ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Cibersegurança

- Trusted Computing
- Trusted Execution Environments (TEEs)
- ARM TrustZone

ISEL – Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, 1 | 1959-007 Lisboa

Trusted Execution Environments (TEEs)

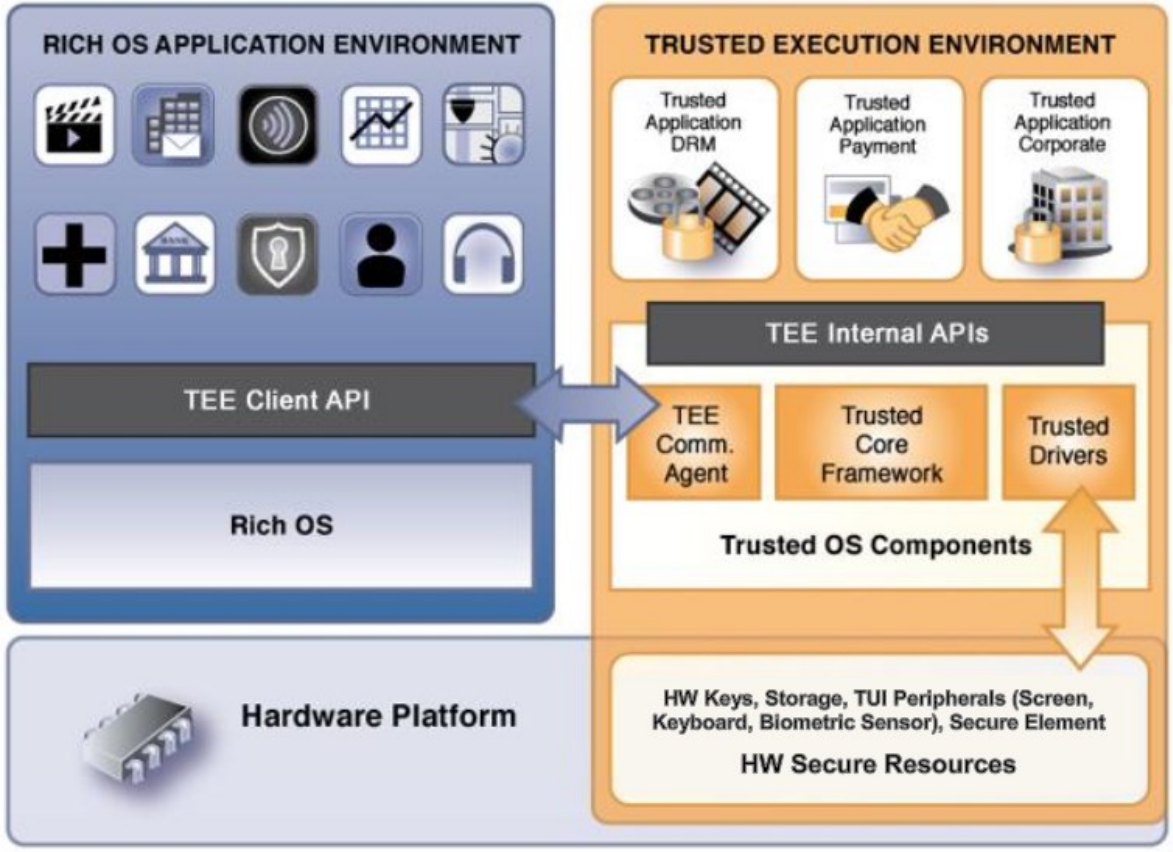
TPMs shortcomings

- Do not naturally provide isolated execution environments
- Only provide limited on-board storage and processing capability
- Do not provide trusted paths with sensor hardware

Trusted Execution Environments (TEEs)

TEE: What is it?

- A TEE is a secure area of the main processor of a connected device that ensures sensitive data is stored, processed and protected in an isolated and trusted environment.
- As such, it offers protection against software attacks generated in the standard Operating System (Rich OS) running on the Rich Execution Environment (REE).



Trusted Execution Environments (TEEs)

Requirements

- A TEE must guarantee
  - The authenticity of the executed code
  - The integrity of the runtime states (e.g. CPU registers, memory and sensitive I/O)
  - The confidentiality of its code, data and runtime states stored on a persistent memory
- It should provide remote attestation that proves its trustworthiness for third-parties.
- The threat model includes all software attacks and the physical attacks performed on the main memory and its non-volatile memory

Trusted Execution Environments (TEEs)

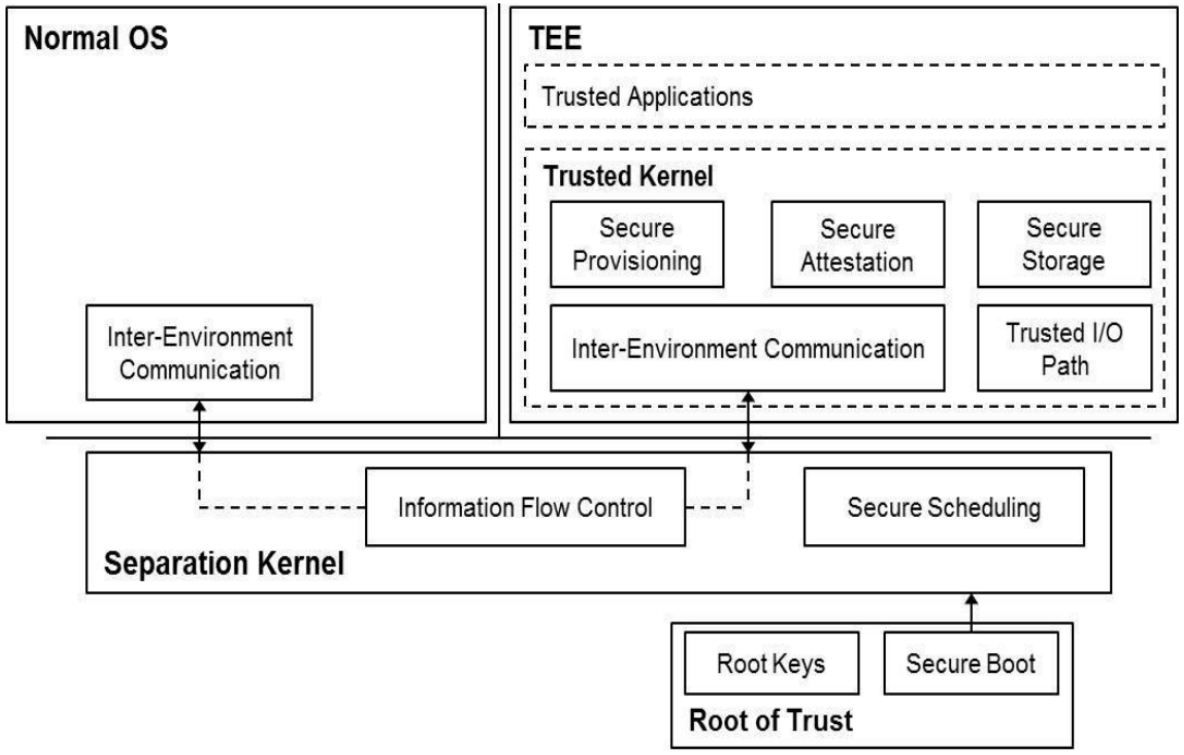
Security principles

- A TEE must adhere to certain basic security principles
  - Be part of the device secure boot chain (based on a RoT) and verify code integrity during each device boot
  - Provide hardware-based isolation from the device's rich OS environment to execute sensitive code
  - Isolate Trusted Applications (TAs) from each other
  - Provide secure data storage, using a hardware-unique key accessible only by the TEE OS to prevent unauthorized access and modification and any possibility of exploiting the data in other devices
  - Provide privileged and secure access to peripherals

Trusted Execution Environments (TEEs)

Building blocks

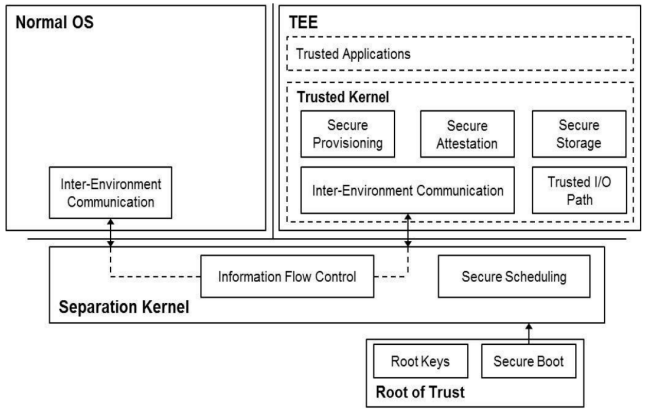
- Secure Boot
- Secure Scheduling
- Inter-Environment Communication
- Secure Storage
- Trusted I/O Path



Trusted Execution Environments (TEEs)

Building blocks

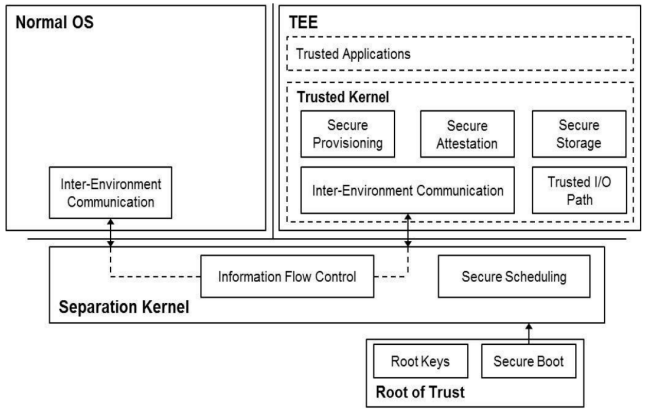
- Secure Boot
  - Assures that only code of a certain property can be loaded
  - If a modification is detected, the bootstrap process is interrupted
  - Involves establishing a chain of trust and a RoT



Trusted Execution Environments (TEEs)

Building blocks

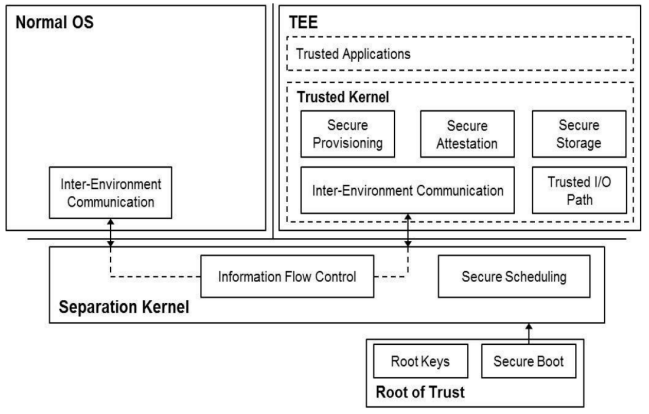
- Secure Scheduling
  - Assures a “balanced” and “efficient” coordination between the TEE and the rest of the system.
    - It should assure that the tasks running in the TEE do not affect the responsiveness of the main OS.
      - Often, the scheduler is designed preemptive!
    - The scheduler should take real-time constraints into consideration.



