Software Guard Extensions (Intel® SGX) Attestation and Sealing

HASP - 2013

June-24-2013

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® 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

- The challenge
- Attestation
- Sealing
- Tying it all together

The Challenge – Provisioning Secrets to the Enclave

- An enclave blob is in the clear before instantiation
  - Sections of code and data could be encrypted, but their decryption key can't be pre-installed
- Secrets must 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

Enclave Lifecycle

Launch

- An enclave is built and launched

Attestation

- The enclave proves its identity to a remote party

Provisioning

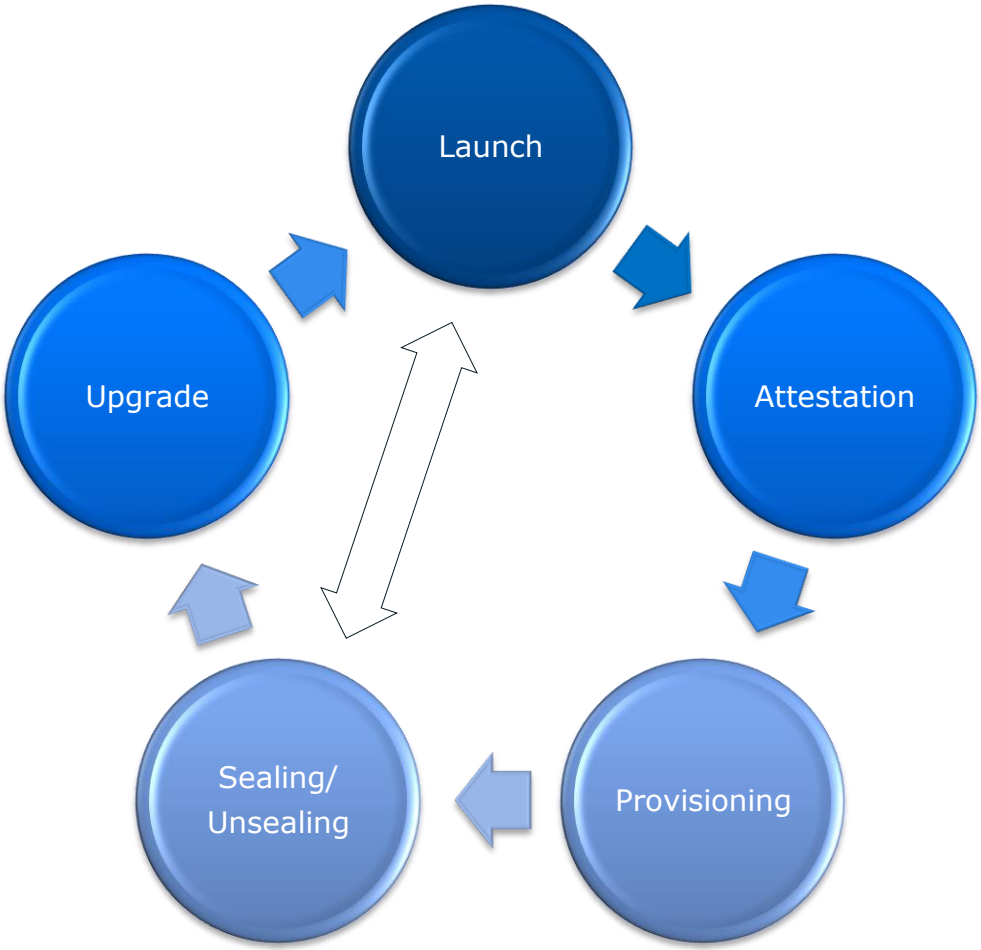
- The enclave receives secrets from the remote party

Sealing/Unsealing

- The enclave securely exports secrets for future use

Upgrade

- From time to time updates to enclave software are installed



Trustworthiness

- A service provider must 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

EREPORT & EGETKEY

- EREPORT
  - Generates a REPORT that is signed by a cryptographic MAC using a REPORT KEY
  - The REPORT includes:
    - MRENCLAVE – software TCB
    - MAC – hardware & firmware TCB
    - User Data – aux trusted information (e.g. ephemeral session key)
- EGETKEY
  - Provides an enclave with a persistent key
    - REPORT KEY for attestation verification
    - SEAL KEY for sealing
      - Enclave Identity sealing key
      - Sealing Identity sealing key

