Introduction to TEE

Amrita Center for Cyber Security Systems and Networks

Overview

What is TEE

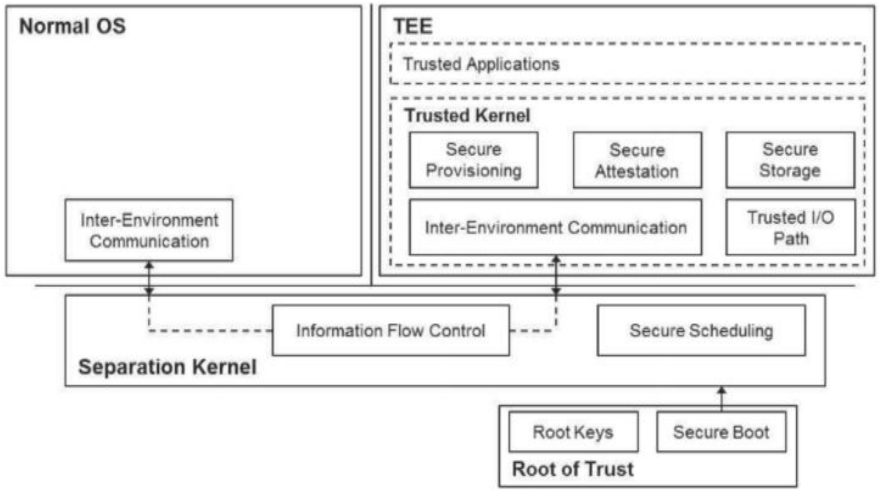
Different Types

Intel SGX

ARM TrustZone

Trusted Execution Environment

- Trusted Execution Environment (TEE) is a tamper-resistant processing environment
- Protects both its runtime states and stored assets
- Run on a separation kernel
    - Security kernel that simulate a distributed environment



TEE features

1. Isolated execution: Separation kernel divides system into partitions. Isolation of data, shared resources, information flow & errors
2. Secure boot: Each step in the boot chain cryptographically verifies the next
3. Secure scheduling: Tasks running in TEE should not interfere or delay tasks in main OS
4. Secure storage: Confidentiality, integrity and freshness of stored data
   Authorization required to access data
5. Remote attestation: Prove trustworthiness to third parties
6. Trusted I/O path: Safe communication between TEE and peripherals

TEE use cases

Content Protection

- IP streaming
- DRM
- Key protection
- Content protection

Mobile Financial Services

- mBanking
- Online payments
- User authentication
- Transaction validation

Corporate/government

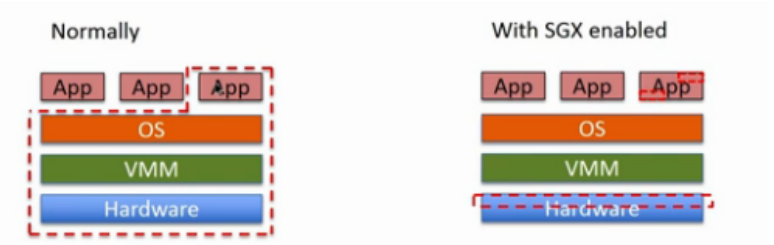
- Secure networking
- Secure email
- BYOD
- User authentication

Different TEE Implementations

TEE	Vendor	Architecture used
Intel SGX	Intel	Intel SGX
QSEE	Qualcomm	ARM Trustzone
KNOX	Samsung	ARM Trustzone
Kinibi	Trustonic	ARM Trustzone
TrustedCore	Huawei	ARM Trustzone
Open Virtualization	Open source	ARM Trustzone
ObC	Nokia	ARM Trustzone
OP-TEE	Open source	ARM Trustzone
AMD Platform Security Processor (PSP)	AMD	AMD PSP

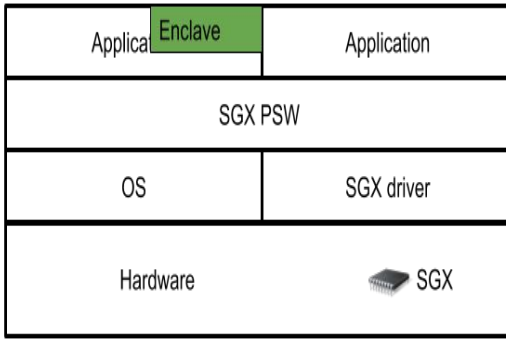
Intel SGX

- Intel Software Guard Extension (SGX) is a set of 18 instructions introduced by Intel to protect applications from privileged processes
- Introduced in Intel Skylake Microarchitecture
- Extra hardware chip inside the CPU
  - Same CPU core used for normal and trusted execution
- Security guarantees: Confidentiality and Integrity
- Application code run inside secure memory area called “enclave”
  - Enclave code - Trusted part of application that run inside enclave



Intel SGX

- Features
  - Isolated Execution: Allows trusted execution of user-level code by providing secure memory area
    - To provide protection against higher privilege software such as rootkits
  - Memory encryption engine to encrypt data when inside RAM
    - To prevent bus snooping and memory dump attacks
  - Attestation: To establish trustworthiness of execution environment
    - To confirm that the processing environment is not tampered
  - Secure Storage: Data store security by sealing data
    - To safely encrypt the application secrets after the execution
  - Code integrity check of trusted code through code signing
    - To detect code tampering at runtime

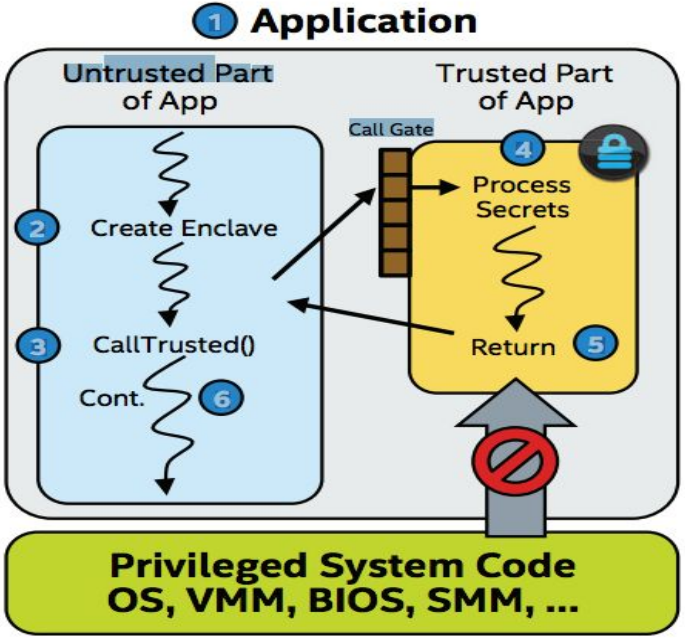


Intel SGX

- Limitations
  - Does not provide Secure Boot, Trusted I/O path, Secure Scheduling
  - No system calls, file operations allowed inside enclave
    - E.g. call to printf, read, write
  - Maximum enclave memory limit is 128MB.
  - Enclave code can only be in C/C++.
  - Requires source code modifications to port already existing applications to SGX.
  - Doesn’t give protection against side channel attacks, cache timing attacks
  - Requires a license from Intel to get full protection by SGX

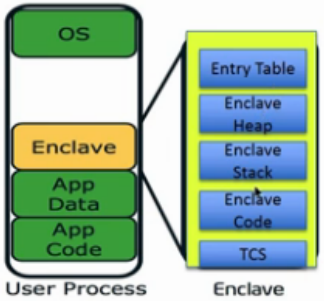
Intel SGX Runtime Execution

1. App built with trusted and untrusted parts
2. App runs & creates the enclave 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



source:
https://software.intel.com/sites/default/files/managed/50/8c/Intel-SGX-Product-Brief.pdf

Enclave



- Enclave has its own code and data areas
  Provides confidentiality and integrity
  With controlled entry points
- However, enclave code and data cannot be accessed from outside the enclave not even by the operating system.
- TCS: Thread control Structure
  (SGX supports multi-threading;
  one TCS for each thread supported)