Trusted Execution Environments (TEEs)

Building blocks

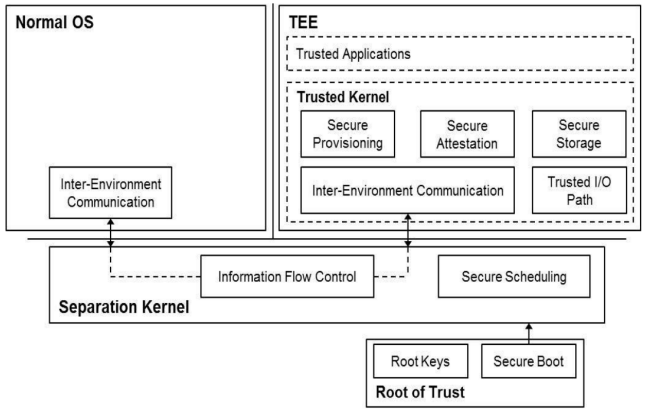
- Inter-Environment Communication
  - Defines an interface allowing the TEE to communicate with the rest of the system.
  - Has numerous benefits, but also introduces new threats
    - Message overload attacks
    - User and control data corruption attacks
    - Memory faults caused by shared pages being removed
    - Unbound waits caused by the noncooperation of the untrusted part of system.
  - 3 key attributes should be satisfied
    - Reliability (memory/time isolation)
    - Minimum overhead (unnecessary data copies and context switches)
    - Protection of communication structures.



Trusted Execution Environments (TEEs)

Building blocks

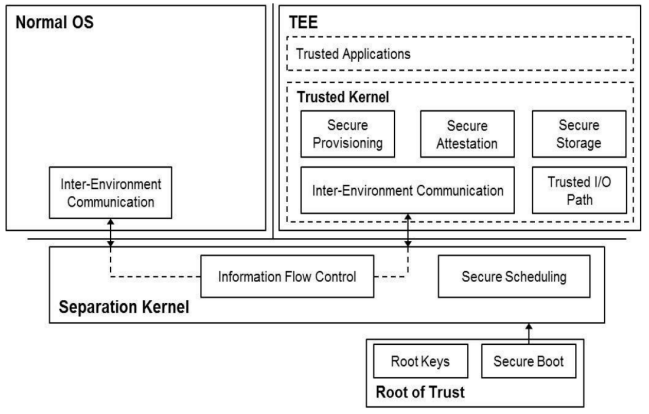
- Secure Storage
  - Storage where confidentiality, integrity and freshness of stored data are guaranteed, and where only authorized entities can access the data
  - A common way to implement secure storage is sealed storage.
  - Sealed storage is based on three components
    - Integrity-protected secret key that can be accessed only by the TEE
    - Cryptographic mechanisms, such as authenticated encryption algorithms
    - Data rollback protection mechanism, such as replay-protected memory blocks (RPMB)



Trusted Execution Environments (TEEs)

Building blocks

- Trusted I/O Path
  - Protects authenticity, and optionally confidentiality, of communication between the TEE and the peripherals, enabling broader functionality within the TEE
    - Allows a human user to directly interact with applications running inside the TEE
  - 4 classes of attacks are considered
    - Screen-capture attack
    - Key logging attack
    - Overlaying attack
    - Phishing attack



Trusted Execution Environments (TEEs)

Standardization

- GlobalPlatform is a non-profit industry association driven by over 100 member companies
- GlobalPlatform's specifications are regarded as the international standard for enabling digital services and devices to be trusted and securely managed throughout their lifecycle
- The first TEE specification was published in July 2010
  - Strongly inspired by/ based on the ARM TrustZone proposal
- Later, GlobalPlatfrom specified a hardware and a software architecture, both of which highlight full isolation between REE and TEE

Trusted Execution Environments (TEEs)

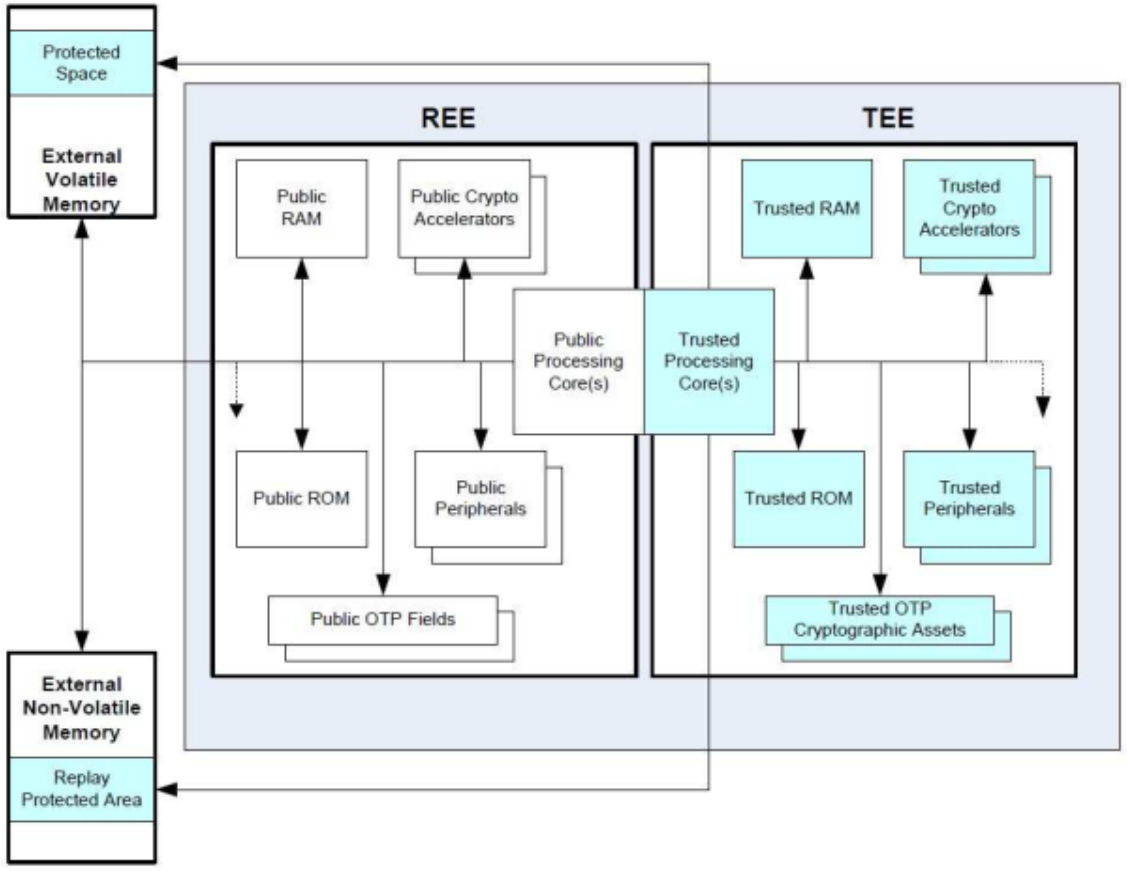
Standardization

- The GlobalPlatform TEE specification defines
  - A TEE system architecture
  - A range of TEE APIs
  - A TEE protection profile for Common Criteria compliance
  - Documentation about functional testing of two TEE APIs
    - The TEE Client API
    - The Internal Core API

Trusted Execution Environments (TEEs)

Standardization

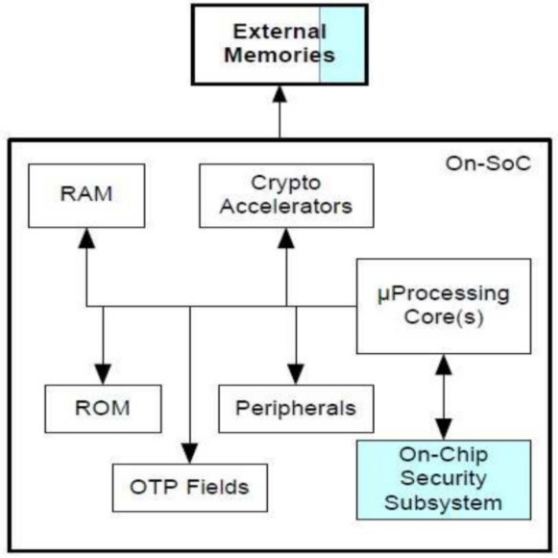
- Hardware architecture
  - The Rich Execution Environment (REE)
  - The Trusted Execution Environment (TEE)
  - Two storage areas
    - The External Volatile Memory
    - The External Non-Volatile Memory
    - Considered external because they are not part of REE or TEE
  - TEE components are considered trusted, while REE components are considered public, hence untrusted.



Trusted Execution Environments (TEEs)

Standardization

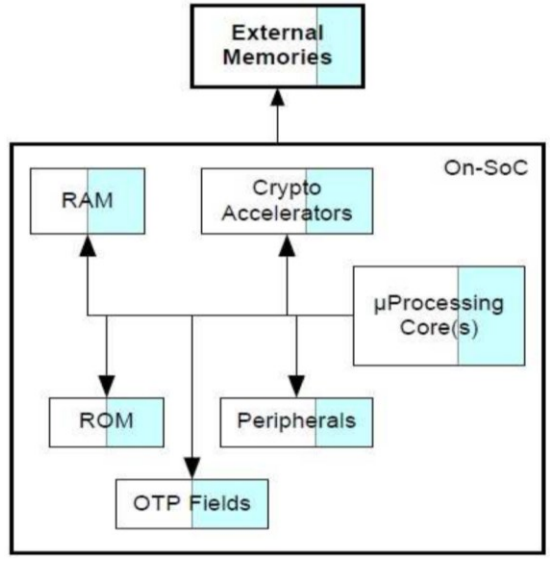
- Hardware architecture :: Possible implementation scenarios (1)
  - TEE is implemented as an On-Chip Subsystem inside the System-on-Chip (SoC)
    - The TEE can be considered as a library or an embedded secure element that includes a set of security services, e.g. key management, secure storage.



Trusted Execution Environments (TEEs)

Standardization

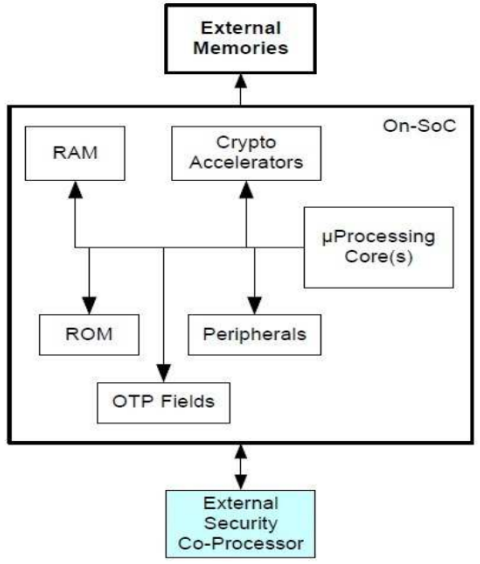
- Hardware architecture :: Possible implementation scenarios (2)
  - The TEE implementation maintains the SoC architecture
    - There are no new physical components and the TEE and REE share all the usual SoC components
    - 2 different ways to achieve such sharing
      - Peripherals are divided into trusted and untrusted peripherals.
      - Two state peripherals
        - Each peripheral has two states: trusted and untrusted
        - Over time, a peripheral switches from one state to another
        - However, at a given time its state is either trusted or untrusted
        - Requires an entity to manage the peripheral states



Trusted Execution Environments (TEEs)

