypts user token using SEALING Key & stores the sealed blob on disk for future use

Build

Example: Secure Transaction Day-to-day



1. Enclave built & measured against ISV's signed manifest
2. Enclave retrieves sealed blob from disk
3. Enclave uses HW to obtain SEALING Key
4. Enclave extracts the User Token from the blob
5. User Token sent to remote platform and ID retrieved.
6. Trusted channel is established. No need to enter passwords

Part II - SGX1 deeper dive

Emulating Intel® SGX

Enclave code is regular software that can run in an SGX ISA emulator

- So what prevents attackers from using a rogue SGX ISA emulator?

Only an enclave running on genuine SGX capable HW (“real” enclave) can:

- Prove to a 3rd party that it’s a “real” enclave
- Access secrets that were previously stored on the platform by a “real” enclave

	Genuine SGX HW	Emulated SGX environment
Enclave validation	Production level	Functional w/ debug content
Sealing	Production level	Functional w/ debug content
Attestation	Production level	Functional w/ debug content

Sealing & Attestation

Attestation: Proving to a 3rd party an enclave is properly running on genuine Intel® SGX enabled HW

- A 3rd party shouldn't supply sensitive information to an enclave without ensuring it's properly running on genuine Intel® SGX enabled HW

Sealing: Encrypting sensitive information when exporting outside of the enclave (e.g. saving tokens for future use)

Sealing and Attestation are based on AES-128/SHA-256 cryptography using 128-bit keys

- Keys are unique per processor and its security status level
- Keys are unique per enclave
- Dedicated keys for Sealing (Seal), Attestation (Report), and platform establishment (Provisioning)

Debugging an Enclave

Production enclaves disable debugging facilities:

- Breakpoints, Single stepping
- Enclave-specific performance monitoring
- Tracing

SGX debug model: minimum intrusiveness

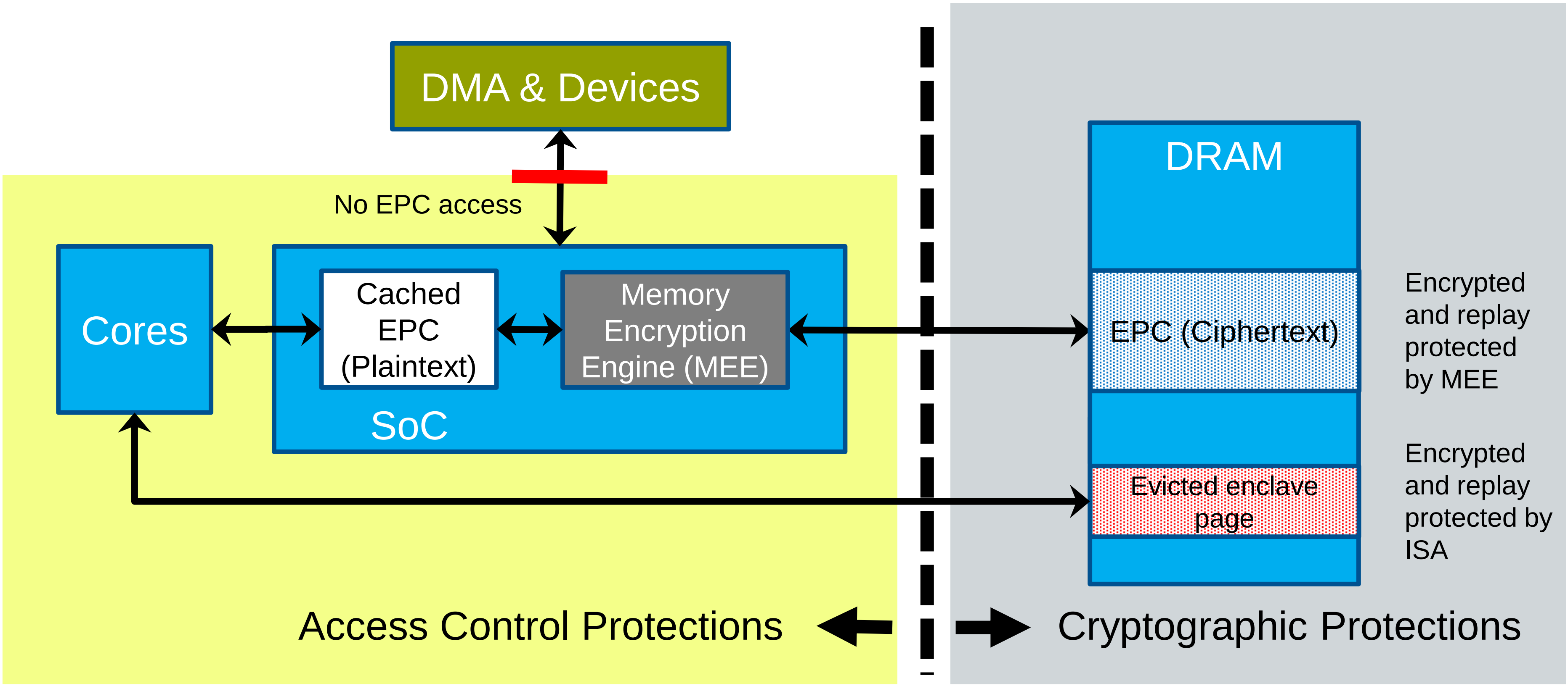
- Debuggable and non debuggable enclaves can co-exist
- Access control semantics remain very similar to production

Any enclave can become debuggable by setting its DEBUG Attribute bit at build time

- The REPORT reflects the debuggability of the enclave
- Enclave receives different keys
- Debug functions (breakpoint, single stepping, etc.) are allowed
- EDBGRD and EDBGWR allow the debugger access to enclave’s memory

EPC Management

EPC Protection Model



EPC Page Swapping - Protection

EPC is a fixed-size portion of memory setup by BIOS

OS must be able to dynamically swap pages into and out of EPC

EPC swapping ISA maintains SGX security objectives:

- Confidentiality, integrity, replay-protection
- 128 Byte MAC-protected metadata payload (PCMD) is generated for every evicted page:
  - Hash of page’s content
  - Page’s security information
  - Page’s version number
- The version always remains in protected memory

EPC Page Swapping - Overview

Two instructions swap a page between EPC and main memory

- EWB – Secure eviction of an EPC page into main memory
- ELD – Secure loading of a previously evicted page back into EPC

But that’s not enough

- Logical processors might still have TLB mappings to the evicted page
- Need to ensure the evicted page’s content can’t be accessed by anyone

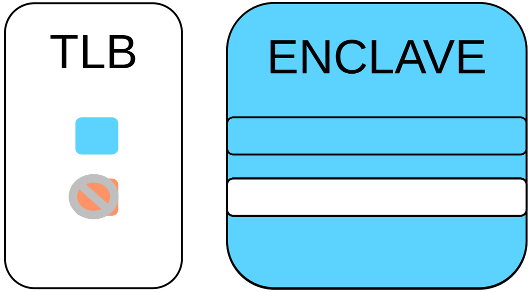
Solution: A staged trust and verify approach

