UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Lecture 10 - OS-Level Protection & Trusted Execution Environments

Prof. Esposito Christian

::.. Summary

- OS-Level Protection Means
- Trusted Execution Environments

::.. Key Lectures

- M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not", Proceedings of the IEEE Trustcom/BigDataSE/ISPA Conference, 2015.
- S. Pinto, T. Gomes, J. Pereira, J. Cabral, A. Tavares, "IIoTEED: an enhanced, trusted execution environment for industrial IoT edge devices", *IEEE Internet Computing*, *21*(1), 40-47, 2017.

OS-Level Protection Means

::.. ContikiSec

The Contiki Operating System provides ContikiSec, a secure network layer for wireless sensor networks, designed to balance low energy consumption and security while conforming to a small memory footprint.

By default, Contiki does not offer any confidentiality or authenticity services for the communication between nodes.

ContikiSec providing three security modes:

- ContikiSec-Enc, which provides confidentiality and integrity;
- ContikiSec-Auth, which guarantees authenticity and integrity;
- ContikiSec-AE, which supports confidentiality, authenticity and integrity.

::.. Security in TinyOS (1/2)

TinyOS has a rich set of security-related libraries:

- Symmetric-key Cryptography
  - NesC implementation of AES block cipher. Link (http://tinyos.cvs.sourceforge.net/viewvc/*checkout*/tinyos/tinyos-2.x-contrib/crypto/index.html).
  - Efficient Assembler implementation of AES on AVR microcontroller. Link (http://point-at-infinity.org/avraes/10).
  - An implementation of the Trivium stream cipher for 8-bit and 16-bit microcontrollers can be found here (http://tinyos.cvs.sourceforge.net/viewvc/*checkout*/tinyos/tinyos-2.x-contrib/crypto/index.html).
  - Two universal hash function families (MMH and Poly32) have been implemented as TinyOS interfaces.

::.. Security in TinyOS (2/2)

- Public-key Cryptography
  - TinyECC : Elliptic Curve Cryptography for Sensor Networks. Webpage (http://discovery.csc.ncsu.edu/software/TinyECC/ver0.3/index.html).
  - TinyPBC : Pairing-based Cryptography in Sensor Networks. Webpage (http://sites.google.com/site/tinypbc/10).
- Secure Communication Libraries:
  - TinySec –protocols implemented by using the default block cipher Skipjack, and to generate a MAC, it uses Cipher Block Chaining Message Authentication Code (CBC–MAC).
  - SenSec - uses a variant of Skipjack, called Skipjack-X.
  - MiniSec - a security library for single-source communi-cations, and multi-source broadcast communication.

::.. Security in Zerynth OS

Zerynth OS has a rich set of security-related modules with:

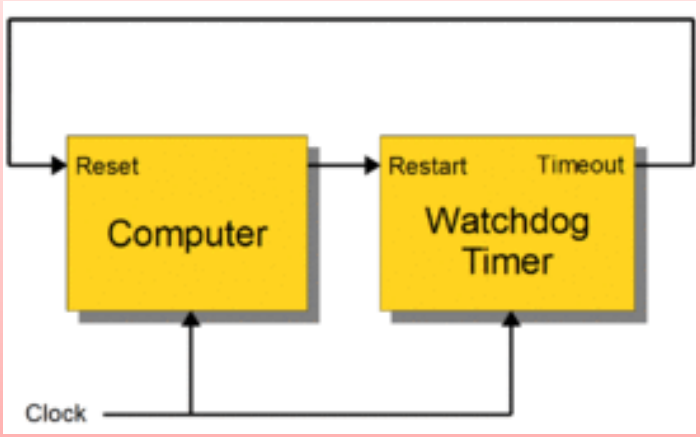
- An almost complete implementation of the SSL/TLS interface.
- Realization of useful operations involving X.509 certificates.
- Useful primitives for data encryption and signature.
- Offering of secure firmware functionalities by disabling access to the flash memory where the firmware resides with external means (e.g. JTAG probes), reserving some memory areas to critical parts of the system (e.g. VM running in mcu Trust Zones), detecting device tampering and taking action accordingly (e.g. erase firmware), detecting and recovering from firmware malfunctions (e.g. watchdogs)

Only watchdogs are implemented in all architectures; the rest of secure firmware features are strongly platform dependent.

::.. Security in Zerynth OS

Zerynth OS has a rich set of security-related modules with:

- An almost complete implementation of the SSL/TLS interface.
- Realization of useful operations involving x509 certificates.
- Useful primitives for data encryption adn signature.



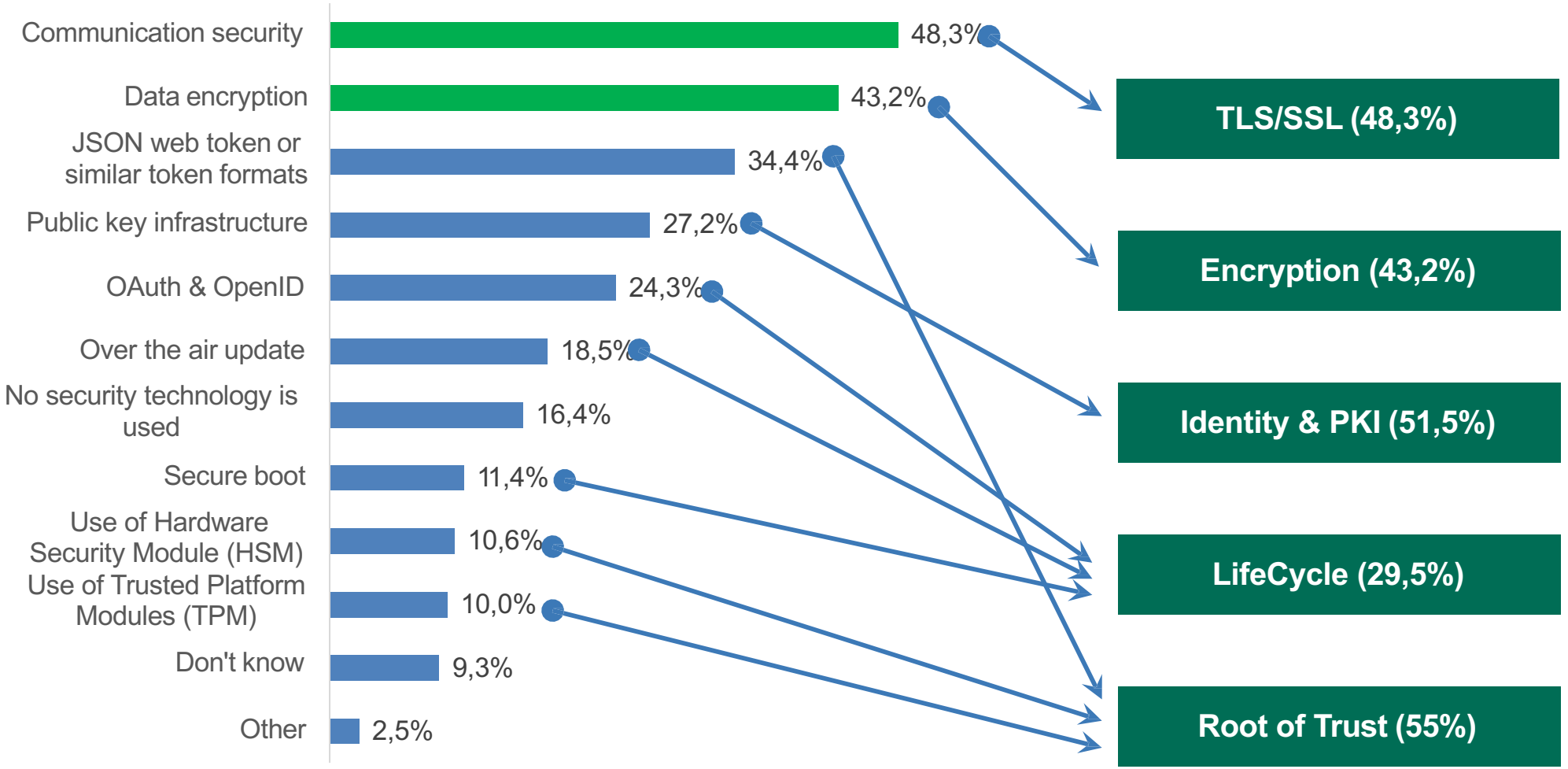
A watchdog is an electronic or software timer used to detect and recover from computer malfunctions. During normal operation, the computer regularly resets the watchdog timer to prevent it from elapsing, or "timing out".
If, due to a hardware fault or program error, the computer fails to reset the watchdog, the timer will elapse and generate a timeout signal. The timeout signal is used to initiate corrective actions.

from h[illegible]unctions (e.g. watchdogs)

Only **watchdogs** are implemented in all architectures; the rest of secure firmware features are strongly platform dependent.

Trusted Execution Environments

::.. IoT Security Technologies (1/2)



iot eclipse.org

IEEE Internet of Things

AGILE

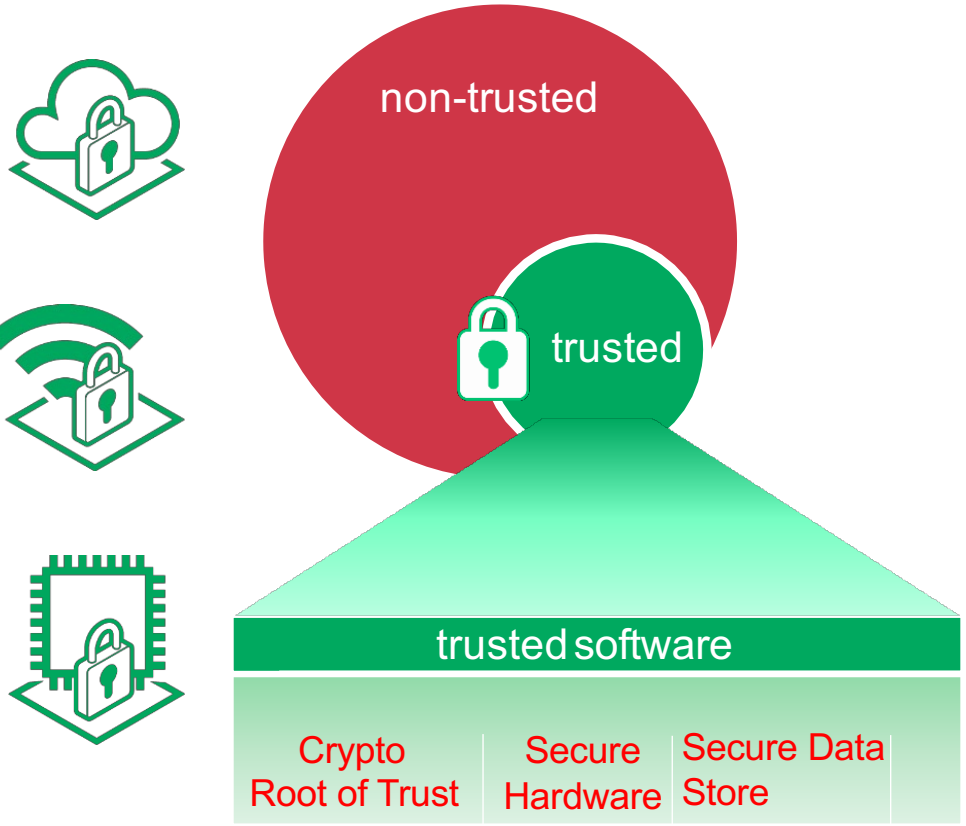
council

::.. IoT Security Technologies (2/2)