Standardization

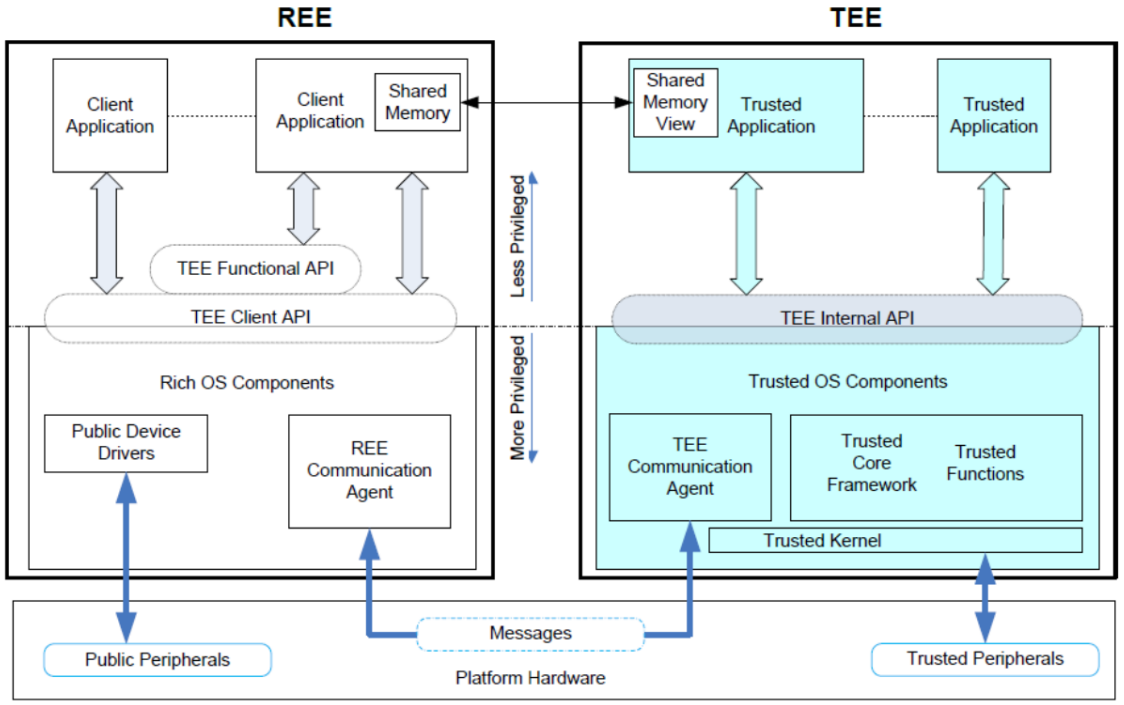
- Hardware architecture :: Possible implementation scenarios (3)
  - The TEE implementation maintains the SoC architecture, but the TEE is described as an external component
    - Typical implementations are TPM or Secure Elements (SEs)



Trusted Execution Environments (TEEs)

Standardization

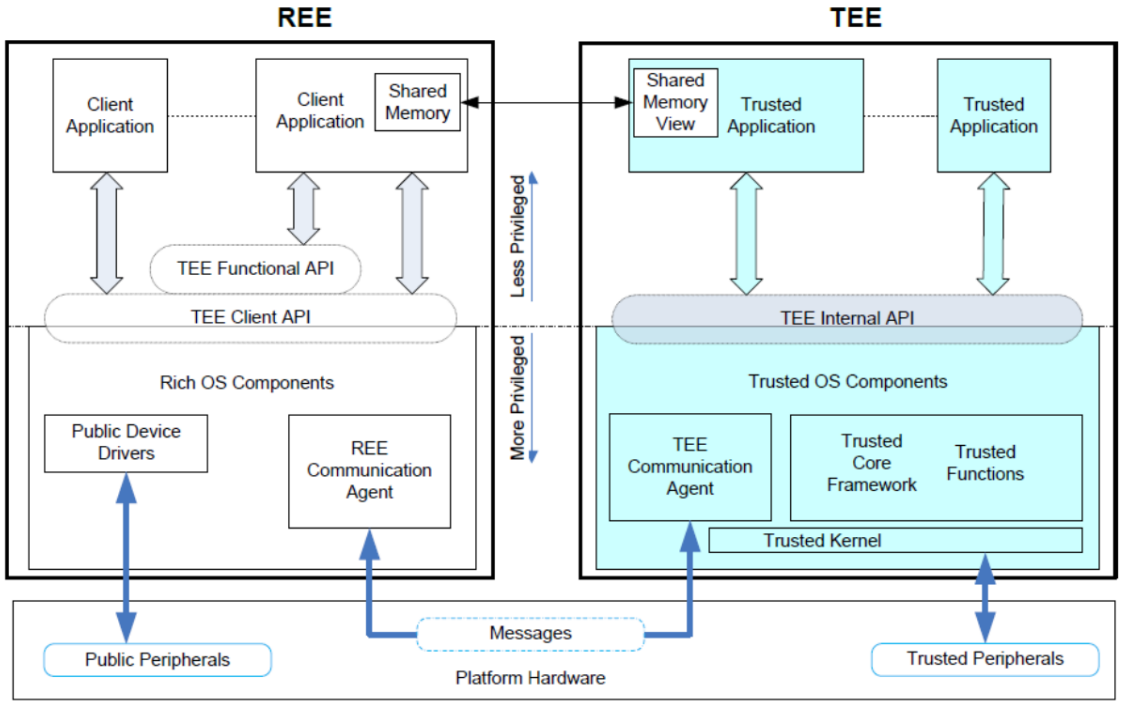
- Software architecture
  - TEE software components
    - The Trusted OS
    - The Trusted Applications (TAs)
  - TEE APIs



Trusted Execution Environments (TEEs)

Standardization

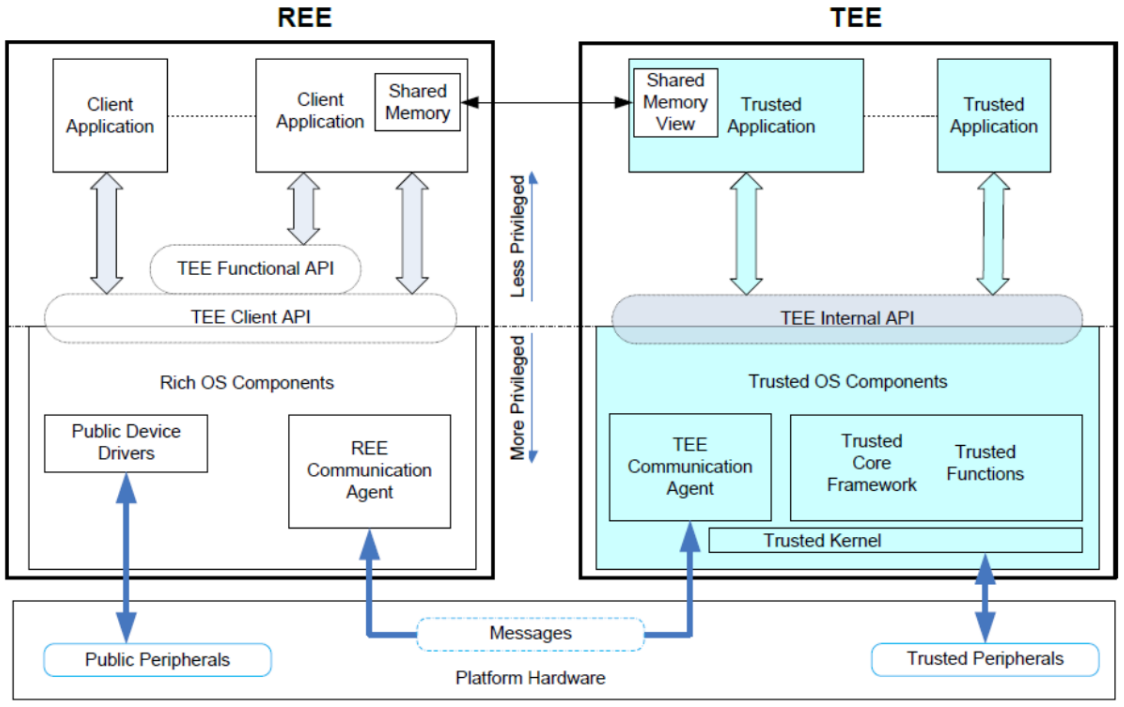
- Software architecture
  - TEE software components
    - The Trusted OS
      - A holder of the TAs which run on top of it
      - Supports facility provider for application developers
      - In order to guarantee the RoT of the TEE, it is authenticated and then isolated from the Rich OS during the secure boot process



Trusted Execution Environments (TEEs)

Standardization

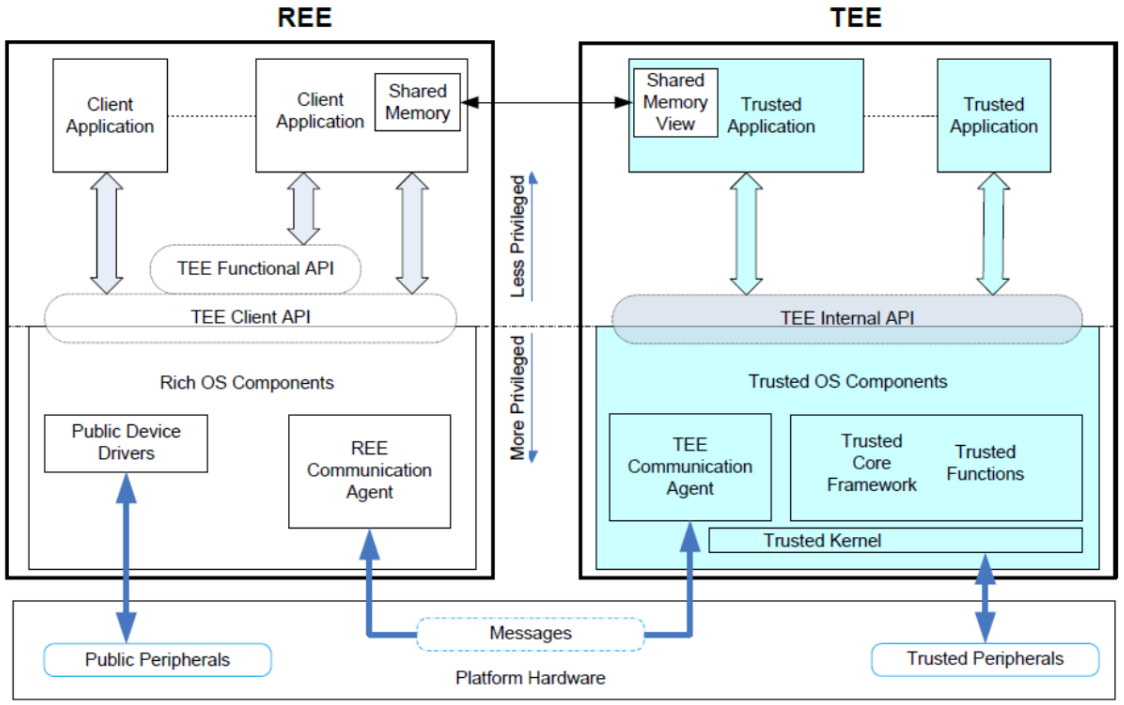
- Software architecture
  - TEE software components
    - The Trusted OS
      - The secure boot comprehends 5 steps
        - The boot starts from the trusted ROM
        - The first initializations are done, and the TEE Trusted OS is authenticated and validated
        - The TEE Trusted OS is set up. Then it prepares the environment to boot the REE
        - The REE takes control and starts the REE OS initializations
        - The REE is set up and the TEE functionalities are ready



Trusted Execution Environments (TEEs)