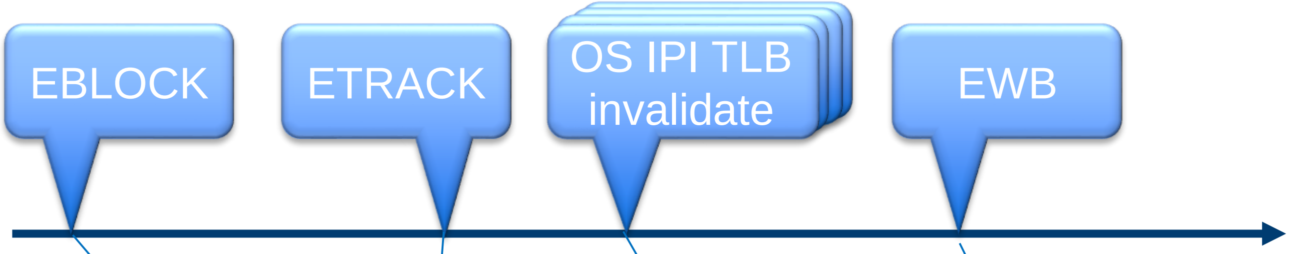
- OS needs to send an IPI to flush TLBs. EWB verifies it has been properly done
- No need for a full rendezvous
- Logical processors are allowed to re-enter the enclave after flushing their stale entries

Build

EPC Page Swapping - Synchronization

Treatment of stale TLB entries





Marks the page as BLOCKED. Attempts to access which miss the TLB will fail.

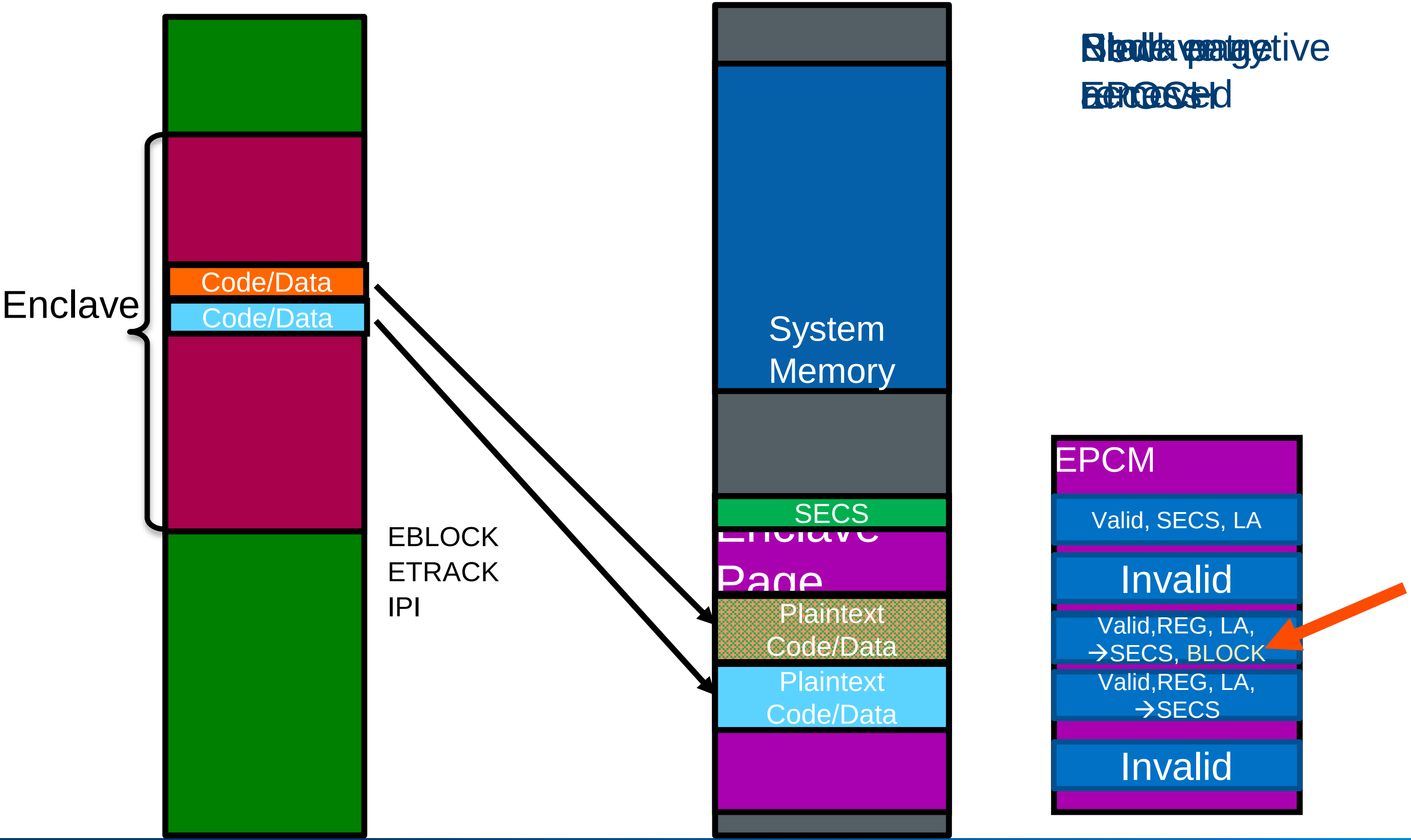
Moves the enclave to a new EPOCH. New entries into the enclave can't access the page but can use other pages