Lifecycle Security

Communications Security

Device Security



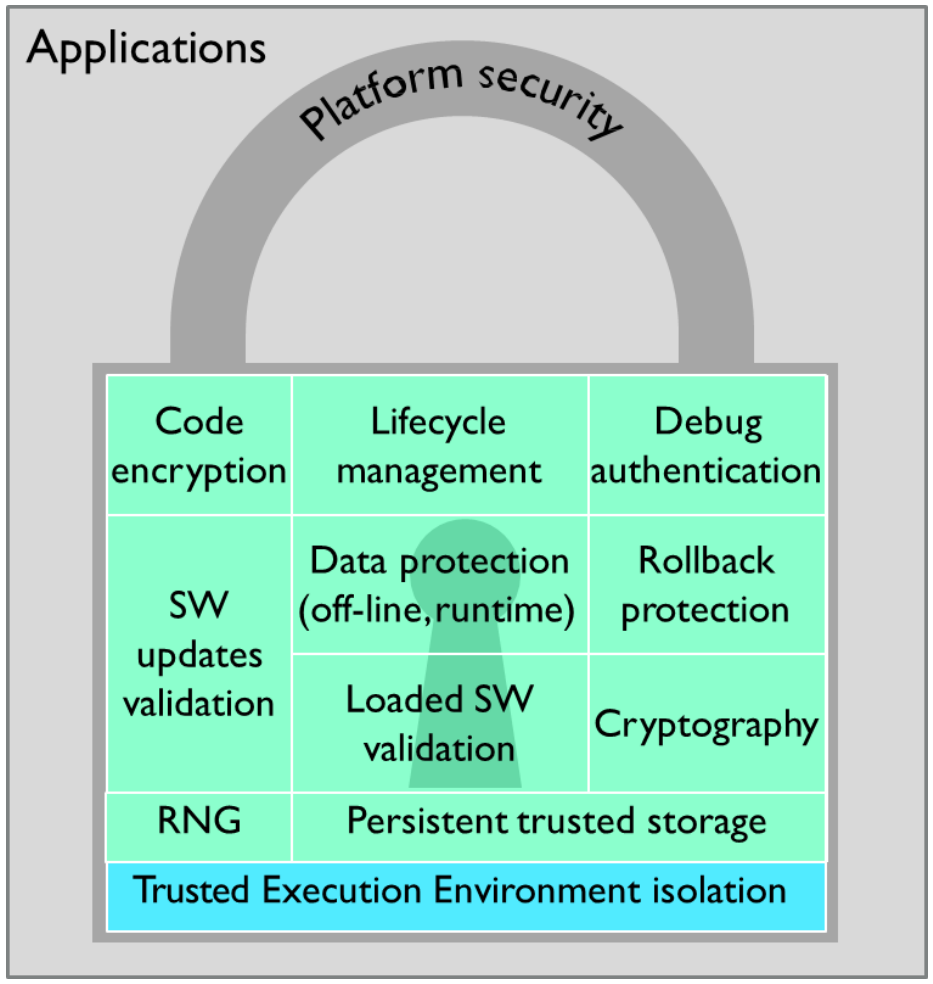
iot eclipse.org

IEEE Internet of Things

AGILE

council

::.. IoT Security Approach



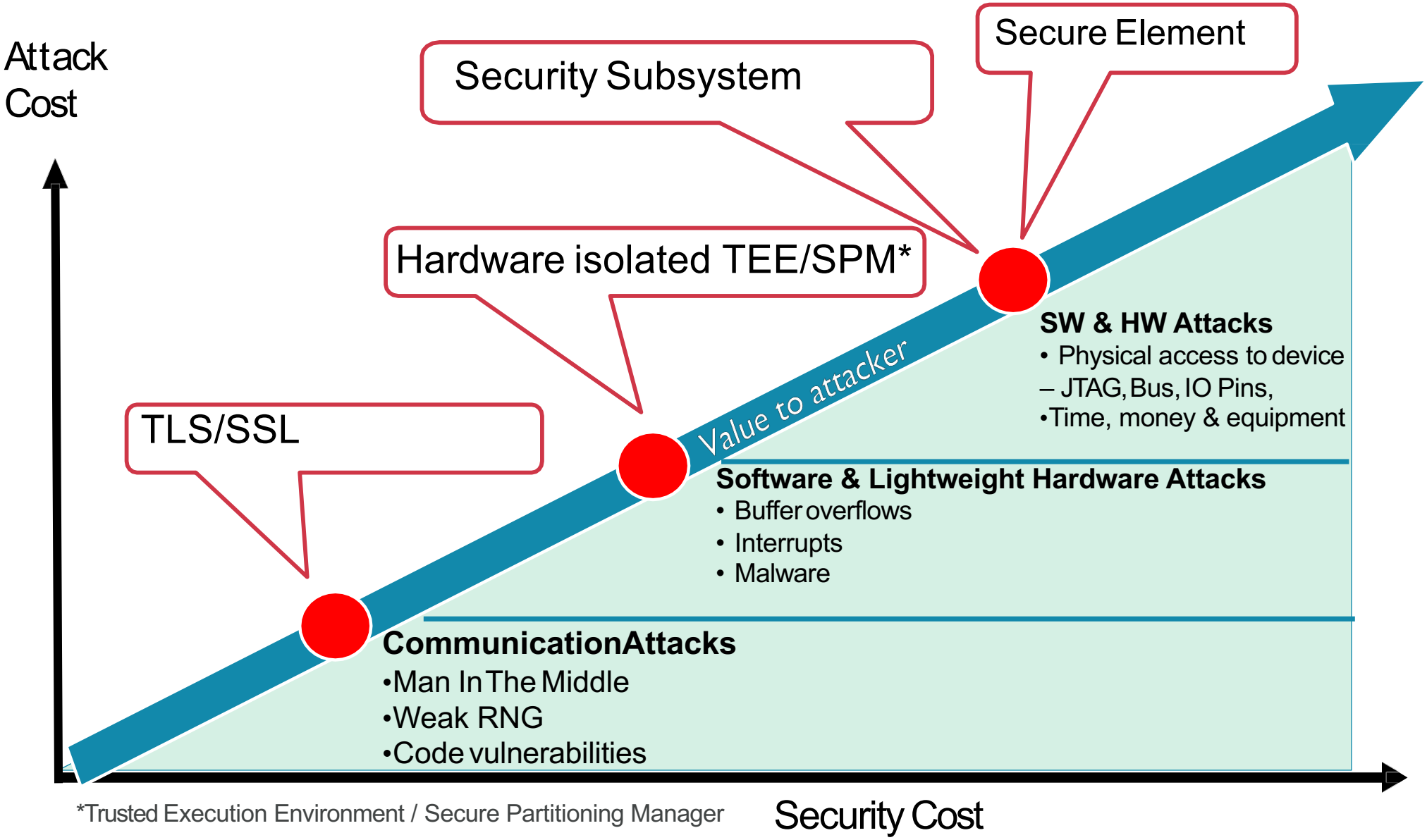
Situation

- Most IoT developers are not security experts
- Little to no knowledge of hardware
- Prior experience in mobile app development
- Time to market & functionality beat security

Strategy

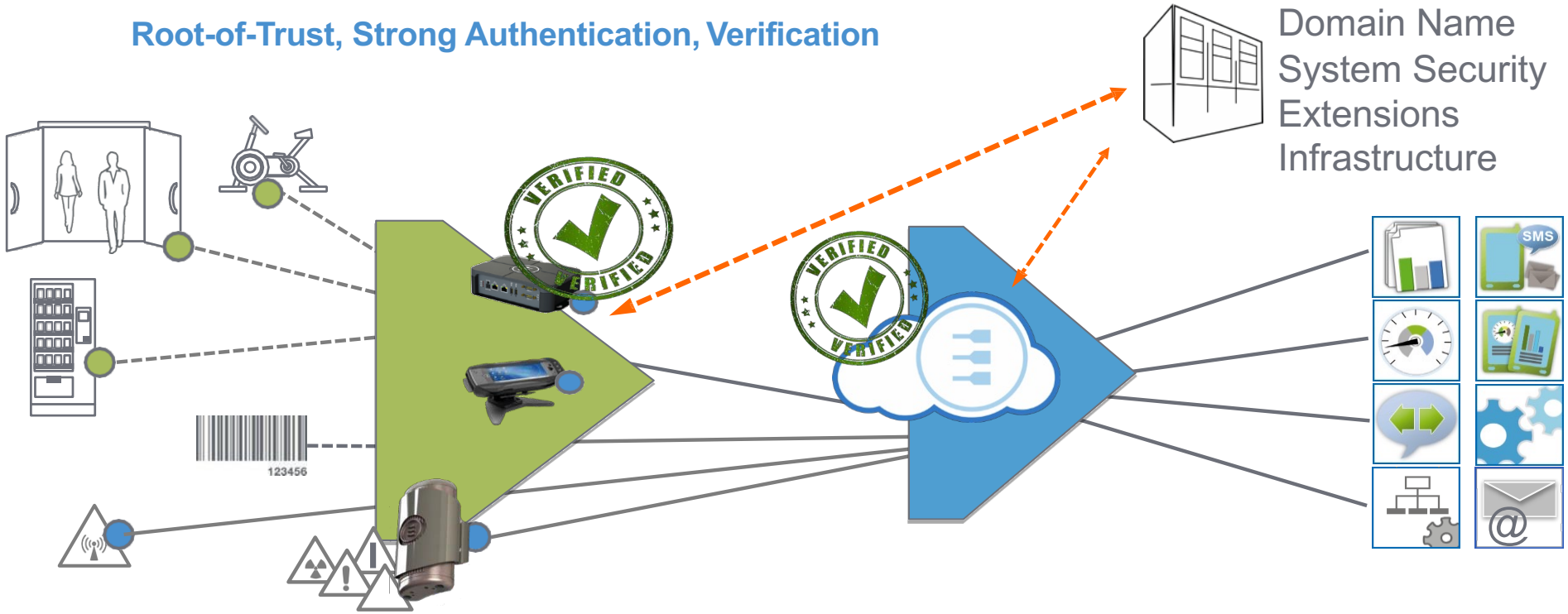
- Ease of use requirements on tools & IoT platform providers
- Hide complexity of hardware based security
- Provide built-in security functions
- Use standard methods and building blocks

::.. IoT Security Trend



::.. IoT Security Landscape

Root-of-Trust, Strong Authentication, Verification



Edge ⟷ **Gateways / Smart Devices** ⟷ **Network** ⟷ **DataCenter/ Cloud**

- Authentication (verified)
- Service discovery / provisioning / pairing
- Trusted execution environment
- Network security / firewall
- Secure Boot

IoT Device Security

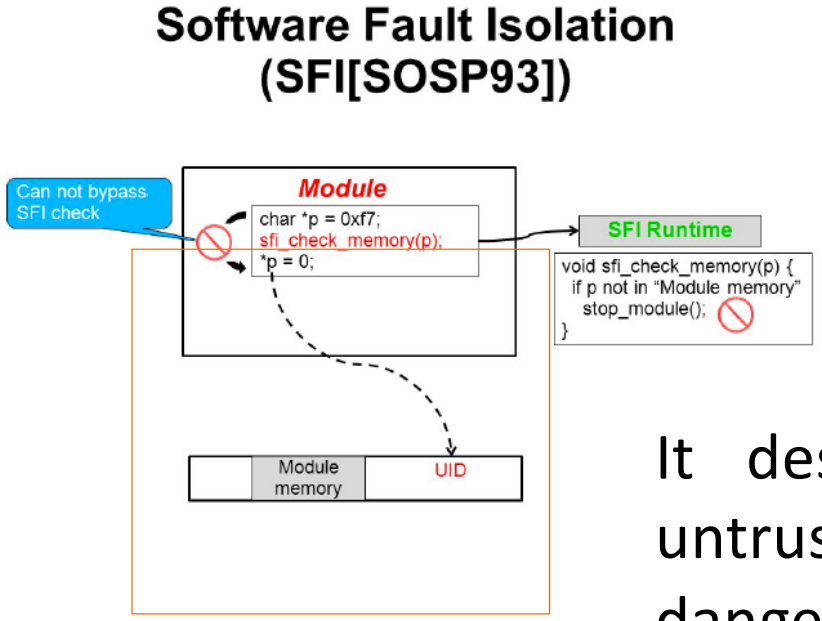
- Authentication (verified)
- Encryption
- Message integrity
- MitM protection
- DNS spoofing protection

Communication Security

- Authentication (verified)
- PKI / certificate management
- Trusted execution environment
- Network security / firewall
- Access control (role based)

Cloud Security

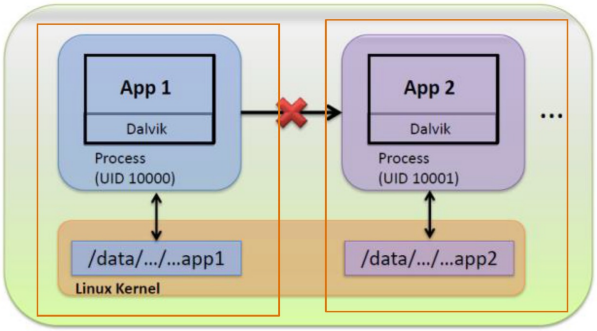
::.. Software fault isolation

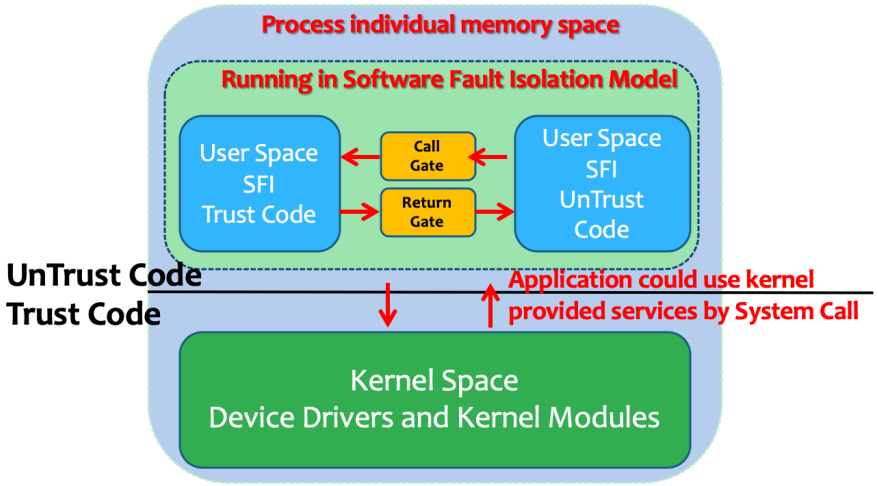


SFI is a software-instrumentation technique at the machine-code level for establishing logical protection domains within a process.

It designates a memory region for an untrusted component and instruments dangerous instructions to constrain its memory access.

It is referred to as code sandboxing.





::.. Isolated Execution (1/4)

Isolating code execution is one of the fundamental approaches to achieving security. Virtualization can create an isolated execution environment for running defensive tools. Existing virtualization-based approaches have limitations including:

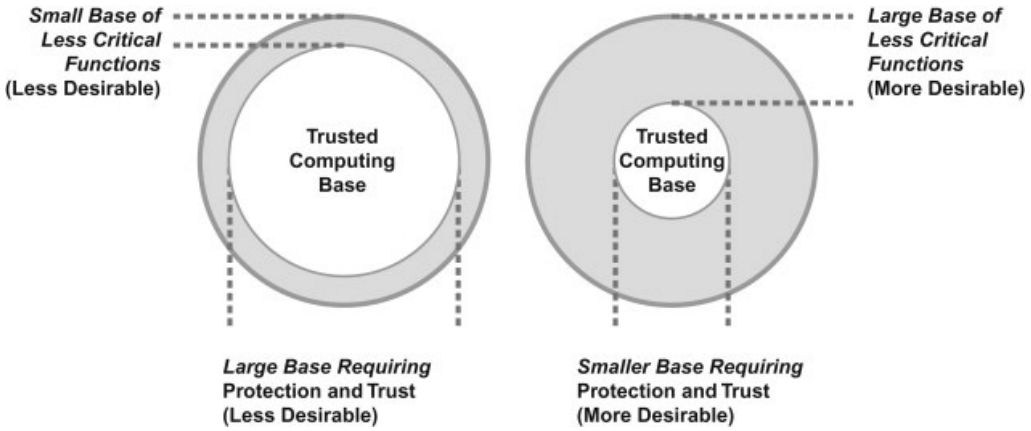
1. Dependence on hypervisors that may have a large Trusted Computing Base (TCB).

::.. Isolated Execution (1/4)

Isolating code execution is one of the fundamental approaches to achieving security. Virtualization can create an isolated execution environment for running defensive tools. Existing virtualization-based approaches have limitations including:

1. Dependence on hypervisors that may have a large Trusted Computing Base (TCB).

The TCB of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security: bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system.

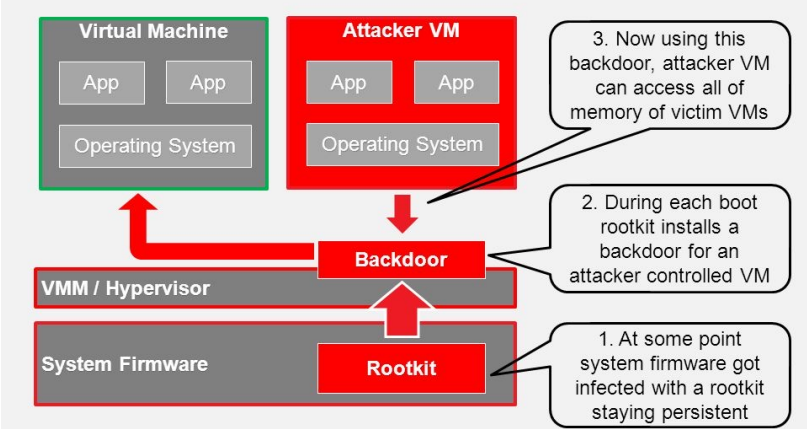


::.. Isolated Execution (1/4)

Isolating code execution is one of the fundamental approaches to achieving security. Virtualization can create an isolated execution environment for running defensive tools. Existing virtualization-based approaches have limitations including:

1. Dependence on hypervisors that may have a large Trusted Computing Base (TCB).
2. Failure to deal with hypervisor or firmware rootkits.