Attestation

Attestation – Software TCB

- When building an enclave, Intel® SGX creates a 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

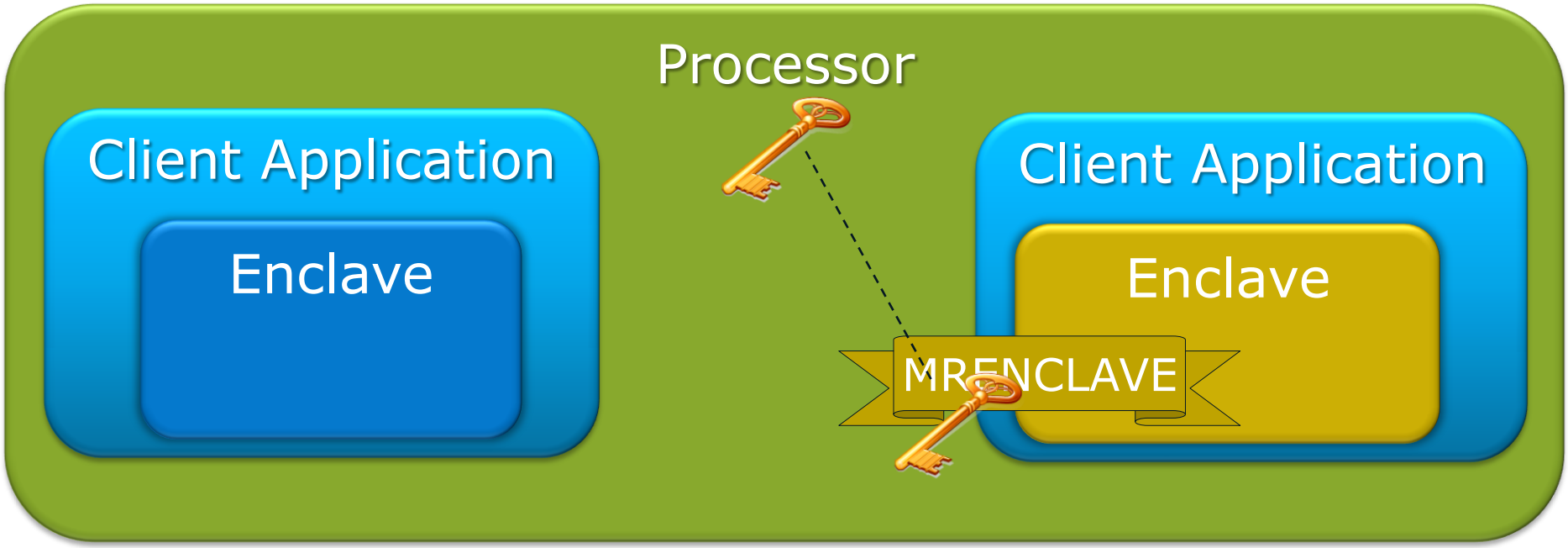
TCB component	Attestation
CPU hardware & firmware	Symmetric - CPU REPORT KEY
Software	MRENCLAVE

