GLOBALPLATFORM®
THE STANDARD FOR SECURE DIGITAL SERVICES AND DEVICES

GlobalPlatform Technology
Root of Trust Definitions and Requirements
Version 1.1

Public Release

June 2018

Document Reference: GP_REQ_025

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

Contents

Figures

Tables

Requirements

1 Introduction

This document provides definitions and requirements for the trusted computing concept of Root of Trust in the context of GlobalPlatform. The document then relates these definitions to existing GlobalPlatform technologies, enabling third parties to better understand those technologies in trusted computing terms.

1.1 Audience

This document is intended for use by the Trusted Computing community and others who need to relate Trusted Computing concepts to GlobalPlatform products.

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://www.globalplatform.org/specificationsipdisclaimers.asp. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.3 References

Table 1-1: Normative References

Standard / Specification	Description	Ref
GlobalPlatform Card Specification	GlobalPlatform Card Specification	[GPCS]
GPD_SPE_010	GlobalPlatform Device Technology TEE Internal Core API Specification	[TEE Core]
GPD_SPE_120	GlobalPlatform Device Technology TEE Management Framework Specification	[TMF]
GPD_SPE_021	GlobalPlatform Device Committee TEE Protection Profile	[TEE PP]
IETF RFC 2119	Key words for use in RFCs to Indicate Requirement Levels	[RFC 2119]
TCG	Trusted Computing Group Glossary, http://www.trustedcomputinggroup.org/developers/glossary, visited 23 February 2016	[TCG]
ISO 7498-2:1989	ISO 7498-2: 1989 Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture, 2 February 1989	[7498-2]
NIST	NIST SP 800-164 Guidelines on Hardware-Rooted Security in Mobile Devices (Draft), National Institute of Standards and Technology, October 2012	[NIST]

Table 1-2: Informative References

Standard / Specification	Description	Ref
GPCS Amendment B	GlobalPlatform Card, Remote Application Management over HTTP, Card Specification v 2.2 – Amendment B	[Amd B]
GPD_SPE_009	GlobalPlatform Device Technology TEE System Architecture	[TEE Sys Arch]
GPD_SPE_007	GlobalPlatform Device Technology TEE Client API Specification	[TEE Client]
GPD_SPE_100	GlobalPlatform Device Technology TEE Sockets API Specification	[TEE Sockets]
GlobalPlatform KMS	GlobalPlatform Key Management System Functional Requirements v1.0	[KMS Rqmts]
Andreas M. Antonopoulos	Andreas M. Antonopoulos, "Cloud security: Root of Trust", Network World, http://www.networkworld.com/columnists/2010/020210-antonopoulos.html, Feb 2, 2010, visited 23 February 2016	[Andreas M. Antonopoulos]
Chris Mitchell	Chris Mitchell, *Trusted Computing*, IEE Professional Applications of Computing Series 6, The Institution of Electrical Engineers, London, United Kingdom, 2005	[Chris Mitchell]
Craig Rawlings	Craig Rawlings, Certicom, "Securing SoC Platform Oriented Architectures with a hardware Root of Trust", http://www.embedded.com/design/mcus-processors-and-socs/4008315/Securing-SoC-Platform-Oriented-Architectures-with-a-hardware-Root-of-Trust, Jul 6, 2009, visited 23 February 2016	[Craig Rawlings]
Graeme Proudler	Graeme Proudler, HP, "What's in a Trusted Computing Platform?", http://www.informit.com/articles/article.aspx?p=28804&seqNum=4, Aug 23, 2002, visited 23 February 2016	[Graeme Proudler]
Kill jeep	Andy Greenburg, "Hackers Remotely Kill a Jeep on the Highway – With Me in It" http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/	[Jeep killed]
MMS attack	Thomas Fox-Brewster, "Stagefright: It Only Takes One Text To Hack 950 Million Android Phones" http://www.forbes.com/sites/thomasbrewster/2015/07/27/android-text-attacks	[MMS Attack]
Patrick Schaumont	Patrick Schaumont, Virginia Tech University, "Tree of Trust", http://www.ece.vt.edu/news/ar07/treeoftrust.html, visited 23 February 2016	[Patrick Schaumont]

Standard / Specification	Description	Ref
CCMB-2012-09-001_002_003	Common Criteria for Information Technology Security Evaluation: 1. Introduction and general model, September 2012, version 3.1 Revision 4 2. Security functional components, September 2012, version 3.1 Revision 4 3. Security assurance components, September 2012, version 3.1 Revision 4	[CCMB-2012-09-001_002_003]
FIPS 140-2	Security Requirements for Cryptographic Modules	[FIPS 140-2]
CPA	Commercial Product Assurance; information available at: https://www.ncsc.gov.uk/	[CPA]
CSPN CER I 02	Critères pour l'évaluation en vue d'une certification de sécurité de premier niveau v1.1	[CSPN CER I 02]
OMTP ATE TR1	Open Mobile Terminal Platform Advanced Trusted Environment TR1 v1.1	[OMTP ATE TR1]
Security IC Platform	Security IC Platform Protection Profile with Augmentation Packages from Eurosmart	[Security IC Platform]
UML	Unified Modeling Language; information available at: http://www.uml.org/	[UML]
TCG Whitepaper	TPM MOBILE with Trusted Execution Environment for Comprehensive Mobile Device Security; available here	[TCG Whitepaper]
BSI-CC-PP-0084	Common Criteria Protection Profile Security IC Platform Protection Profile with Augmentation Packages	[PP-0084]

1.4 Terminology and Definitions

The following meanings apply to SHALL, SHALL NOT, MUST, MUST NOT, SHOULD, SHOULD NOT, and MAY in this document (refer to [RFC 2119]):

- **SHALL** indicates an absolute requirement, as does **MUST**.
- **SHALL NOT** indicates an absolute prohibition, as does **MUST NOT**.
- **SHOULD** and **SHOULD NOT** indicate recommendations.
- **MAY** indicates an option.

Selected terms used in this document are defined in Table 1-3.

Table 1-3: Terminology and Definitions

Term	Definition
Actor	Unique entity participating in a GlobalPlatform compliant environment. These entities can take the form of issuers, application developers, production centers, service providers, provisioning systems, etc.
Bootstrapped Root of Trust	A Root of Trust whose implementation required several components. It is composed of one Initial Root of Trust Component and one or more Extended Root of Trust Components.
Chain of Trust	A transitive trust relationship starting from a Root of Trust that is propagated to the Validated/Measured Modules when a software module verifies/measures the next software module and keeps a reportable record of this verification.
Cold Boot	When power is cycled “off” and then “on” on a specific platform and subsequent execution does not utilize data in volatile storage from a previous execution.
Computing Engine	In the context of this document, any system of hardware that will consistently change in a deterministic manner from one state to another. That deterministic manner is governed by the computing engine’s current state, instruction code, and/or data. A platform has one computing engine. This definition does not limit Computing Engines to CPUs, although a Computing Engine must have a minimum of circuitry to perform specific tasks. Note: Hardware entropy sources are acted upon as an aspect of data, and so while they themselves are non-deterministic, the resultant value at a given point in time is acted upon by the computing engine in a deterministic manner.
Credential	A key and/or password and/or any other data used in an authentication protocol.
Critical Security Parameters	In the context of this document, security-related information (e.g. secret and private cryptographic keys, and authentication data such as passwords and PINs) whose disclosure or modification can compromise the Root of Trust Security Services.
Device	In the context of this document, an end-user product that includes at least one platform.

Term	Definition
Executable Code	In the context of this document, a set of instructions in native or interpreted language. These instructions are executed directly by the hardware composing the computing engine itself, or indirectly if they are instead interpreted by another executable code block.
eXecute in Place	Code that is designed such that it may be run directly from ROM or FLASH (or other non-volatile storage) without loading into RAM before execution.
Extended Root of Trust Component (eRoTc)	A block of executable code whose integrity is verified by either an Initial Root of Trust Component or another Extended Root of Trust Component at any point during the life cycle of the platform (see section 3.3).
Firmware	Software dedicated to interacting with device specific hardware.
Hardware Security Module (HSM)	A physical computing device considered as tamper resistant that safeguards and manages digital keys (e.g. for strong authentication, encryption, and authorization), and that provides cryptoprocessing to external entities.
Initial Root of Trust Component (iRoTc)	A block of code that a platform manufacturer provisions and initializes during the manufacturing process and that is the first code executed on the platform (see section 3.3).
Measured Module	A module measured by the RoT service for measurement or by any component in a Chain of Trust for Measurement.
Measurement	In the context of this document, an operation that produces a platform characteristic.
Message Authentication Code	A symmetric cryptographic transformation of data that provides data origin authentication and data integrity.
Module	A block of code and/or data and/or keys.
Next Executable Code	In the context of this document, the executable code whose execution is initiated by the current executable code block. Might or might not be processed by the same computing engine as the current executable code block.
Non-Bootstrapped Root of Trust	A Root of Trust that is composed of only one block of code.
Platform	In the context of this document, one computing engine and executable code that provides a set of functionalities. SE, TEE, and REE are examples of platforms.
Platform Characteristic	A value that can be used to confirm the security condition of a platform; for example, a cryptographic hash, code verification assertion, or hardware state assertion.
Policy	A set of rules that defines access to services or assets.
Primary Root of Trust (pRoT)	A RoT executed on a platform.
Proof of possession	A means of proving that a party sending a message is in possession of a particular cryptographic key.