All threads exit the enclave and flush their stale TLB entries

Page is evicted. EWB fails if any logical processor hasn't flushed the TLBs

EPC Page Swapping

Build



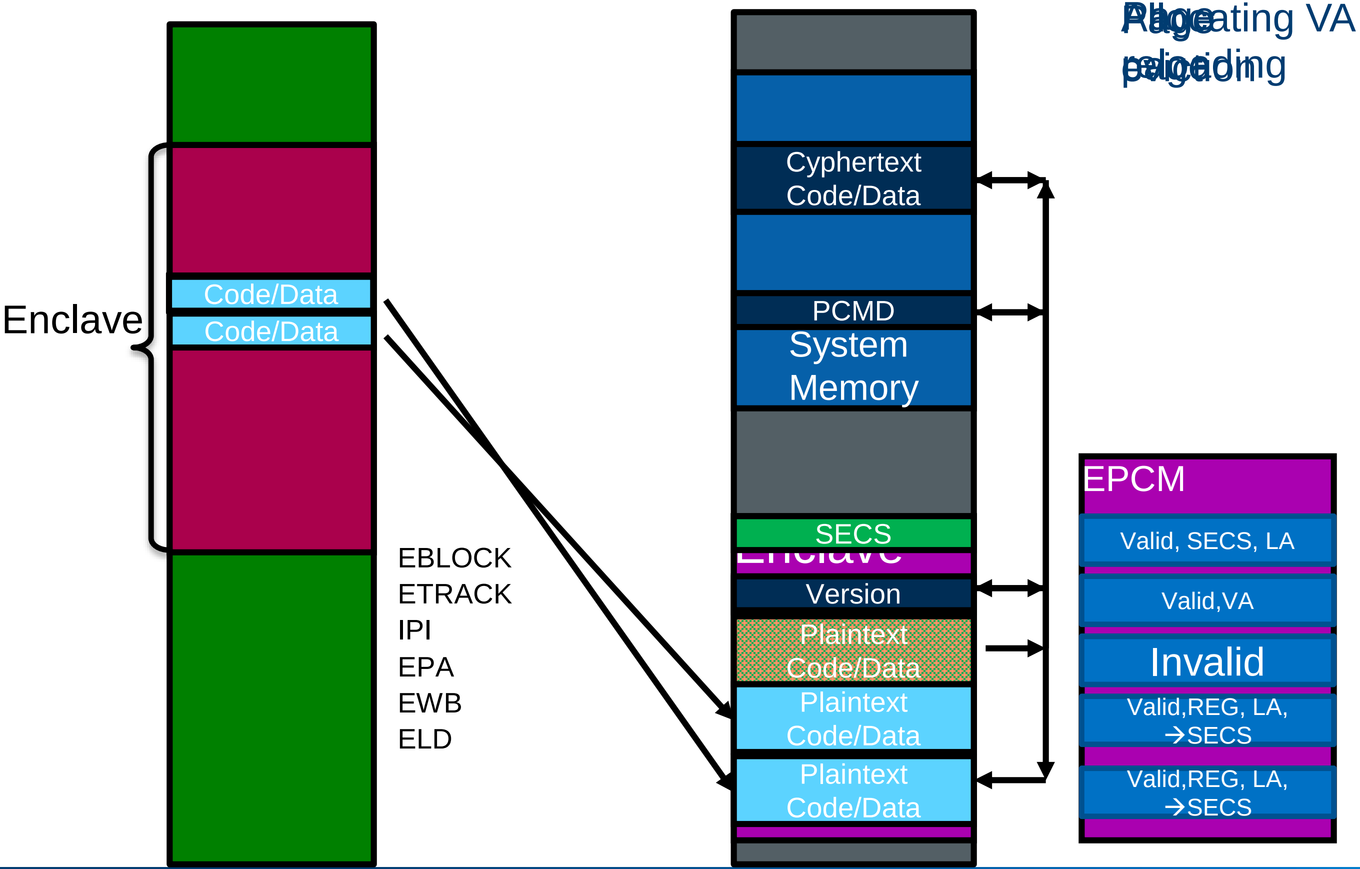
EPC Page Swapping

Build

7/7

Page reloading

Allocating VA page



Keys

The Importance of SGX Keys

SGX keys are the foundation for Sealing and Attestation

- Provide isolation between enclaves and platforms
- Provide separation after security bug fixes
- Provide proof of security posture

Keys are 128 bits and unique per:

- CPU and its Security Version Number
- Enclave and its Security Version Number
- Platform owner's provided entropy (OWNEREPOCH)

Uni-directional - A higher security enclave can recreate keys of a lesser secure enclave, but not the other way around

Sealing key

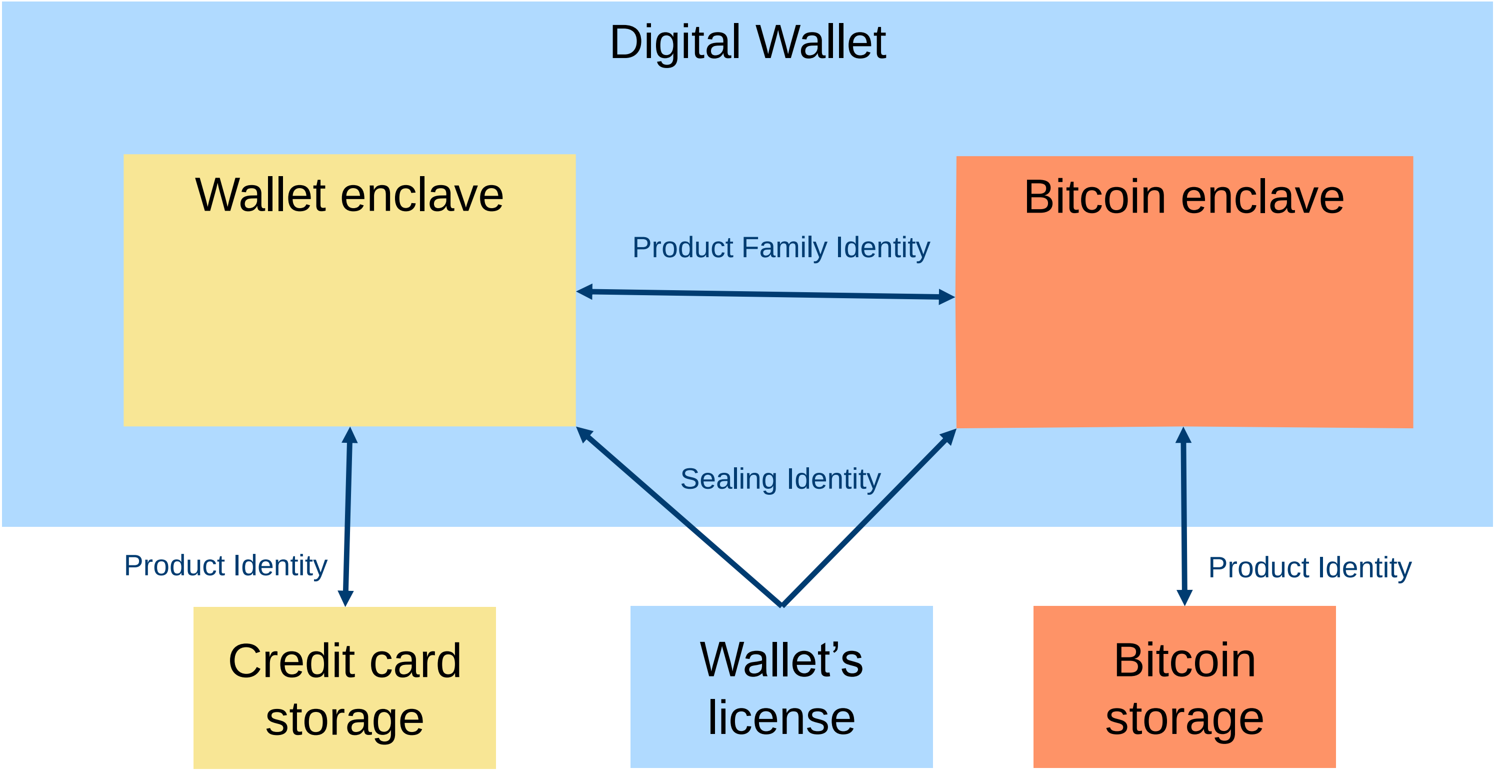
A persistent general purpose 128-bit key obtained from the EGETKEY leaf function

Key can be obtained based on different policies:

- Enclave identity – Enclave’s content
- Sealing identity – Enclave’s creator
- Product identity – Enclave product
- Product family identity – Enclave product’s family

ISV defines the key policy based on usage model and security requirements

Sealing Example – A digital Wallet



Report key

The key is used for generating a MAC on the enclave's REPORT information.