Standardization

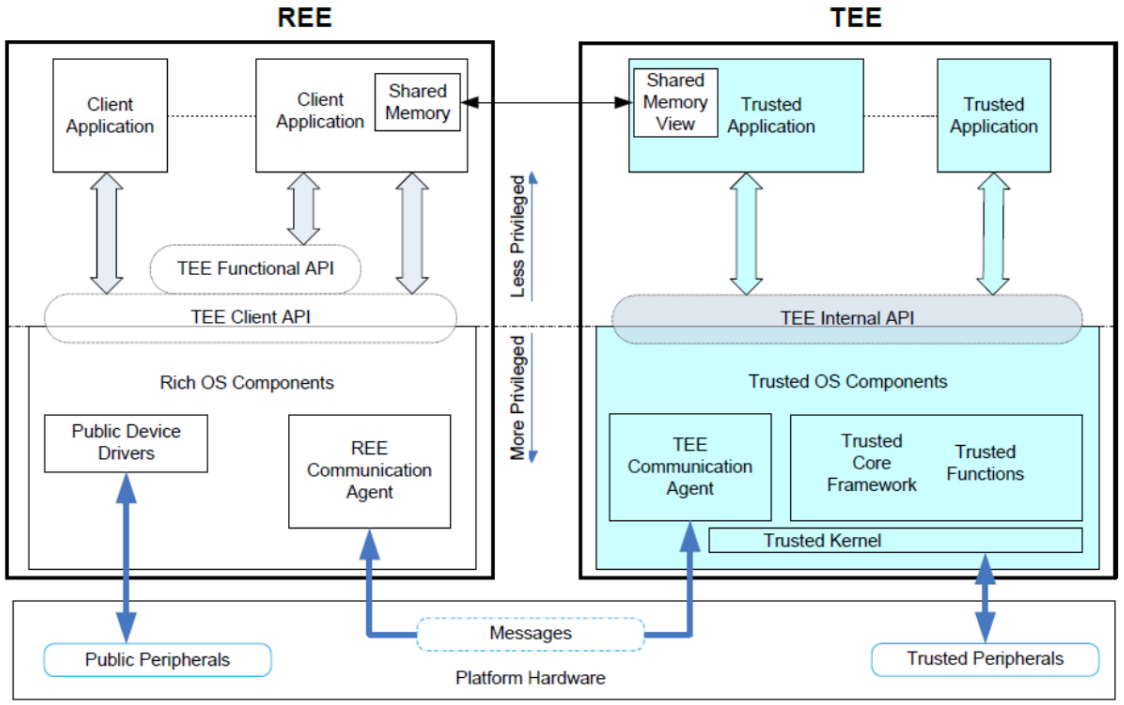
- Software architecture
  - TEE software components
    - The Trusted Applications (TAs)
      - Applications running inside the TEE, which provide security services to Client Applications



Trusted Execution Environments (TEEs)

Standardization

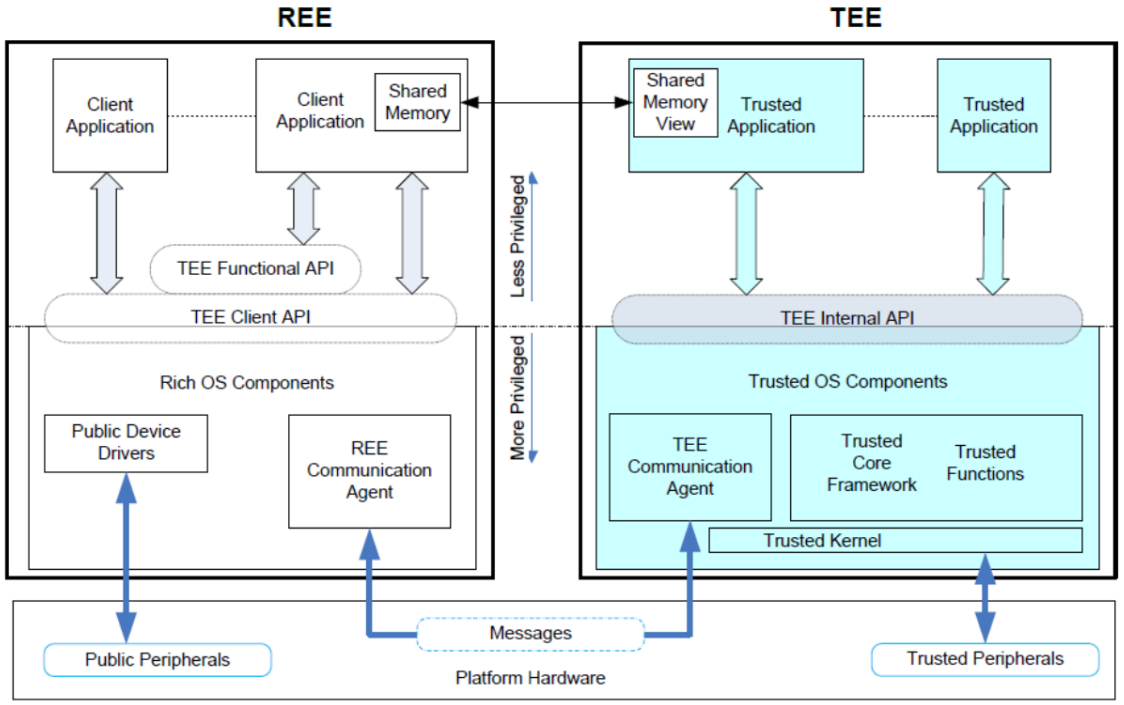
- Software architecture
  - TEE APIs
    - Public APIs
      - Provide a communication interface to Client Application enabling exchanges with TEE
    - Private APIs
      - Provide a communication interfaces to TAs enabling exchanges with Secure Element Applications and usage of TEE resources



Trusted Execution Environments (TEEs)

Standardization

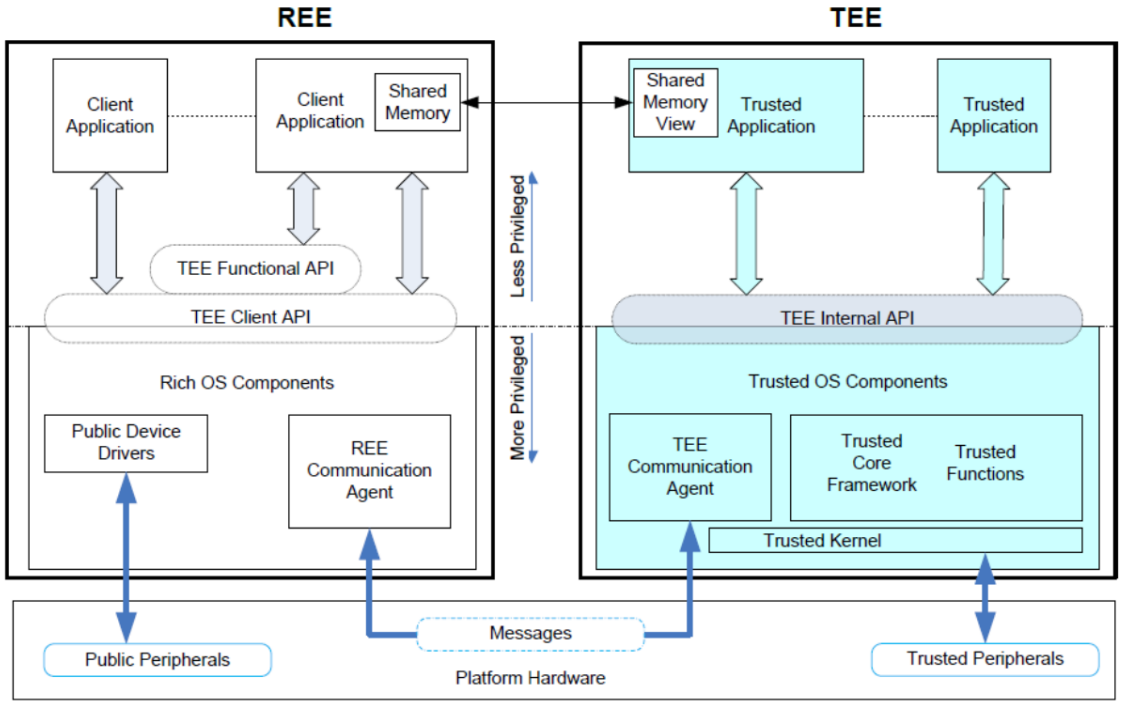
- Software architecture
  - TEE APIs :: Private APIs
    - TEE Internal API
    - Trusted User Interface API
    - TEE Secure Element API



Trusted Execution Environments (TEEs)

Standardization

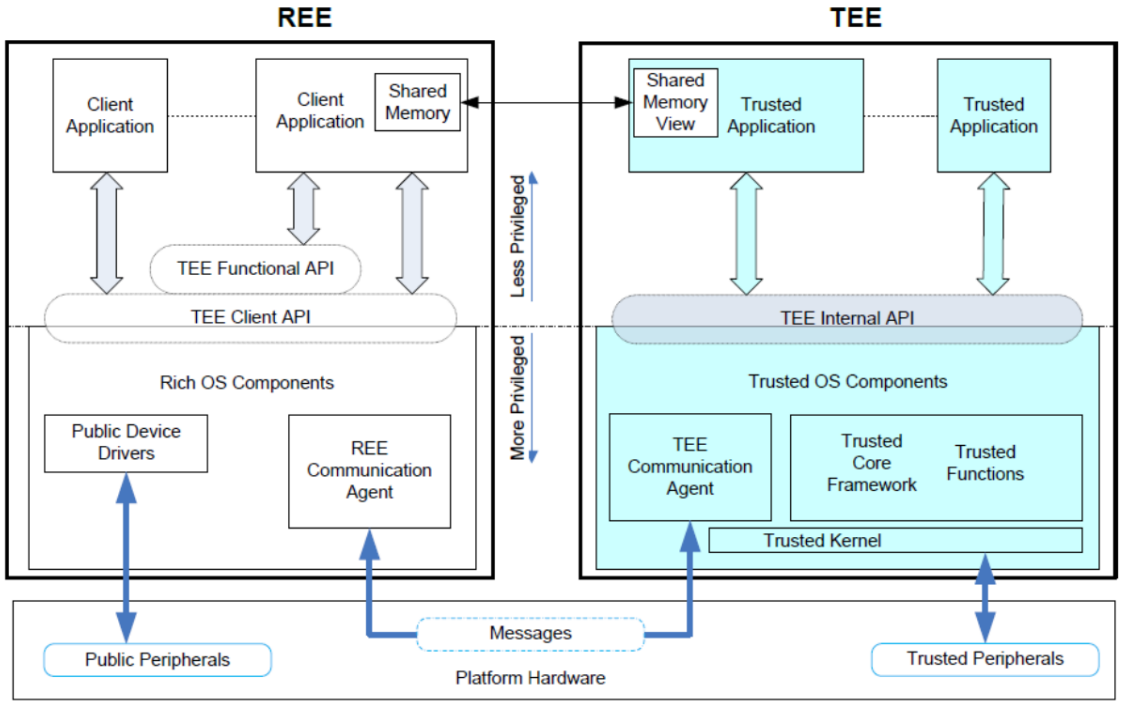
- Software architecture
  - TEE APIs :: Private APIs
    - TEE Internal API
      - The interface used by TAs running within the TEE to benefit from TEE resources, such as cryptographic capabilities and Trusted Storage
      - Introduces 5 sub-sets of functionalities
        - Trusted Core Framework API
          - Provides integration, scheduling, communication, memory management, and system information retrieval interfaces



Trusted Execution Environments (TEEs)