Enclave Properties

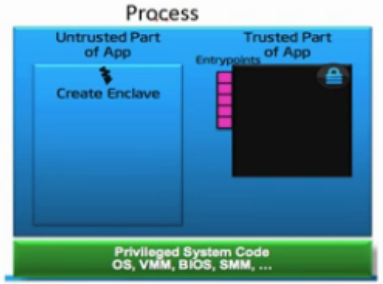
- Achieves confidentiality and integrity
  - Tampering of code / data is detected and access to tampered code / data is prevented.
- Code outside enclave cannot access code/data inside the enclave
- Even though OS is untrusted, it should still be able to manage page translation and page tables of the enclave
- Enclave code and data
  - Enclave code and data is in the clear when in the CPU package (eg. Registers / caches), but unauthorized access is prevented
  - Enclave code and data is automatically encrypted it leaves the CPU package

Instruction set for Enclave

- Privileged Instructions
  - Creation related: to create, add pages, extend, initialize, remove enclave
  - Paging related: evict page, load an evicted page
- User level instructions
  - Enter enclave, leave enclave
  - Interrupt related: asynchronous exit, resume

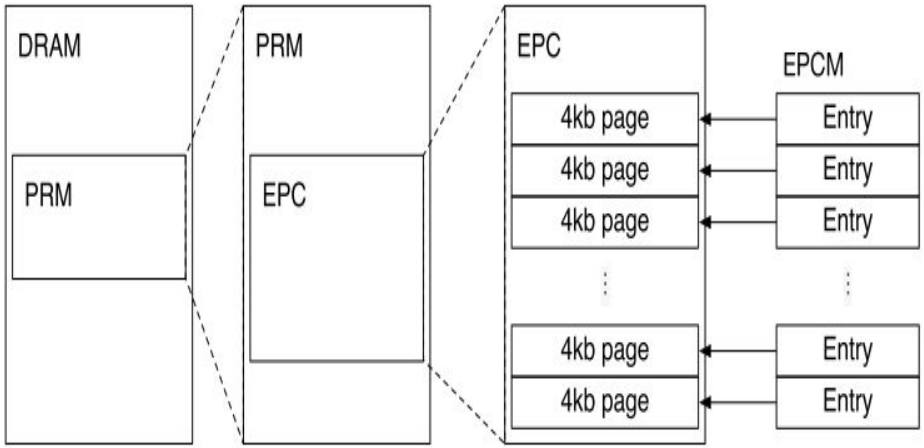
ECREATE Instruction

- Creates a SECS (SGX enclave control structure)
  - Contains global information about the enclave
- System software can choose where (in the process virtual space) the enclave should be present
- Also specifies
  - Operating mode (32/64 bit)
  - Processor features that is supported
  - Debug allowed



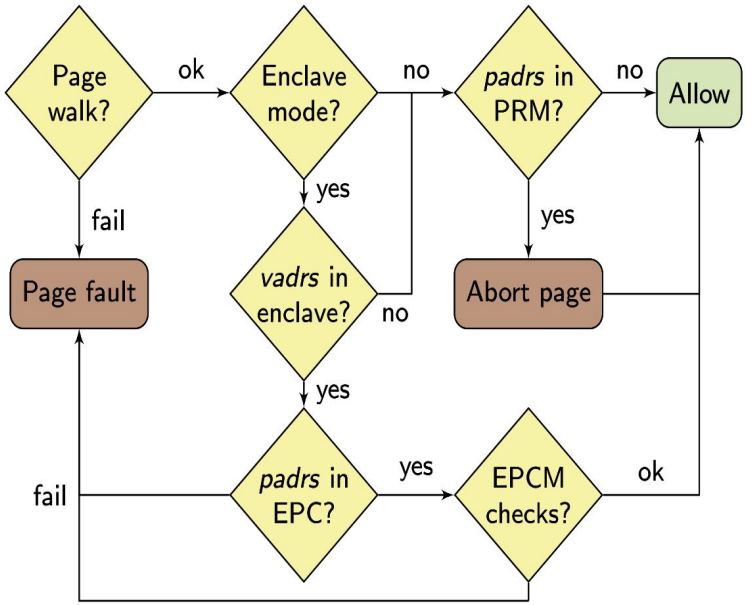
Isolated Execution

- Memory for enclaves are allocated from reserved portion (PRM) of DRAM
  - PRM - Processor Reserved Memory
- Enclave Page Cache (EPC) - 4KB pages for enclave code/data
- EPCM - Bitmap for EPC
  - To keep track of EPC



Isolated Execution

1. Only enclave code can access enclave data
2. OS manages the virtual to physical address translation
    a. Additional checks to prevent unauthorized access



Memory Encryption Engine

- Dedicated hardware to encrypt EPC pages
  - Implemented as a extension of Memory Controller
- To prevent memory tapping, cold boot attacks and replay attacks
  - Memory Tapping - Dumping the RAM contents/monitoring the system bus
  - Cold Boot - Attacker tries to dump RAM contents by hard reset the machine
  - Replay attack - Replacing a data block with another valid data block (especially during bus transfers)
- Encryption/Decryption keys generated during boot time
  - Changes during every boot
- Decryption happens inside CPU just before execution
- Enforces integrity checks while enclave page is written back.

Attestation

- Attestation is used to prove the authenticity (trustworthiness) of an enclave prior to provisioning a secret
- Use Cases
  - Banking application provides the transaction PIN after verifying the platform
  - DRM application provide media files to trustworthy customers
- Provisioning - Process of delivering the secret (e.g. PIN) after verification
- Secret: anything developer wants to hide
  - Data e.g. biometric data or decryption keys
  - Code e.g. payload of a malware
- Trusted code is built without the secret
- Attestation is an optional step - developer’s choice

Attestation

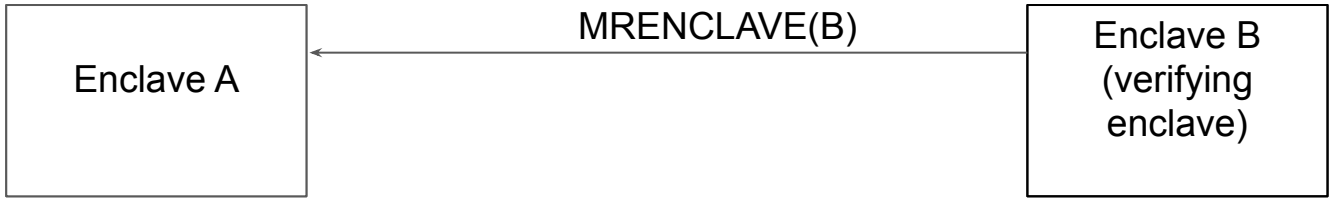
- 2 types
  - Local Attestation - Between 2 enclaves on the same platform
  - Remote Attestation - Confirm to a remote server that the application is running on a trustworthy environment
    - Local attestation using Quoting enclave
    - Verification by Service Provider and IAS
- Two SGX instructions - EGETKEY, EREPORT
- Terminology
  - MRENCLAVE - Hash digest that identifies an enclave
  - MRSIGNER - Hash digest that identifies an enclave developer

Use Case

- Remote Attestation
  - Media Service Application provide the user with the “License Key” to proceed only after confirming that the
    - Media service application requesting the key itself is not tampered
    - Executing platform is authentic and can be trusted
- Local Attestation
  - 2 Enclaves of the same Application/Vendor have to exchange secrets

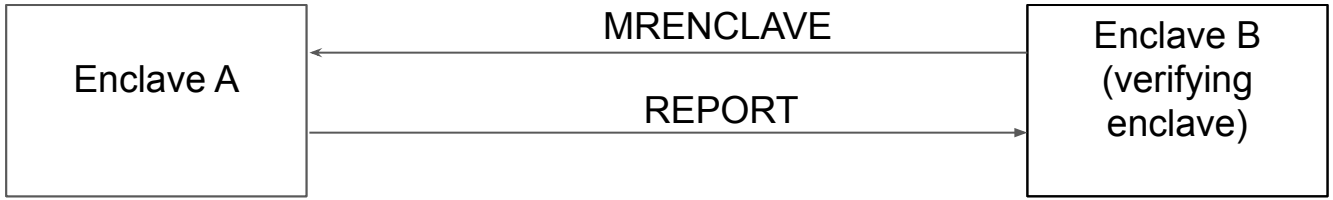
Local Attestation

Enclave A request the secret from Enclave B. B sends its MRENCLAVE



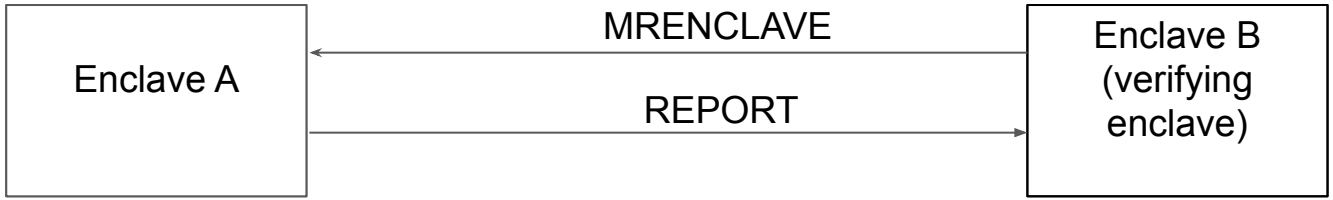
- MRENCLAVE - SHA256 (Log)
- Log - contents of page
  - relative position of the pages
  - security flags of the pages