A **rootkit** is a collection of computer software, typically malicious, designed to enable access to a computer or an area of its software that is not otherwise allowed

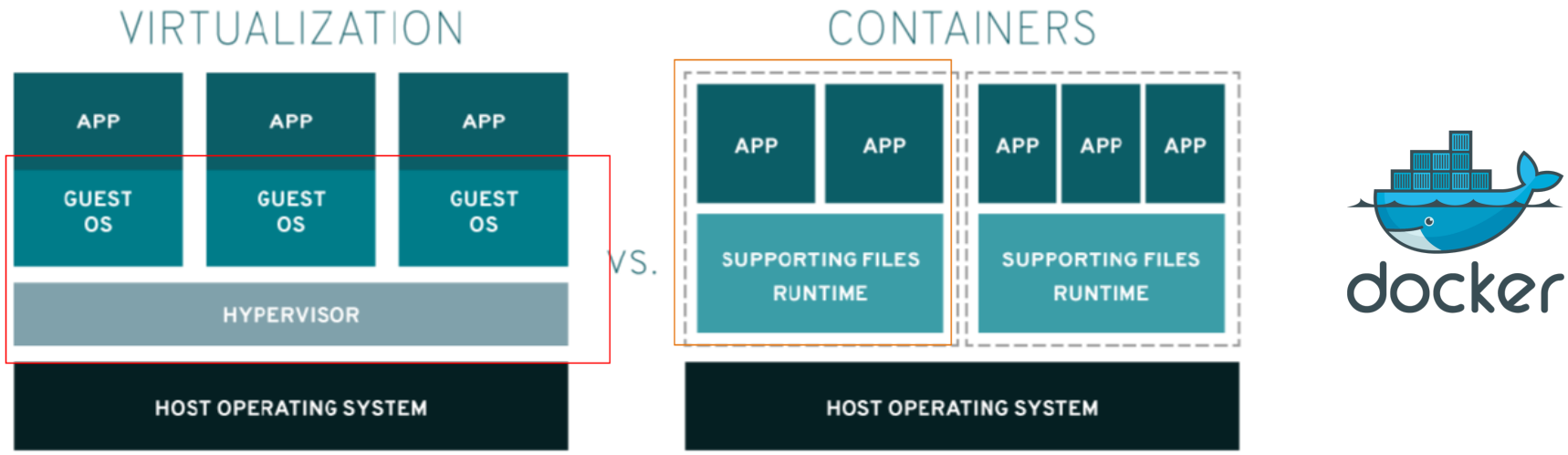


::.. Isolated Execution (1/4)

Isolating code execution is one of the fundamental approaches to achieving security. Virtualization can create an isolated execution environment for running defensive tools. Existing virtualization-based approaches have limitations including:

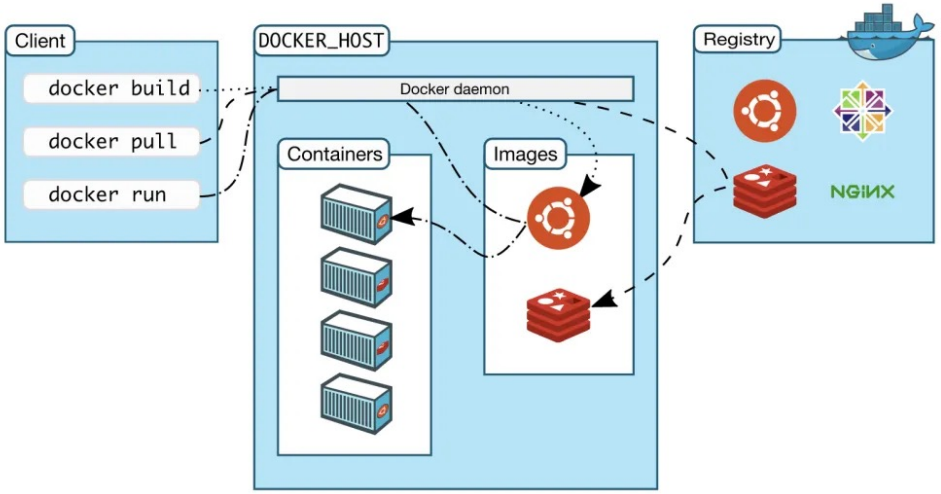
1. Dependence on hypervisors that may have a large Trusted Computing Base (TCB).
2. Failure to deal with hypervisor or firmware rootkits.
3. Suffering from system performance overhead (e.g., context switches from a VM to a hypervisor).

::.. Isolated Execution (2/4)



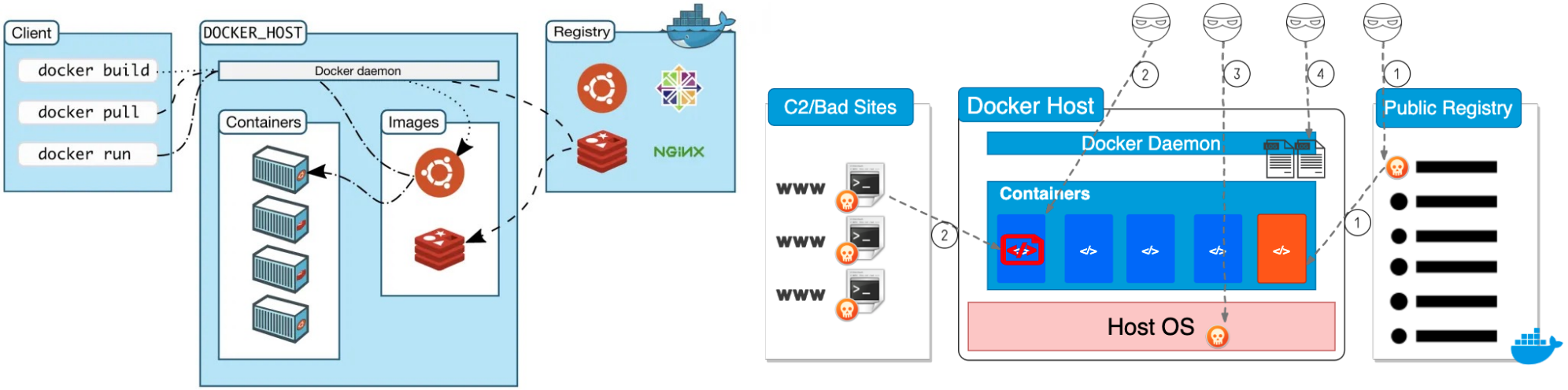
A different approach is realized with container-based solutions like Docker, which unlike, the hypervisor virtualization for which you have to create completely new machines to isolate them from each other and ensure their independence, Docker will allow you to create containers that will contain only your app.

::.. Isolated Execution (3/4)



Container-based virtualization is based on daemons to manage the containers and application images. Attacks are based on exploited misconfigured open Docker daemons, to hijack environments.

::.. Isolated Execution (3/4)

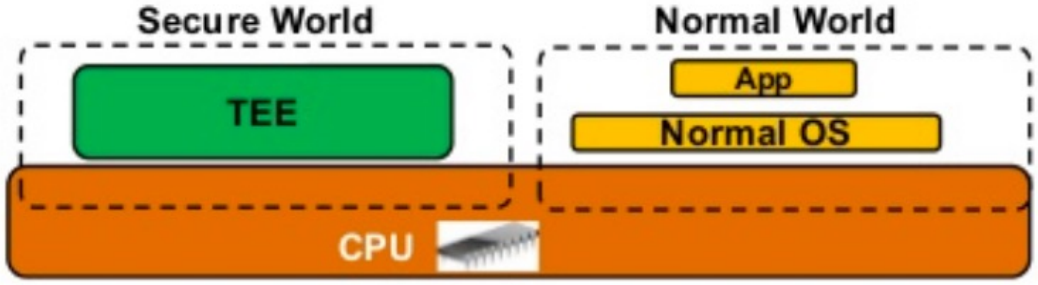


1. **Deploy Container Images with Malicious Code** - Malicious images are first pushed to a public registry to be pulled and deployed on the unsecured Docker hosts.
2. **Deploy Benign Container Images and Download Malicious Payloads at Run Time** - Benign images are deployed on the Docker hosts, where malicious payloads are then downloaded and executed.
3. **Deploy Malicious Payloads on the Host** - Adversaries mount the entire host file system to a container and access to it from the container.
4. **Obtain Sensitive Information from the Docker Log** - Adversaries scrape the Docker logs to find sensitive information.

::.. Isolated Execution (4/4)

Hardware-assisted Isolated Execution Environments has been realised for securing systems. This approach combines the isolated execution concept with hardware-assisted technologies. Both are crucial to secure computer systems:

- The isolated execution concept provides a Trusted Execution Environment (TEE) for running defensive tools on a compromised system.
- Using hardware-assisted technologies excludes the hypervisors from TCB, achieves a high level of privilege (i.e., hardware-level privilege), and reduces performance overhead giving that context switches are performed faster in hardware.



Separate the computing in normal and secure contexts, where in the second context critical software is run.

::.. Secure Execution Environment

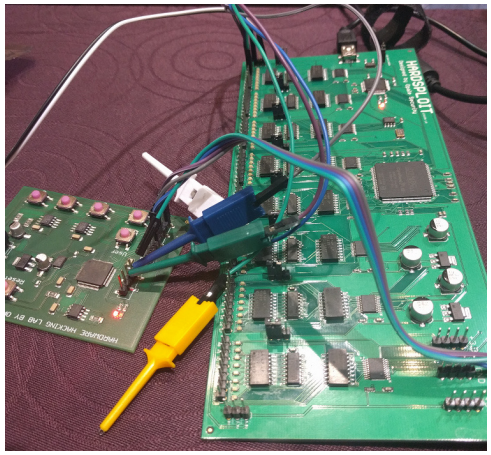
A secure environment allows the secure storage and execution of applications.

- Isolated execution: Every application should run independently of other applications.
  - a malicious application cannot access sensitive data hold by other secure applications within the memory.
  - the malicious application cannot access the code or data of an application while it is running and also cannot alter the execution.
- Secure storage: The integrity and secrecy of all data are guaranteed, including the binaries representing the applications to be run. the most sensitive data to protect prove to be the passwords, encryption keys and certificates.
- Secure provisioning: This property guarantees both the capability to securely send data to a specific software in the secure environment and the ability to remotely install sensitive applications and transfer encryption keys or certificates.

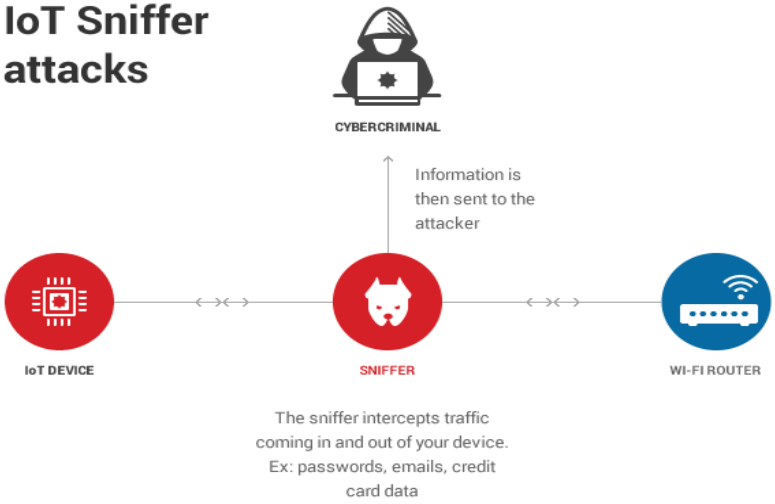
An application running in an environment which guarantees these three characteristics is called a secure application.

::.. Hardware-Based Solutions

Hardware security solutions have the advantage of greatly reducing intrusions and attacks.



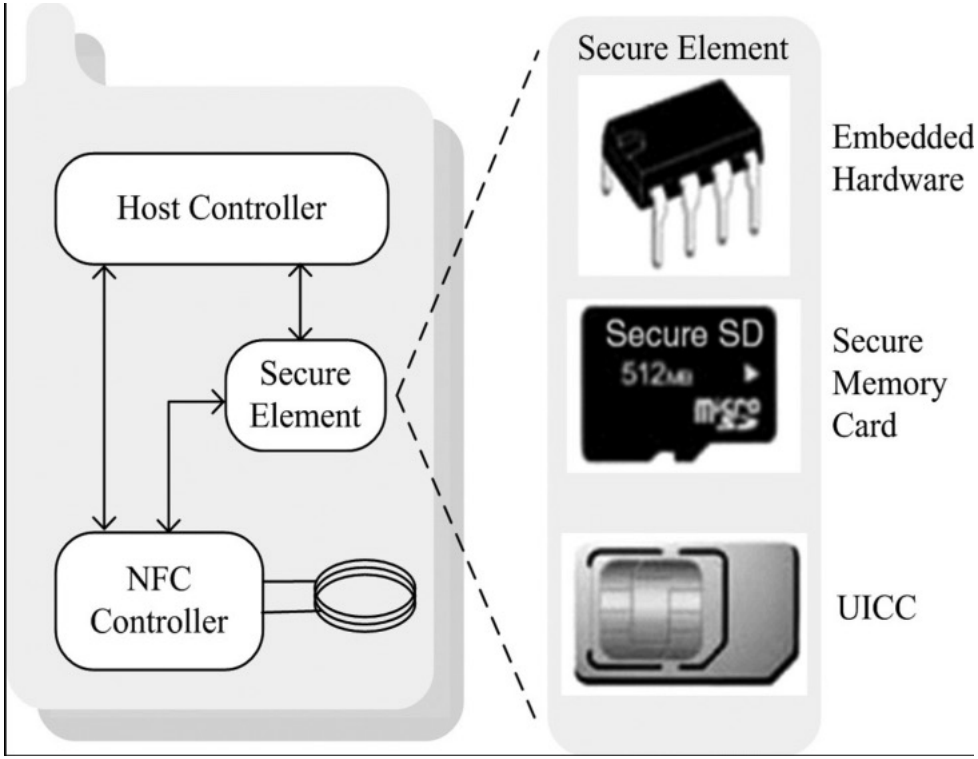
Malicious adversaries may run attacks able to inspect the storage memory or intercept the key during the execution process, making encryption not fully able to address the security issues in the IoT world.



The massive deployment of SIM card incorporating a Secure Element is the origin of the hardware-based solutions.

::.. Secure Elements

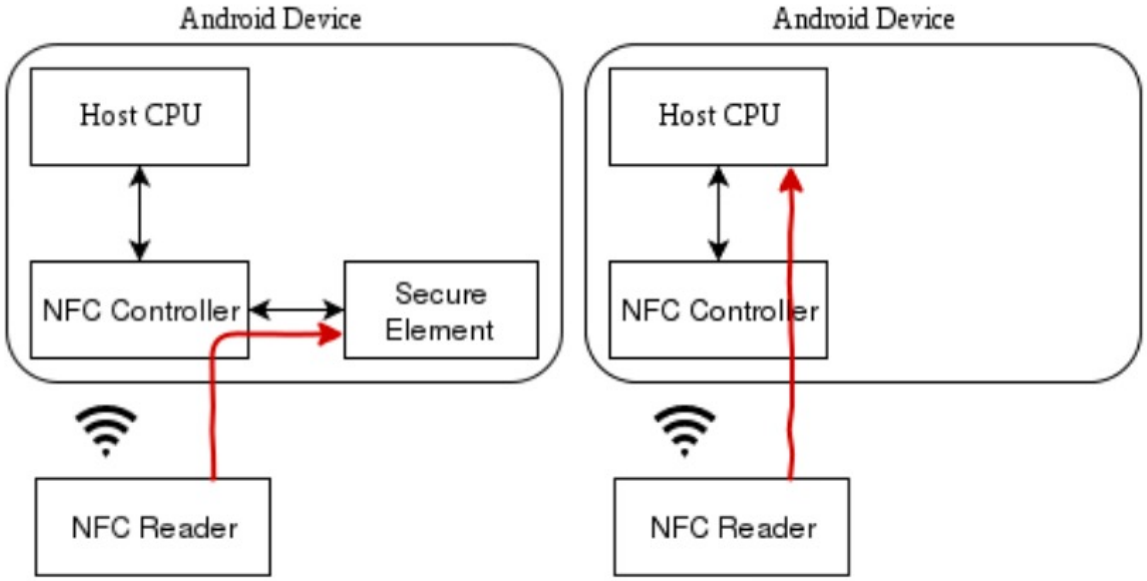
A Secure Element (SE) is a microprocessor chip which can store sensitive data and run secure apps such as payment. It acts as a vault, protecting what's inside the SE (applications and data) from malware attacks that are typical in the host (i.e. the device operating system).



Only the applications signed by the manufacturer are allowed to run in a SE, which limits the possibilities of a malicious app attacker and of a root attacker. This is a big restriction for developers and the main reason that hindered the development of this technology.

::.. SE Application

SEs find application to device the integration of smart cards within embedded system, so that NFC-enabled devices to act as contactless smart card. This can be used by mobile applications to perform transactions through NFC, such as banking and transport-related applications. Its ability to emulate contactless smart cards facilitates interoperability with existing card-reader infrastructures based on NFC.