Standardization

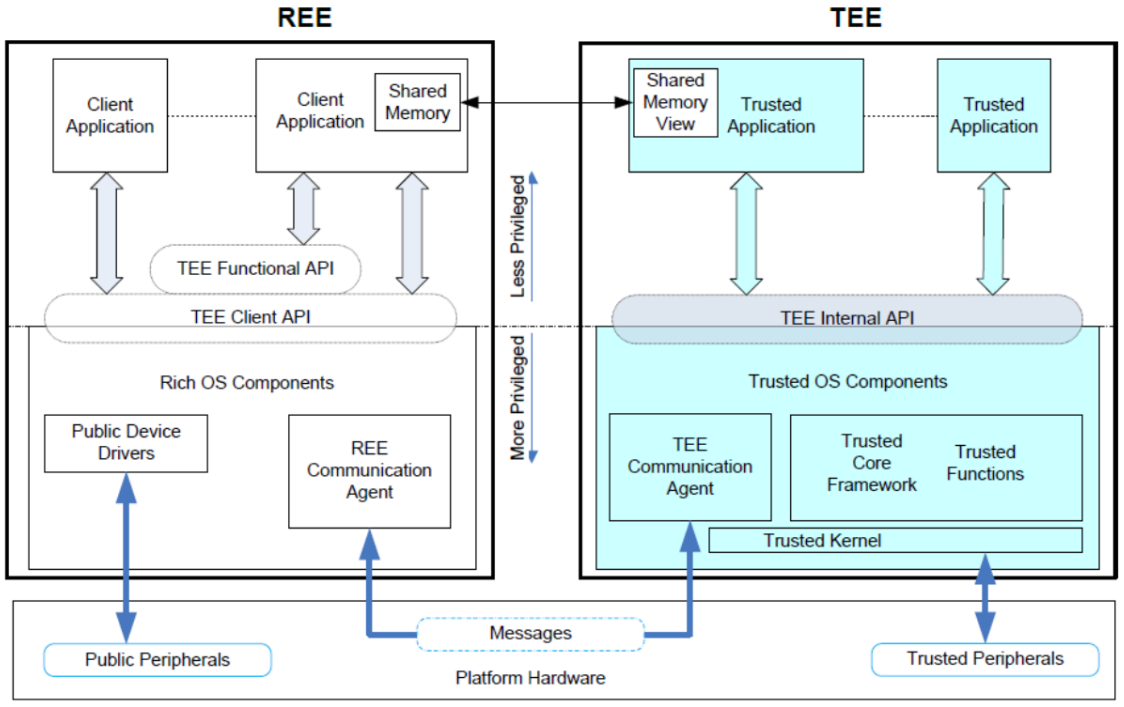
- Software architecture
  - TEE APIs :: Private APIs
    - TEE Internal API
      - The interface used by TAs running within the TEE to benefit from TEE resources, such as cryptographic capabilities and Trusted Storage
      - Introduces 5 sub-sets of functionalities
        - Trusted Storage API
          - Provides Trusted Storage for keys and general data, i.e. each TA has its own trusted storage space, which is available only to the TA that created it, where it saves its cryptographic keys and data



Trusted Execution Environments (TEEs)

Standardization

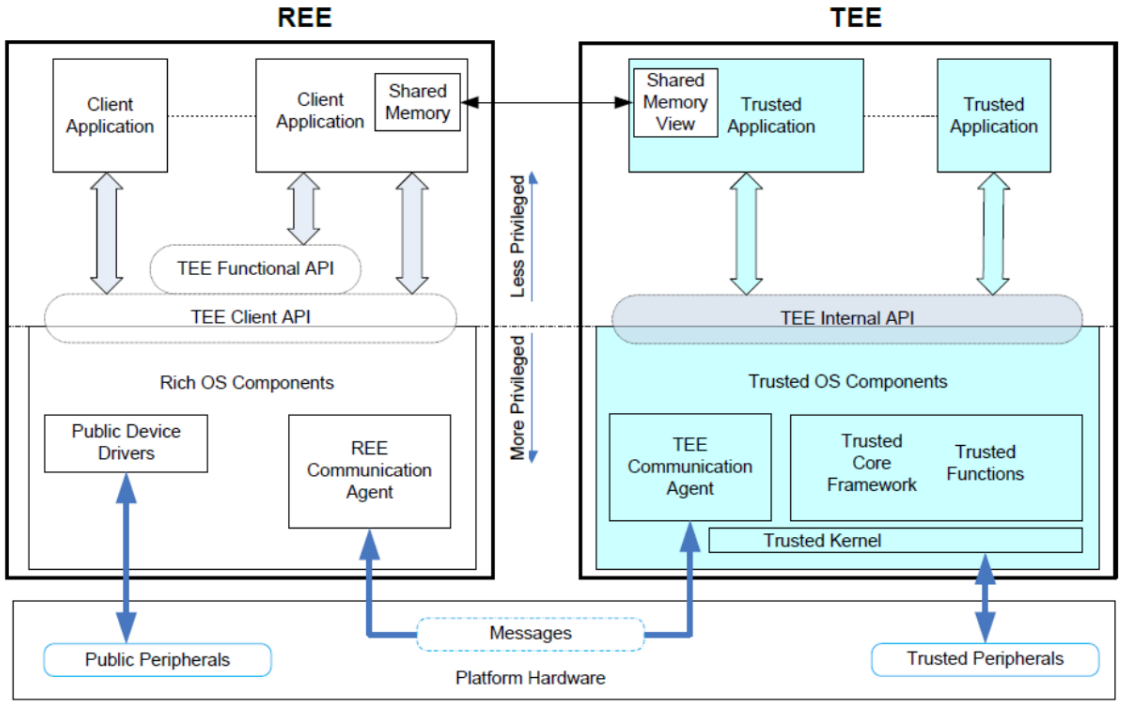
- Software architecture
  - TEE APIs :: Private APIs
    - TEE Internal API
      - The interface used by TAs running within the TEE to benefit from TEE resources, such as cryptographic capabilities and Trusted Storage
      - Introduces 5 sub-sets of functionalities
        - Cryptographic Operations API
          - Provides cryptographic capabilities E.g. digests, symmetric ciphers, message authenticated codes, authenticated encryption, asymmetric encryption schemes, asymmetric signature schemes and key exchange algorithms



Trusted Execution Environments (TEEs)

Standardization

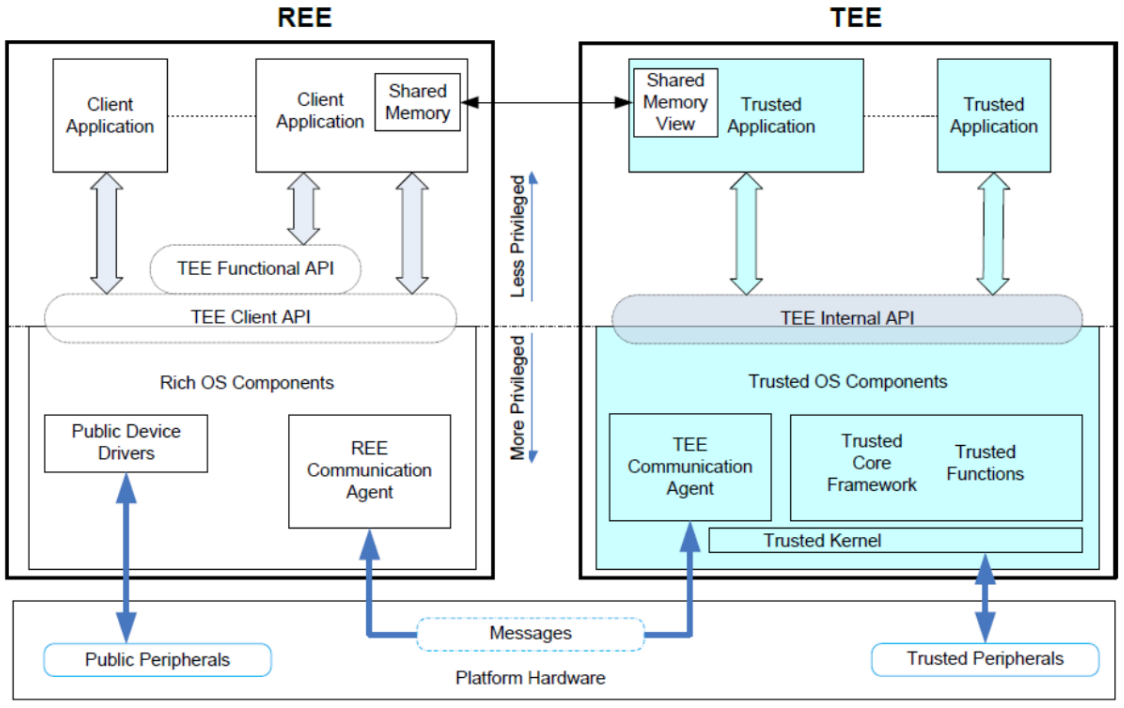
- Software architecture
  - TEE APIs :: Private APIs
    - TEE Internal API
      - The interface used by TAs running within the TEE to benefit from TEE resources, such as cryptographic capabilities and Trusted Storage
      - Introduces 5 sub-sets of functionalities
        - TEE Arithmetical API
          - Provides arithmetical primitives to create cryptographic functions not found in the Cryptographic Operations API



Trusted Execution Environments (TEEs)

Standardization

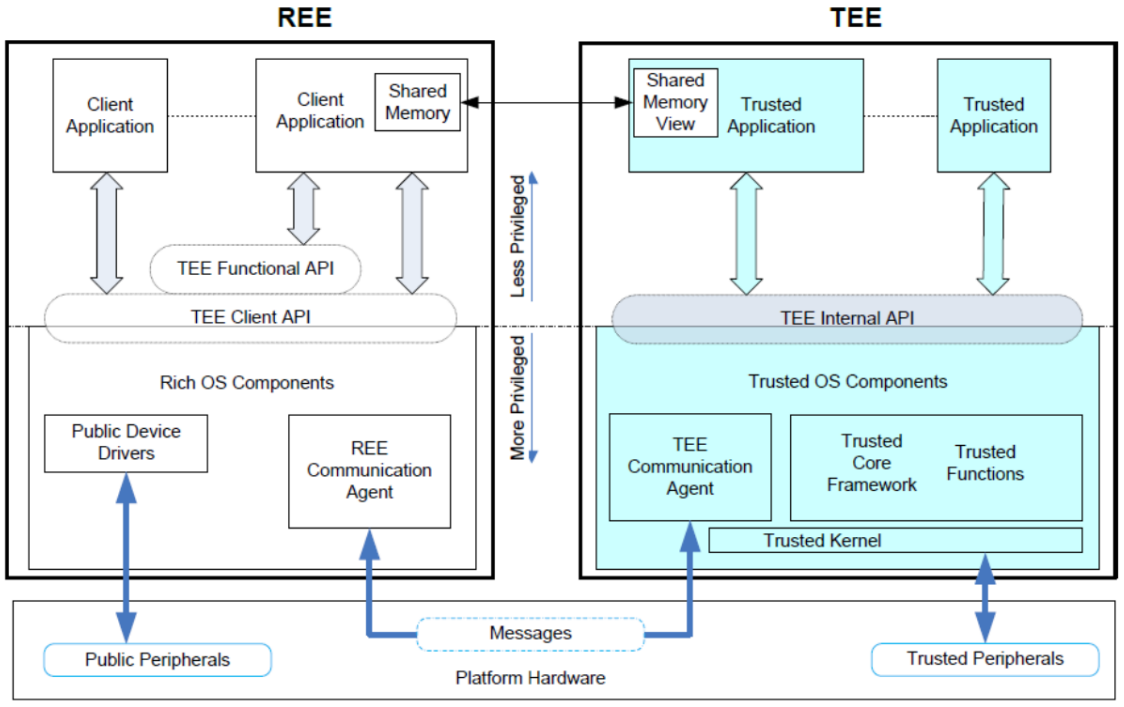
- Software architecture
  - TEE APIs :: Private APIs
    - TEE Internal API
      - The interface used by TAs running within the TEE to benefit from TEE resources, such as cryptographic capabilities and Trusted Storage
      - Introduces 5 sub-sets of functionalities
        - Time API
          - Provides trusted time support for various time-based functionality such as DRM