- EREPORT uses the key to generate the MAC
- EGETKEY provides the key to the verifying enclave to verify the MAC

The REPORT's information is the reporting enclave's information

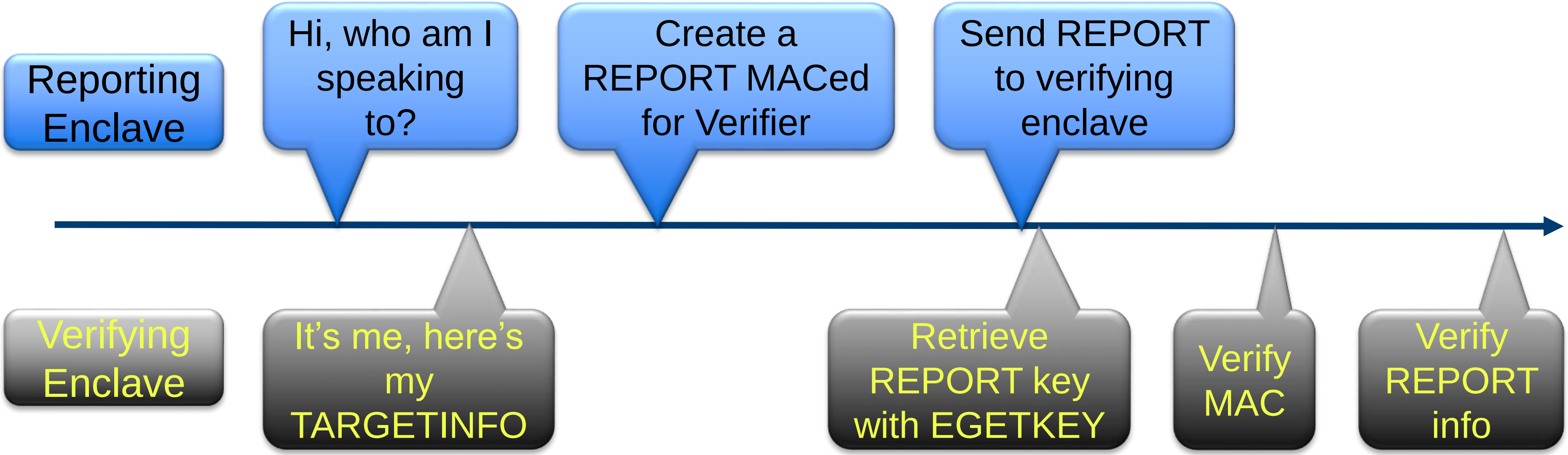
The REPORT's MACing key is the verifying enclave's REPORT key

- The 64-Byte REPORTDATA structure in the REPORT can be used to securely pass information from the enclave to the verifying enclave

Local Attestation and Report Key

How one enclave proves its identity to another enclave on the same platform

- Only works on the same platform
- Typically done both ways for bi-directional trust



Remote Attestation

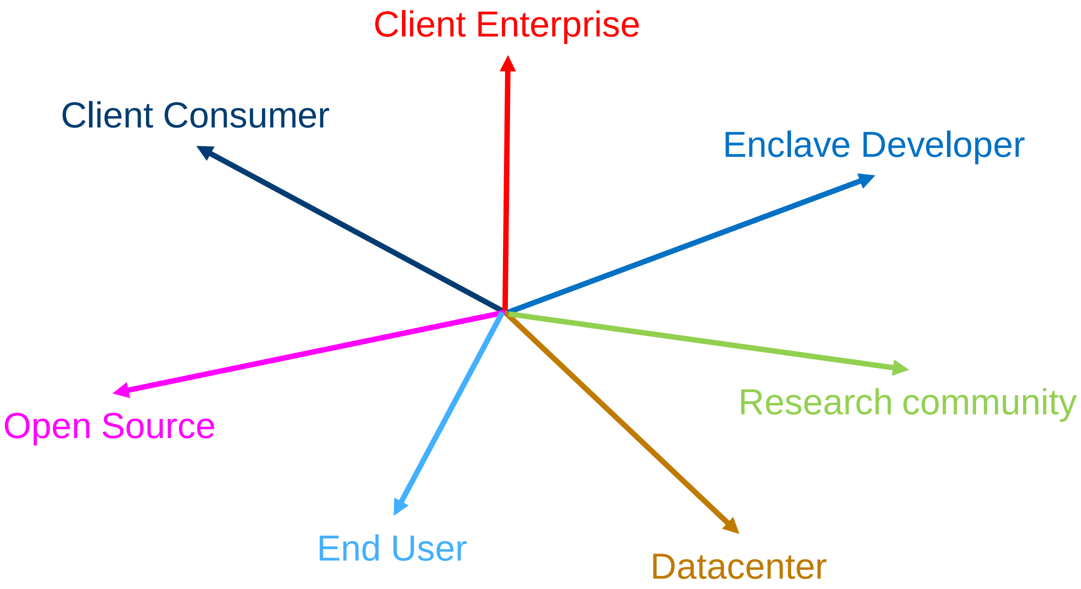
- Enclave creates REPORT
- Quoting enclave verifies REPORT
- Quoting enclave signs report with certified signing key
- Remote party verifies the Quote
- Remote party checks the REPORT
- Remote party trusts enclave

Part III – SGX2, Launch Config, VMM Oversubscription

SGX2

Optimization vectors

Different usage models drive for different feature extensions



SGX2 – Enclave Dynamic Memory Management

6 new SGX operations that allows enclaves to securely change after EINIT

- Add memory
- Remove memory
- Change page type or access rights

Examples

- Add more stack/heap
- Garbage collection
- Spawn more threads
- Load a library

SGX2 – Security model

EPC operations are performed by the untrusted OS

- Enclave must approve changes made by the OS using the EACCEPT leaf function

As with SGX1’s page swapping, stale TLB entries must be flushed using the EPOCH tracking mechanism before access is granted to modified pages

Security model maintained with confidentiality, integrity and anti-replay

- Model allows freeing a page and re-allocating it without allowing replay attacks