Term	Definition
Root of Trust (RoT)	A computing engine, code, and possibly data, all co-located on the same platform; provides security services (as discussed in Chapter 3). No ancestor entity is able to provide a trustable attestation (in Digest or other form) for the initial code and data state of the Root of Trust. Depending on the implementation, the Root of Trust is either a Bootstrapped or a Non-Bootstrapped Root of Trust.
Root of Trust Component	A block of code, and possibly data, that could be either an Initial Root of Trust Component or an Extended Root of Trust Component. If a platform includes multiple Root of Trust Components, all of them together build a Bootstrapped Root of Trust.
Root of Trust Security Service	In the context of this document, a service provided by a Root of Trust, such as authentication, authorization, confidentiality, identification, integrity, measurement, reporting, update, or verification.
Secondary Root of Trust (sRoT)	A RoT providing RoT Security Services used by another platform.
Secure Component	GlobalPlatform terminology to represent either a Secure Element or a Trusted Execution Environment.
Secure Element (SE)	A tamper-resistant secure hardware component which is used in a device to provide the security, confidentiality, and multiple application environment required to support various business models. May exist in any form factor, such as embedded or integrated SE, SIM/UICC, smart card, smart microSD, etc.
Shielded Location	In the context of this document, a location specifically designed and used for the purpose of protecting the integrity and optionally the confidentiality of data and that permits only the data owner to access the data. Note: Usually a ROM provides integrity but does not protect the confidentiality of its content.
Tamper-resistant secure hardware	Hardware designed to isolate and protect embedded software and data by implementing appropriate security measures. The hardware and embedded software meet the requirements of the latest Security IC Platform Protection Profile ([PP-0084]) including resistance to physical tampering scenarios described in that Protection Profile.
TEE Run Time Environment (RTE)	In the context of this document, includes all the functionality a TEE may offer after the boot phase. This includes (but is not limited to): • A selection of the TEE APIs defined by GlobalPlatform (as described in TEE System Architecture [TEE Sys Arch]) • Additional APIs defined by the Trusted OS vendor • Additional APIs defined by the Silicon Provider / OEM / Original Design Manufacturer
Token	Data derived from a credential.

Term	Definition
Trusted Execution Environment (TEE)	An execution environment that runs alongside but isolated from an REE. A TEE has security capabilities and meets certain security-related requirements: It protects TEE assets from general software attacks, defines rigid safeguards as to data and functions that a program can access, and resists a set of defined threats. There are multiple technologies that can be used to implement a TEE, and the level of security achieved varies accordingly. For more details of the TEE architecture and system requirements, see GlobalPlatform TEE System Architecture [TEE Sys Arch]. TEE security requirements are defined by the GlobalPlatform TEE Protection Profile [TEE PP].
Trusted OS	A specific part of the TEE RTE that provides GlobalPlatform compliant interfaces.
Trusted Service Manager	An entity trusted by other actors to be in charge of service management and delivery.
Validated Module	Module verified by the RoT service for verification or by any component in a Chain of Trust for Verification.

1.5 Abbreviations and Notations

Selected abbreviations and notations used in this document are defined in Table 1-4.

Table 1-4: Abbreviations and Notations

Abbreviation / Notation	Meaning
APDU	Application Protocol Data Unit
API	Application Programming Interface
BTEE	Boot time Trusted Execution Environment
CASD	Controlling Authority Security Domain
CPA	Commercial Product Assurance
CRC	Cyclic Redundancy Check
CSPN	Certification de sécurité de premier niveau
CVM	Cardholder Verification Method
DEK	Data Encryption Key
eRoTc	Extended Root of Trust Component
eSE	embedded Secure Element
FIPS	Federal Information Processing Standard
HSM	Hardware Security Module
HUK	Hardware Unique Key
iRoTc	Initial Root of Trust Component
ISD	Issuer Security Domain
ISO	International Organization for Standardization
JCRE	Java Card Runtime Environment
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
ODM	Original Design Manufacturer
OEM	Original Equipment Manufacturer
OMTP	Open Mobile Terminal Platform
OPEN	Open Platform Environment
OTA	Over The Air
OTP	One-Time Programmable
PCB	Printed Circuit Board
PIN	Personal Identification Number
PKI	Public Key Infrastructure
pRoT	Primary Root of Trust

Abbreviation / Notation	Meaning
REE	Rich Execution Environment
ROM	Read Only Memory
RoT	Root of Trust
rSD	root Security Domain
RTE	Run Time Environment
SCP	Secure Channel Protocol
SD	Security Domain
SE	Secure Element
SoC	System-On-Chip
sRoT	Secondary Root of Trust
SSD	Supplementary Security Domain
TA	Trusted Application
TAM	Trusted Application Manager
TCG	Trusted Computing Group
TCP/IP	Transmission Control Protocol / Internet Protocol
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TMF	TEE Management Framework
TPM	Trusted Platform Module
TSM	Trusted Service Manager
TUI	Trusted User Interface
UDP/IP	User Datagram Protocol / Internet Protocol
UICC	Universal Integrated Circuit Card
XiP	eXecute in Place

1.6 Revision History

GlobalPlatform technical documents numbered *n*.0 are major releases. Those numbered *n*.1, *n*.2, etc., are minor releases where changes typically introduce supplementary items that do not impact backward compatibility or interoperability of the specifications. Those numbered *n.n*.1, *n.n*.2, etc., are maintenance releases that incorporate errata and precisions; all non-trivial changes are indicated, often with revision marks.

Table 1-5: Revision History

Date	Version	Description
October 2016	1.0	Public Release
March 2017	1.0.1	Maintenance Release, including the following changes: The term “Extended Root of Trust” has been modified to “Enhanced Root of Trust”. Figure 5-1 has been modified.
June 2018	1.1	Alignment with TCG The terms “Extended Root of Trust” from v1.0 and “Enhanced Root of Trust” from v1.0.1 (both abbreviated “eRoT”) have been eliminated. Instead, this version discusses Initial Root of Trust Component (iRoTc) and Extended Root of Trust Component (eRoTc).

2 Overview

Our lives are increasingly dependent on smart connected mobile devices. We use them to conduct business, maintain social relationships, make purchases, and enjoy media content. Despite the benefits, these devices hold significant security assets and personal data that are susceptible to attacks from hackers. Furthermore, the sheer number of applications that are easily available for download represents an even larger opportunity for fraudsters.

Similarly, automotive and home devices are becoming increasingly connected and providing more capabilities than their original core functionality.[1] Consumers are increasingly using their devices in new ways—organizing a trip from one's TV, streaming music while driving, and using a smartphone to pay for petrol. These expanded practices create new security vulnerabilities, which in turn emphasize the need for mechanisms that allow trusted parties to have access to applications without granting hackers, rogue applications, etc., the same opportunity.

Two examples of increased complexity creating security holes:

- A hacker was able to kill a jeep on the highway (see [Jeep killed]).
- A hacker used an MMS to install spyware on a mobile device (see [MMS Attack]).

The increased security needs are ultimately driven by new characteristics of consumers' smart connected devices. In order to prevent the theft of personal data, hijacking of a connected device, and other security hazards, security solutions need to authenticate the user, securely store personal data, validate transactions, and so on.

The complexity of final solutions increases with the multiple actors that are used to provide a customer service. A service provider that would like to deploy a service on mobile devices needs to rely on mobile characteristics that it cannot manage. The maker of the mobile device, who is responsible for the security architecture, should prevent security weaknesses. Within the mobile ecosystem, the handset maker might not be able to manage all items. For example, a part of the OS might be provided by a third company. If the service provider needs to connect to its service, it must rely on a connectivity service provider (e.g. Mobile Network Operator or Internet Service Provider).

All of these factors present security concerns that must be addressed in the market. One way of resolving these issues is to provide a small, isolated execution environment for sensitive assets and code that would allow service providers and OEMs to improve the user experience while increasing the security to prevent fraud, guarantee privacy, and so on.

Many security components claim to be Roots of Trust or to contain Roots of Trust. However, each vendor of these components has distinct and often independent ideas about what constitutes a Root of Trust. This document explores the commonality among these Roots of Trust and defines their requirements and properties. It also introduces a few use cases that rely on Roots of Trust to provide security services to platforms and remote service providers (e.g. Mobile Network Operators).