Trusted Execution Environments (TEEs)

Standardization

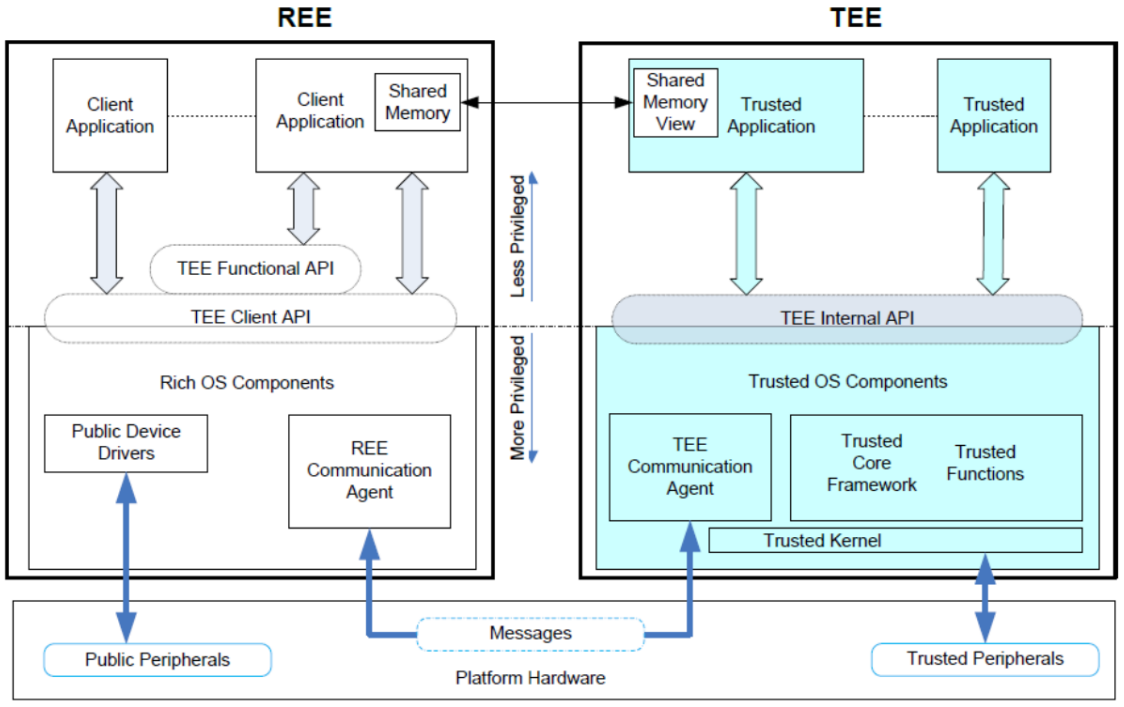
- Software architecture
  - TEE APIs :: Private APIs
    - Trusted User Interface API
      - Enables a Secure Indicator to indicate that the screen and the input fields are controlled by a TA
      - Enables a Secure Display/Input, which means that viewed/entered data cannot be eavesdropped or modified by an unauthorized application either from the REE or the TEE
      - Can be used to protect a PIN entry or a login/ password entry, to display sensitive information to the user such as the amount of a transaction and potentially to require confirmation



Trusted Execution Environments (TEEs)

Standardization

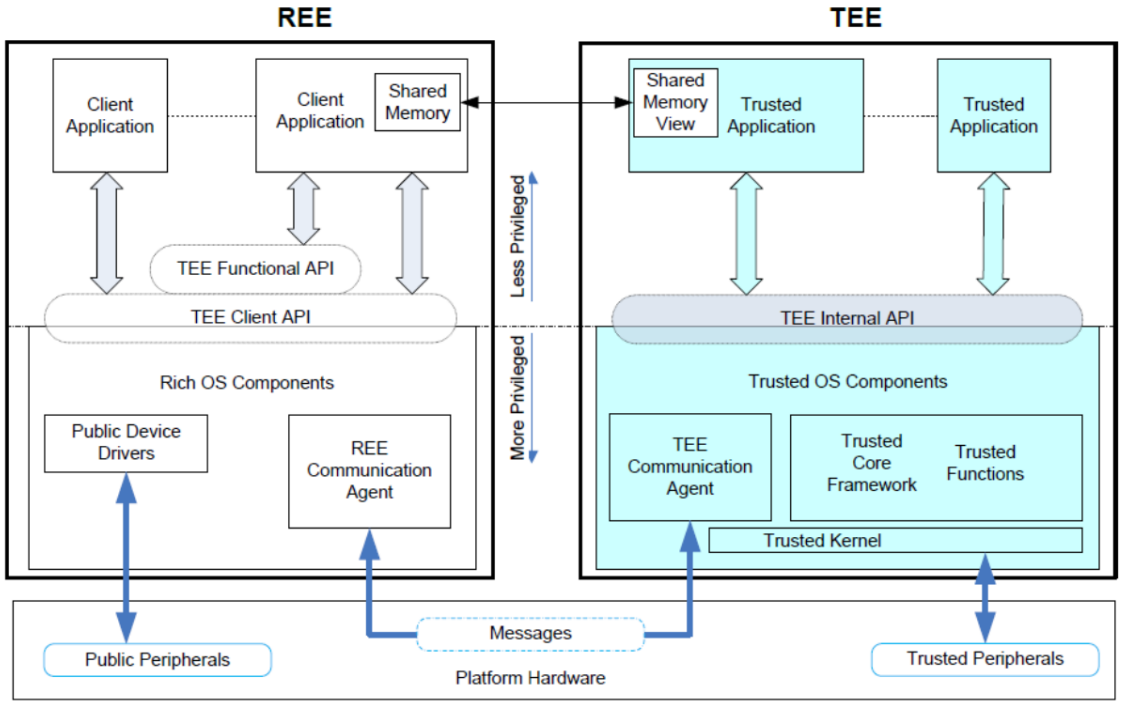
- Software architecture
  - TEE APIs :: Private APIs
    - TEE Secure Element API
      - Allows a TA to communicate with a Secure Element Application (SEA)
      - If the Secure Element is connected exclusively to the TEE, connection between a TA and a SEA is considered trusted
      - If the Secure Element is connected to the REE, connection between a TA and a SEA must be protected



Trusted Execution Environments (TEEs)

Standardization

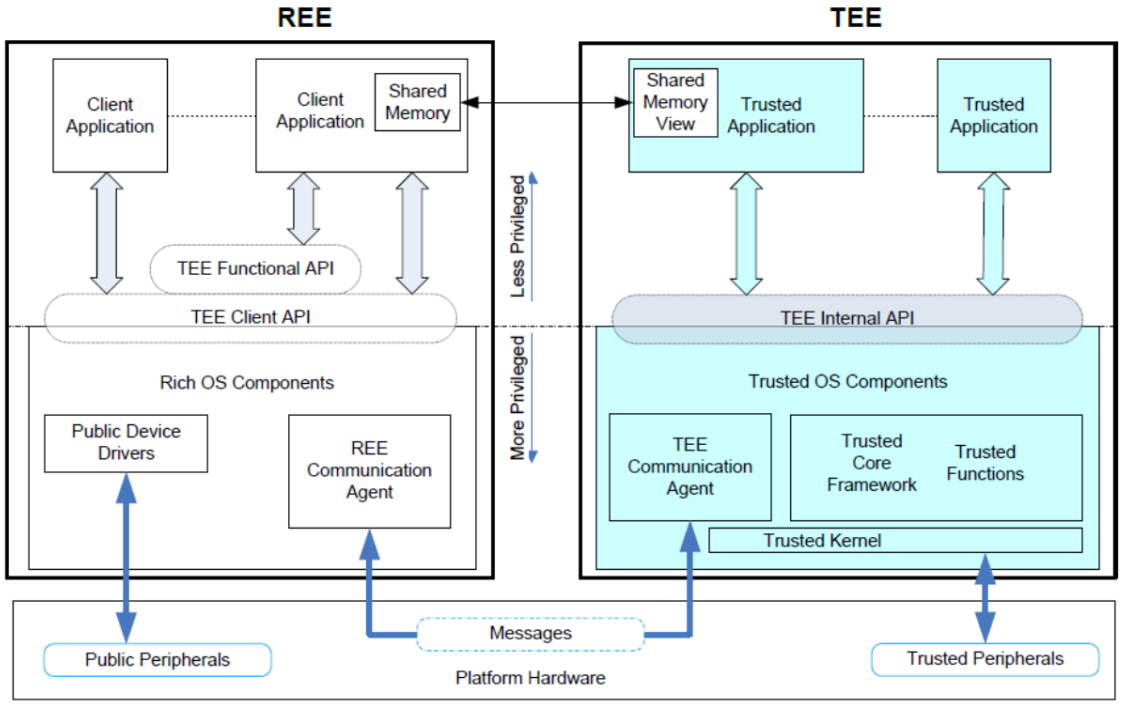
- Software architecture
  - TEE APIs :: Public APIs
    - TEE Client API
      - A low-level communication interface designed to enable Client Applications (running in the Rich OS) to exchange data with a TA (running within a TEE)
      - TEE and Trusted Applications must treat any input from REE as potentially malicious
        - So, CAs communicates with an instance of a TA!
      - During a communication between a CA and a TA, if a suspicious behavior is detected, the TA instance and the related session are immediately destroyed



Trusted Execution Environments (TEEs)

Standardization

- Software architecture
  - TEE APIs :: Public APIs
    - TEE Client API
    - TEE functional API
      - Offers to CA a set of Rich OS-friendly APIs



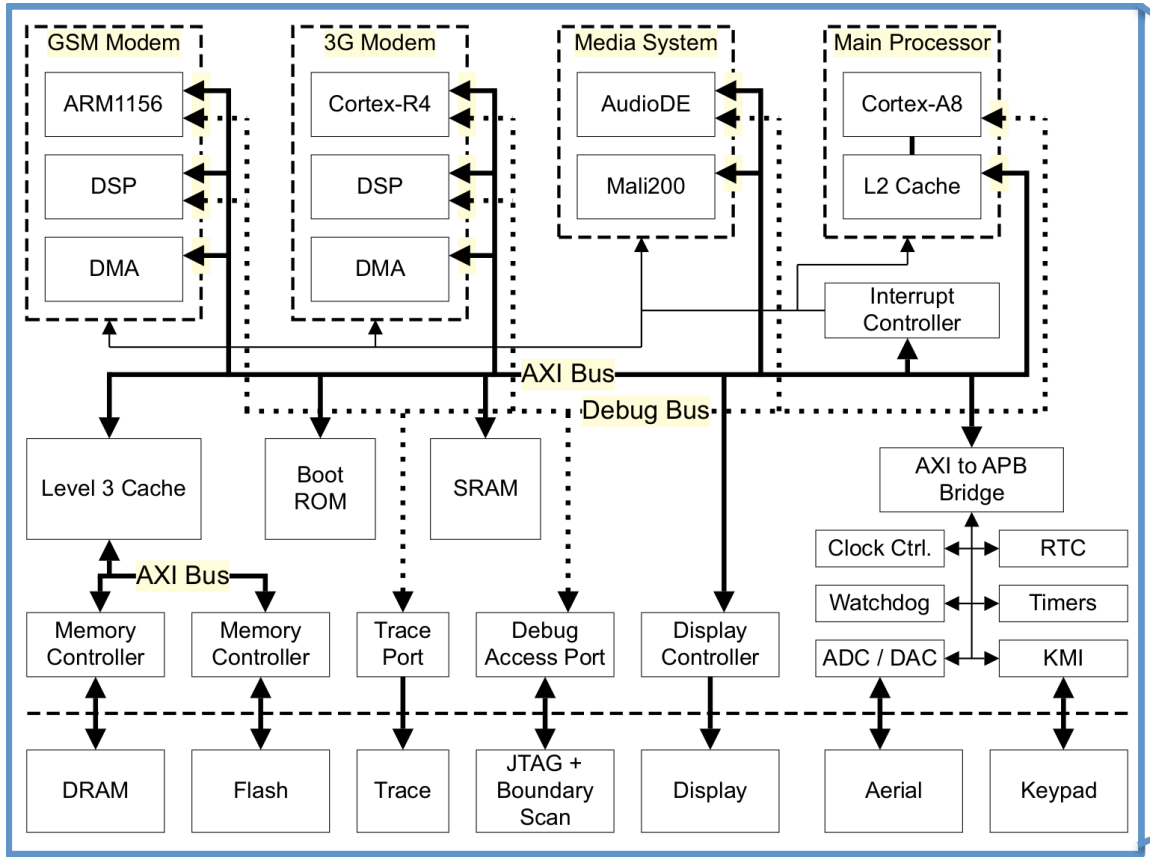
Trusted Execution Environments (TEEs)