SGX2 instructions

EAUG – Add a page of zeros

EACCEPT – Approve a change from within the enclave

EACCEPTCOPY – Approve a change and populate a page

EMODT – Modify page type

EMODPR – Modify access rights - Restrict

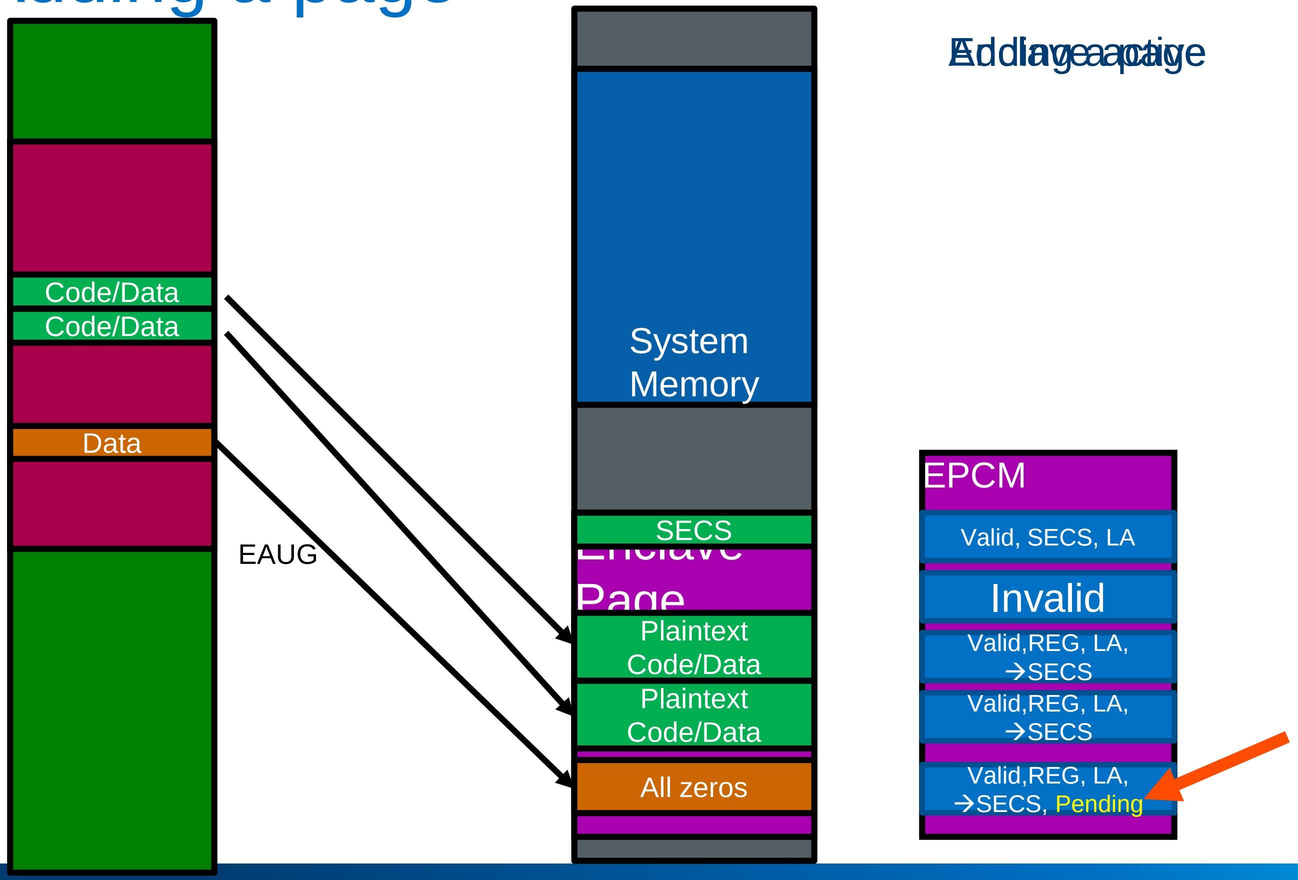
EMODPE – Modify access rights – Extend

IPIs required after EMODT and EMODPR to ensure stale TLB entries are flushed out.