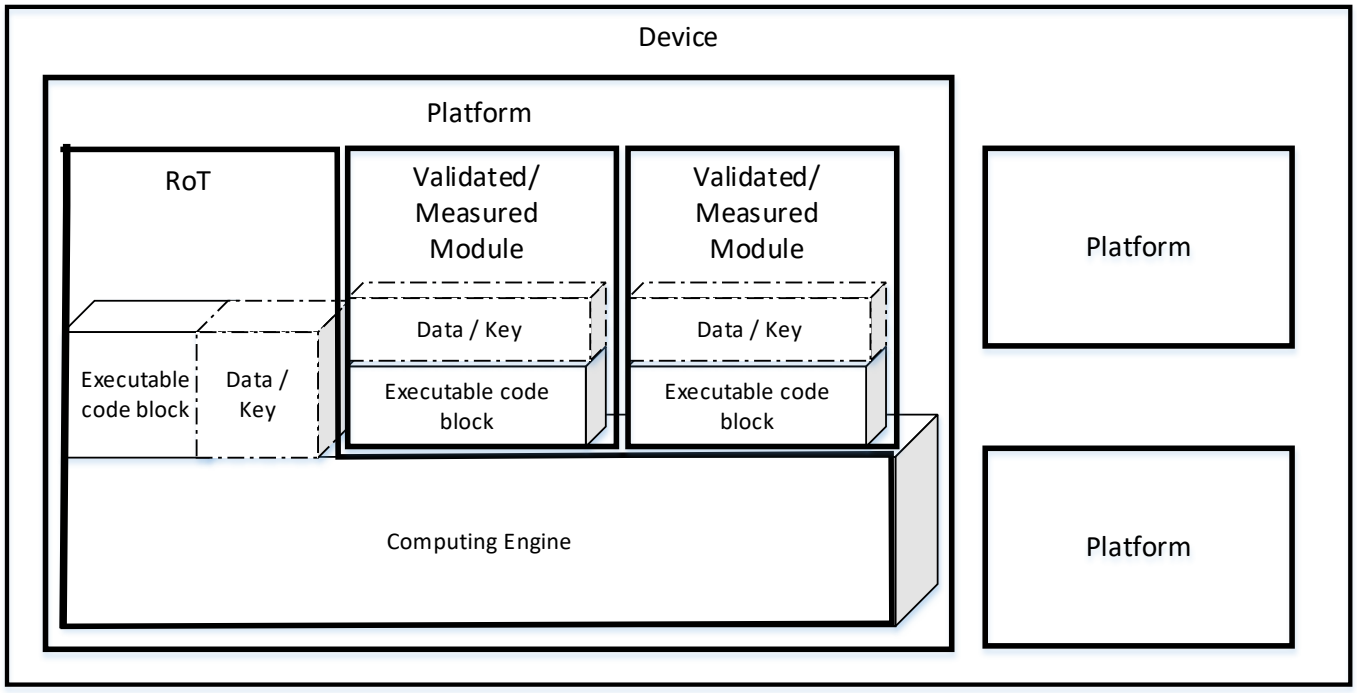
[1] Visit http://www.globalplatform.org/documents/whitepapers/IoT_public_whitepaper_v1.0.pdf.

In the context of this document, Device, Platform, and Shielded Location are used with the following meanings:

- A Device is an end-user product that includes at least one platform; for example, a UICC or a mobile phone without UICC.
- A Platform is composed of one computing engine and executable code that provides a set of functionalities; for example, a Trusted Execution Environment (TEE) or an embedded Secure Element (eSE). When implementing a Chain of Trust, a platform contains only one Root of Trust (RoT). [2]
  - The RoT may be composed of an Initial Root of Trust Component (iRoTc) and one or more Extended Root of Trust Components (eRoTc); the combination is referred to as a Bootstrapped Root of Trust.
  - A RoT that is composed of only one block of code is referred to as a Non-Bootstrapped Root of Trust.
- A Shielded Location is a location specifically designed and used for the purpose of securely storing and protecting sensitive data; for example, a tamper-evident or tamper-resistant memory or register.

Figure 2-1 shows an example architecture overview and the relationship between the different entities of a Non-Bootstrapped RoT.

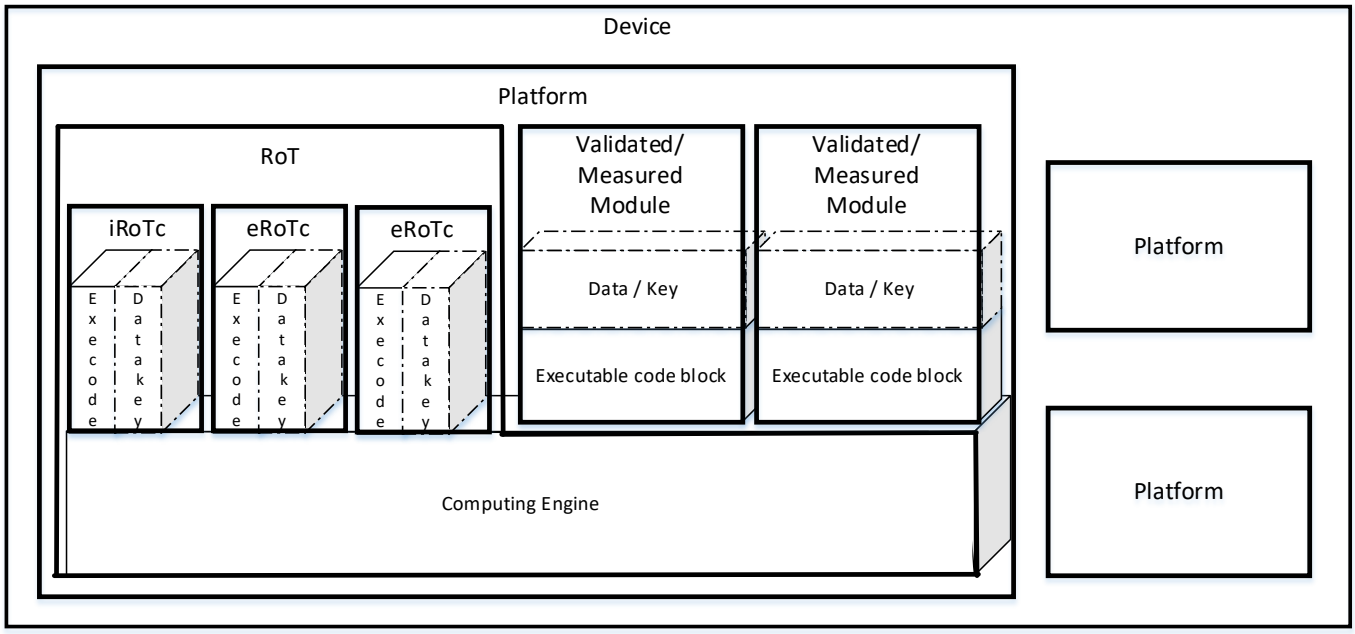
Figure 2-1: Architecture Overview of a Non-Bootstrapped RoT



[2] A platform that does not implement a chain of trust has no RoT.

Figure 2-2 shows an example architecture overview and the relationship between the different entities of a Bootstrapped RoT.

Figure 2-2: Architecture Overview of a Bootstrapped RoT



3 Root of Trust

Many designers of security protocols, schemes, products, and services rely on a root of trust to provide security. However, each designer has its own definition for a root of trust. Upon examining a variety of definitions for Roots of Trust, one begins to see common properties and attributes for them.

The following are descriptive quotes from publications on roots of trust.

> TCG says: *A component that must always behave in the expected manner, because its misbehavior cannot be detected. The complete set of Roots of Trust has at least the minimum set of functions to enable a description of the platform characteristics that affect the trustworthiness of the platform.* [TCG]
>
> Graeme Proudler says: *A root-of-trust is a set of unconditionally trusted functions and must be a computing engine, because it must perform actions. It must work properly no matter what software is executing on the platform, in order to be immune to software attacks. Ideally, it should also be immune to physical attack, to avoid the need to trust an owner or user of a platform (who might otherwise physically meddle with it). Otherwise, a third party cannot unreservedly trust a remote platform. Unfortunately, physical immunity is impossible, and we have to settle for mere physical protection. (It's the same for smart cards and crypto-coprocessors.) For most commercial purposes, it's sufficient to have a root-of-trust that's built to resist a modest level of physical attack.* [Graeme Proudler]
>
> Andreas M. Antonopoulos says: *What that means is that trust flows down a chain until it reaches the root of trust -- the foundation on which I have built my little tower of trust. I trust a door because I trust the lock because I trust the key because it is in my pocket. If I lose the key, the Chain of Trust unravels and I no longer trust the door.* [Andreas M. Antonopoulos]
>
> Patrick Schaumont says: *[…] your root of trust […] is the element that is implicitly trusted, such as the key of an encryption algorithm. Implementing the root of trust is always expensive, so the smaller you can make this, the cheaper your system is. Indeed, the non-trusted elements can use standard components or software.* [Patrick Schaumont]
>
> Craig Rawlings says: *It has long been understood by security experts that a root of trust (ROT) is essential to system security. A ROT is a secret that if kept will not compromise the security of the overall system. A ROT is necessary to provide: (1) a safe place for system-critical secrets, (2) secure processes and (3) extended trust to internal and external entities.* [Craig Rawlings]
>
> Chris Mitchell says: *[…] Roots of Trust are functions that must operate as expected, irrespective of any other process in a platform, because without them there is no way to believe anything reported by a platform.* [Chris Mitchell]

Note: As should be clear from the quotations above, a key is not a Root of Trust (although a Root of Trust may include a key).