Technologies

- Three main technological implementations of TEEs have been presented
  - ARM TrustZone
  - Intel Software Guard eXtension (SGX)
  - AMD Memory Encryption Technology (MET)

Trusted Execution Environments (TEEs)

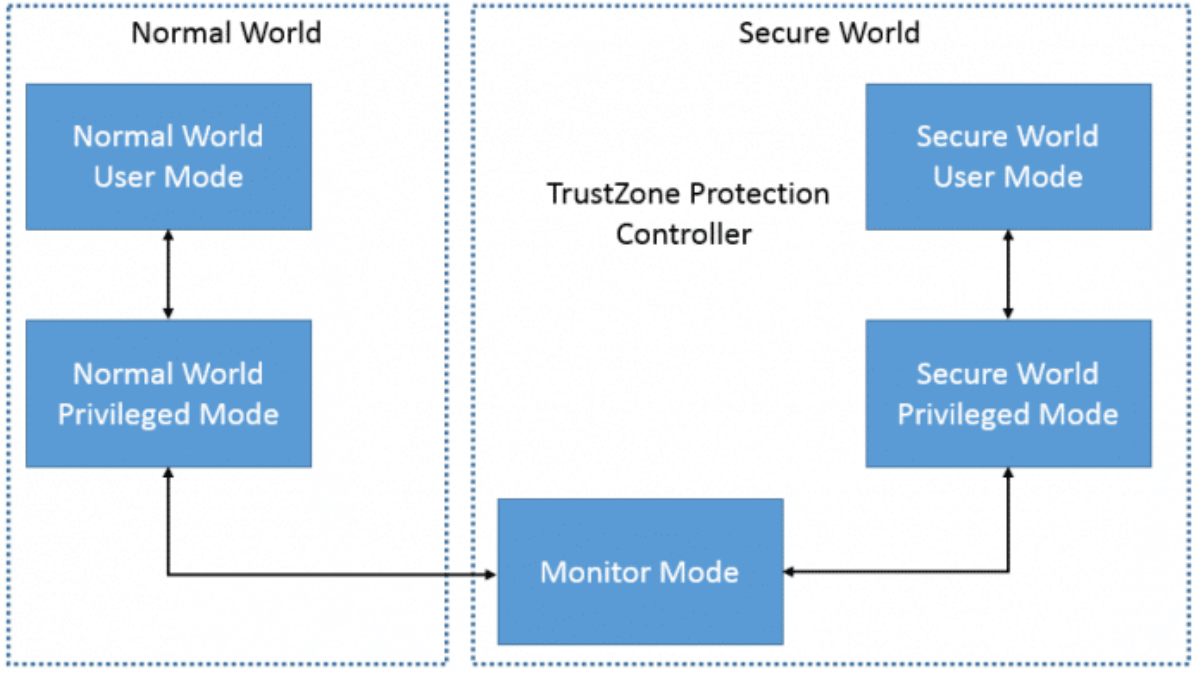
Technologies :: ARM SoC



Trusted Execution Environments (TEEs)

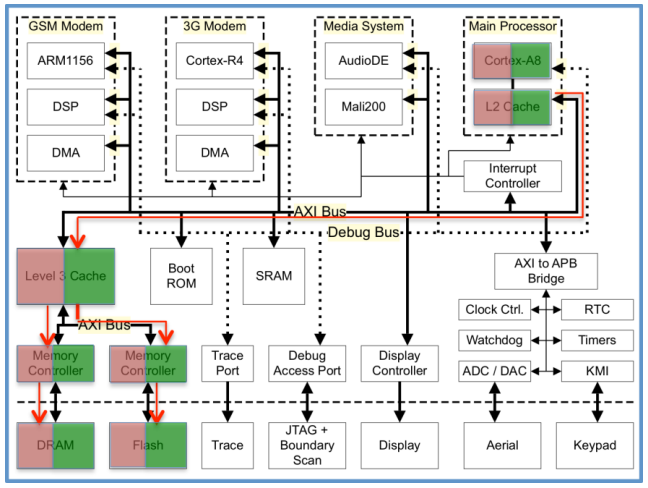
Technologies :: ARM TrustZone

- A flexible hardware-assisted security technology proposed by ARM
- Partitions the hard- and software-resources of a SoC by distinguishing the context execution in two worlds
  - The secure world
  - The normal world



Trusted Execution Environments (TEEs)

Technologies :: ARM TrustZone

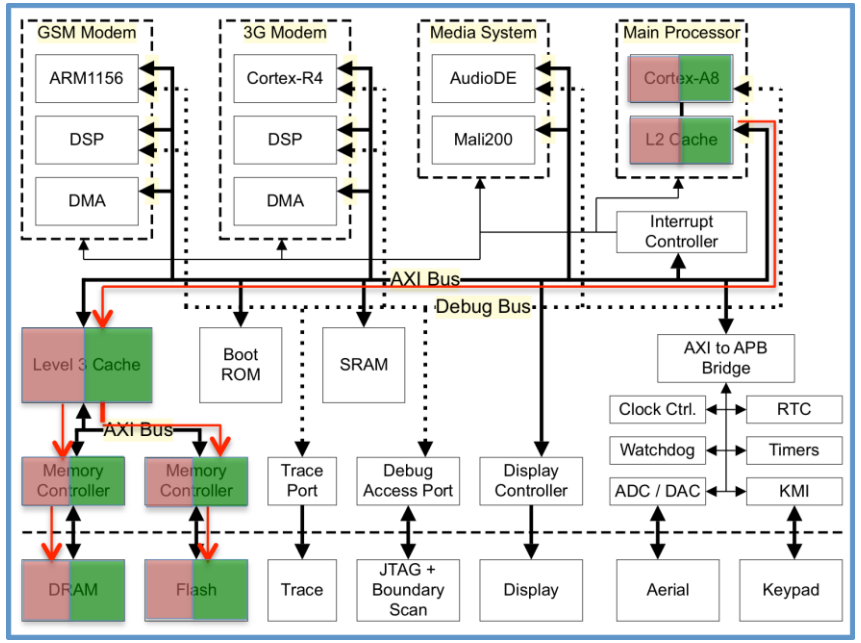


- Separation is achieved by two main features
  - A TrustZone-enabled processor core
    - Uses time-slicing to execute code in either the secure or normal world
  - SoC's AXI bus
    - Ensures that secure world resources cannot be accessed from normal world resources
    - A control signal known as the Non-Secure (NS) bit was added to the AXI specification
      - This bit is used to communicate the security state of a master component to a slave component
      - The bus or slave logic uses this bit to ensure that the security separation is not violated
        - When an untrusted master attempts to access a secure slave, the transaction should fail and an error may be raised

Trusted Execution Environments (TEEs)

Technologies :: ARM TrustZone

- TrustZone security features extensions :: Memory Management Unit
  - The TrustZone-aware Memory Management Unit (MMU) allows for each world to have its own virtual-to-physical memory address translation tables
  - To provide memory isolation at the cache-level, the processor's L1 and L2 caches use a bit to store the security state of the transaction that accessed the memory
  - The cache controllers are then assumed to be responsible for ensuring that only secure masters can access memory that was fetched from a secure slave



Trusted Execution Environments (TEEs)

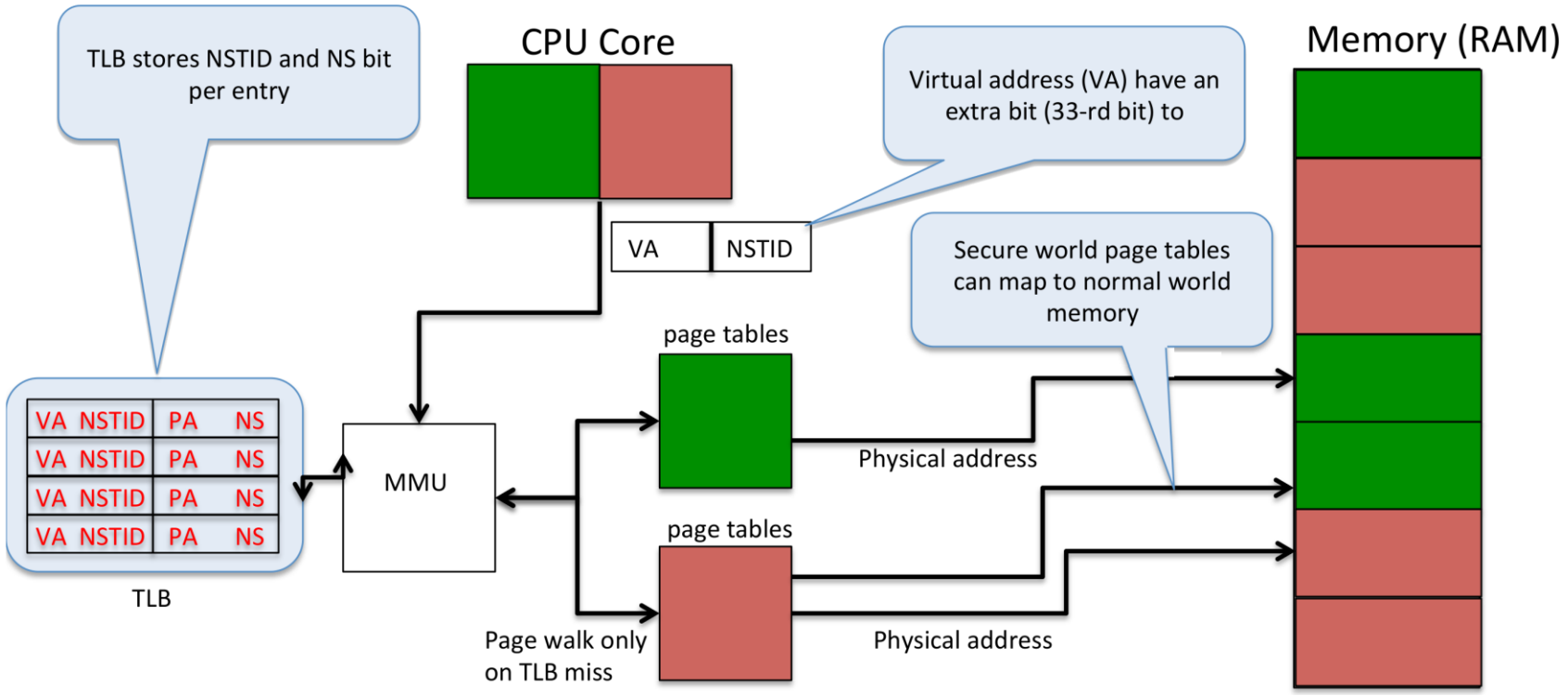
Technologies :: ARM TrustZone

- TrustZone security features extensions :: Memory infrastructure
  - TrustZone Address Space Controller (TZASC)
    - Can be used to configure specific memory regions as secure or non-secure, such that applications running in the secure world can access memory regions associated with the normal world, but not the otherwise
    - Such DRAM partitioning is performed by the TZASC under the control of a programming interface restricted to the software running with secure world privileges
  - TrustZone Memory Adapter (TZMA)
    - Provides similar memory partitioning functionality, but targeting off-chip ROM or SRAM
  - Both the TZASC and the TZMA are optional components