Local Attestation



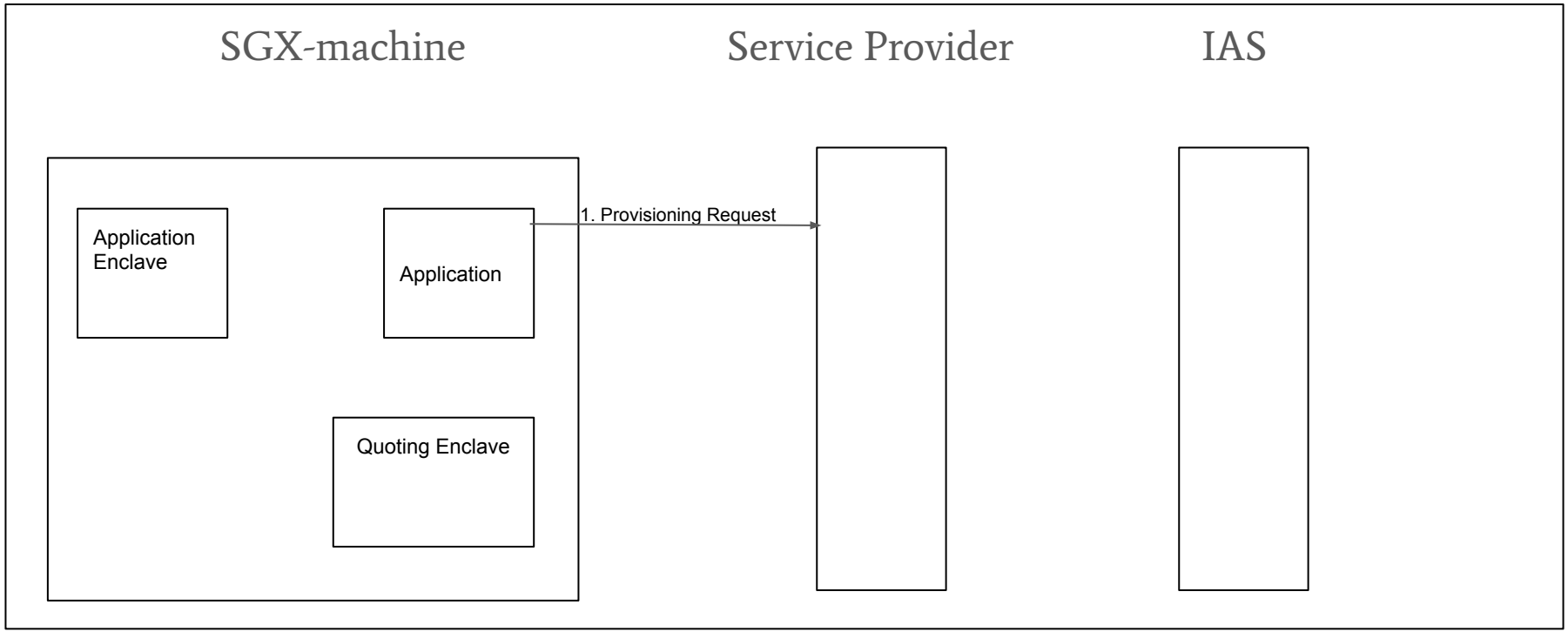
- Enclave A invoke EREPORT using MRENCLAVE of Enclave B
- REPORT
  - two identities of the enclave (MRSIGNER & MRENCLAVE)
  - the attributes associated with the enclave
  - a message authentication code (MAC) tag
  - additional information

Local Attestation

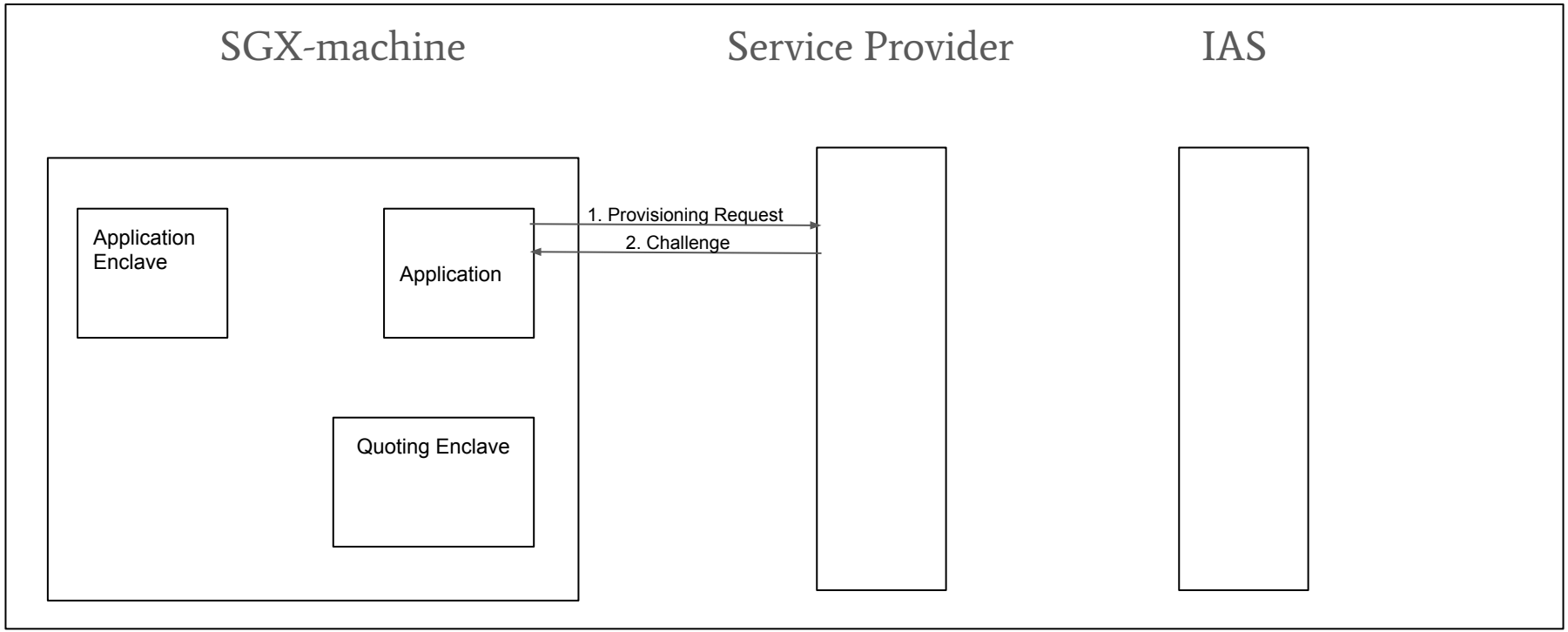


- B invokes EGETKEY to get Report key
- Recomputes the MAC and compare it
- If MAC matches, implies both enclaves run on the same platform