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

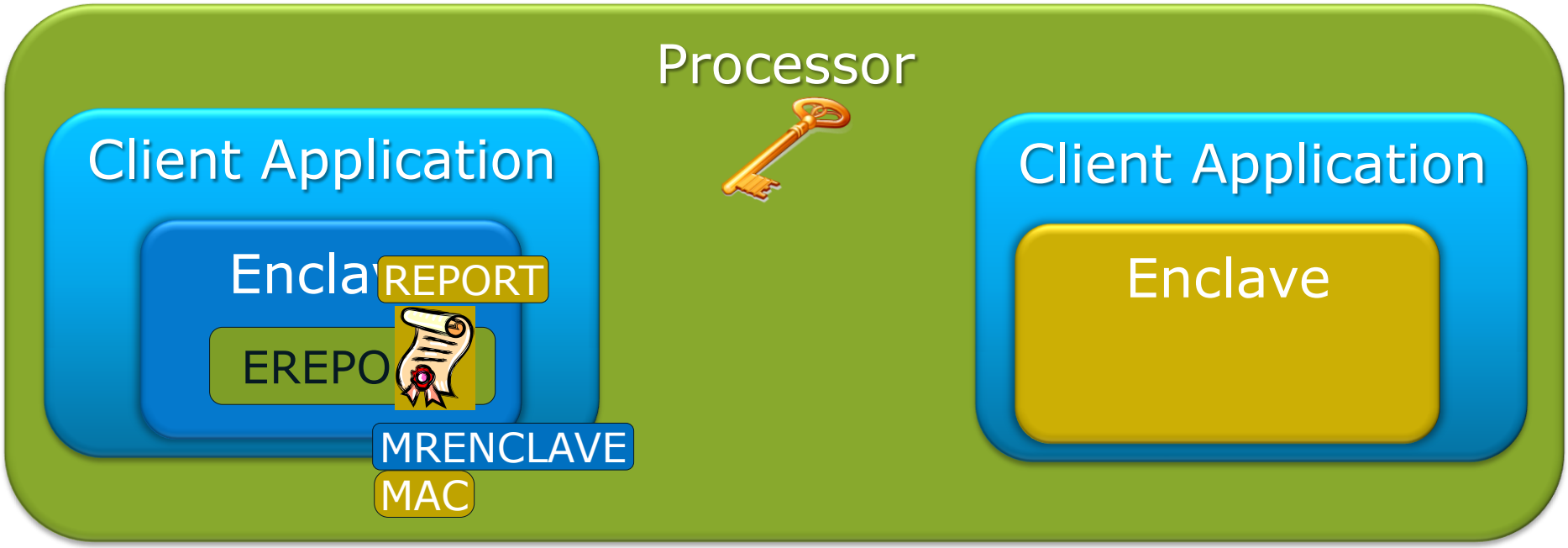
TCB component	Attestation
CPU hardware & firmware	Asymmetric - Intel® EPID
Software	MRENCLAVE

Local Attestation - Flow



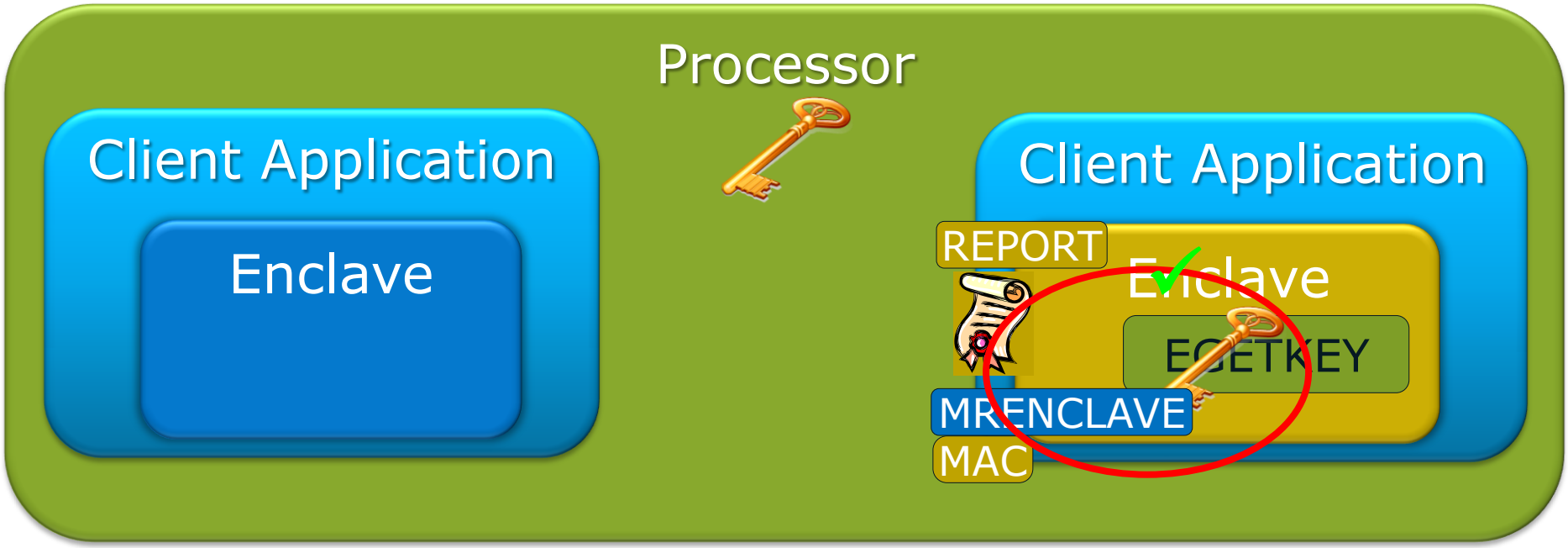
1. Verifying enclave sends its MRENCLAVE to reporting enclave
2. Reporting enclave creates a cryptographic REPORT that includes its MRENCLAVE and sends to verifier
3. Verifying enclave obtains its REPORT key and verifies the authenticity of the REPORT