EACCEPT enforces compliancy

Build

SGX2 – Adding a page

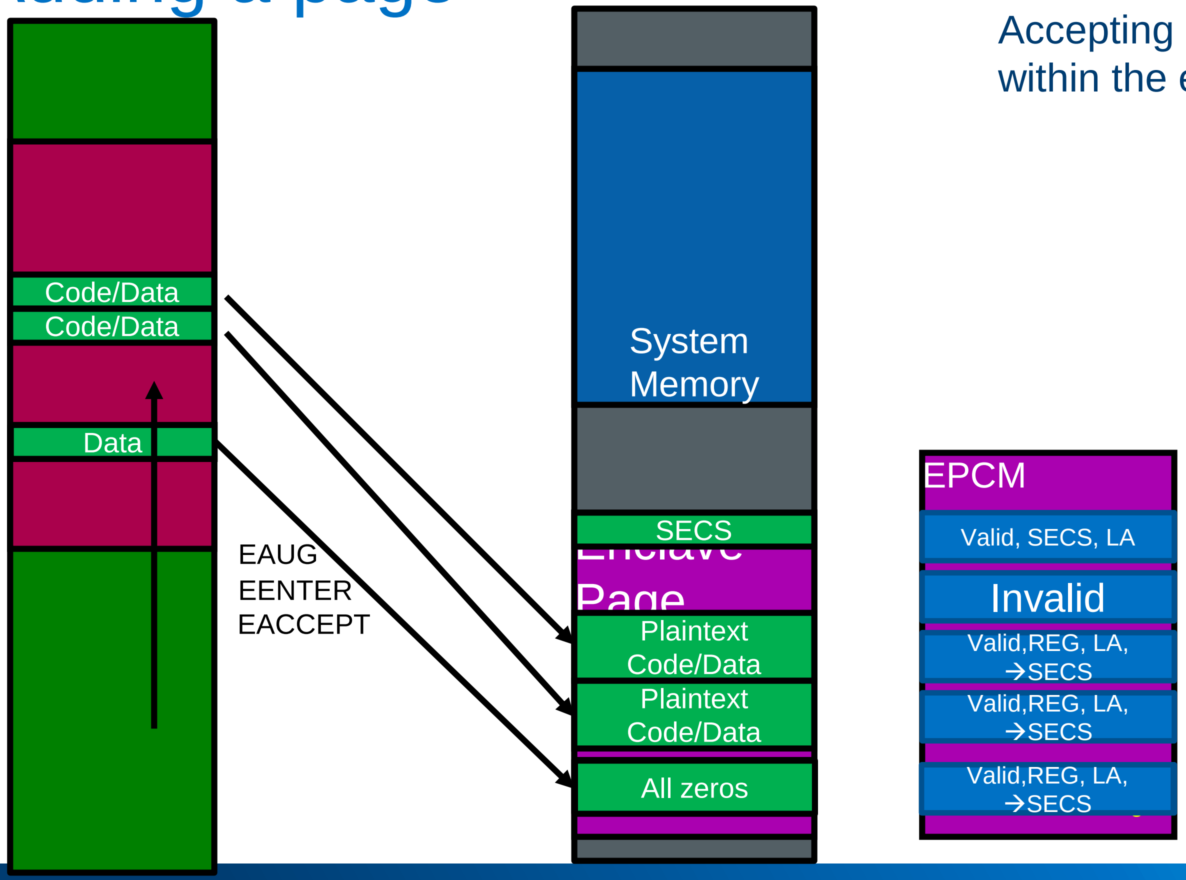


Build

3 / 4

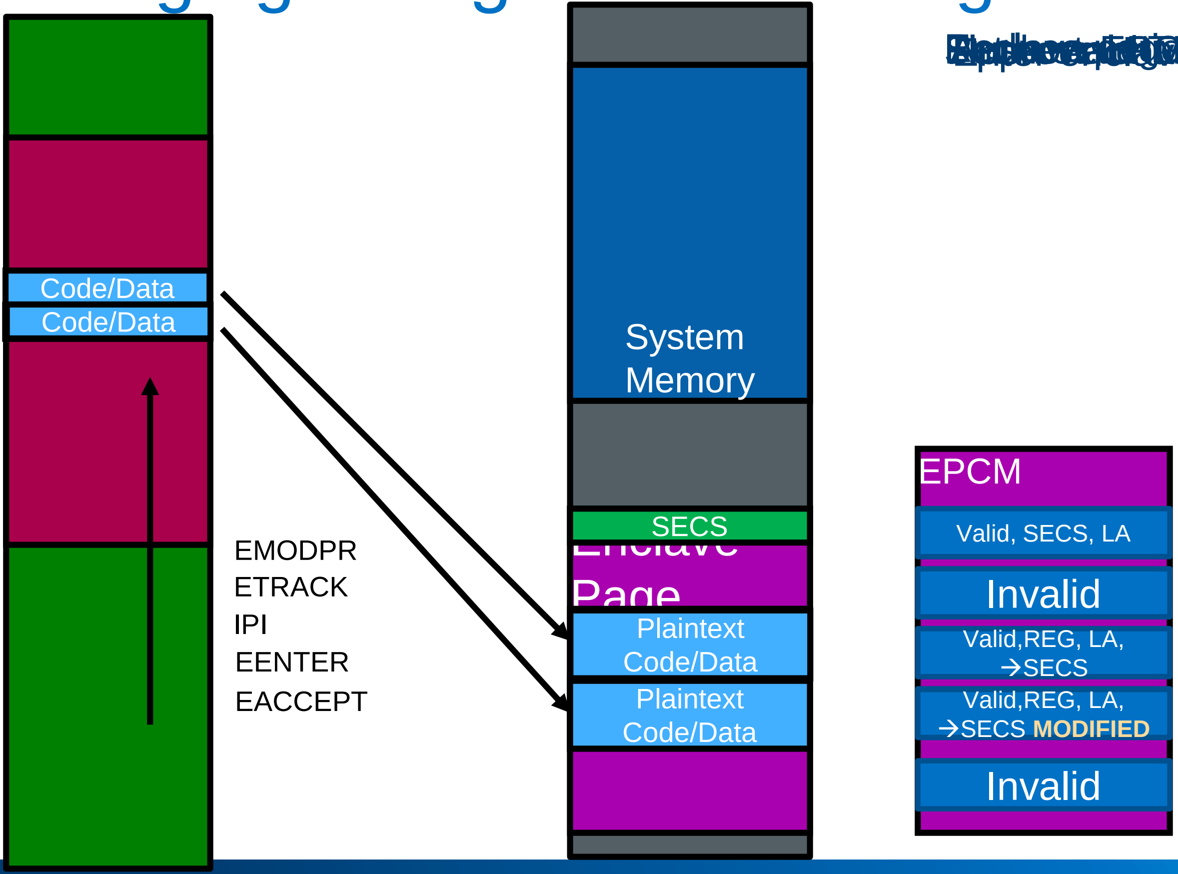
SGX2 – Adding a page

Accepting a page from within the enclave



Build

SGX2 – Changing a Page’s Access Rights



Launch Configuration

Launch Control - EINITOKEN key

SGX requires a Launch Enclave to approve each enclave that runs on the platform

- Launch Enclave after Reset is Intel's, but it can change if CPU supports SGX Launch Configuration
- The Launch Enclave generates an EINIT token that is used by EINIT to initialize enclaves

The EINIT token

- Includes information about the approved enclave
- MACed with the EINITOKEN Key
- Verified by EINIT during enclave initialization

A Launch Enclave is special:

- It has access to the EINITOKEN Key
- It doesn't require an EINIT token
- It must be signed by the approved author of the Launch Enclave

SGX Launch Configuration

Goals

- Allow a platform owner to designate his/her own Launch Enclave
- Allow different enclave launch policies on different virtual machines

Architecture

- Four 64-bit Machine-Specific-Registers hold a 256-bit digest of the approved Launch Enclave’s public key
- Only the approved Launch Enclave can ask for the EINITTOKEN key
- The digest can be locked at boot time or remain open to allows switching of Launch Enclaves, e.g. in a virtualized environment

VMM Oversubscription

Virtualizing EPC

Partitioned model - Simple, but not efficient

- Each VM receives a fixed portion of EPC and has to live with it
- A VM is allowed to swap its EPC pages
- VMM doesn’t need to intercept guest activity
  - On VM teardown that VMM needs to ensure all the VM’s EPC pages are free.
  - Removing EPC pages is hierarchical, so will might need to perform two passes

Oversubscription model – Allows better datacenter load balancing

- VMM dynamically swaps and allocates pages to VMs on demand
- While the VM might be swapping its EPC pages
- Single EPOCH tracking mechanism can get the VMM confused
  - SGX1 VMM will need to intercept VM’s activities. OVERSUB extension reduces the overhead

Virtualizing EPC - Challenges

A VMM must know where an enclave's SECS is to be able to reload its EPC pages back in

- A VMM would need to intercept all VM's ECREATE and ELD operations

A VMM must prevent a VM from removing SECS pages, so it could reload child pages back in

- A VMM would need to intercept all VM's EWB and EREMOVE operations

Shared use of EPOCH tracking can cause conflict faults in guest VM or in VMM

- A VMM would need to intercept all VM's ETRACK operations
- A VMM would need to cope with #GP in case of a conflict

Virtualizing EPC – OVERSUB Architecture

SGX extension to simplify the Oversubscription virtualization model.

ERDINFO – Provides info about an EPC page and the location of its parent SECS

- VMM will use ERDINFO before evicting a page to know how to put it back

EINCVIRTCHLD, EDECVIRTCHLD – Virtualize the # of child pages of an enclave

- VMM will use EINCVIRTCHLD every time it evicts an EPC page to prevent the guest from evicting the SECS

ETRACK, ELDU, ELDB – New VMExit on conflicts

- VMM will set this up to be informed whenever the VM is experiencing a resource conflict with the VMM.

ETRACKC, ELDUC, ELDBC – Conflict safe versions of ETRACK, ELDU and ELDB

- VMM will use these to avoid receiving a General Protection fault in case of a conflict

Part IV - Provisioning, Attestation and Recovery

Provisioning - Concept

For remote attestation, platform must certify a signing key-pair with the Provisioning/Attestation Server

- Intel's provisioning process uses EPID groups for anonymity.