Trusted Execution Environments (TEEs)

Technologies :: ARM TrustZone

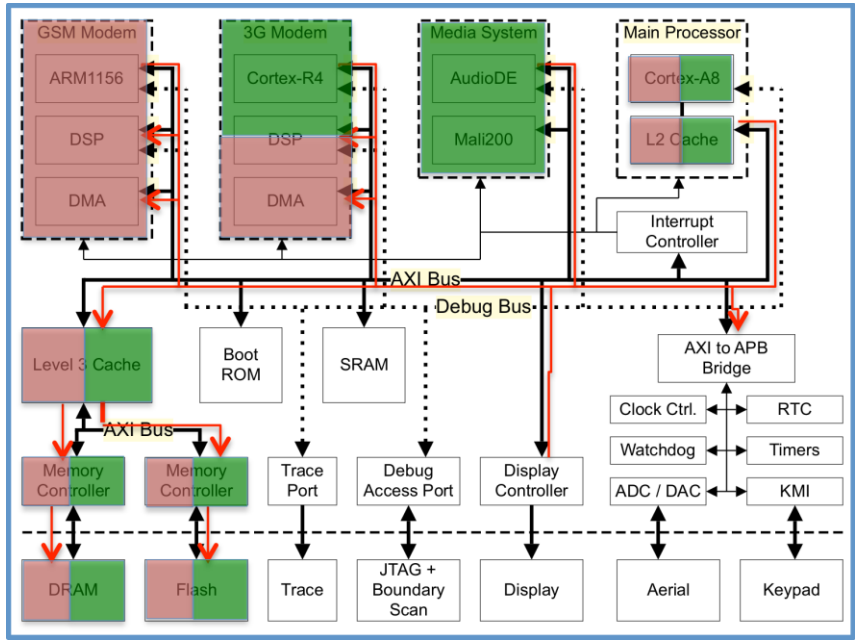
- TrustZone security features extensions :: Memory infrastructure



Trusted Execution Environments (TEEs)

Technologies :: ARM TrustZone

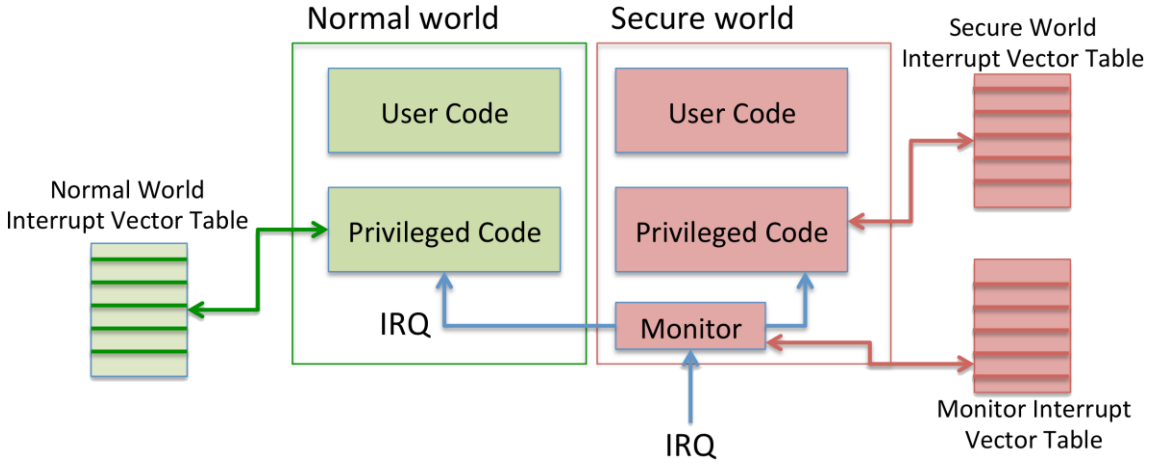
- TrustZone security features extensions :: System devices management
  - TrustZone Protection Controller (TZPC)
    - Allows for system devices to be restricted to the secure or normal worlds
    - Optional and implementation specific component



Trusted Execution Environments (TEEs)

Technologies :: ARM TrustZone

- TrustZone security features extensions :: Interrupt management
  - Generic Interrupt Controller (GIC) has been extended with support for prioritized secure and non-secure sources



  - This enables secure interrupts to be handled with higher priority than the non-secure interrupts
    - Important to prevent denial-of-service (DoS) attacks by non-secure software

Trusted Execution Environments (TEEs)

Technologies :: ARM TrustZone

- A TrustZone core can switch between security states at runtime
  - In the secure state, the processor generates AXI transactions with the NS bit set to zero, allowing it to access resources in both security domains
  - In the normal world, the processor can only access normal world resources
  - To perform a context switch to the other world, the processor first has to pass through a new mode called monitor mode

Trusted Execution Environments (TEEs)

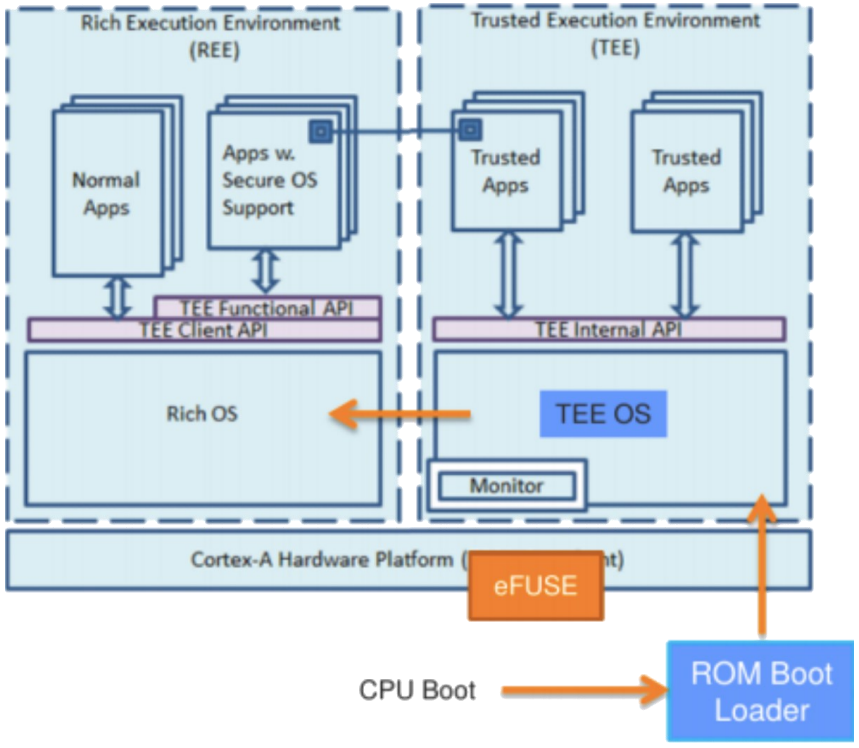
Technologies :: ARM TrustZone

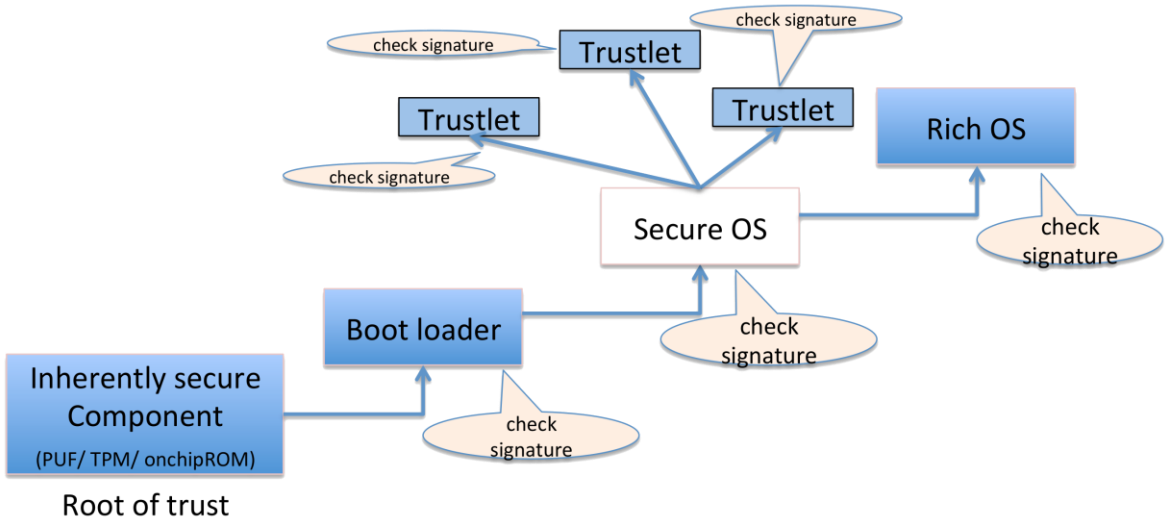
- Monitor mode
  - Monitor mode exists in the secure world, and both privileged and user mode exist in each world
  - Serves as a gatekeeper that manages context switches between the two worlds
    - Saving the state of the current world
    - Restoring the state of the world being switched to
  - Normal world entry to monitor mode is only possible via an interrupt, external abort, or explicit call of the Secure Monitor Call (SMC) instruction
  - Secure world entry to monitor mode can additionally be invoked by writing into the Current Program Status Register (CPSR)
  - ARM recommends to execute monitor code with interrupts disabled

Trusted Execution Environments (TEEs)

Technologies :: ARM TrustZone

- Secure Boot





Computer security

References

- Coppolino, L. & D'Antonio, S. & Mazzeo, G. (2019). A Comprehensive Survey of Hardware-assisted Security: from the Edge to the Cloud. Internet of Things. 100055. 10.1016/j.iot.2019.100055.
- Sabt, M. & Achemlal, M. & Bouabdallah, A. (2015, August). *Trusted Execution Environment: What It is, and What It is Not*. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA. Paper presented at Trustcom, Helsinki (pp. 57-64). doi: 10.1109/Trustcom.2015.357.
- Arfaoui, G. & Gharout S. & Traoré, J. (2014, April). *Trusted Execution Environments: A Look under the Hood*. In Proceedings of the 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, Oxford (pp. 259-266). doi: 10.1109/MobileCloud.2014.47
- Pinto, S. & Santos, N. (2019). Demystifying ARM TrustZone: A Comprehensive Survey. *ACM Computing Surveys*, 51(6), 130:1-130:36. doi: https://doi.org/10.1145/3291047