When a service is built on a device, several operations need to be executed before the application providing the service is ready to use. The application must trust that operations executed before its execution are authentic and that no malware code has been executed. It is best to verify the executable code with its data and keys and leave a record of its verification that could be verified during application execution.

Each module acting as a RoT must be as small as possible and should be implemented in a straightforward manner to limit its complexity, so that its correctness can be easily assessed and certified. Reducing the RoT to the smallest size and certifying it makes it more difficult to hack and/or load malware inside the Chain of Trust (see Chapter 5).

Vendors will make various choices in designing and implementing Roots of Trust. When a block of code is the first code to run on the platform and this block of code is able to verify but not to record the execution of the next code block, the Root of Trust is composed of several executable blocks. In some cases, an initial block of code starts executing but lacks the resources to record platform characteristics. In particular, if the initial block of code must pass execution control on to an independent block of code, and it verifies the integrity and authenticity of the new code without recording that event before passing control, then we say we are expanding the Root of Trust from the first block to the next. As mentioned in Chapter 2, we call the first block the Initial Root of Trust Component or iRoTc. We call the subsequent block, or each subsequent block if there is more than one, an Extended Root of Trust Component or eRoTc. The platform is executing several blocks of executable code in a bootstrapping sequence to create a Root of Trust; this collection of iRoTc and eRoTc is called a Bootstrapped Root of Trust.

The following sections list the requirements that characterize a Root of Trust, clarifying expectations and functional behavior that the security industry needs of Roots of Trust.

3.1 Requirements for a Root of Trust

The following requirements apply to every Root of Trust (RoT) regardless of the security services it provides.

Requirement 3-1: Computing Engine, Code, and Data

A Root of Trust SHALL consist of a computing engine and executable code (providing the root of trust security function(s)), co-located on the same platform.

A Root of Trust MAY require data and/or key(s); if so, the data and/or key(s) SHALL be co-located on the same platform as the computing engine and executable code.

***Note**: Italicized paragraphs in the remainder of this chapter are informative only.*

The computing engine could be of any architecture, and the code is not limited to binary instructions. The data may be an optional component for some Roots of Trust, but necessary for others. The data may include an integrity measurement (such as a CRC or a cryptographic hash), a public key, a private key, a secret key, or other security assets.

Implementation note: A Root of Trust may have several types of implementations. An alternative to the computing engine, executable code, and data (if needed) co-located on the same platform, is hardware logic that performs the equivalent Root of Trust Security Service. Examples include Field Programmable Gate Arrays (FPGAs) and special cryptographic accelerators. The same requirements apply to FPGAs and cryptographic accelerators that apply to other Roots of Trust.

Requirement 3-2: Security Services

A Root of Trust SHALL provide one or more security services.

Any executable module in a Chain of Trust for the security services offered may rely on the Root of Trust for the security services it offers.

It is not necessary that the security services provided by the Root of Trust be unique to the Device or platform it supports. Indeed, a Device may have multiple Roots of Trust providing the same service. Nonetheless, a Root of Trust must provide a security service for the benefit of the platform. Typical security services include, but are not limited to, authentication, authorization, confidentiality, identification, integrity, measurement, reporting, update, and verification.

Requirement 3-3: Certification

The Vendor/Manufacturer SHALL design a Root of Trust for a certification process for a platform or for a device.

In other words, an independent party, such as a testing lab, is able to assess the computing engine, code, and data, and to verify that it performs the advertised service and satisfies the security claims of its security target.

Following a successful assessment, an authority may issue a certificate proclaiming that the Root of Trust meets a published set of security requirements.

Notable certification processes include: Common Criteria [CCMB-2012-09-001_002_003], recognized by multiple countries; NIST FIPS 140-2 [FIPS 140-2]; and national recognition schemes such as CPA [CPA] in the UK and CSPN [CSPN CER I 02] in France.

GlobalPlatform is launching a TEE Security Evaluation Secretariat to provide certification based on the GlobalPlatform TEE Protection Profile [TEE PP].

A platform may have only one RoT (either bootstrapped or not). If the RoT is bootstrapped, then only the entire RoT can be certified; Root of Trust Components of a Bootstrapped RoT cannot be certified individually.

A device may have two or more platforms and each can be certified.

Requirement 3-4: Unique Identifiable Ownership

A Root of Trust SHALL have a single identifiable owning entity.

Authorized Users, including end users, administrators, service providers, and content providers, must be able to attribute the security services provided by the Root of Trust to the owning entity.

This entity accepts responsibility for the Root of Trust, and if it can be updated, also for its update.

The owner of the Root of Trust is not necessarily the owner of the platform containing the Root of Trust.

Ownership may be asserted through a form of ID retrieval (which may be anonymity preserving) and/or through the provision of provably correct key relationships.

Requirement 3-5: Mutability

Code and/or data of a Root of Trust SHALL be immutable or its mutability SHALL be controlled only by the unique identifiable owner.

Requirement 3-6: Ownership Transfer

If a Root of Trust implements an ownership transfer mechanism designed by the initial owner/provider of the RoT, then the current owner of the Root of Trust SHALL provide a mechanism to authorize the transfer of ownership to the new owner.

Requirement 3-7: One RoT per Platform

A Platform SHALL contain one and only one Root of Trust.

A platform has one computing engine, which may have only one Root of Trust. However, the Root of Trust may provide several security services.

Requirement 3-8: Temporal

The Root of Trust SHALL include the code which executes first upon the initialization of the computing engine during cold boot in that platform.

Requirement 3-9: Manufacturer Identity

A Root of Trust SHALL have an identifiable manufacturer.

There are many options for the creation and provisioning of the Root of Trust, depending on the lifecycle of the components used in constructing a platform. These include, but are not limited to, creation by SoC vendor, creation by TEE vendor, and creation by device manufacturer.

3.2 Requirements for a Non-Bootstrapped Root of Trust

Requirement 3-10: Provenance

The platform manufacturer SHALL create and provision the Root of Trust during the manufacturing process.

3.3 Requirements for a Bootstrapped Root of Trust

In some device design scenarios, the platform manufacturer supplies some components of the Root of Trust, and the device manufacturer supplies some components of the Root of Trust. To build a Root of Trust from smaller blocks of code, the initial block of code passes execution control to an independent block of code. Before it passes execution, the initial block of code verifies the integrity and authenticity of the new code. The initial block of code is called the parent and the independent block of code will be the child. If the child later activates the next independent block of code, it becomes a parent and the next independent block code becomes a child. When the parent verifies the integrity and authenticity of the child without recording that event before passing control, then we say we are bootstrapping the Root of Trust from the initial blocks to the independent blocks. When a block of code measures or verifies the integrity of the next block of executable code and records the result in a shielded location accessible to subsequent blocks of code before passing control, then the Root of Trust bootstrapping has completed and execution has entered into a Chain of Trust (see Chapter 5).

We introduce the concepts of Initial Root of Trust Components and Extended Root of Trust Components to bootstrap blocks of code together to create a Root of Trust. In this scenario, the parent Root of Trust Components may be unable to access some shielded locations to preserve the reportable verification of subsequent blocks of executable code, and consequently cannot leave a record of the verification of the code (and associated data) before it passes execution control to the next block of code.

- The Initial Root of Trust Component and the Extended Root of Trust Components may come from different vendors; bootstrapping provides a way to build the Root of Trust with components from different vendors.
- Extended Root of Trust Components give vendors an opportunity to provide additional features or replace existing security services with more efficient ones.

An Extended Root of Trust Component is a block of code and data whose integrity a parent block of code can verify prior to its execution without recording the result of the verification in a shielded location accessible to subsequent blocks of code.

When a parent transfers execution control to the next block of code outside of itself while not preserving reportable verification, that new code and data are an Extended Root of Trust Component. Note that the transfer of execution control to an Extended Root of Trust Component may remove the availability of the services of its parent Root of Trust Component.