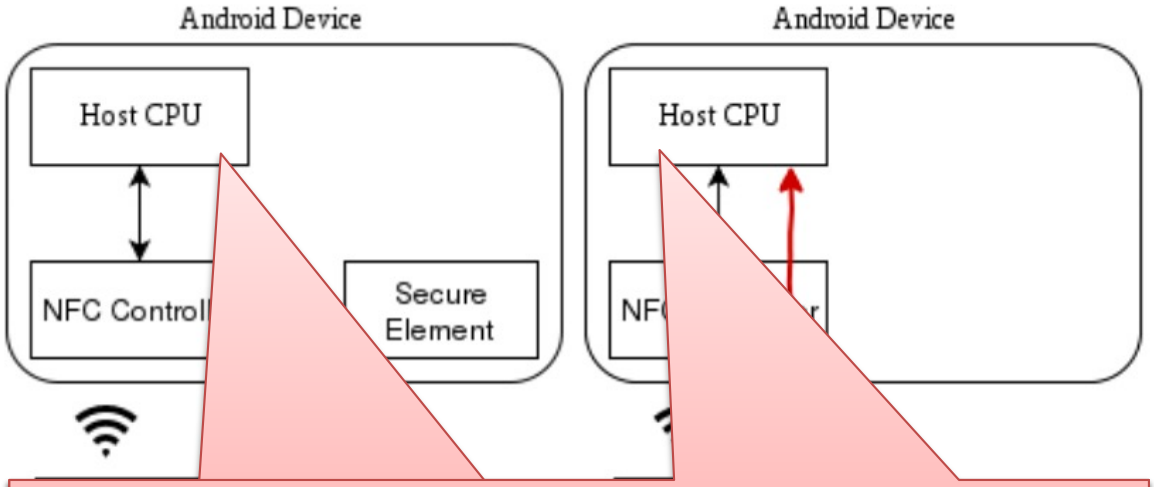
(a) SE-based (b) HCE-based

When an SE is used, typically all of the data received by the NFC controller is routed directly to the SE.

Host-based Card Emulation (HCE) is a different solution where data is directed to the device's CPU, which is responsible for handing it to the corresponding application via the operating system.

::.. SE Application

SEs find application to device the integration of smart cards within embedded system, so that NFC-enabled devices to act as contactless smart card. This can be used by mobile applications to perform transactions through NFC, such as banking and transport-related applications. Its ability to emulate contactless smart cards facilitates interoperability with existing card-reader infrastructures based on NFC.



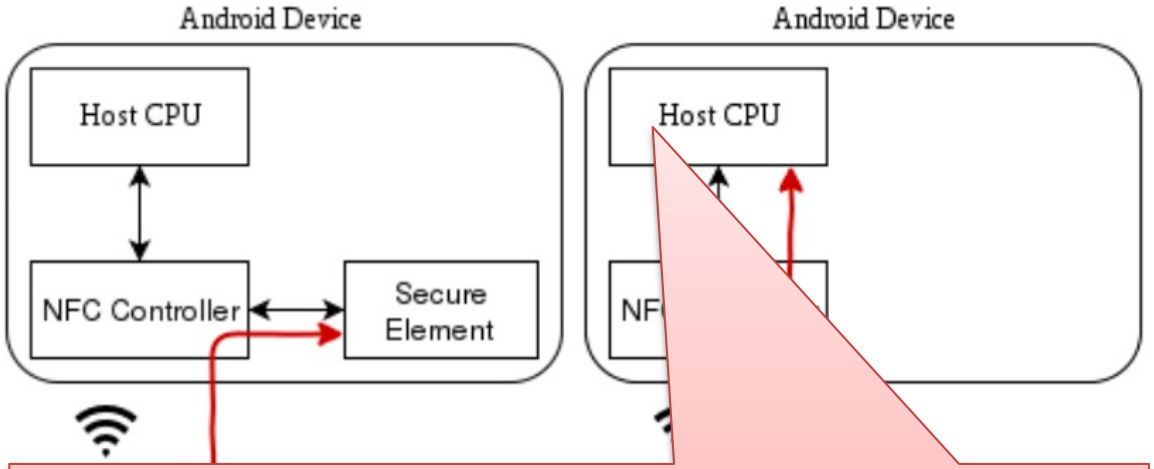
In both cases, data is passed to an application based on an Application ID (AID), as stipulated in the ISO/IEC 7816-4 specification.

When an SE is used, typically all of the data received by the NFC controller is routed directly to the SE.

Host-based Card Emulation (HCE) is a different solution where data is directed to the device's CPU, which is responsible for handing it to the corresponding application via the operating system.

::.. SE Application

SEs find application to device the integration of smart cards within embedded system, so that NFC-enabled devices to act as contactless smart card. This can be used by mobile applications to perform transactions through NFC, such as banking and transport-related applications. Its ability to emulate contactless smart cards facilitates interoperability with existing card-reader infrastructures based on NFC.



Because HCE is software-based, it cannot provide the level of tamper-resistance that a hardware SE can. Device storage may only be protected by the security mechanisms of the operating system, such as application sandboxing.

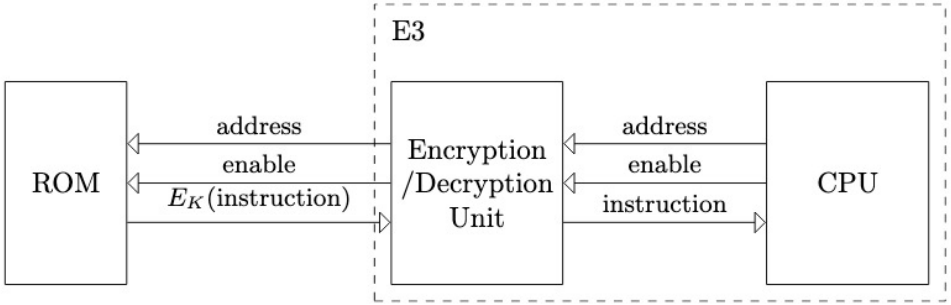
When an SE is used, typically all of the data received by the NFC controller is routed directly to the SE.

Host-based Card Emulation (HCE) is a different solution where data is directed to the device's CPU, which is responsible for handing it to the corresponding application via the operating system.

::.. E3

An Encrypted Execution Environment (E3) is an execution environment in which the application software is encrypted, and has the goal of enabling execution without revealing the instructions composing the application.

This is likely to be employed a company that allo-cates a large amount of monetary value and time to protectits investment from counterfeiting and other unauthorisedduplication.



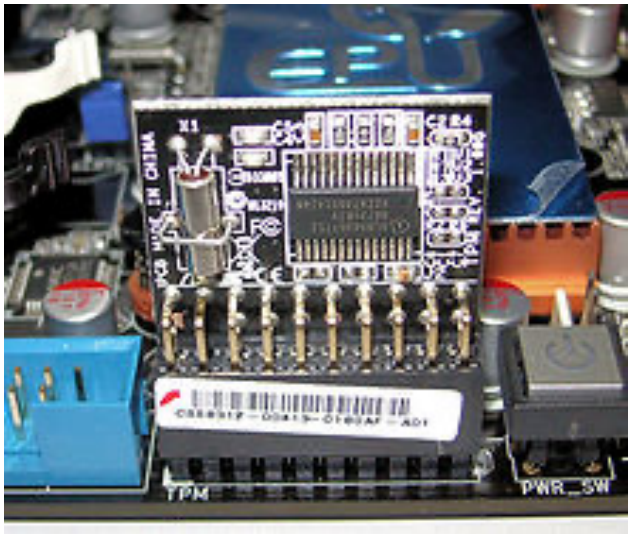
This does not necessarily imply that the instructions can be executed directly from their encrypted state. Indeed, it is acceptable for the E3 to decrypt and execute instructions so long as plain-text instructions are not revealed to the E3 environment.

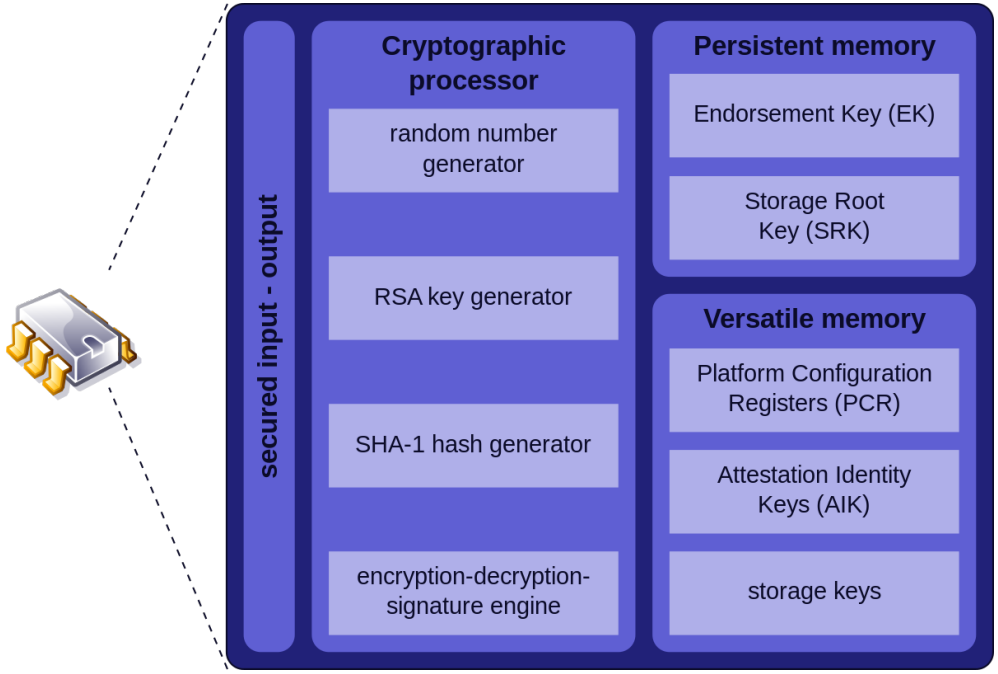
If each device provisioned with software has its own E3 key, then E3 software for one device will not execute on another.

::.. Trusted Platform Module (1/4)

The TPM consists of a micro-controller with additional cryptographic capabilities. The TPM can be bound to the device using a Low Pin Count (LPC) bus, and can perform very complex cryptographic operations from the symmetric encryption to RSA asymmetric encryption.

The advantage of using a TPM is that the developer has not to know anything about the implementation of these algorithms, as the TPM provides an Application Programming interface (API).





::.. Trusted Platform Module (2/4)

The cryptographic features of a TPM are the following:

- RSA accelerator: An engine module performs RSA encryption/decryption with a maximum key length of 2048 bits. A built-in RSA engine is used during digital signing and key wrapping operations.
- The TPM is capable of computing hash values of pieces of data, but it is not sufficient to hash large pieces of data, only small sensitive data such as encryption keys and certificates.
- Generating pseudo random numbers: This feature is very important and useful to generate encryption keys especially for RSA for instance.

The TPM is characterised by the endorsement key, created randomly on the chip at manufacture time, cannot be changed and never leaves the chip, while the public key is used for attestation and for encryption of sensitive data sent to the chip.

::.. Trusted Platform Module (3/4)

The Attestation Identity Keys are hold by the TPM to perform an authentication with a service provider. This authentication affects the platform and differs from the user authentication.

The TPM stores three kinds of certificates:

- Endorsement certificate ensures the integrity of the Endorsement Key. This certificate can be provided by the same issuer as the EK but it is not mandatory.
- Platform certificate is provided by the platform vendor and ensures that all the security components provided with the platform are genuine. This certificate enables the platform trust.
- Conformance certificate is provided by a third party evaluation lab or by the platform vendor itself. It certifies that the security properties claimed by the manufacturer are genuine.

::.. Trusted Platform Module (4/4)

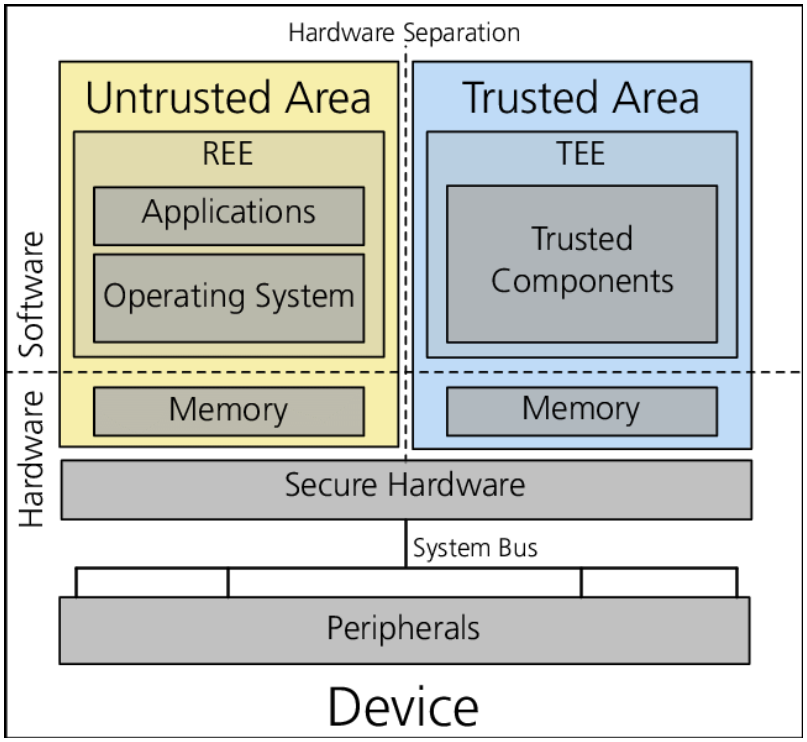
The TPM protects data by means of two possible solutions:

- **Memory curtaining** extends common memory protection techniques to provide full isolation of sensitive areas of memory—for example, locations containing cryptographic keys. Even the operating system does not have full access to curtained memory.
- **Sealed storage** protects private information by binding it to platform configuration information including the software and hardware being used. This means the data can be released using a key that derives from the state of the system, i.e. the software used and the hardware on which it is running. This means that this information can only be used with the same combination of software and hardware.

As a use case, a TPM can be used as a PKI that provides the required keys and certificates for establishing secure communications and signing documents.

::.. TEE (1/10)

Trusted Execution Environments (TEE) combine hardware and software parts, and let the system to be divided into two execution environments.



- The first environment is the Rich Execution Environment (REE) represents the standard OS. The term Rich describes the extensive features of the OS, which significantly increase the attack surface.
- The second environment is the TEE. And represents the Secure OS responsible for performing sensitive operation such as cryptographic operations. It also has the capability to secure the display and the input by using a secure mode of the buses connecting the processor to the I/O peripherals.

The TEE splits the processor in two zones with a secure OS and other mechanisms to enhance the security of sensitive data processing.

::.. TEE (2/10)

The other essential feature is the secure storage, where critical and sensitive data are isolated from the user data to enhance its security. In opposition to the SE, the TEE provides a secure communication channel between the processor and the external peripheral especially input and display.

- If a malicious application can intercept the input, it can have access to sensitive data like pass phrase. The secure display is also primordial because the user has to be sure that what he sees on the screen is really sent by the secure world.

A secure boot process is also needed:

1. Read a trusted ROM (locked at manufacturing),
2. Signature and integrity checking of the Secure OS,
3. Setting up the Secure OS which takes the control.

TEE has also a third mode called Monitor mode. This mode is used to perform context saving and switching between the Rich OS and the Secure OS. This mode is only accessed by the Secure OS to request a context switching when all the operations are performed.

::.. TEE (3/10)

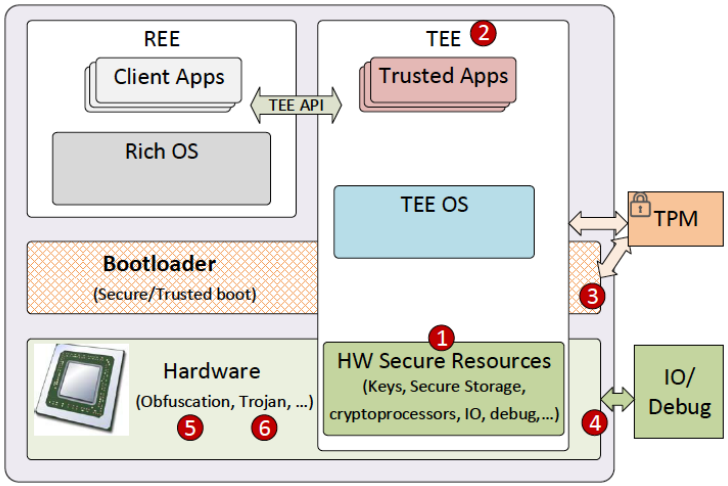
The Secure OS has a limited instruction set, necessary to reduce the attack surface, and schedules sensitive applications running on it to execute secure instructions like cryptographic operations: key generation, encryption and decryption.