The PROVISION key is used as the CPU's identifier for the Provisioning Server

- Intel's Provisioning infrastructure keeps the PROVISION Keys of all the CPUs it manufactures

Access to key is limited to the Provisioning Enclave

Once provisioning completes, the Provisioning Enclave holds a certified signing key-pair that confirms the CPU's authenticity for the specific SVN

- The Provisioning Enclave can use its private key to sign messages to prove to 3rd parties they originated from a genuine SGX enabled platform at a specific SVN
- In particular, the Provisioning Enclave can sign a REPORT generated by another enclave on the platform

Provisioning process

Symmetric identification

- Provisioning Enclave identifies itself using a derivation of the PROVISION key
- Provisioning Server keeps a copy of all PROVISION keys

Certificate-based identification

- Provisioning Certificate Enclave (PCE) uses a PROVISION-key based private key to sign a Provisioning blob
- Provisioning Server keeps certificates of all PROVISION-key based public keys
- Sets up the grounds for Datacenter based attestation
- Stronger security allows better scaling of the provisioning infrastructure

Attestation

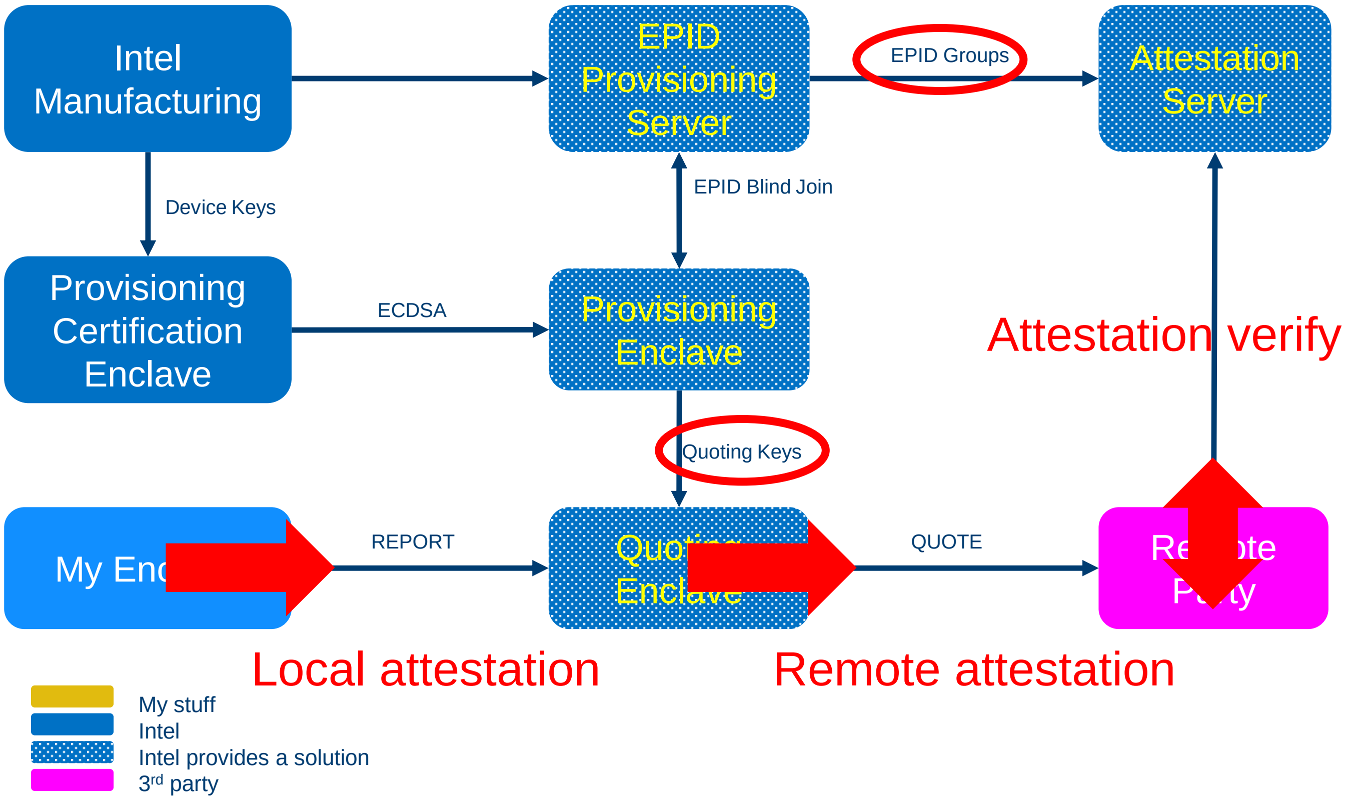
Intel Attestation Service provides a service to 3rd parties to verify Quotes

The server can respond with the following status (sample):

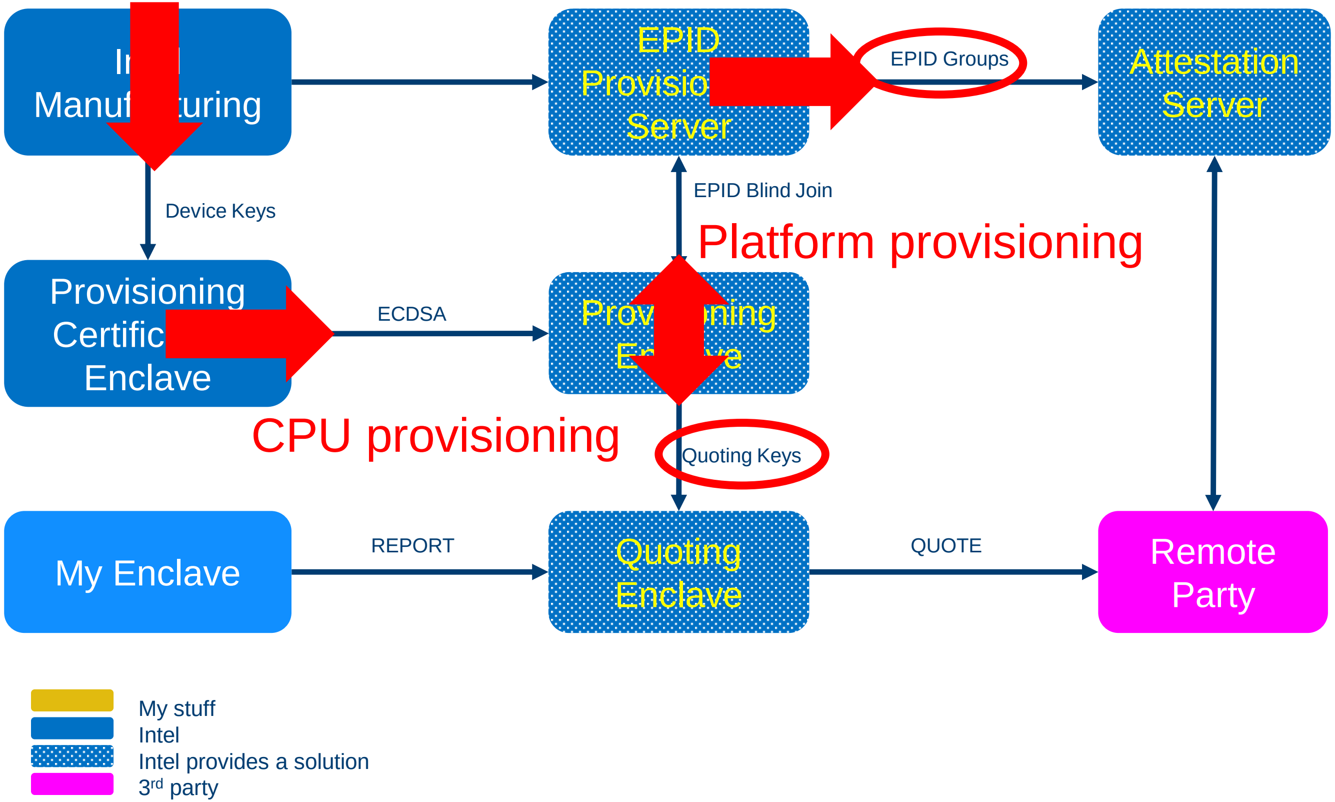
- OK
- GROUP_REVOKED
- GROUP_OUT_OF_DATE
- CONFIGURATION_NEEDED

In case of an issue, the response may include also a Platform Info Blob that Platform SW can interpret and advise on corrective actions