The document relies on Requirement 3-3: Certification to mitigate concerns about a large RoT resulting from the bootstrapping of independent blocks of code, which seemingly violates an implicit requirement that each RoT should be as small as possible. Using a reputable third party to assess the RoT against a published set of security requirements provides some assurance that the RoT will behave as expected.

If the Root of Trust does preserve reportable verification, then the next block of code is a non-Root-of-Trust component and a link in a Chain of Trust.

The following requirements apply to Initial and Extended Root of Trust Components regardless of the security services they provide.

Requirement 3-11: Provenance (iRoTc)

The platform manufacturer SHALL create and provision the Initial Root of Trust Component during the manufacturing process.

In some implementations, the platform maker provides the iRoTc but the next executable code acting as eRoTc may be provided by another provider (e.g. the device provider) and cannot be loaded during the manufacturing process of the iRoTc.

Requirement 3-12: Temporal (iRoTc)

The Initial Root of Trust Component SHALL include the code which executes first upon the initialization of the computing engine during cold boot in that platform.

An Initial Root of Trust Component is a root of trust whose code and data integrity cannot be measured and/or verified immediately prior to its execution and consumption.

Requirement 3-13: Verification (eRoTc)

A parent Root of Trust Component (i.e. either an Initial RoT Component or another eRoTc) SHALL verify the integrity of the code and data of an Extended Root of Trust Component before the first execution of the eRoTc. A parent Root of Trust Component does not preserve the reportable verification of the code and data of an eRoTc.

The following list of examples is non-exhaustive:

- *An owner of the parent Root of Trust Component may authorize the instantiation of an eRoTc at any point during the life cycle of the platform (including manufacturing process).*
- *A parent Root of Trust Component may be either an iRoTc or an eRoTc.*
- *A parent Root of Trust Component does not keep a record such as a hash of the code and data in a shielded location before the eRoTc's execution. Following the eRoTc's execution, it will not be possible to access a record of the verification of the eRoTc.*
- *Many embedded devices use this method to instantiate higher level functionality. Lower level components perform security checks of higher level components, but the space dedicated to the higher level component's code and data does not leave enough storage for the results.*
- *A Boot ROM code acting as iRoTc that verifies the next code that is a Trusted OS is not able to keep a reportable verification of the Trusted OS. In this example, the Trusted OS acts as an eRoTc.*

Requirement 3-14: Location (Bootstrapped RoT)

The iRoTc and all the eRoTc(s) that compose a Bootstrap Root of Trust SHALL be located on the same platform.

3.4 Primary and Secondary Root of Trust

The term Primary Root of Trust (pRoT) usually refers to the first platform-based RoT instantiated on a device. However, a device could have more than one computing engine, and thus more than one RoT. Primary Root of Trust may also refer to the platform-based RoT instantiated on the main central processing unit (CPU) (sometimes called an application programming unit or APU) on a device. In either case, each other computing engine on the device that satisfies RoT requirements is called a Secondary RoT (sRoT). The notion of Primary and Secondary may become relative, especially when either RoT is capable of providing unique security services to the other.

In the context of this document, the pRoT or the platform may benefit from the RoT Security Services provided by another device-based RoT; i.e. an sRoT.

4 Root of Trust Security Services

This chapter provides definitions and requirements for the Root of Trust Security Services relevant to GlobalPlatform. A Root of Trust may contain any combination of these services.

Recommendation for designers:

When calling attention to a Root of Trust, designers are encouraged to refer to the service or services provided by the RoT in order to set the proper context and expectations. For example, say “The Root of Trust for Verification is provided by this ROM code and data” or “That firmware provides a Root of Trust for Update”.

With the proper context, software designers and evaluators can more readily assess whether or not a Root of Trust provides the security services relevant to their security needs.

Several standards organizations define sets of security services. For example:

- ISO 7498-2 standardizes and manages the following security services:
  authentication, access control, confidentiality, integrity, and non-repudiation (see [7498-2]).
- The Trusted Computing Group standardizes and manages the following security services as Roots of Trust: measurement, reporting, and storage (see [TCG]).
- NIST standardizes and manages the following security services as Roots of Trust:
  integrity, measurement, reporting, storage, update, and verification (see [NIST]).

Based on its experience with the devices and markets that GlobalPlatform addresses through its members, GlobalPlatform has identified nine services that could be provided by Roots of Trust: authentication, confidentiality, identification, integrity, measurement, authorization, reporting, update, and verification.

One may construct a Root of Trust with one or more security services. However, the Root of Trust cannot claim to offer the service unless it exposes the interface to the service in some way. For example, consider a Root of Trust implementing several Security Services where the Verification Service can maintain the shielded location for the public key and be able to verify the integrity of the next component in the Chain of Trust (see Chapter 5) without exposing those protected capabilities offered by the Integrity Service and Measurement Service. In this case, the RoT is only a RoT for Verification and cannot claim to be a RoT for Confidentiality or Measurement since those services are tailored for and reserved exclusively for use by the RoT for Verification.

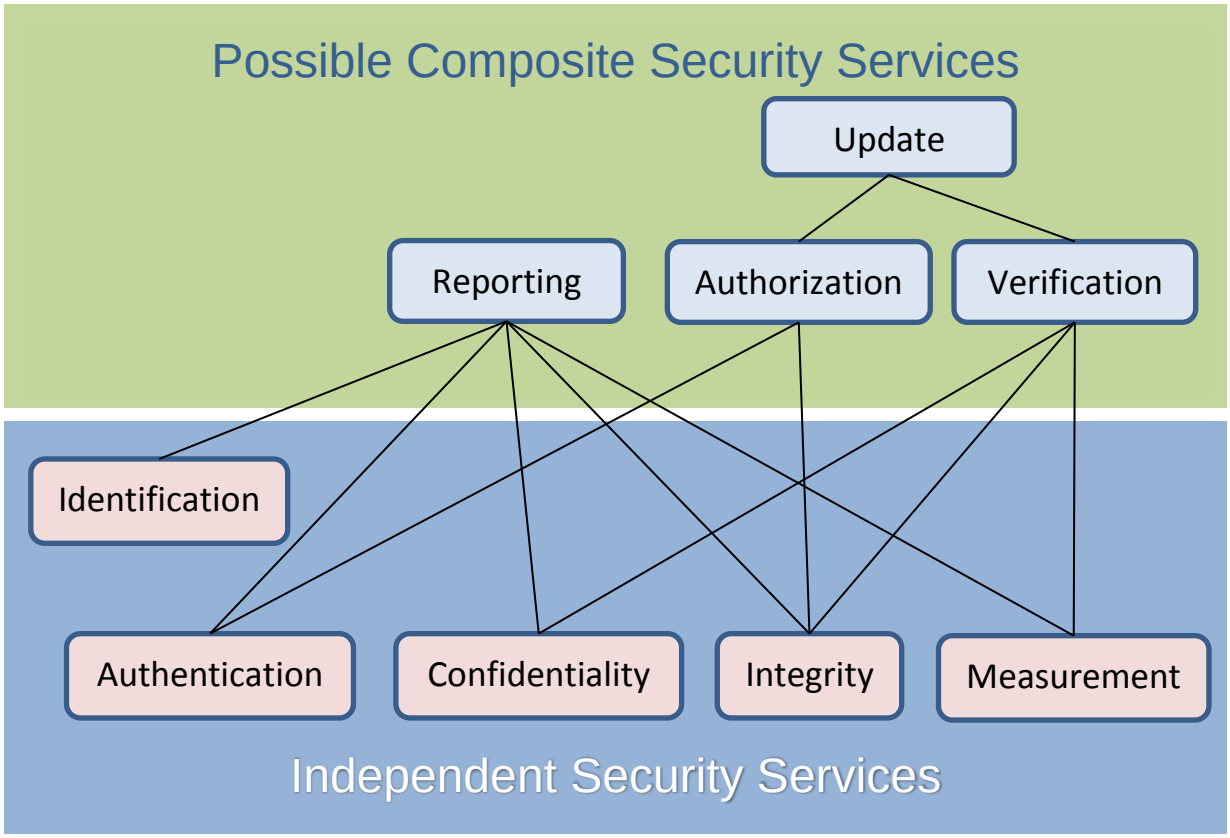
A Root of Trust may maintain one or more shielded locations used for the purpose of securely storing and protecting sensitive data. A shielded location should be accompanied by an interface which provides the appropriate level of protection to limit access to sensitive information and avoid unauthorized disclosure, use, and access (e.g. in an implementation, the sensitive data stored in a shielded location may only be accessed by an API after the API checks whether access to this sensitive data is authorized). The level of physical and logical protection of the shielded location may be dictated by certification policies, or by a vendor’s desire to offer value-added features for their products.