Local Attestation - Flow



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

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

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

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

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

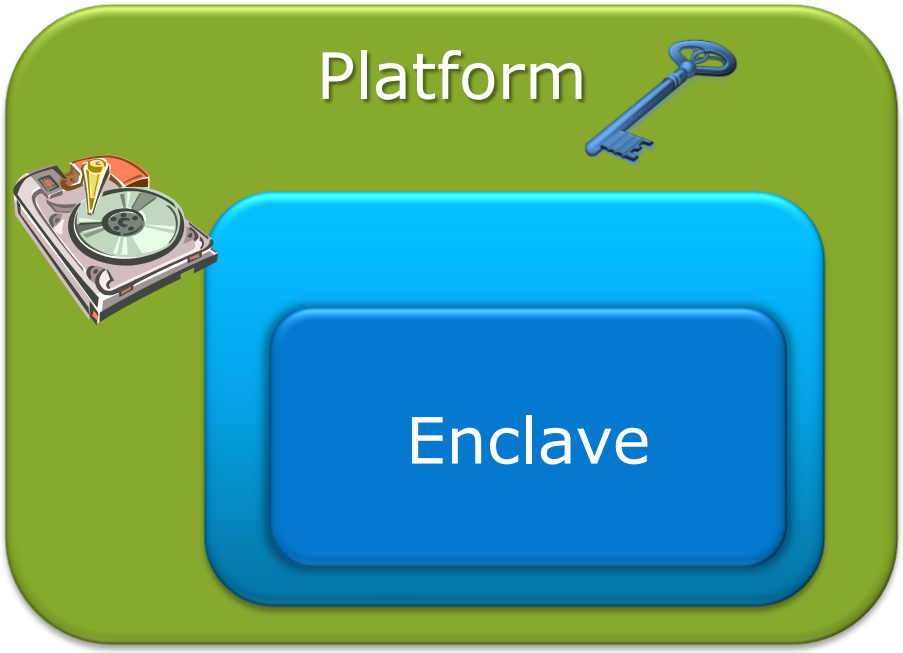
- "Sealing": Cryptographically protecting data when it leaves the enclave.
  - Totally under software's control
  - Many policies possible
- "Sealing Identity" of an enclave is a combination of:
  - Sealing Authority
  - Product ID
    - Typically the applications Product ID
  - Security Version Number (SVN)
    - Product's security level
    - Incremented by the authority as security vulnerabilities get fixed