Attestation & Provisioning



Attestation & Provisioning



Recovery and Attestation

A fix is released for a buggy component in the TCB

- It can be SW or FW

The updated component is released with a higher SVN value

- ISV-SVN or CPU-SVN

New keys isolate between the old TCB and the new TCB

- At the new SVN level old sealed secrets can still be accessed by the enclave

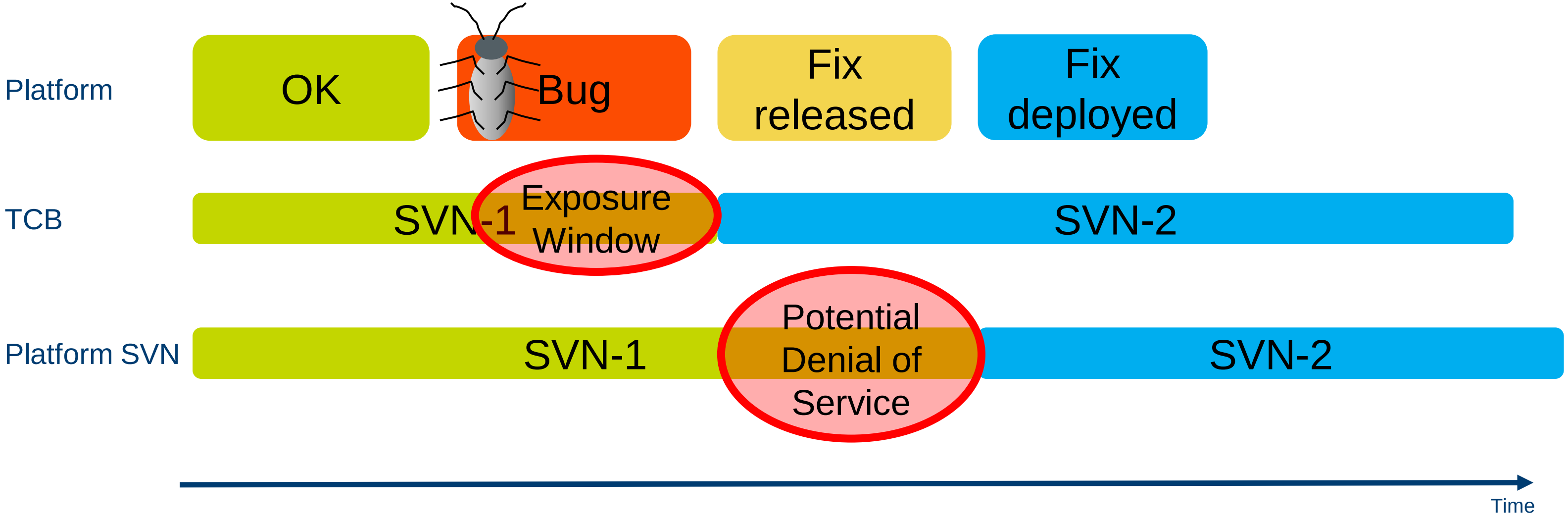
Platform SW initiates re-provisioning to join the EPID group that reflects the new security level

- Without re-provisioning, the platform might be up to date, but 3rd parties won't know about it

TCB update timelines

Timely bug fixes is important to reduce the denial-of-service window

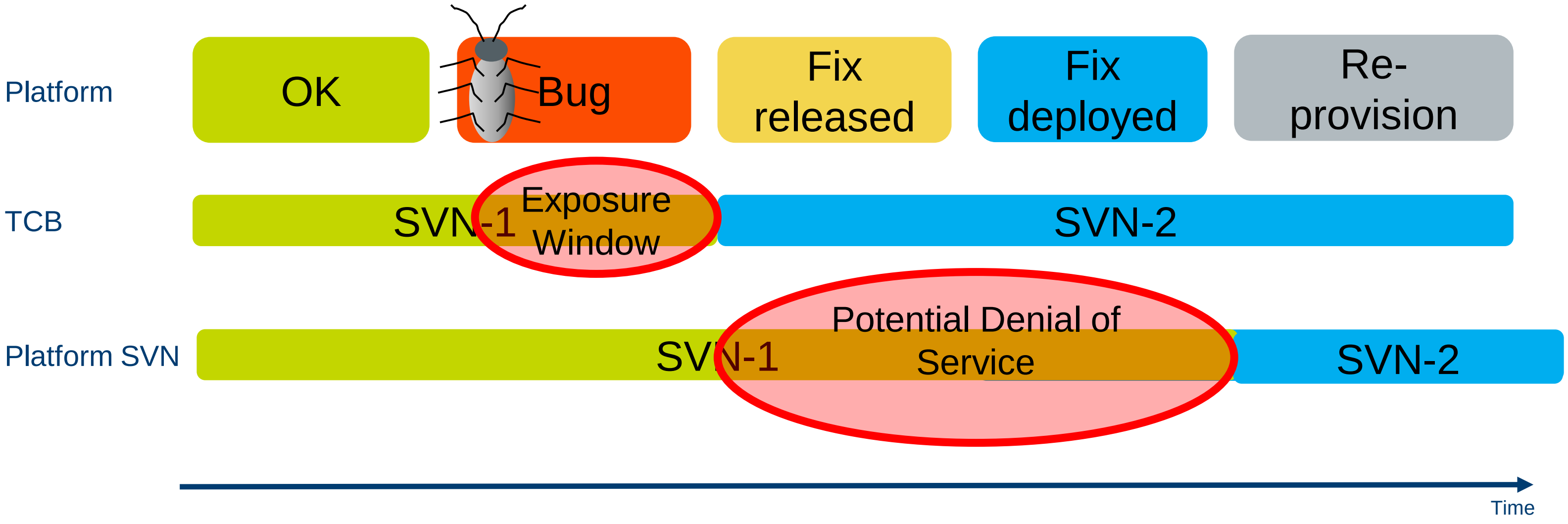
However, need also to reprovision platform, or else 3rd party vendors won't know the true status of the platform



TCB update timelines

Timely bug fixes is important to reduce the denial-of-service window

However, need also to reprovision platform, or else 3rd party vendors won't know the true status of the platform



Summary

From Dream to Reality

Intel started to work on SGX a very long time ago

After several iterations, SGX1 architecture was defined and we started implementation on the 6th generation Core (codenamed Skylake)

On September 2015 6th generation Core with SGX was announced

Since then

- SGX architecture continues expanding
- Academic researches work hard to evaluate the new technology
- SW vendors start implementing solutions based on SGX for clients and servers
- Cloud Service Providers started offering solutions based on SGX

intel®

experience what's inside™