If the platform implements a Chain of Trust, the initial boot code executed during the cold boot of a platform shall contain RoT Security Services. The initial boot code may use these services to perform security-related tasks, such as verifying the integrity of other components before handing control to them or recording platform characteristics for future reference and reporting. As the device bootstraps its way through initialization, each component in the sequence may instantiate additional RoT Security Services and use them to perform security-related tasks, such as authorizing access to protected resources, updating critical device firmware, or reporting platform characteristics. Other security services can use RoT Security Services to provide protection of keys, data stored on the device, and data in transit to and from back-end services (e.g. Trusted Service Manager). Application developers can use RoT Security Services either directly or indirectly through the use of those other security services. Service providers or content providers of the device and recipients of the platform characteristics (e.g. firmware measurements) can have confidence that the platform provides state-of-the-art security services using certified Roots of Trust.

One could define some independent services and some composite services. We call them independent because one can implement their assigned service without the help of other services; that is, they are not dependent on the other services. The implementer of composite services decides how dependent they will be on other services: They could be completely self-contained, or they could use the assigned services of other Roots of Trust, which makes them dependent on the other services.

The following figure illustrates a possible combination of RoT Security Services.

Figure 4-1: Possible Relationship of Independent and Composite RoT Security Services



Requirement 4-1: Security Service Interface(s)

The interface (in this context, also called the entry point) SHALL provide the ability to exercise the capability or function of the Root of Trust.

The security service interface (or entry point) maintains the integrity of the contents of the shielded locations.

Requirement 4-2: Shielded Location Integrity Protection

The interface SHALL provide the appropriate level of protection to the shielded locations to maintain the integrity of the contents.

Any verification service that relies on an uncorrupted public key relies on the preservation of the integrity of the contents of the shielded location that contains the public key. The same holds true for other security services that use private keys to encrypt other objects, or to create signatures.

Requirement 4-3: Shielded Location Confidentiality Protection

The interface MAY provide the appropriate level of protection to the shielded locations to maintain the confidentiality of the contents.

Confidentiality is not required for public keys, but is required for private keys.

Requirement 4-4: Shielded Location Access

The interface SHALL enforce internal policy regarding access to and use of the shielded locations.

The security service interface restricts access to authorized entities, thus protecting data held in shielded locations from unauthorized use and disclosure.

4.1 Authentication Service

The Root of Trust for Authentication maintains one or more shielded locations for the purpose of securely storing and preserving the integrity of at least one credential. A service user stores one or more authentication credentials here for use at a later time to assist in authenticating compliance with a policy, such as performing access control over the use and/or release of keys, or authorizing access to data. The Root of Trust for Authentication also includes an interface to maintain authorized access to and use of the contents of the shielded locations as well as protecting it from unauthorized use and disclosure. In this case, authorized use is for authentication purposes only.

Requirement 4-5: Authentication Service Storage

The Root of Trust for Authentication MAY maintain one or more shielded locations for storing credentials to be used in authentication protocols.

The RoT may use this service to authenticate an external entity and vice versa; an external entity may use this service to authenticate the RoT.

Requirement 4-6: Authentication Service Access Control

The Root of Trust for Authentication SHALL maintain an interface to control access and use of the contents of shielded locations.

4.2 Confidentiality Service

The Root of Trust for Confidentiality maintains shielded locations for the purpose of storing sensitive data, such as secret keys and passwords.

Requirement 4-7: Confidentiality Service Storage

The Root of Trust for Confidentiality SHALL maintain one or more shielded locations for the purpose of storing secret values.

Requirement 4-8: Confidentiality Service Access Control

Since the values are secret, the Root of Trust for Confidentiality SHALL protect the contents from unauthorized disclosure and use.

4.3 Identification (of a Root of Trust) Service

The Root of Trust for Identification maintains a shielded location for storing a secret value, such as a symmetric key or an asymmetric private key, for the purpose of establishing the identity of the Root of Trust. This identity should not be confused with the owner of the Root of Trust, the owner of the platform, or the owner of the device. This identity is only for the Root of Trust. Through its signing capability, the Identification Service provides message origin authentication.

An implementation may store the hash of a public key certificate in a shielded location in which the Root of Trust protects it from unauthorized modification. In the case of an Initial Root of Trust Component, this identity may serve as a proxy for the identity of the platform, since a platform contains one and only one Initial Root of Trust Component.

Requirement 4-9: Identification Service Storing

The Root of Trust for Identification SHALL maintain a shielded location for storing a secret value.

Requirement 4-10: Identification Service Access

The Root of Trust for Identification SHALL maintain an interface that allows it to authenticate that it is authorized to perform the services it provides.

Requirement 4-11: Authentication Protocols

Root of Trust Identification Services SHOULD support standard protocols for authenticating the RoT's identity to other entities.

Requirement 4-12: Root of Trust Identity Authentication

The owner of a Root of Trust SHOULD vouch for the identity of the RoT, e.g. in the form of a public key certificate signed by the manufacturer or vendor of the Root of Trust.

4.4 Integrity Service

The Root of Trust for Integrity maintains shielded locations for the purpose of storing and protecting the integrity of non-secret critical security parameters and platform characteristics. Critical security parameters include, but are not limited to, authorization values, public keys, and public key certificates.

Requirement 4-13: Integrity Service Storage

The Root of Trust for Integrity SHALL maintain shielded locations for the purpose of storing and protecting the integrity of non-secret but critical security parameters and platform characteristics.

Requirement 4-14: Integrity Service Access

The Root of Trust for Integrity SHALL maintain an interface to protect shielded locations from unauthorized modifications.

Certificates, which are signed, may need additional integrity protection to protect against unauthorized substitution in Roots of Trust.

4.5 Measurement Service

The Root of Trust for Measurement provides the ability to reliably create platform characteristics. The Root of Trust for Measurement may calculate the cryptographic hashes of code and data. The Root of Trust for Measurement may convey such platform characteristics to other Roots of Trust or may allow the platform characteristics to be used by other RoT Security Services, such as the Root of Trust for Integrity, the Root of Trust for Verification, and the Root of Trust for Update. An implementer using the Root of Trust for Measurement may perform a measurement once prior to using the measured resource for the first time, or periodically, as policy dictates.

Requirement 4-15: Measurement Service

The Root of Trust for Measurement SHALL reliably create platform characteristics.

The Root of Trust for Measurement typically does not contain a shielded location for the measurement.

4.6 Authorization Service

The Root of Trust for Authorization provides reliable capabilities to assess authorization tokens and determine whether or not they satisfy policies for access control. It may store authorization tokens in shielded locations or accept them through an interface. Policies may be resident in the Root of Trust for Authorization. However, more sophisticated implementations may have the ability to ingest policies and tokens, do the computations to determine whether or not the tokens satisfy the policies, and return a simple Boolean to signal the result (e.g. true = tokens presented satisfy the policy, false = tokens presented do not satisfy the policy).

Optional Composite Root of Trust

A possible composite Root of Trust for Authorization implementation would contain the features required for a Root of Trust for Authentication and a Root of Trust for Integrity. Such a composite Root of Trust for Authorization must maintain shielded locations for storing tokens and must use the same interface as the Roots of Trust for Authentication and Integrity to control access to the use and modification of the content of the shielded locations. If a composite Root of Trust for Authorization exposes the interface of one or both of the Roots of Trust for Authentication and Integrity, then it can claim to be a Root of Trust for each such service as well. If the composite Root of Trust for Authorization does not expose the interface of one of those Roots of Trust for external use, then it cannot claim to be a Root of Trust for that service.

Requirement 4-16: Authorization Service

The Root of Trust for Authorization SHALL reliably assess authorization tokens and determine whether or not they satisfy policies for access control.

Requirement 4-17: Authorization Service Composition

The Root of Trust for Authorization MAY provide the same services as the Root of Trust for Authentication, the Root of Trust for Integrity, or both, in which case(s) it SHALL also satisfy the requirements for each such Root of Trust.

4.7 Reporting Service

The Root of Trust for Reporting reliably reports platform characteristics. It provides an interface that limits its services to providing reports on its platform characteristics authenticated by a platform identity.

Optional Composite Root of Trust

A possible composite Root of Trust for Reporting implementation would contain the features required for Roots of Trust for Authentication, Confidentiality, Identification, Integrity, and Measurement. Such a composite Root of Trust for Reporting must maintain shielded locations for storing platform characteristics and platform identity; must maintain the capabilities and interfaces that provide and protect the services of Authentication, Confidentiality, Identification, Integrity, and Measurement; and must control access to the use and modification of the contents of the shielded locations. This implementation need not expose the other services, but may maintain them strictly for the use of the Root of Trust for Reporting.

Alternatively, a composite Root of Trust for Reporting implementation may contain any subset of these services, and rely on independent RoT Security Services to complete the set.