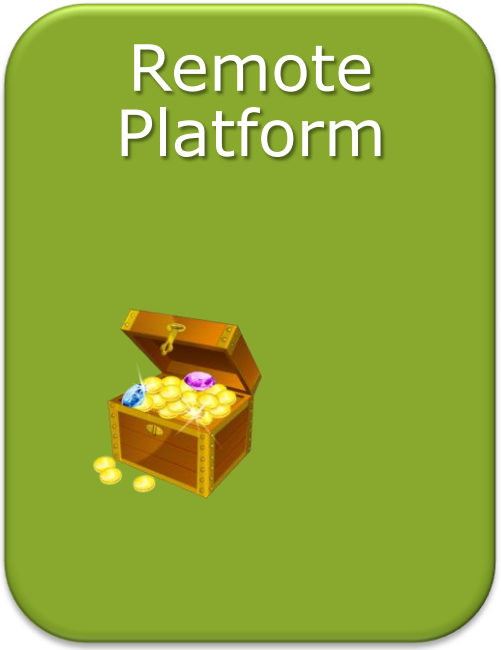
Intel® SGX provides the way to obtain persistent keys

Sealing Policies

- Sealing to Enclave Identity
  - EGETKEY bases the key on the value of the enclave's MRENCLAVE
  - Ensures that only this specific software instance will be able to regenerate the same SEAL KEY
- Sealing to Sealing Identity
  - EGETKEY bases the key on the value of the Sealing Identity
  - Ensures that the same SEAL KEY can only be regenerated by:
    - The same Sealing Authority
    - The same Product ID
    - An SVN >= Current SVN

Sealing - Flow





1. Secret provided to the Enclave
2. Enclave calls EGETKEY and obtains a SEAL KEY
3. Enclave wraps the secret with the SEAL KEY and stores the blob on disk

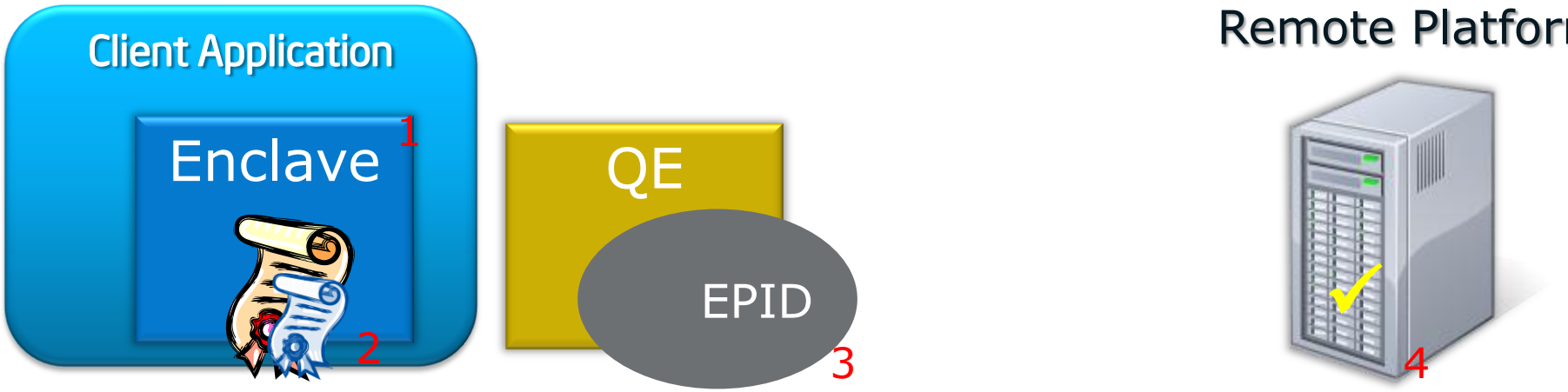
Sealing - Flow

1. Secret provided to the Enclave
2. Enclave calls EGETKEY and obtains a SEAL KEY
3. Enclave wraps the secret with the SEAL KEY and stores the blob on disk