Comparison between the hardware solutions.

Criteria	SE	TEE	TPM
Tamper resistance	✓		✓
Secure input and display		✓	
High computation power		✓	✓
High storage capacity		✓	
Dependency to manufacturer	✓	✓	✓
Proven security level	✓	✓	✓

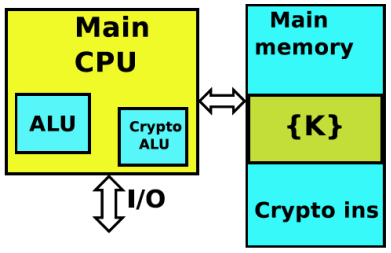
SE and the TPM have more physical security features than TEE. However, TEE provides a bigger computation power which enables more complex security models.

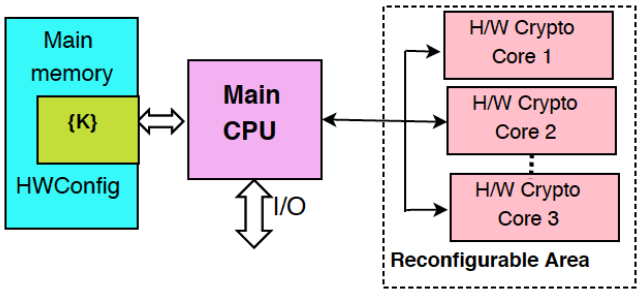
::.. TEE (4/10)



The base for the hardware-level realization of TEE is made of cryptoprocessors, which are a specialized processor that executes cryptographic algorithms on hardware level and accelerates the encryption and decryption process for better data security and secret key protection.

A general-purpose processor (GPP) is customized and used for implementing some cryptographic algorithms on it. The main customization are instruction set extensions (may be called cryptographic instruction) for cryptographic applications. In this case, secret keys are saved in the main memory and used like other normal data.





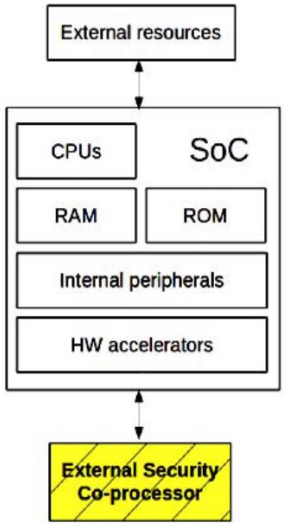
A cryptographic coprocessor is a module outside the GPP that accelerates cryptographic computations. Secret keys are normally not stored in the cryptographic coprocessor memory, they are stored as data in the processor data registers or main memory. The cryptographic coprocessor can be controlled or parametrised using the host GPP

::.. TEE (5/10)

There is a number of ways in which these TEEs can be realized.

::.. TEE (5/10)

There is a number of ways in which these TEEs can be realized.



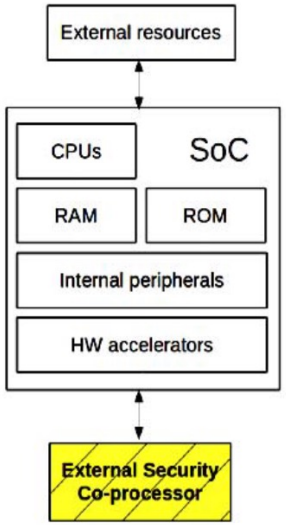
The first realization uses a security coprocessor to offload the security critical tasks from the main operating environment.

- The benefits are the operation can generally be completely isolated and it can run simultaneously with the main core.
- The drawback is that there is an overhead associated with transferring the data to and from the core.

It can have an external (with HSM) or embedded (TPM) security coprocessor.

::.. TEE (5/10)

There is a number of ways in which these TEEs can be realized.



The first realization uses a s
tasks from the main operati

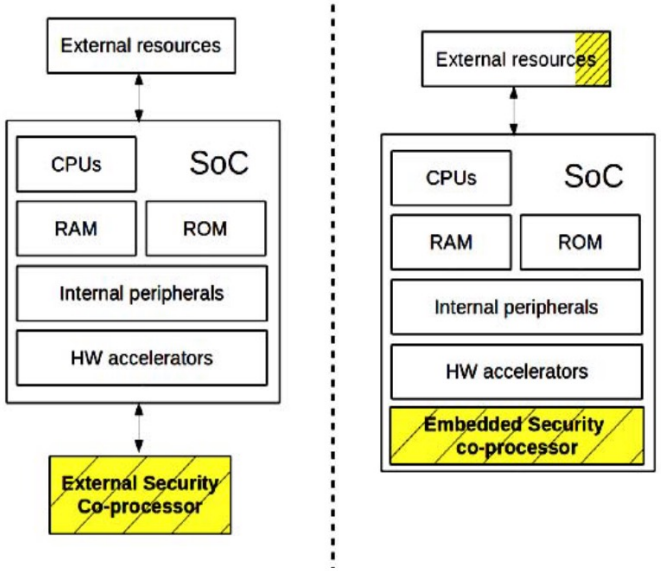
- The benefits are the op
  it can run simultaneous
- The drawback is that t
  the data to and from the core.

A TPM is a hardware chip on the motherboard and is capable to provides full disks encryption.
TPM keeps hard drives locked until the system completes a system verification or authentication process. On the other hand, HSM is a removable or external device that generates, stores and manages cryptographic keys.

It can have an external (with HSM) or embedded (TPM) security coprocessor.

::.. TEE (5/10)

There is a number of ways in which these TEEs can be realized.

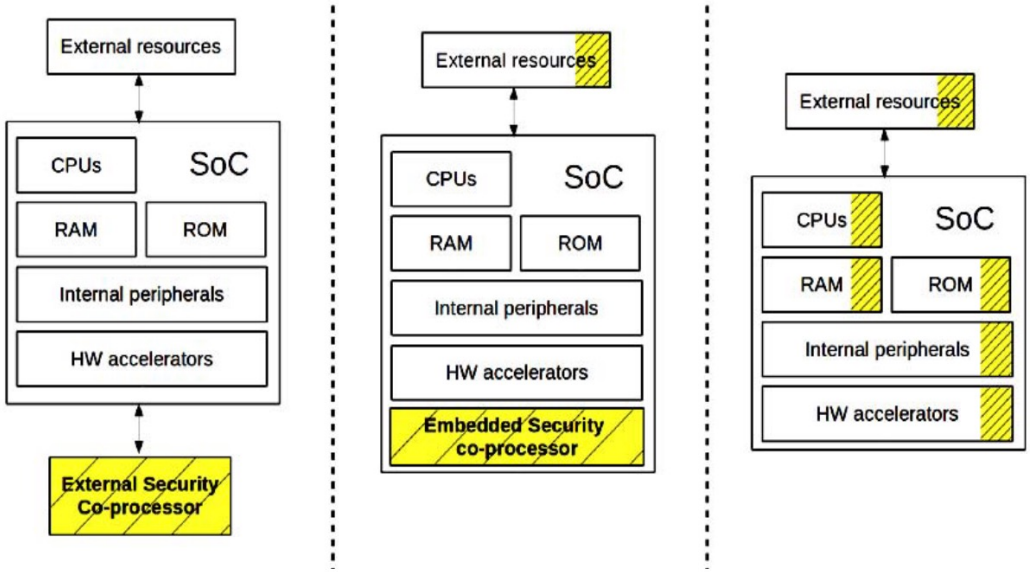


Many popular TEE architectures support a new kind of configuration where a single core supports multiple virtual cores that are mutually exclusive of one another i.e. when one is running the other is suspended. Normally this transition from one state to other is done by some sort of monitor/trigger. This configuration is sometimes referred to as the “processor secure environment”.

Examples are ARM TrustZone and Intel Trusted Execution Technology.

::.. TEE (5/10)

There is a number of ways in which these TEEs can be realized.



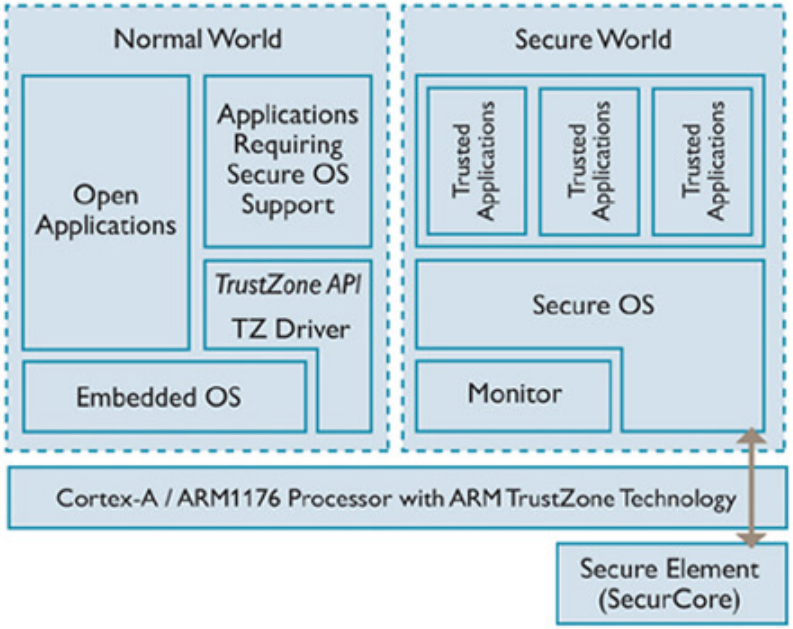
Beside the hardware based techniques, other approaches were presented, at the architecture level XOM (eXecute-Only-Memory), platforms achieves TEE by architectural separation, it prevents information from leaking out of the XOM applications by using compartment, where a compartment is a logical container that prevents information from flowing into or out of it.

Additional examples of software TEE include Docker containers.

::.. TEE (6/10)

ARM TrustZone processor has a specific instruction called Secure Monitor Call (SMC) is used to trigger the processor running in normal world to enter "monitor mode" that marshals the transition to secure world.

- No need to unload the data to/from the secure world. There is an extra cost to store/restore the device state on entry/exit from a given mode.
- Disadvantage is that when one world is active the other world must be completely halted, thus complicating interrupt handling.



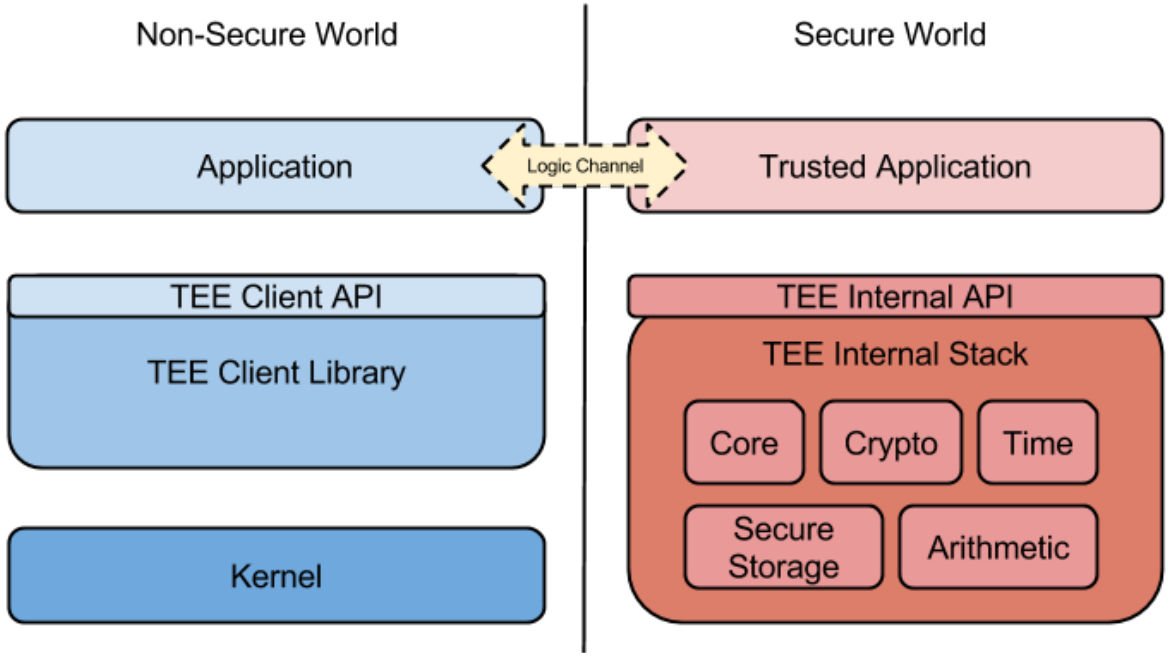
The most significant architectural innovation is the addition of a new 33rd processor bit, the non-secure bit, which indicates in which world the processor is currently executing.

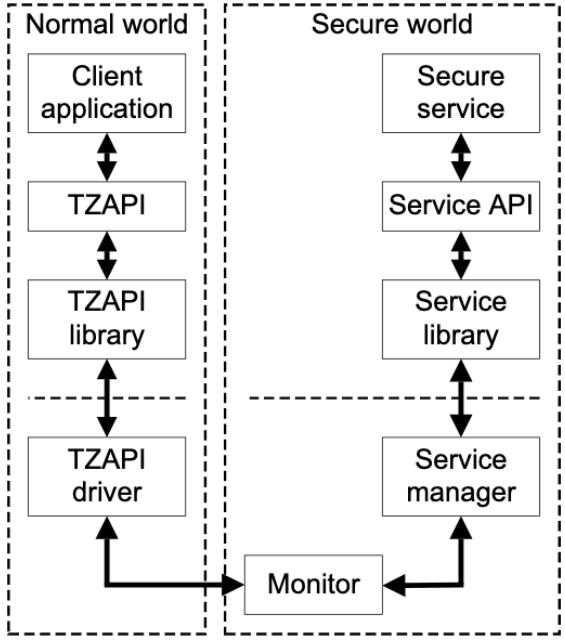
The TrustZone address space controller (TZASC) extends security at the memory level, by enabling partition of DRAM into different memory regions. The Trust-Zone-aware memory management unit (MMU) provides two distinct MMU interfaces.

::.. TEE (7/10)

The TrustZone API (TZAPI) specifies how non-secure applications (NSAs), running on the rich environment, interact with the isolated execution environment.