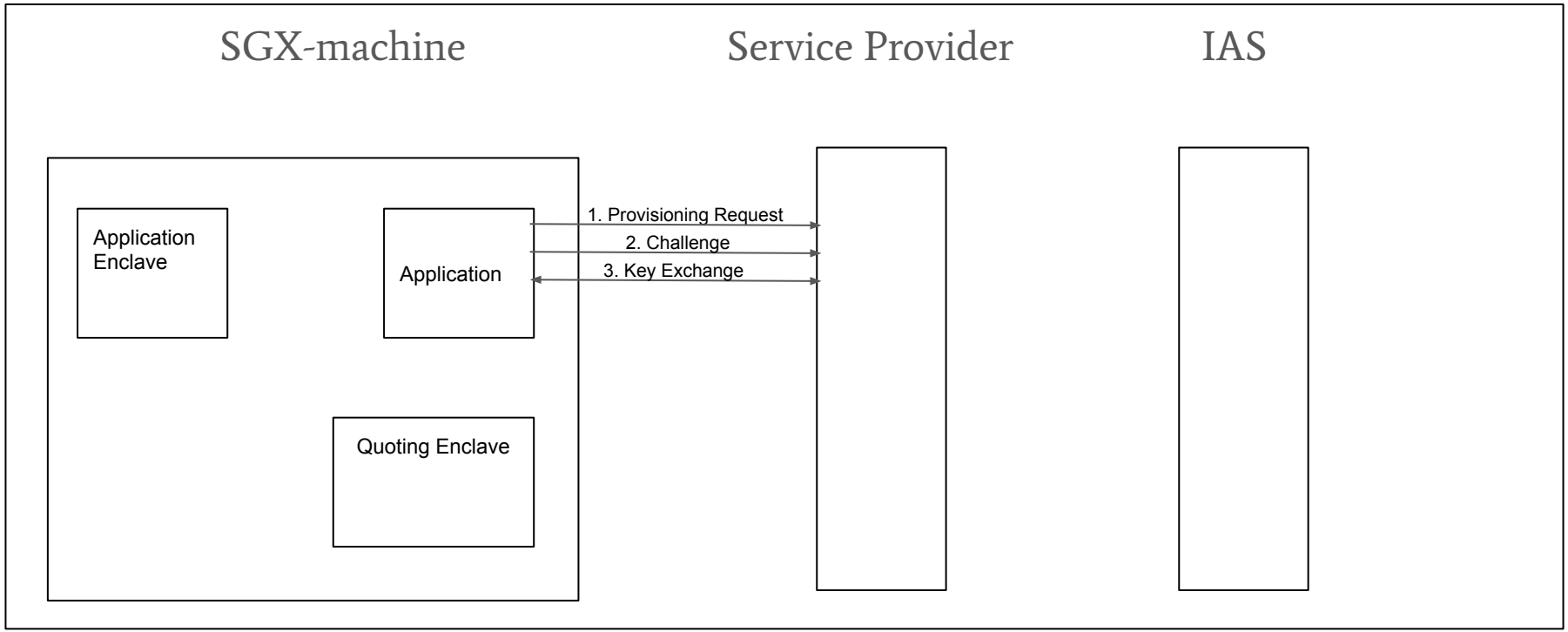
Remote Attestation



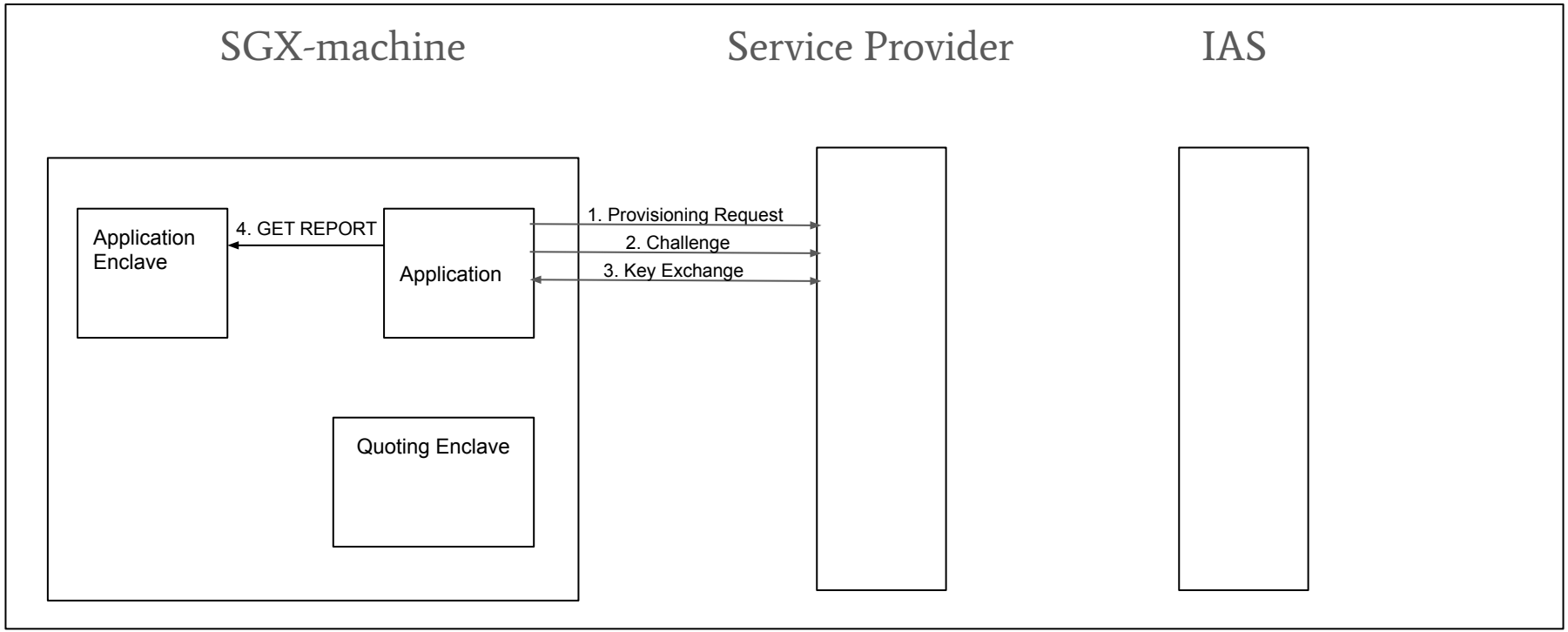
Remote Attestation



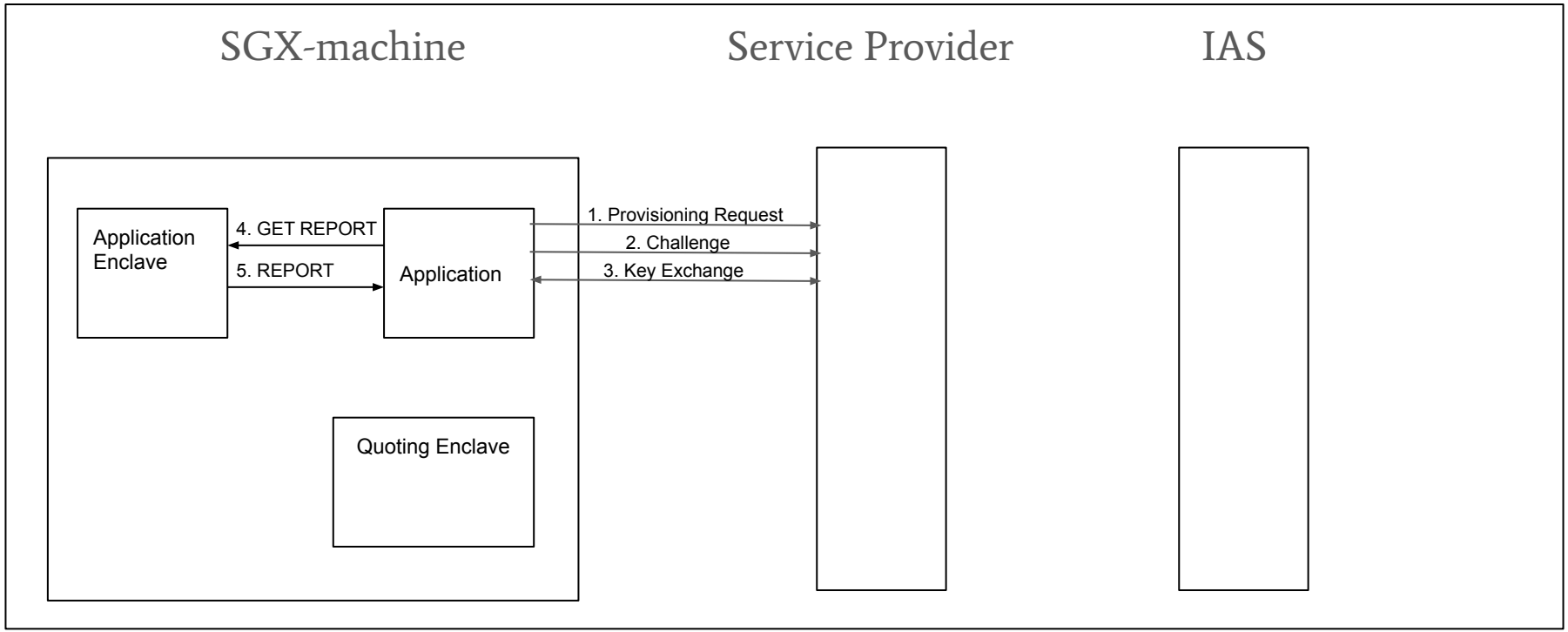
Remote Attestation



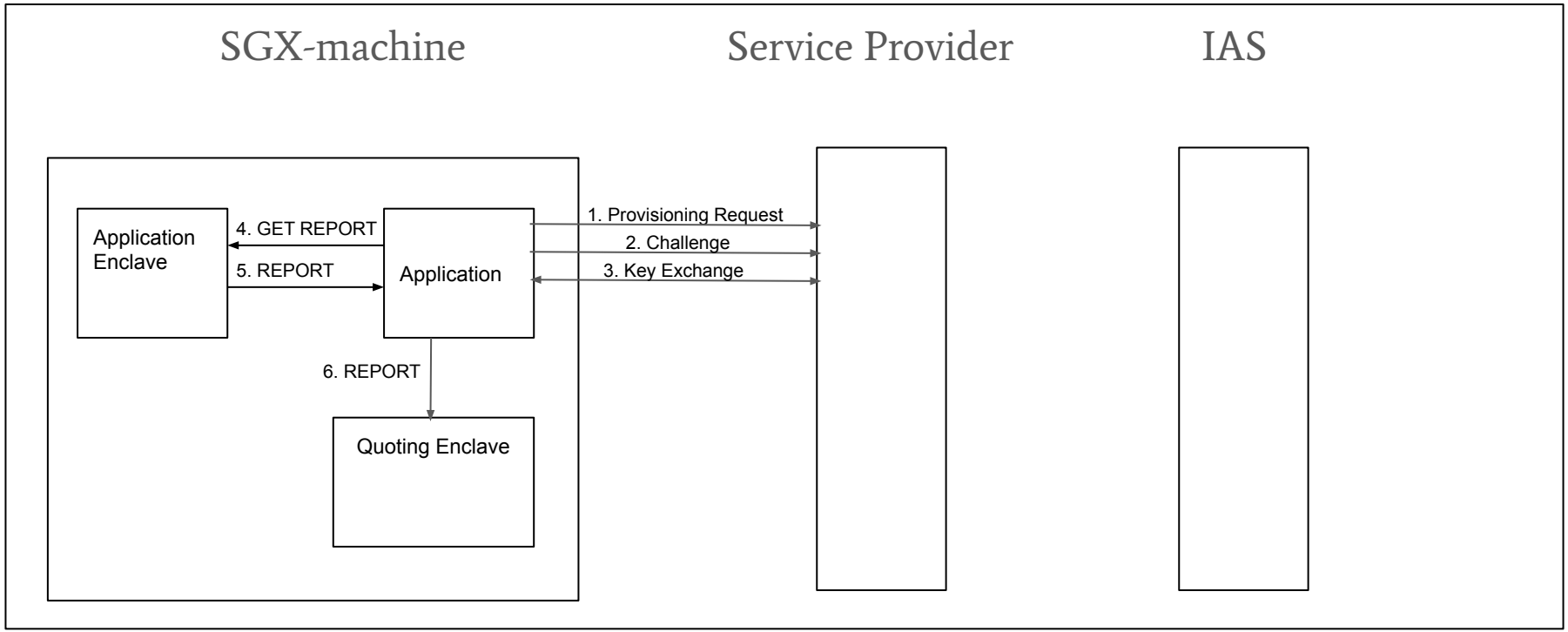
Remote Attestation



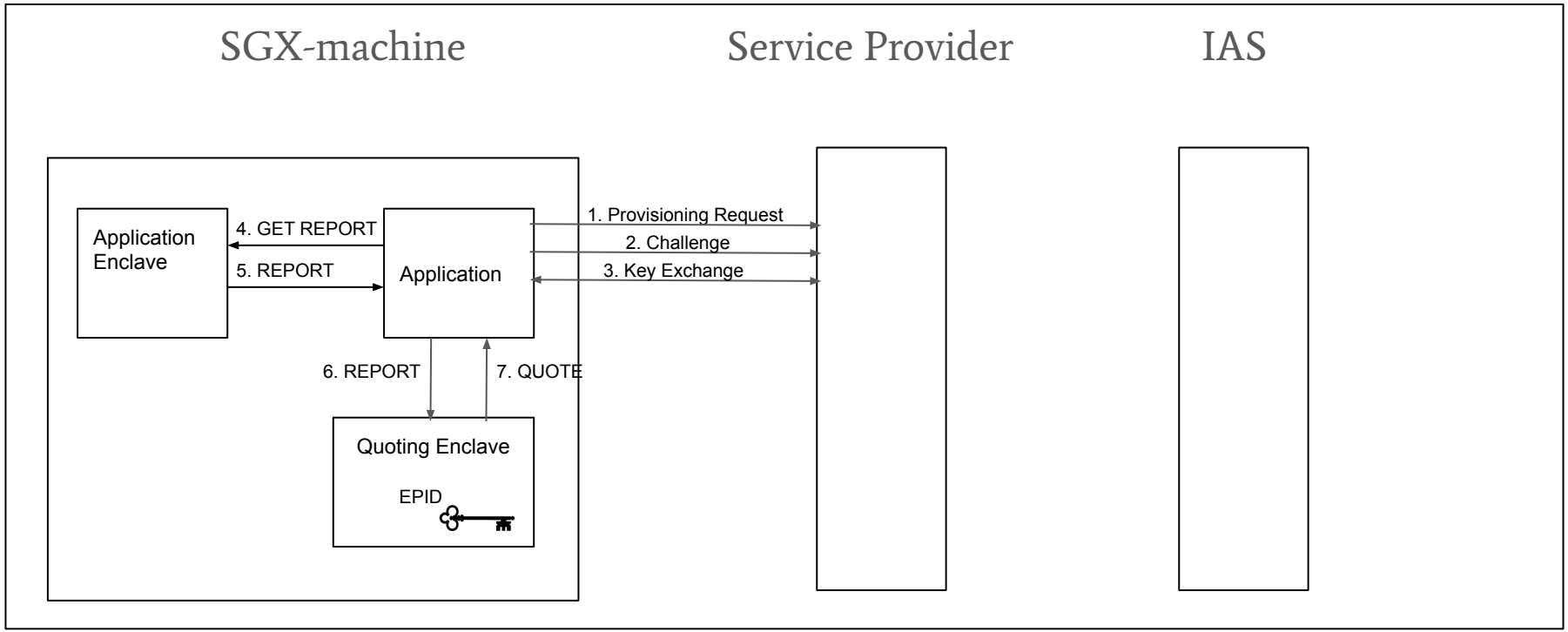
Remote Attestation



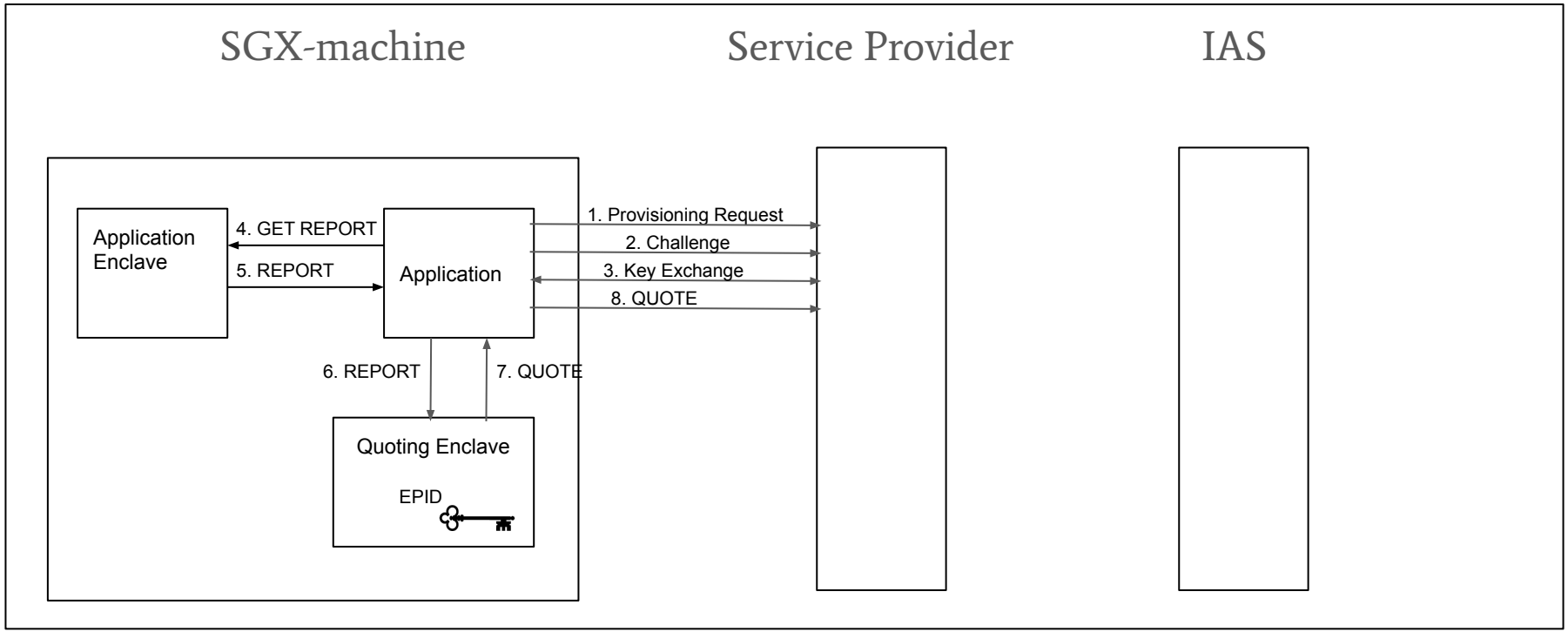
Remote Attestation



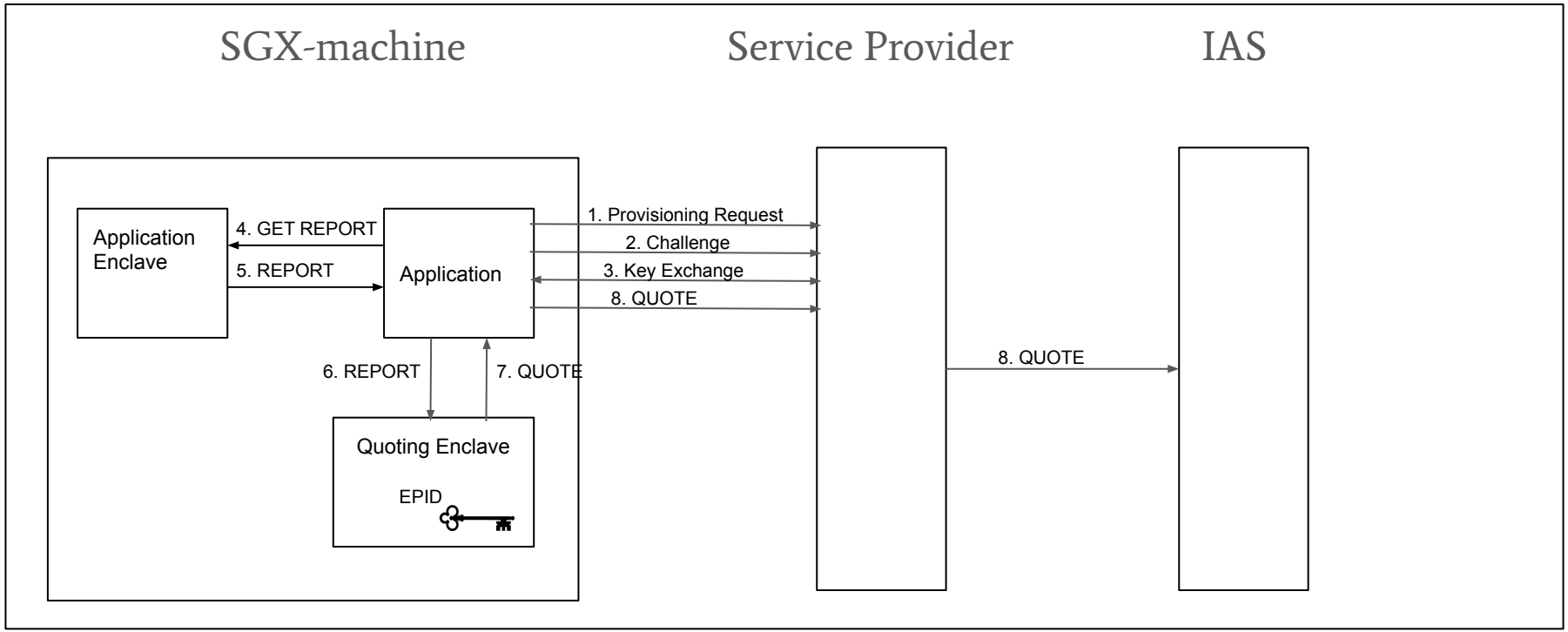
Remote Attestation



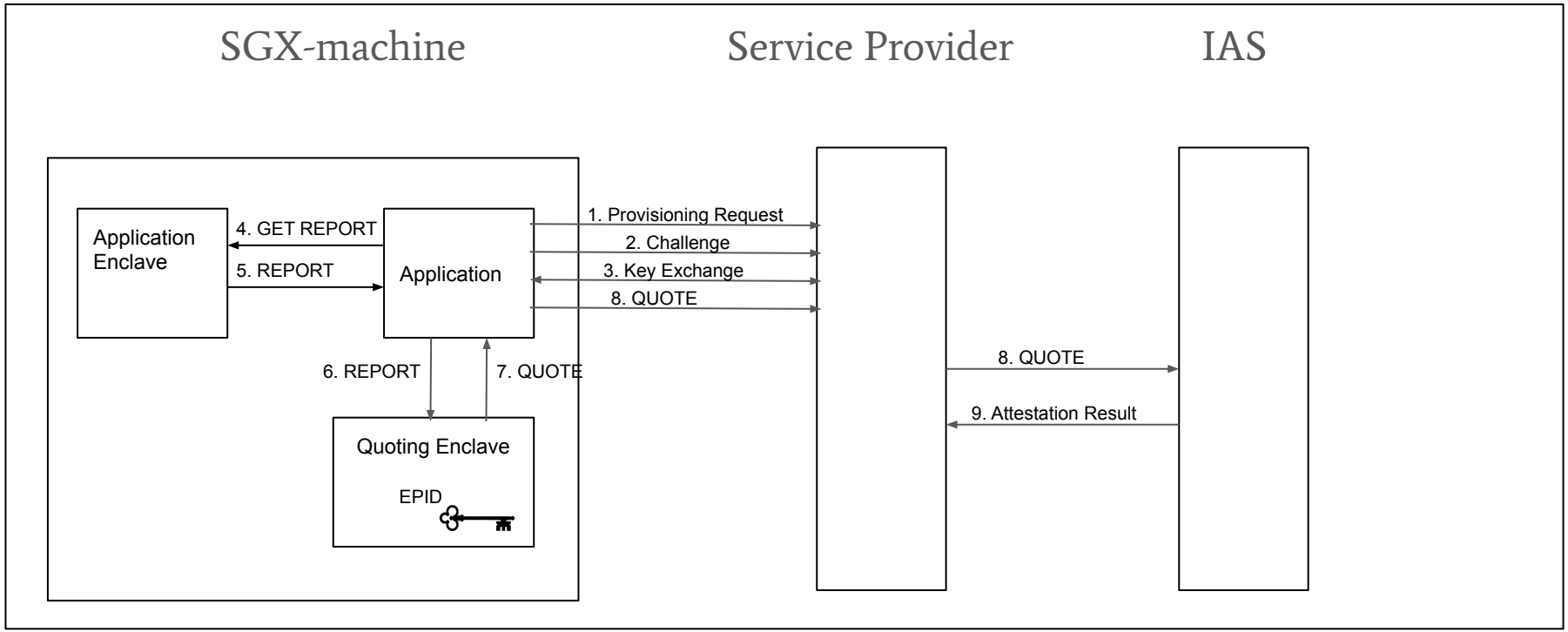
Remote Attestation



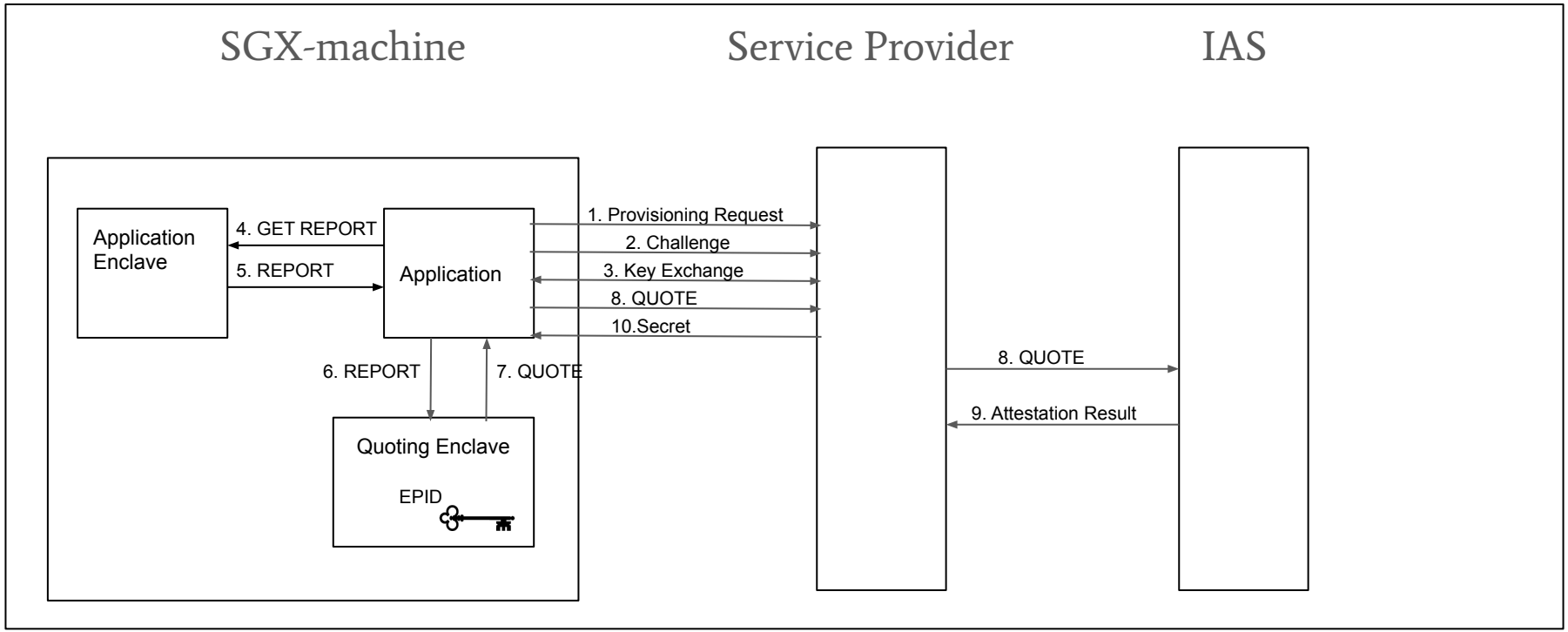