Sealing - Flow

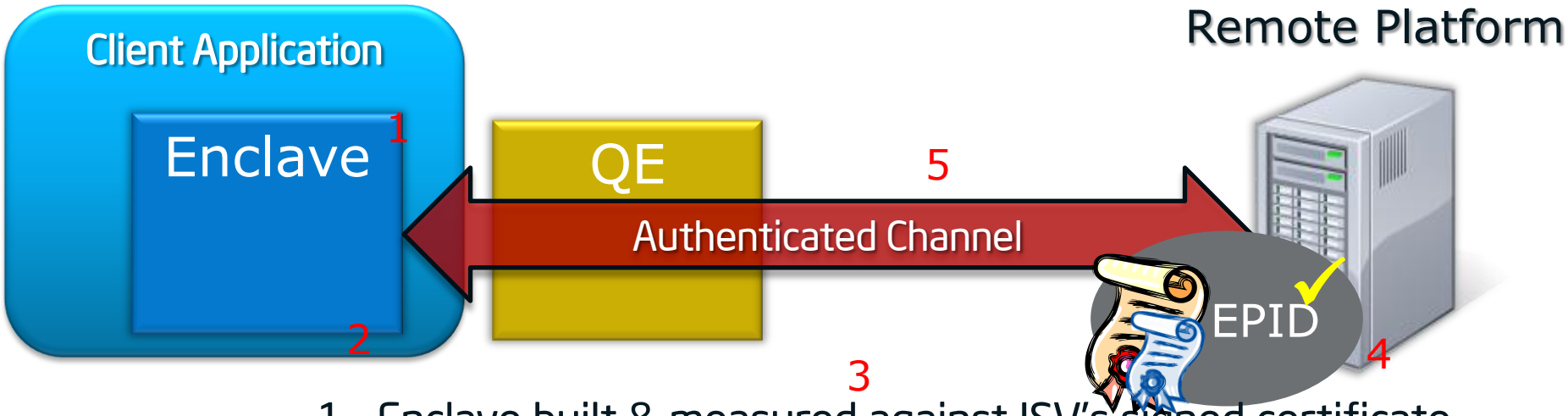
1. Secret provided to the Enclave
2. Enclave calls EGETKEY and obtains a SEAL KEY
3. Enclave wraps the secret with the SEAL KEY and stores the blob on disk

Example: Secure Transaction



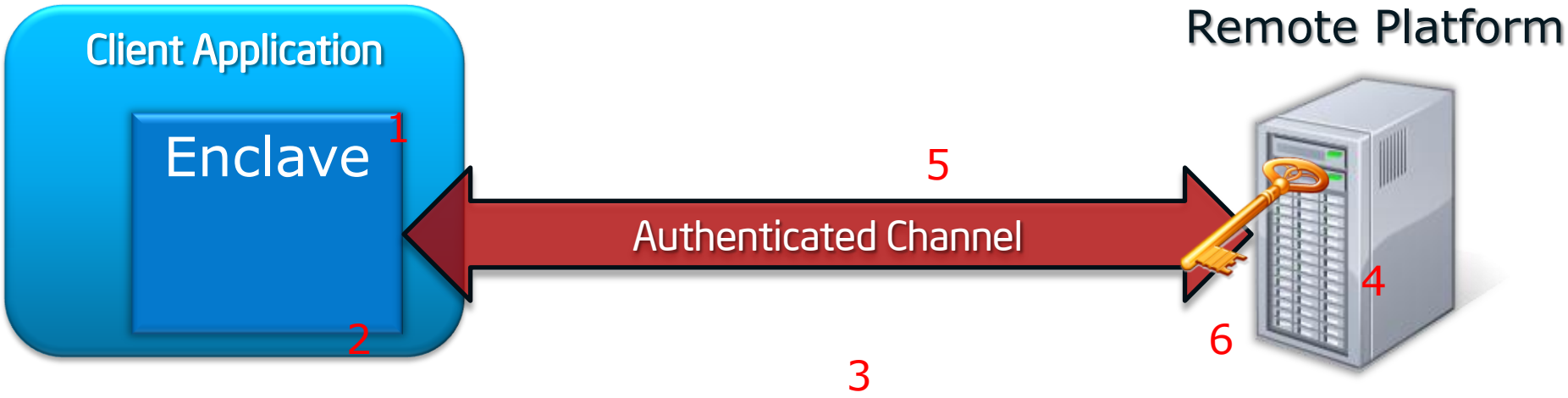
1. Enclave built & measured against ISV's signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key 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