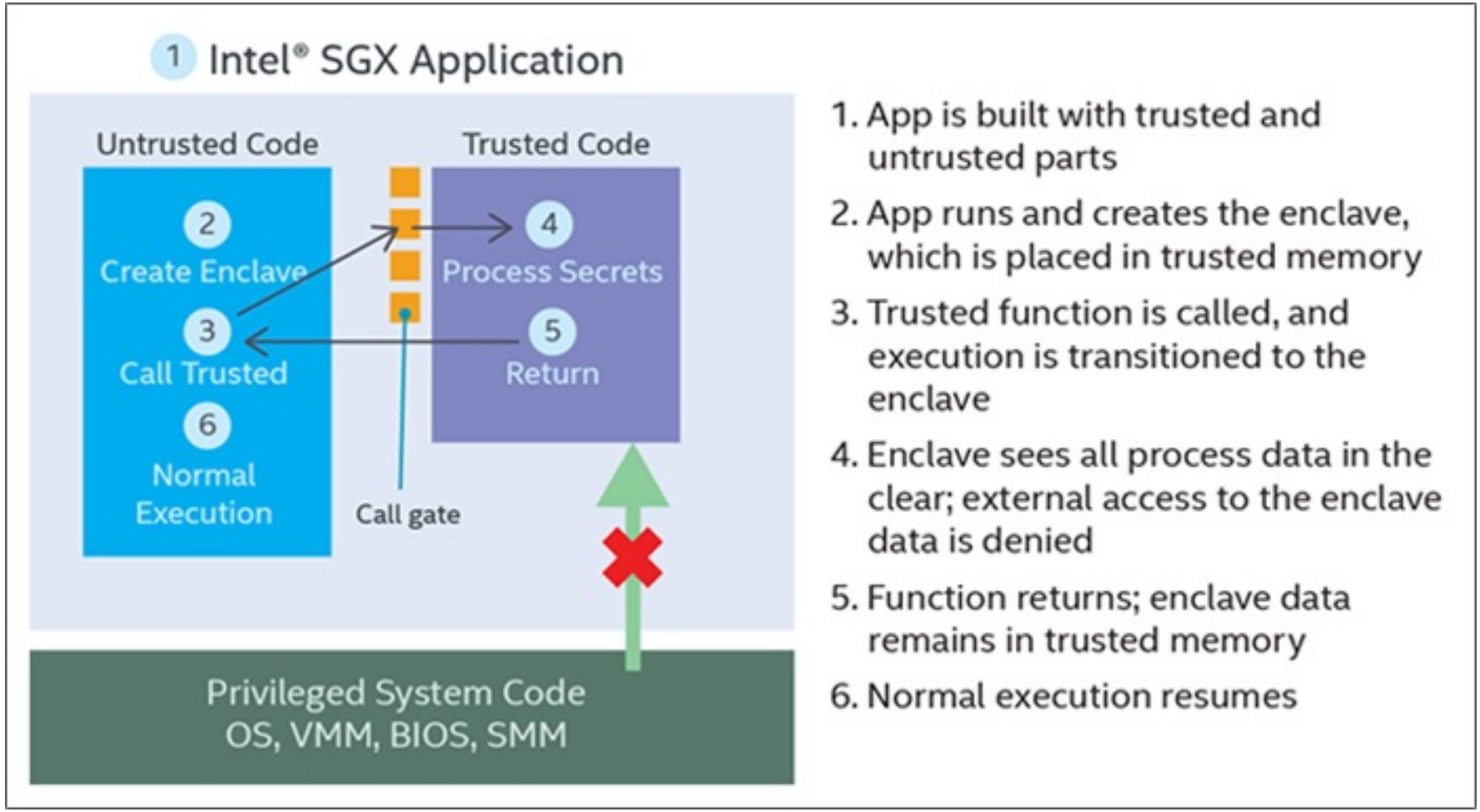
Following a client−server model, the API defines a set of abstract software interfaces by which an NSA can interact with a TA. The API allows clients to send commands and requests to a TA, and exchange data between both worlds. The TrustZone API doesn't include any specification about how to develop applications running inside the isolated execution environment.





::.. TEE (8/10)

Techniques such as Intel Software Guard Extensions (Intel SGX) use both hardware and software to implement TEE.

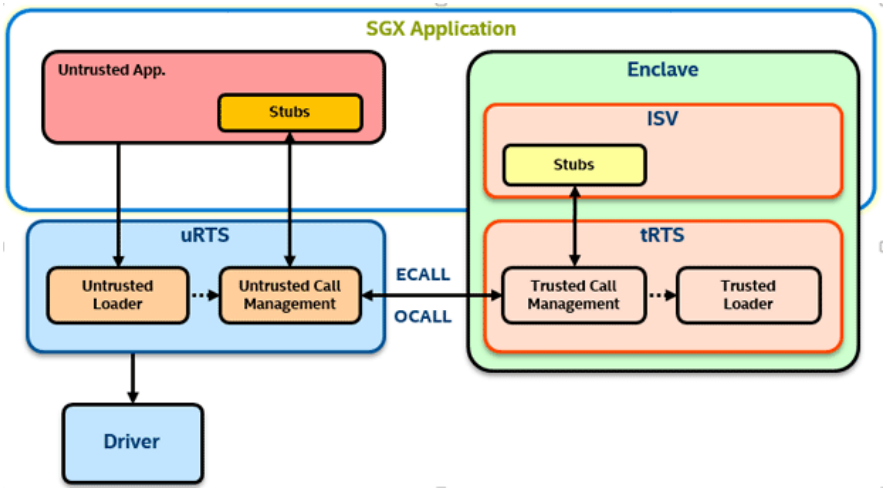


::.. TEE (9/10)

SGX is a set of CPU instructions that allow an application to instantiate a protected container, referred to as an enclave.

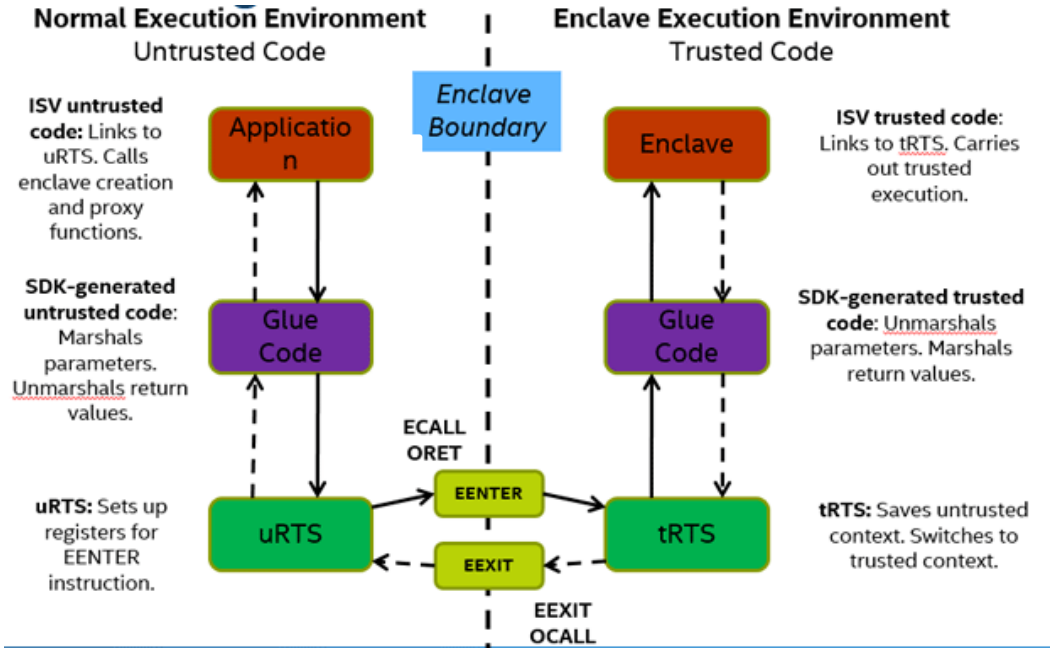
- An enclave is defined as a protected area in the application's address space which cannot be altered by code outside the enclave, not even by higher privileged code (e.g., kernel, virtual machine monitors, BIOS).
- The core does not perform a full transition, but parts of a standard application are protected by hardware mechanisms in the core.
- The benefits are that there is no need to transfer data back and forth between cores or to setup complicated transitions to and from a secure world, and there is no additional need for a separate operating environment as is required in other styles of TEE configuration.
- The trusted component should be as small as possible: a large enclave with a complex interface doesn't just consume more protected memory: it also creates a larger attack surface.
- Enclaves should also have minimal trusted-untrusted component interaction: limiting these dependencies will strengthen the enclave against attack.

::.. TEE (10/10)



Untrusted Run-Time System (uRTS) indicates the code that executes outside the enclave environment and performs loading and manipulating an enclave and making calls (ECALLs) to an enclave and receiving calls (OCALLs) from an enclave.

Trusted Run-Time System (tRTS) represents the code that executes within the enclave environment and performs receiving calls (ECALLs) from the application and making calls outside (OCALLs) the enclave, or managing the enclave itself.



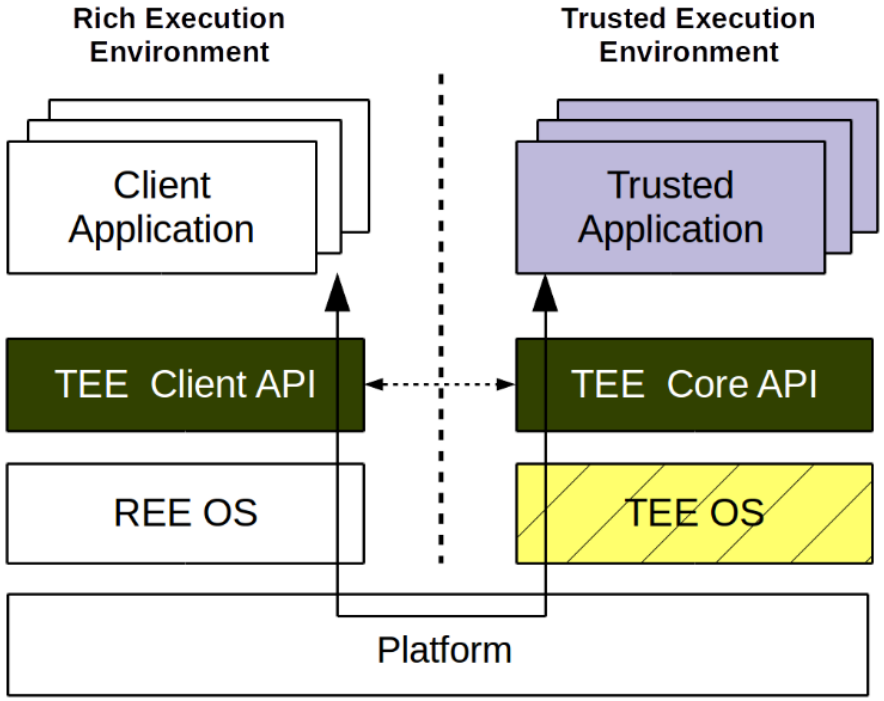
::.. Open-TEE (1/3)

Application developers have lacked the interfaces to use hardware-based TEE functionalities. In fact, their use has been limited primarily to applications developed by the device vendors. Recent standardization of TEE interfaces by Global Platform (GP) promises to partially address this problemby enabling GP-compliant trusted applications to run on TEEs from different vendors.

Ordinary developers wishing to develop trusted applications face significant challenges. Access to hardware TEE interfaces are difficult to obtain without any support from vendors. Tools and software needed to develop and debug trusted applications may be expensive or non-existent, with only primitive debugging techniques like “print tracing”.

A virtual TEE called Open-TEE, which conforms to Global Platform Specifications, has been proposed as a virtual standards-compliant TEE implemented entirely in software will allow developers to build TEE applications using tools and development environments that they are already familiar with.

::.. Open-TEE (2/3)



GlobalPlatform specification is made of

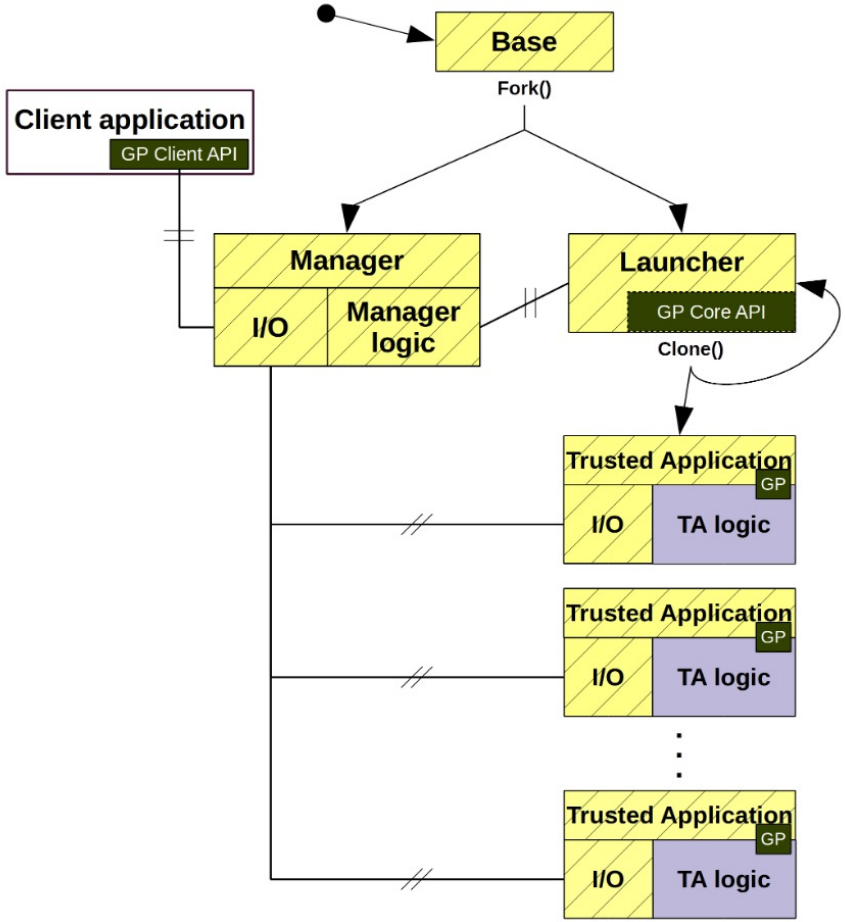
- The TEE Core API provides an extensive set of features, such as a crypto API and secure storage, that can be used to implementa TA.
- The TEE Client API is a very generic and thin layer consisting of a small number of functions and definitions to transfer data back and forth from the REE to a TA.

Between the TEE Client API running on the REE and TEE Core API running on the TEE, we have an effective Remote Procedure Call (RPC) where a process running in the REE can invoke tasks in the TEE.

These standardization efforts in Global Platform could resolve the issue of interoperable TEEs. However, they do not remove the obstacle in gaining access to the requisite hardware nor does simplify the task of developing and testing TAs.

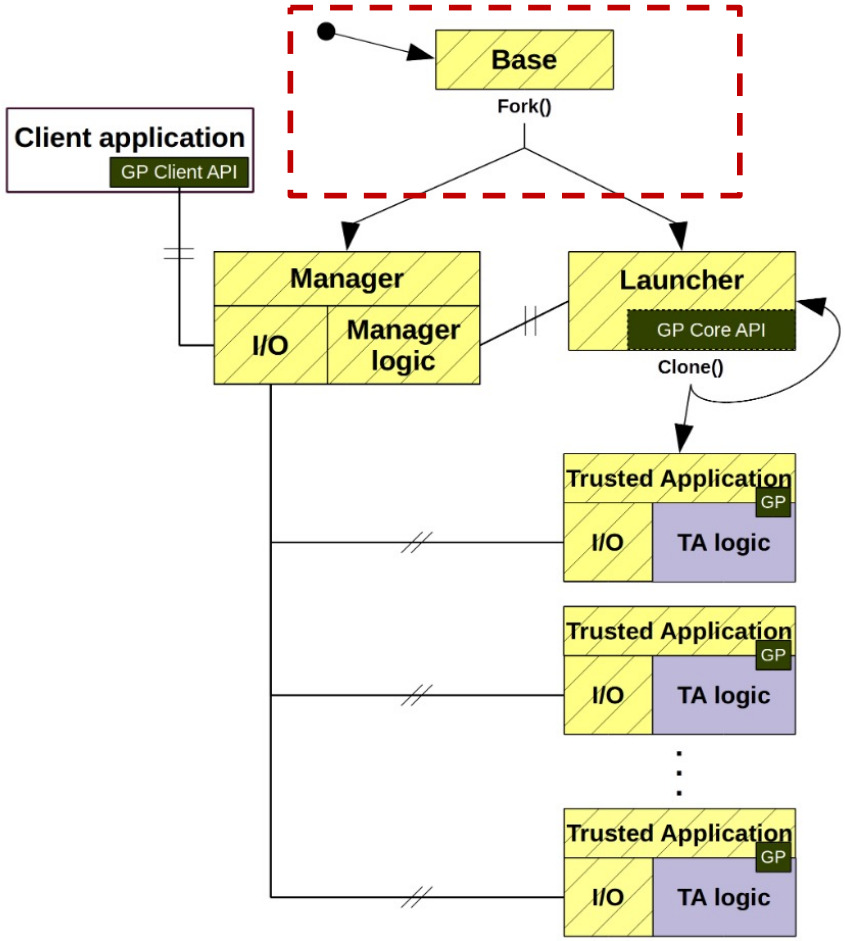
::.. Open-TEE (3/3)

Open-TEE provides an architecture and a software development kit (SDK) that implements GP specification as a framework a top a set of tools that are familiar to the developer, thus removing the need for specialized hardware and the overheads that it incurs.