Remote Attestation



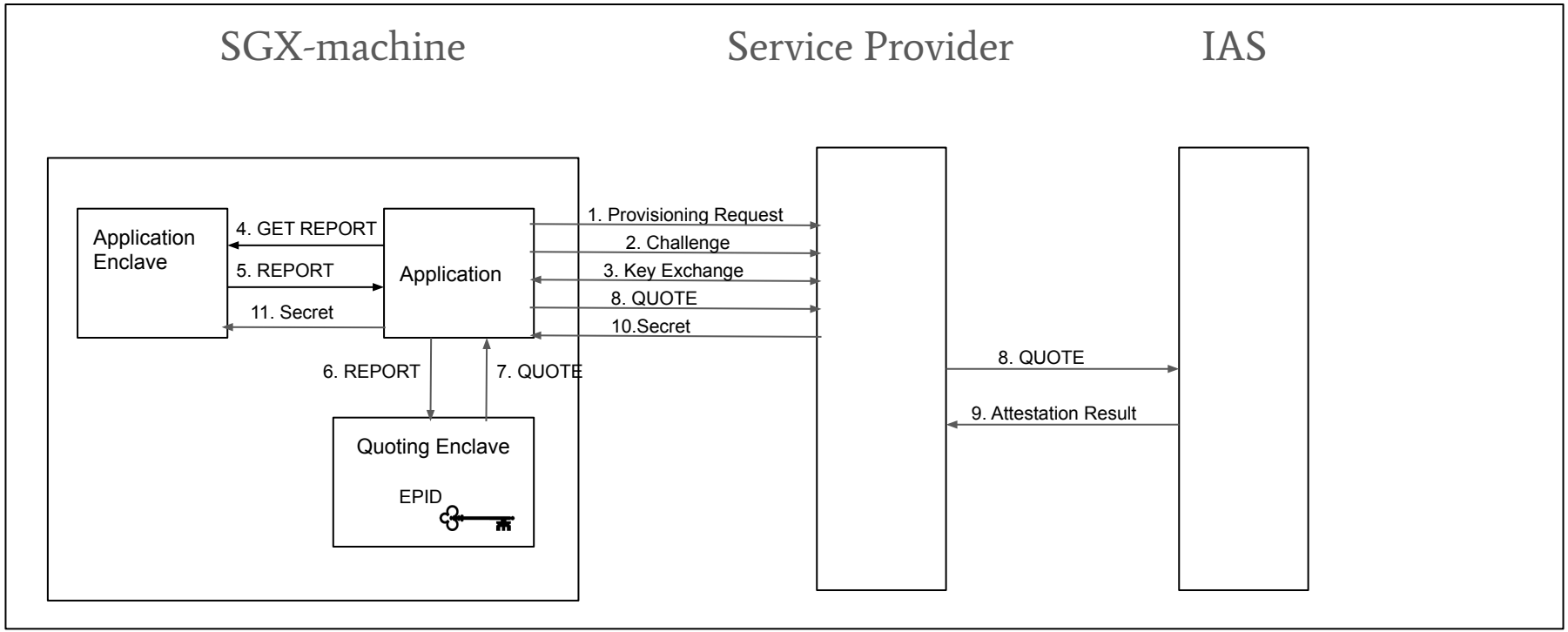
Remote Attestation



Remote Attestation



Remote Attestation



Security Mechanisms in SGX: Sealing

- To securely store application secrets into the disk after enclave execution
  - Encryption
  - SGX instruction *EGETKEY* generates encryption/decryption key.
- Two ways: Based on enclave identity or sealing identity.
  - Enclave identity
    - To EGETKEY, pass the enclave hash (MRENCLAVE)
    - The key can be regenerated back by an enclave with similar MRENCLAVE value
  - Sealing Identity
    - To EGETKEY, pass the public key hash of the enclave builder (MRSIGNER)
    - The key can be regenerated back by all enclave released by the enclave builder.
- Developers choice

Security Mechanisms in SGX: Code Signing

- Mechanism to enforce integrity of enclave code
- Developer signs the enclave code using Intel SGX SDK signing tool
- Signature: Enc_{priv}(SHA2 (trusted code + enclave attributes))
  - Enclave attributes: Product id, CPU version no, Vendor
- At runtime:
  - Decrypt (Signature) = Hash
  - Success if (Hash == ComputeHash (Enclave code + EA))

SGX Enclave Modes

SGX supports 3 different modes for application enclave - Specified during enclave build

1. Debug - Default mode
    a. For debugging SGX application
    b. Customized debugger can step into the enclave code
    c. All debug symbols present
    d. Compiler optimizations disabled
2. Release
    a. Used for release of SGX application
    b. Provides complete Intel SGX protection guarantees
    c. Compiler optimizations enabled
3. Simulation
    a. Does not require SGX hardware