If a composite Root of Trust for Reporting exposes the interface of one or more of the Roots of Trust for Authentication, Confidentiality, Identification, Integrity, and Measurement, then it can claim to be a Root of Trust for each such service as well. If the composite Root of Trust for Reporting does not expose the interface of one of those Roots of Trust for external use, then it cannot claim to be a Root of Trust for that service.

Requirement 4-18: Reporting Service

The Root of Trust for Reporting SHALL have the capability to reliably report platform characteristics authenticated by the platform's identity in a non-repudiable way.

Requirement 4-19: Reporting Service Composition

The Root of Trust for Reporting MAY provide the same services as one or more of the Roots of Trust for Authentication, Confidentiality, Identification, Integrity, and Measurement, in which case(s) it SHALL also satisfy the requirements for each such Root of Trust. It MAY provide these services in addition to the Root of Trust for Reporting, or MAY leverage these services from Roots of Trust that are physically independent.

Requirement 4-20: Reporting Service Content

The Root of Trust for Reporting SHALL provide freshness for reports of the platform characteristics.

Requirement 4-21: Reporting Service Non-repudiation

The Root of Trust for Reporting SHALL maintain an interface that limits its services to providing reports authenticated in a non-repudiable way.

4.8 Update Service

The Root of Trust for Update verifies the integrity and authenticity of signed updates, and upon successful verification, authorizes the initiation of the update process. A platform may employ a Root of Trust for Update to update the code and data for Roots of Trust, including itself. Non-Root-of-Trust platform software may also utilize the Root of Trust for Update for updates.

Optional Composite Root of Trust

A possible composite Root of Trust for Update implementation would contain the features required for Roots of Trust for Authentication (to authenticate the identity of the owner of the update), Authorization (to authorize initiation of the update after satisfying the policy), Confidentiality (to protect secret keys, if needed), Integrity (to protect public keys and platform characteristics, if needed), Verification (to check the integrity of the update) and Measurement (to recreate platform characteristics, e.g. cryptographic hash of the update payload). Such a composite Root of Trust for Update must maintain shielded locations for storing owner identity and authorization tokens, if any; must maintain the capabilities and interfaces that provide and protect the services of Authentication, Authorization, Confidentiality, Integrity, and Measurement; and must control access to the use and modification of the contents of the shielded locations. Alternatively, a composite Root of Trust for Update implementation may contain any subset of these services, and rely on independent RoT Security Services to complete the set.

If a composite Root of Trust for Update exposes the interface of one or more of the Roots of Trust for Authentication, Authorization, Confidentiality, Integrity, and Measurement, then it can claim to be a Root of Trust for each such service as well. If the composite Root of Trust for Update does not expose the interface of one of those Roots of Trust for external use, then it cannot claim to be a Root of Trust for that service.

Requirement 4-22: Update Service

The Root of Trust for Update SHALL verify the integrity and authenticity of signed updates, and upon successful verifications, SHALL authorize the initiation of the update process.

Requirement 4-23: Update Service Composition

The Root of Trust for Update MAY contain the RoT Security Services, such as Authentication, Authorization, Confidentiality, Integrity, and Measurement, necessary to perform the update, in which case the Root of Trust for Update SHALL satisfy the requirements which correspond to those services, even if it does not expose the service via an interface.

4.9 Verification Service

The Root of Trust for Verification verifies the integrity and authenticity of signed objects. Objects may include, but are not limited to, public keys, code, and data. A platform may employ an Initial Root of Trust Component for Verification to verify the code and data for Extended Root of Trust Components as well as non-Root-of-Trust code and data. Ideally, a platform uses a Root of Trust for Verification to verify non-initial-Root-of-Trust-component code and data prior to its execution. Non-Root-of-Trust platform software may also utilize the Root of Trust for Verification.

Optional Composite Root of Trust

A possible composite Root of Trust for Verification implementation would contain the features required for Roots of Trust for Authentication (if needed to authenticate the identity of the owner of the object being verified), Confidentiality (if needed to protect secret keys), Integrity (if needed to protect public keys and platform characteristics), and Measurement (to recreate platform characteristics, e.g. cryptographic hash of the update payload). Such a composite Root of Trust for Verification must maintain shielded locations for storing owner identity, keys, and platform characteristics, if any; must maintain the capabilities and interfaces that provide and protect the services of Authentication, Confidentiality, Integrity, and Measurement; and must control access to the use and modification of the contents of the shielded locations. Alternatively, a composite Root of Trust for Verification implementation may contain any subset of these services, and rely on independent RoT Security Services to complete the set.

If a composite Root of Trust for Verification exposes the interface of one or more of the Roots of Trust for Authentication, Confidentiality, Integrity, and Measurement, then it can claim to be a Root of Trust for each such service as well. If the composite Root of Trust for Verification does not expose the interface of one of those Roots of Trust for external use, then it cannot claim to be a Root of Trust for that service.

Requirement 4-24: Verification Service

The Root of Trust for Verification SHALL verify the authenticity of digital signatures and verify the integrity of the objects protected by the signatures.

Requirement 4-25: Verification Service Composition

The Root of Trust for Verification MAY contain any subset of the RoT Security Services for Authentication, Confidentiality, Integrity, and Measurement necessary to perform signature verification, in which case(s) it SHALL satisfy all the requirements which correspond to each such service, even if it does not expose the service via an interface. It MAY provide all these capabilities in addition to the Root of Trust for Verification, or it MAY leverage these services from physically independent Roots of Trust.

5 Trust Relationships

This document illustrates several forms of trust relationships: Bootstrapped Root of Trust, a Chain of Trust for Verification, a Chain of Trust anchored in a Bootstrapped Root of Trust, a Chain of Trust for Confidentiality, and Intrinsic trust between platforms.

5.1 Bootstrapped Root of Trust

A platform may instantiate a Root of Trust when it executes a sequence of code modules in an environment in which it is infeasible to record the outcome of any code or data integrity verification that may take place. On the other hand, a platform instantiates a Chain of Trust when it purposely extends a security service in such a way that an application can inspect the links in the chain created by the extensions at any time and verify their validity, thus verifying the validity of the security service it extends from end to end.

The Bootstrapped Root of Trust occurs through a sequence of code modules in which a RoT Component (either an iRoTc or an eRoTc) performs verification and authorization checks on the next code module. If it passes those checks without the parent RoT Component leaving a record behind, it may be executed and becomes an eRoTc in a Bootstrapped Root of Trust. The RoT Component performing the verification is called the parent RoT Component and the eRoTc is called the child RoT Component. The child RoT Component gains an implicit trust from the previous module.

Figure 5-1 illustrates an example of a Bootstrapped Root of Trust using verification services between RoT Components.

Figure 5-1: Example of a Chain of Trust

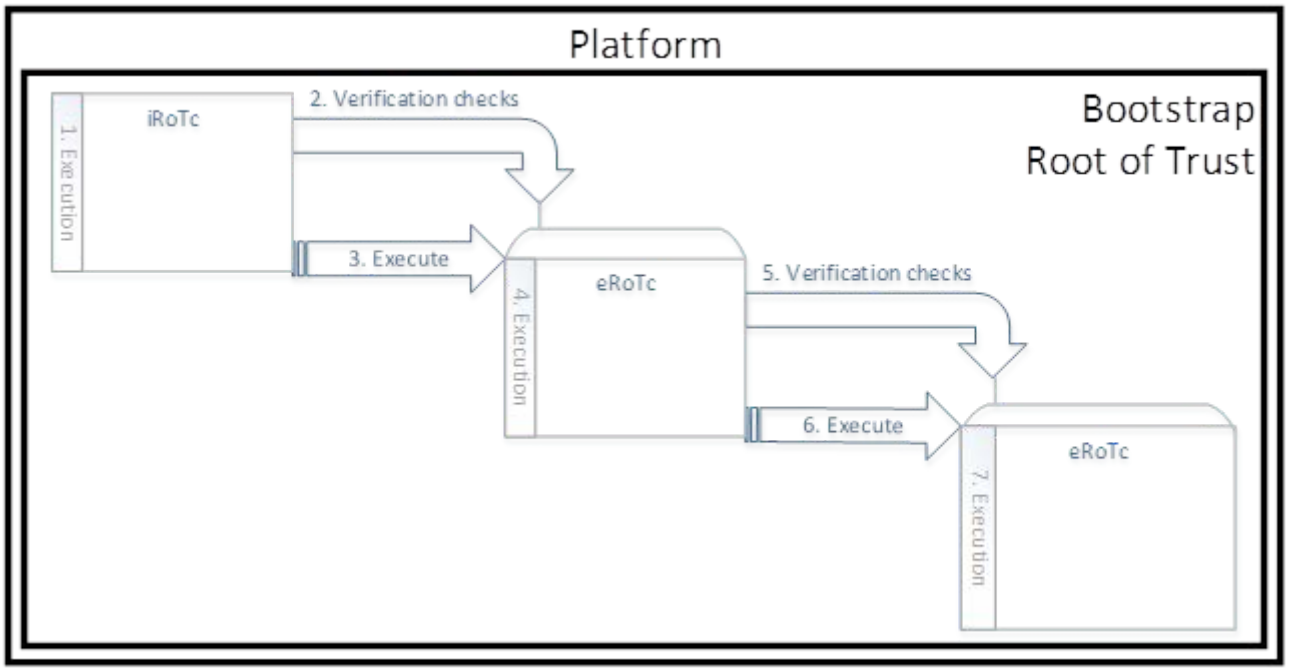


Figure 5-1 illustrates an initial module, labeled iRoTc, and two others, labeled eRoTc, as a chain of modules in which each of the Extended RoT Components (eRoTc) are verified before execution, and in which these verifications are hidden from the platform and the platform users and administrators.