::.. Open-TEE (3/3)

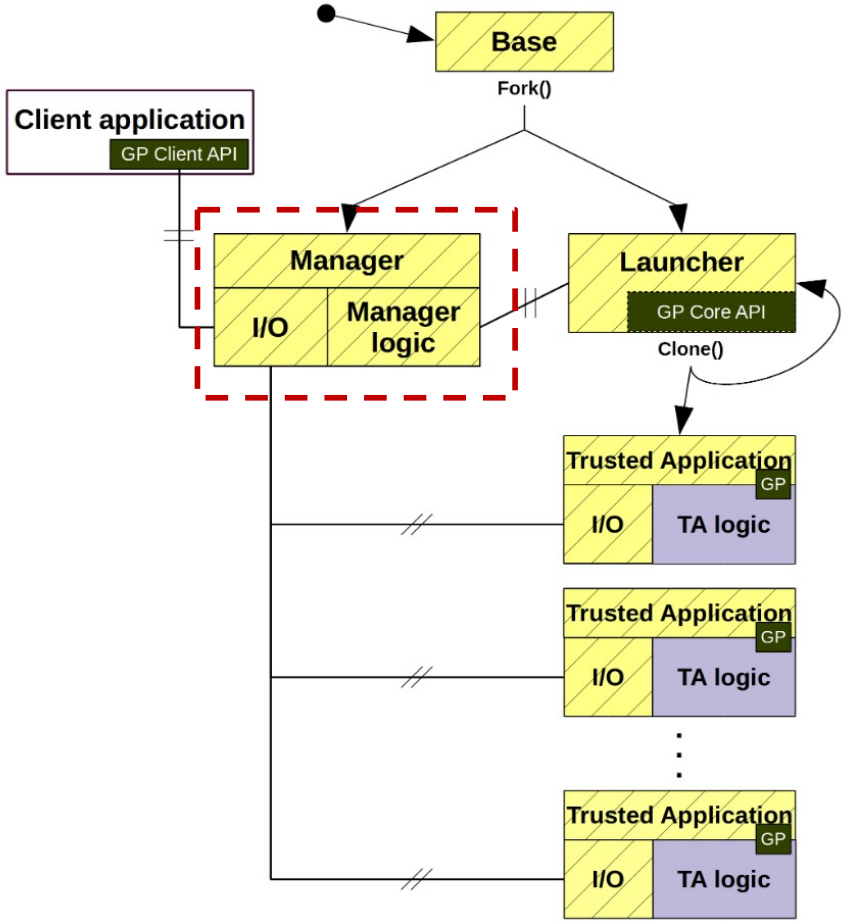
Open-TEE provides an architecture and a software development kit (SDK) that implements GP specification as a framework a top a set of tools that are familiar to the developer, thus removing the need for specialized hardware and the overheads that it incurs.



- Open-TEE is designed to function as a daemon process in user space. It starts executing Base, a process that encapsulates the TEE functionality as a whole. Once initialized the Base will fork to create two independent but related processes.

::.. Open-TEE (3/3)

Open-TEE provides an architecture and a software development kit (SDK) that implements GP specification as a framework a top a set of tools that are familiar to the developer, thus removing the need for specialized hardware and the overheads that it incurs.

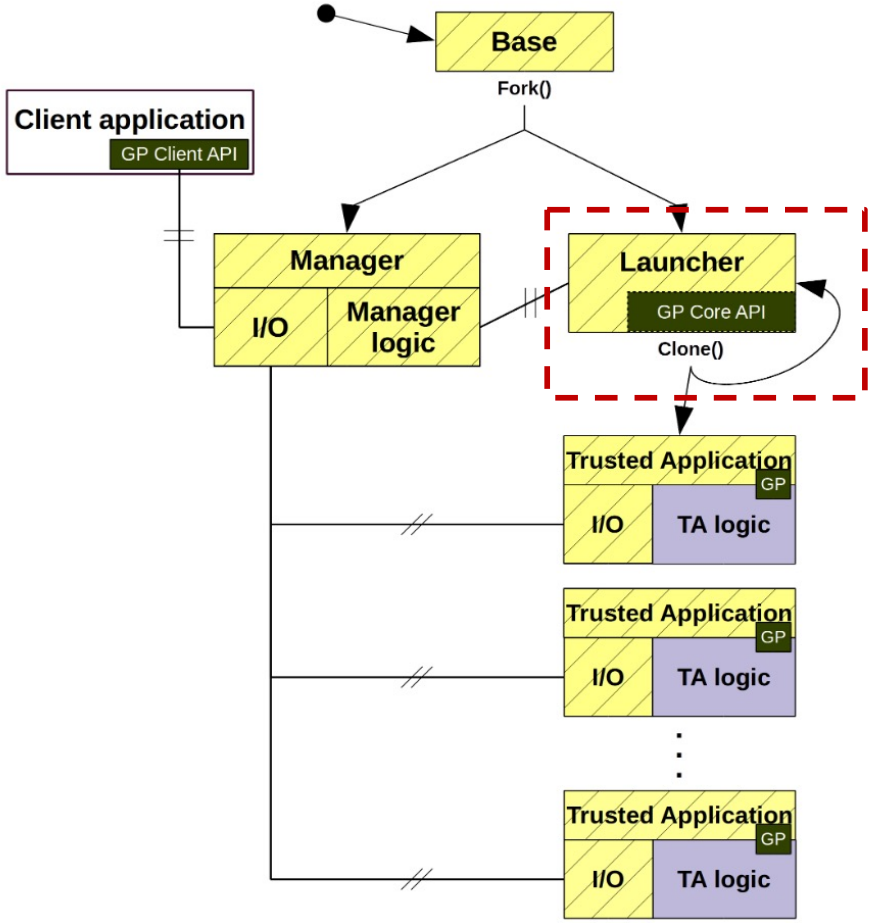


- Manager can be visualized as Open-TEE's "operating system":
  1. Managing connections between applications,
  2. Monitoring TA state,
  3. Providing secure storage for a TA,
  4. Controlling shared memory regions for the connected applications.

Centralizing this into a control process can also be seen as a wrapper abstracting the running environment and reconciling it with the requirements imposed by the GP TEE standards.

::.. Open-TEE (3/3)

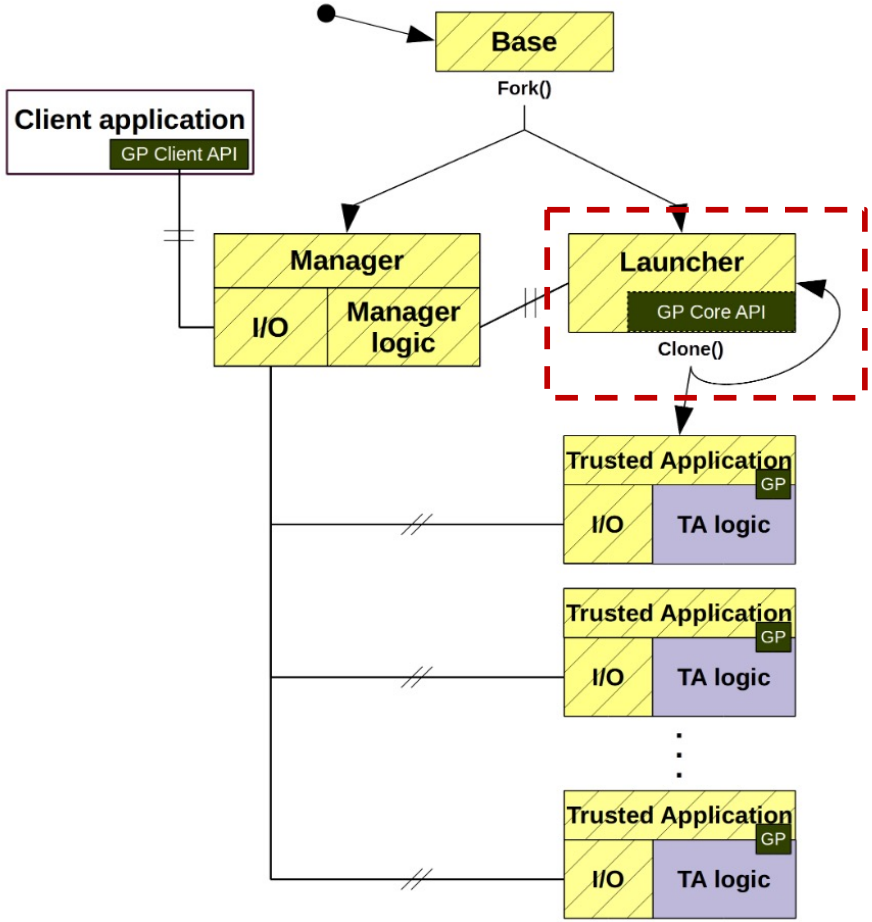
Open-TEE provides an architecture and a software development kit (SDK) that implements GP specification as a framework a top a set of tools that are familiar to the developer, thus removing the need for specialized hardware and the overheads that it incurs.



- The sole purpose of Launcher is to create new TA processes efficiently. When it is first created, Launcher will load a shared library implementing the TEE Core API and will wait for further commands from Manager.
- Manager will signal Launcher when there is a need to launch a new TA.
- Upon receiving the signal, Launcher will clone itself. The clone will then load the shared library corresponding to the requested TA.

::.. Open-TEE (3/3)

Open-TEE provides an architecture and a software development kit (SDK) that implements GP specification as a framework a top a set of tools that are familiar to the developer, thus removing the need for specialized hardware and the overheads that it incurs.



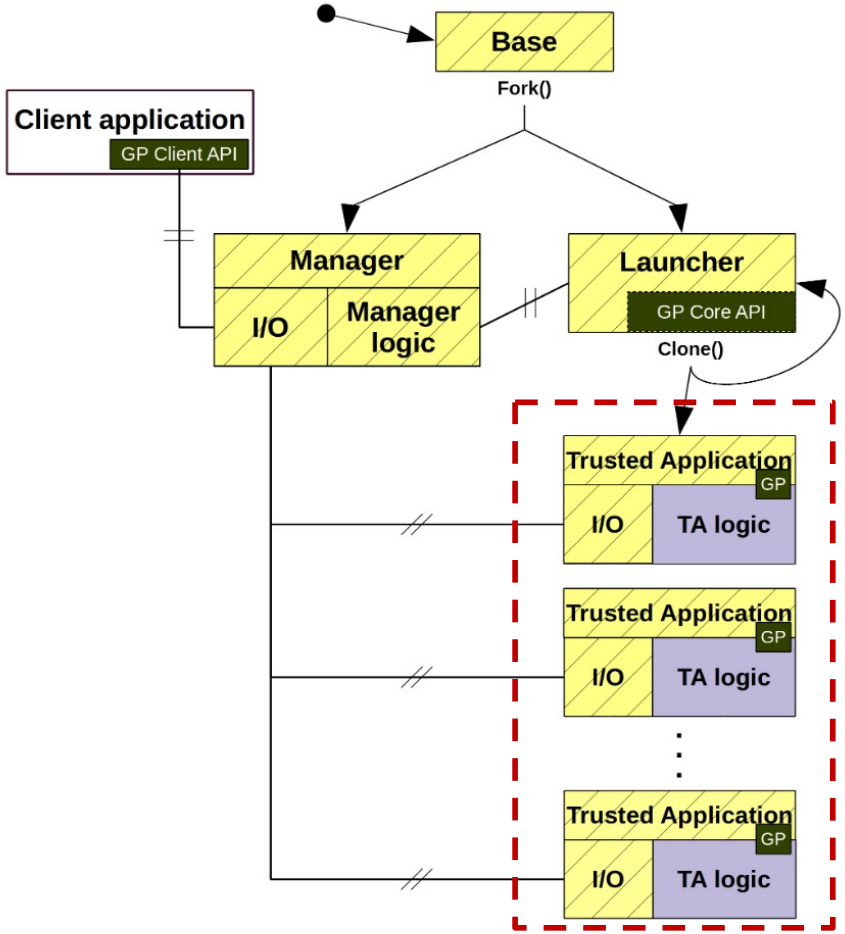
- The design of Launcher follows the "zygote" design pattern.

  Zygote is a special Android OS process that enables shared code across Dalvik/Art VM in contrast with Java VM where each instance has its own copy of core library class files and heap objects.

- A newly created TA process is then re-parented onto Manager so that it is possible for it to control the TA.

::.. Open-TEE (3/3)

Open-TEE provides an architecture and a software development kit (SDK) that implements GP specification as a framework a top a set of tools that are familiar to the developer, thus removing the need for specialized hardware and the overheads that it incurs.

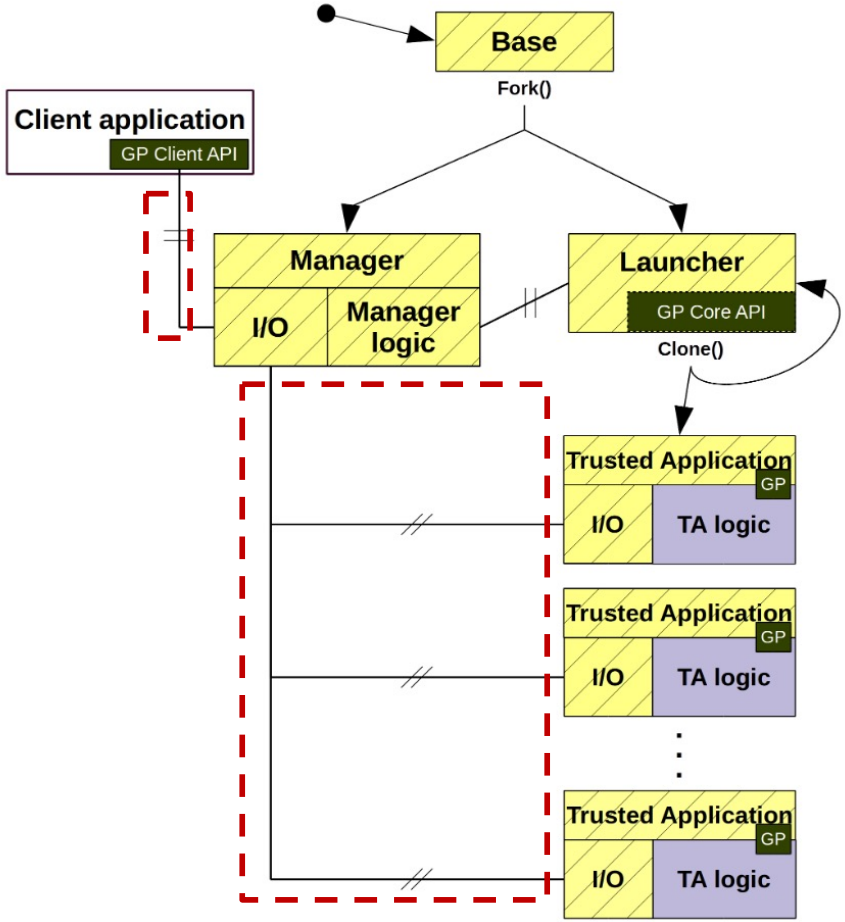


- TA processes have been divided into two threads. The first handles Inter-ProcessCommunication (IPC) and the second is the working thread, referred to respectively as the IO and TA Logic threads.

- This architectural model enables the process to be interrupted without halting it, and allows greater separation and abstraction of the TA functionality from the Open-TEE framework.

::.. Open-TEE (3/3)

Open-TEE provides an architecture and a software development kit (SDK) that implements GP specification as a framework a top a set of tools that are familiar to the developer, thus removing the need for specialized hardware and the overheads that it incurs.