Intel SGX Instruction Set

Supervisor Instruction	Description	EAX value
ENCLS[ECREATE]	Create an enclave	00H
ENCLS[EADD]	Add a page	01H
ENCLS[EINIT]	Create an enclave	02H
ENCLS[EREMOVE]	Remove an EPC page	03H
ENCLS[EDBGRD]	Read data by debugger	04H
ENCLS[EDBGWR]	Write data by debugger	05H
ENCLS[EEXTEND]	Extend EPC page measurement	06H
ENCLS[ELDB]	Load an EPC page as blocked	07H
ENCLS[ELDU]	Load an EPC page as unblocked	08H
ENCLS[EBLOCK]	Block an EPC page	09H
ENCLS[EPA]	Add version array	0AH
ENCLS[EWB]	Write back/invalidate an EPC page	0BH
ENCLS[ETRACK]	Activate EBLOCK checks	0CH

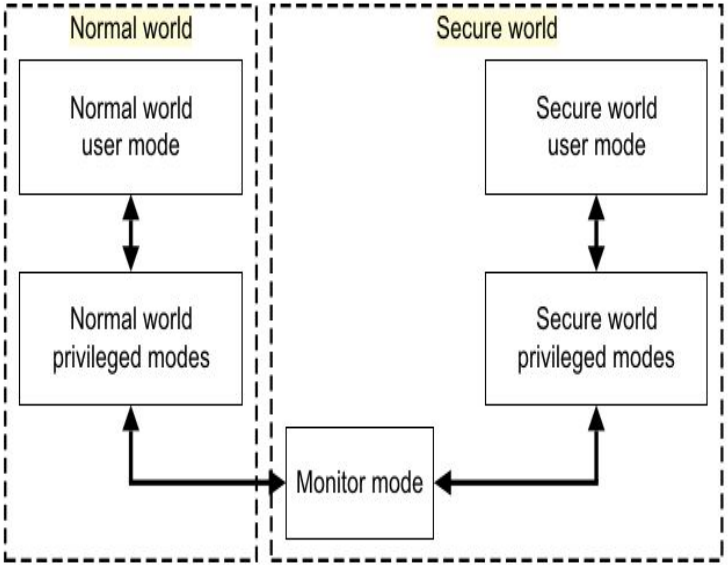
Intel SGX Instruction Set

User Instruction	Description	EAX value
ENCLU[EREPORT]	Create a cryptographic report	00H
ENCLU[EGETKEY]	Create a cryptographic key	01H
ENCLU[EENTER]	Enter an Enclave	02H
ENCLU[ERESUME]	Re-enter an Enclave	03H
ENCLU[EEXIT]	Exit an Enclave	04H

ARM TrustZone

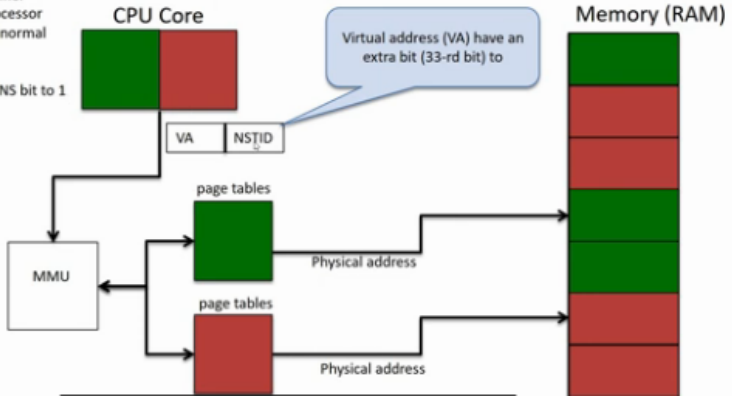
ARM TrustZone

- Supports two isolated environments: normal world & secure world
- Secure Monitor Call (SMC) instructions to switch between these worlds
- Non-secure (NS) bit in SCR indicates the world
- Trustlet - Trusted Application in Secure World



Memory Management

- Non Secure Table Identifier current state of the processor (0 if secure world / 1 if normal world
- If NSTID = 1 then force NS bit to 1



ARM TrustZone: Hardware Extensions

- Advanced Extensible Interface (AXI):
  - Extension to the system bus,
  - Used for data transfer
  - Active high non-secure bit that indicates whether the read or write is from the secure or non secure world
- Advanced Peripherals Bus (APB):
  - The peripherals are connected via APB
  - No provision to recognize the security state of the transaction
- AXI-to-APB bridge:
  - Checks the validity of the security state and accepts or rejects the access requests

ARM TrustZone: Hardware Extensions

- Cache Controller:
  - Both the worlds share the same cache
  - uses the NS bit to distinguish between cache lines that belong to each world
- Memory aliasing:
  - 32 bit physical address space for each of the worlds
  - 33rd bit is used to distinguish the world from which the transaction is from
- Memory management unit:
  - translation table accommodates the NS bit so as to indicate the world it belongs to

ARM TrustZone: Hardware Extensions

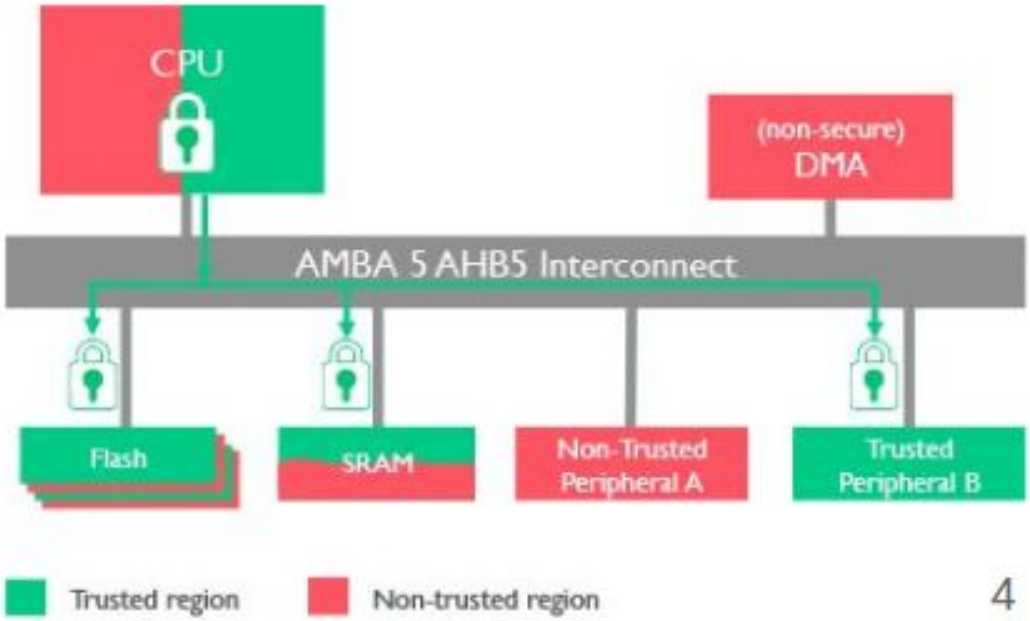


*figure from https://microsoftrnd.co.il/Press%20Kit/BlueHat%20IL%20Decks/GalBeniamini.pdf

ARM TrustZone: Software Extensions

- Separate OSs run in secure as well as non secure world
- OSs communicate with each through the monitor mode
- Normal world can enter the monitor world by interrupts, aborts or SMC instruction
- IRQ is the source interrupt for the normal world and FIQ is the source interrupt for the secure world
- Three vector tables for interrupt handling: secure world, normal world and monitor mode

SGX vs TrustZone

Features	Intel SGX	ARM TrustZone
Isolated Execution	Secure containers called enclaves created at runtime for execution of programs.	System divided into two: Normal world and secure world. Trustlets execution happen secure world.
Memory Limitation	Enclave stored in PRM. Max Enclave size 128 MB	Trustlets stored in “secapp-region” 100MB size
Context switching	Special instructions are used to switch into execution in an enclave.	Monitor mode facilitates switch between normal and secure world.
Communication with peripherals	No generic method for the enclaves to communicate with the peripherals.	Peripherals check the security bit on the communication bus to determine security state of transaction and accordingly share data.
Exception handling	Mostly handled outside the enclave.	3 different vector tables for each of normal world, secure world and monitor mode exceptions.