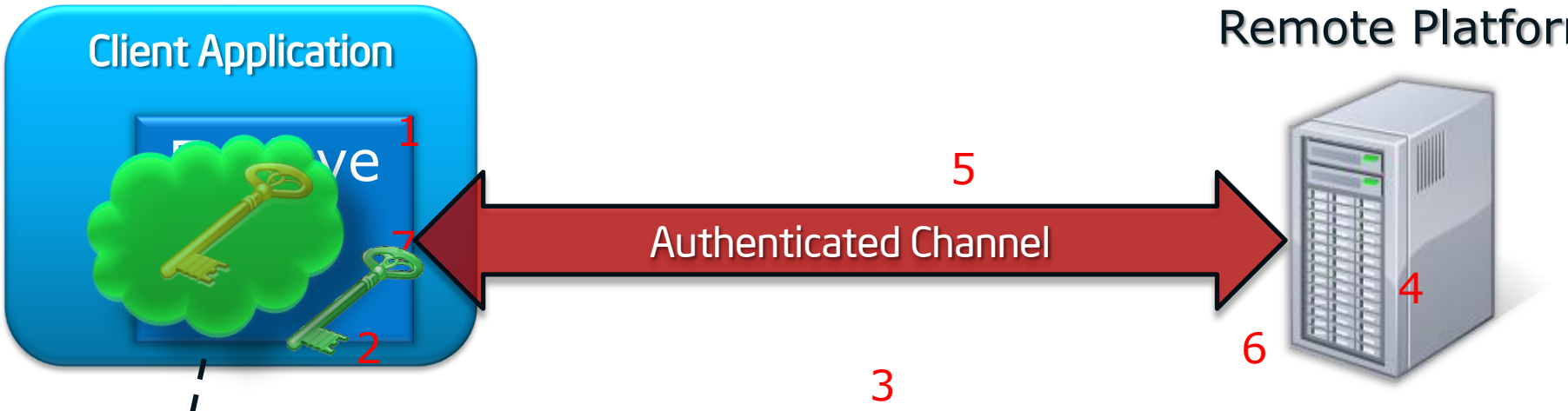
1. Enclave built & measured against ISV's signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key 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

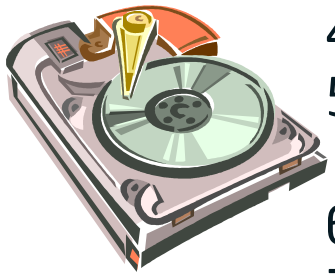
Example: Secure Transaction



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



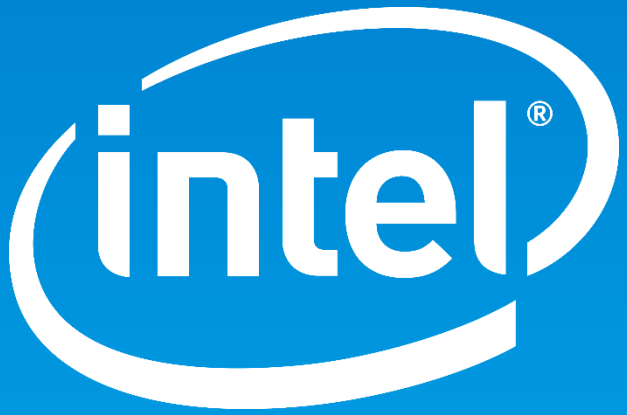


1. Enclave built & measured against ISV's signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key 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

Summary

- Intel® SGX's *EREPORT* and *EGETKEY* instructions provide unique attestation and sealing capabilities to protected software containers (a.k.a enclaves)
- Attestation
  - Local attestation between 2 entities running on the same platform
  - Remote attestation between an entity running in an enclave and a remote 3rd party entity
- Sealing
  - A sealing key that is unique to a software's instance
  - A sealing key that is shared between different versions of the same software as long as they have equal or better Security Version Numbers
    - For offline migration of data to newer versions of the application

Thank You


intel