- The TEE Client API and TEE Core API are implemented as shared libraries in order to reduce code and memory consumption.
- Open-TEE implements a communication protocol on top of Unix domain sockets and inter-process signals as the means to both control the system and transfer the messages between the CA and TA.

::.. Samsung Artik (1/8)

SAMSUNG ARTIK™ Modules

The ARTIK platform security, while designed to provide end to end security, leverages strong hardware security capabilities and support through the use a Common Criteria EAL5 Secure Element available on most of ARTIK modules.

The Artick modules provide key and certificate for secure device registration with ARTIK Cloud, which ensures the device is genuine, and messages exchanged are verified. Additionally, they offer support for secure boot, secure JTAG, and full access to the security library, which is based on TrustZone and a Secure Element, and provides secure storage and cryptographic algorithms running in a secure environment without exposure to the user execution environment.

ARTIK Device	PKI	Secure Onboarding	Secure Element	Security API	Secure Boot	KMS	SEE/TEE	SSS	Secure JTAG	PUF
520	X	X	X	X*						
530	X	X	X	X*						
710	X	X	X	X*						
520S	X	X	X	X	X	X	X		X	
710S	X	X	X	X	X	X	X		X	
053	X	X		X	X	X		X		X

::.. Samsung Artik (2/8)

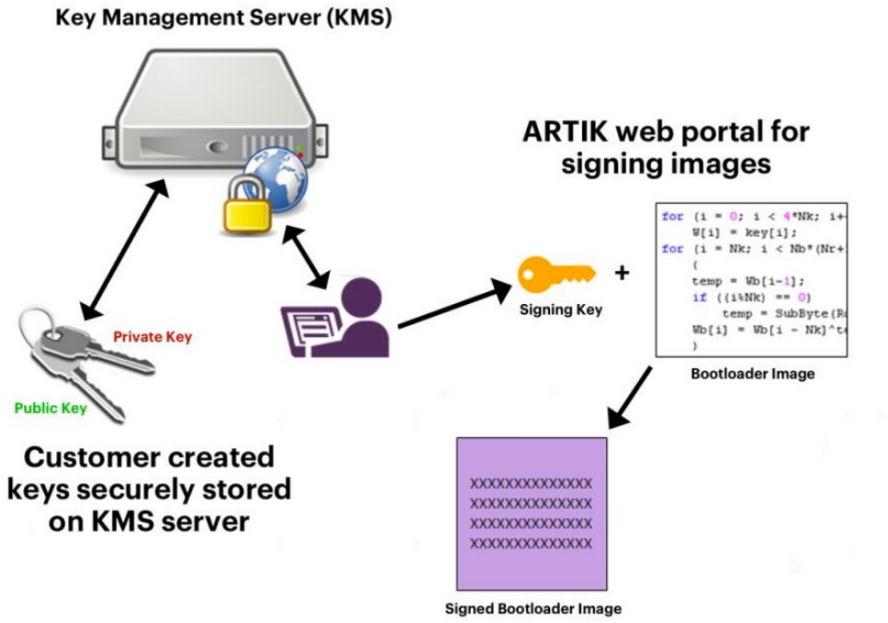
One of the most basic security measures is to provide communication security. Two most important requirements of communication security are Encryption and Authentication.

- ARTIK provides hardware acceleration (Crypto Engine) for AES and RSA encryption and decryption, and uses Elliptic Curve Diffie–Hellman (ECDH) for encryption key sharing.
- Authentication is based on the knowledge of a shared secret between the two parties initiating the link or by using public cryptography based signatures and certificates. ARTIK provides a Public Key Infrastructure (PKI).
- ARTIK Cloud and devices useTransport Layer Security (TLS) and Datagram TLS (DTLS) protocols for a secure communication channels. Over TLS/DTLS, ARTIK Cloud authenticates ARTIK devices using a certificate/key-pair stored on each ARTIK device. The certificate/key-pair is stored in a highly-secure storage, the Secure Element. The certificate/key-pair cannot be altered or tampered with once they have been flashed onto the ARTIK device.

::.. Samsung Artik (3/8)

Secure Boost - To ensure authenticity, software needs to be signed by the software provider who owns it, and has verified the software's execution on the target device. A secure boot process consists of several stages, called bootloader stages, where software is verified and installed at each stage.

A Secure Boot scheme adds cryptographic checks to each stage of the boot process. This process aims to preventany unauthorized software from running by assuring the integrity of all boot images when each boot loader is called by the previous one.

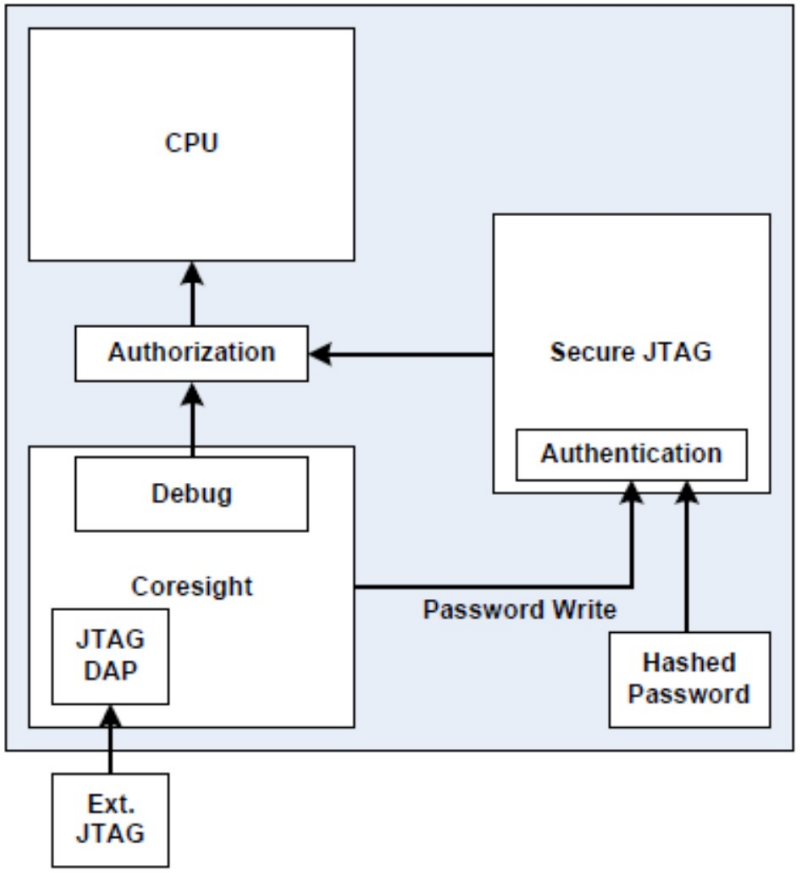


To ensure resilience against mass-compromises, an asymmetric key signature algorithm is used for signing and verification.

Images are signed through a Key Management System (KMS), where keys are stored and operated within certified Hardware Security Modules (HSM) based on strict access control policies.

::.. Samsung Artik (4/8)

The Samsung ARTIK devices provide JTAG fordebugging ofthe platform. However, JTAG is a known vulnerability since access via JTAG opens up methods to bypass internally defined security mechanisms.



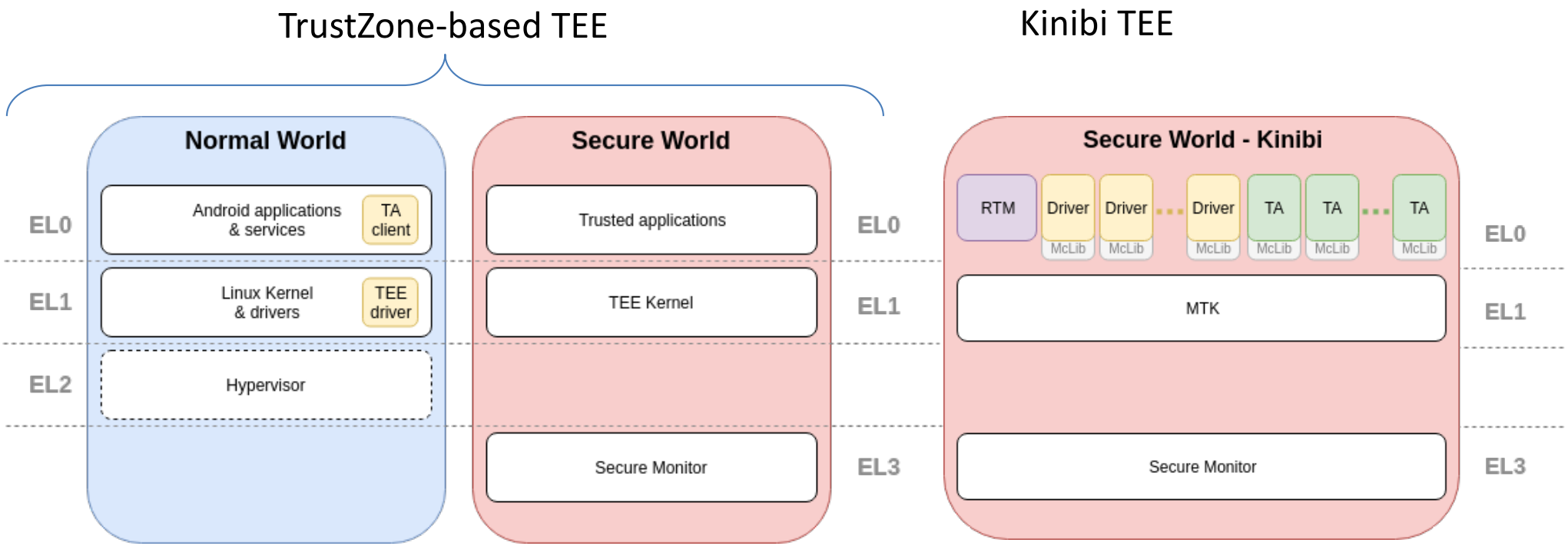
A secure JTAG protects the device from unauthorized access, and requires the use of a password, specific to a particular manufacturing lot of modules, to gain access the JTAG chain. The password is based on the serial number of module, which is provided on special request.

The Samsung ARTIK 5 and 7 family modules support TEE, based on the ARM® TrustZone® hardware architecture.

The required software support for a hardware-based TEE implementation is available within its firmware support.

::.. Samsung Artik (5/8)

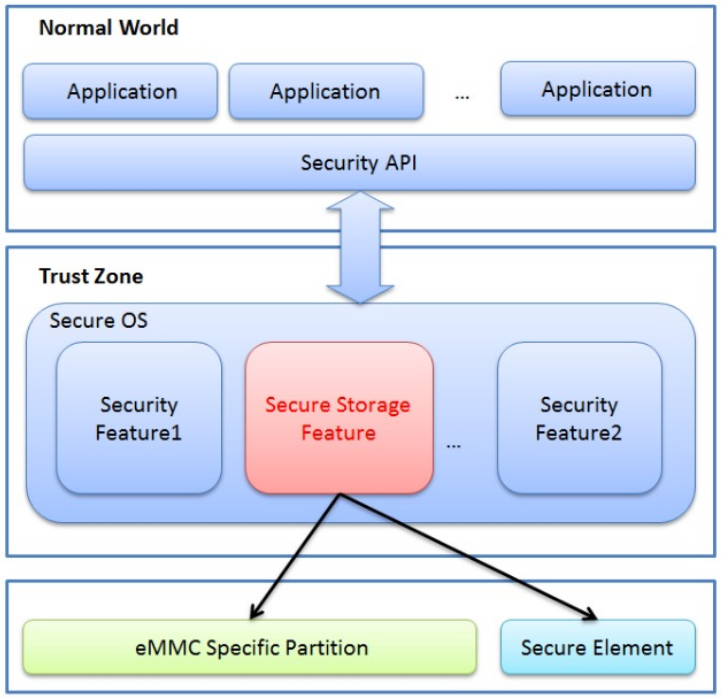
- Samsung Secure Execution Environment (SEE) provides a full TEE implementation of a Secure-OS providing a set of APIs for access and communication from a non-secure operating system based on TrustZone.
- The alternative TEE solution is the one of Trustonic named Kinibi, which is based on a micro-kernel approach.



::.. Samsung Artik (6/8)

A Secure Storage feature allows to securely store data, uses two means:

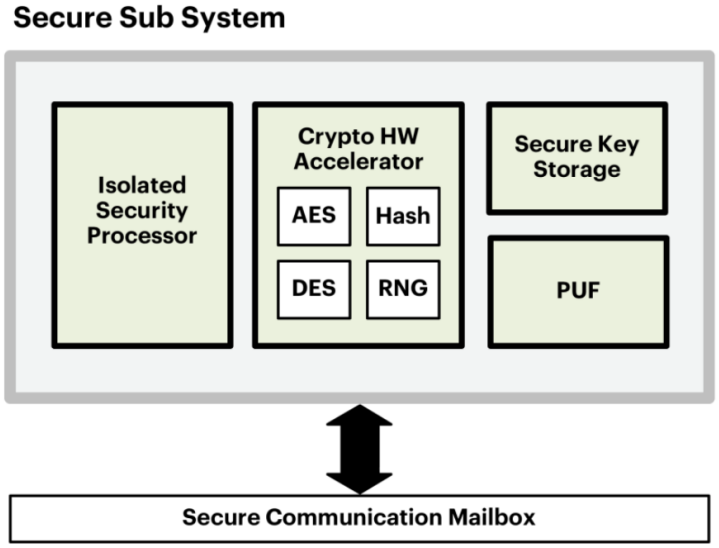
- The eMMC file system uses the same storage as the normal operating system. However, a specific partition is managed by the TrustZone based Secure OS. All data in this partition is encrypted with a unique key generated runtime and is stored as a file unit of 32KB with a maximum number of files that may be stored is 1024.



- The Secure Element is an isolated storage device that supports the storage of up to 16 AES 128 bits keys. The Secure Element provides high levels of security as hardware with anti-temper measures. All communication from the Secure Element to the processor is secured and encrypted.

- A Security Library is used to communicate with the Samsung SEE from the non-secure OS (Linux), and all security-related operations.

::.. Samsung Artik (7/8)

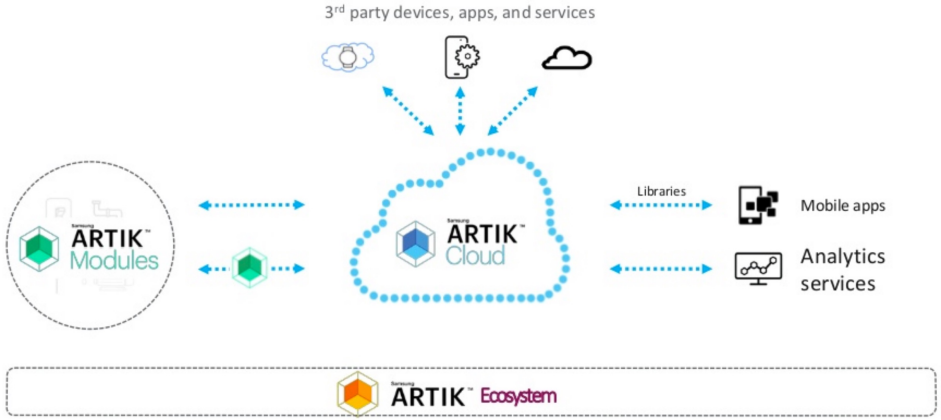


On the board, security related hardware funzions are provided by the security subsystem. This Security SubSystem (SSS) provides secure code execution, secure storage, an isolated security processor, cryptographic hardware acceleration, and a Physically Uncloneable Function(PUF). In addition, it contains unique keys and certificates created during the module manufacturing.

It is also include a hardware-based Secure Element to provide full services for the secure storage of cryptographic material and other protected content.

Samsung ARTIK™

Modules + IoT Cloud + Security + Ecosystem



::.. Samsung Artik (8/8)