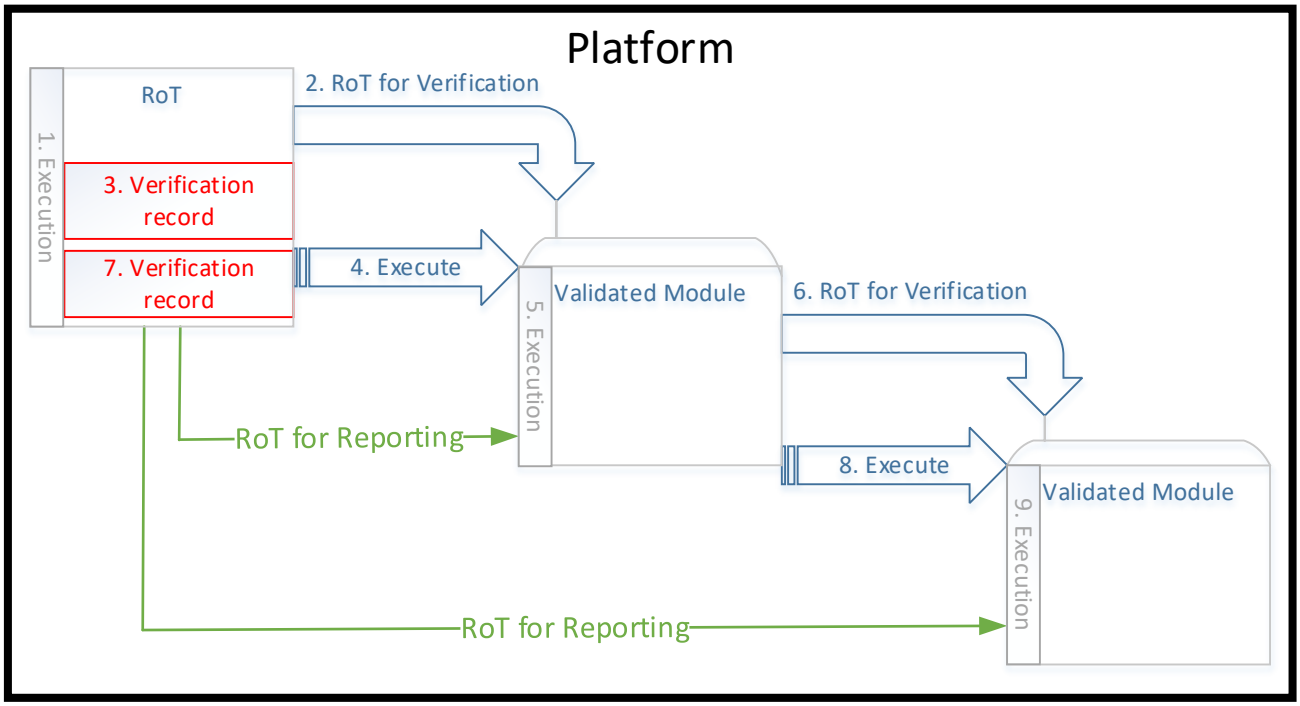
- In Step 1, the Initial Root of Trust Component (iRoTc) begins execution. Since this block of code is the first to execute on this platform, it is either not able to measure or verify itself before execution, or measuring itself after execution provides little security value to the platform. At some point, the iRoTc may expand platform's available resources and capabilities, including security services, by transferring control to another block of code.
- In Step 2, the iRoTc initiates the verification of the integrity of the next block of code in the execution chain. In cases in which the iRoTc lacks the ability (perhaps due to limited resources) to memorialize or retain the result of the verification, the iRoTc will not store or retain this result. In lieu of resources, the vendor of the platform may have chosen to bootstrap a series or collection of modules to create a Root of Trust for the platform.
- In Step 3, the iRoTc is sufficiently satisfied with the verification of the next block of code and passes execution to it.
- In Step 4, the eRoTc executes.
- In Step 5, the eRoTc initializes verification of the next block of code in the chain.
- In Step 6, the eRoTc is satisfied with the verification and passes execution on to the final eRoTc.
- In Step 7, the final eRoTc begins execution, and presumably possesses all the resources and security services necessary to function as a full-fledged Root of Trust.

5.2 Chain of Trust

A Chain of Trust actually extends a service from a RoT (either a Primary or a Secondary RoT) or module to other modules. A good example of this is a Chain of Trust for Verification, in which the RoT contains a verification service, and uses that service to verify a module that contains a verification service similar to the verification service in the parent. A Chain of Trust shall always start in a RoT. Like Roots of Trust, Chains of Trust shall contain at least one security service. The method of extending the service from a Root of Trust to subsequent modules is highly implementation-dependent.

Figure 5-2 illustrates an example of a Chain of Trust when the RoT for Verification is extended.

Figure 5-2: Example of a Chain of Trust

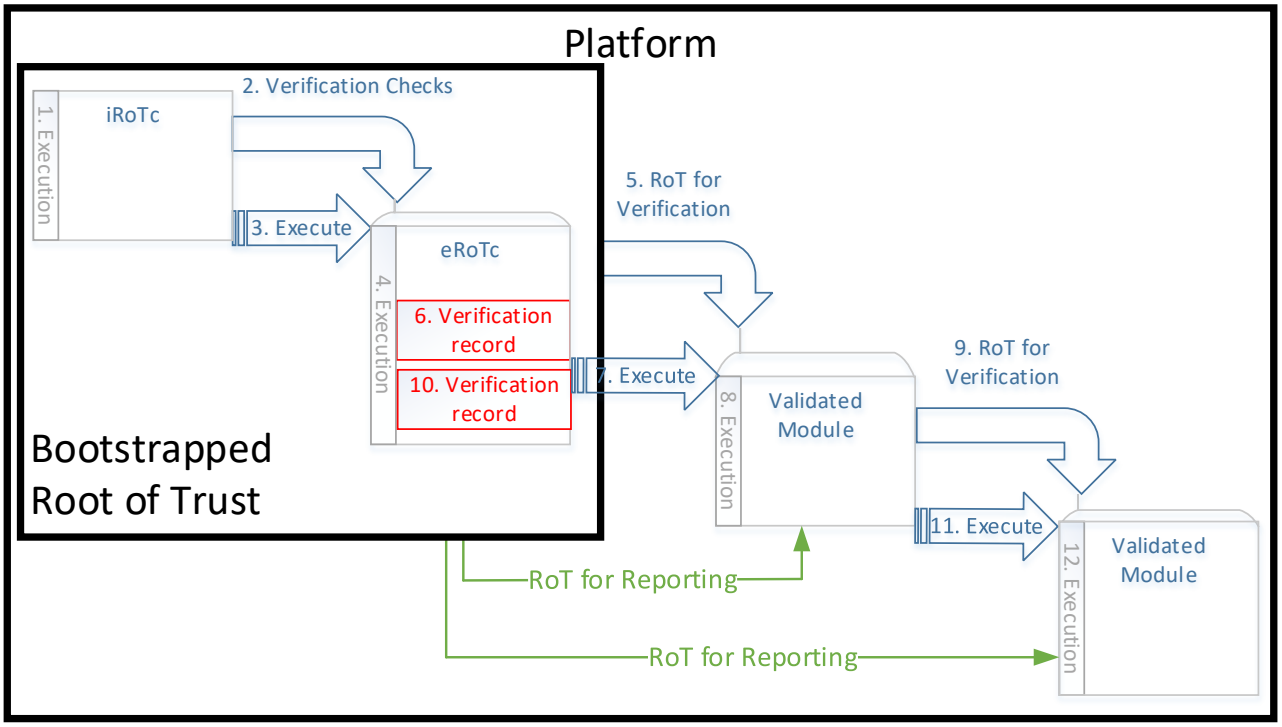


- In Step 1 of Figure 5-2, the RoT begins execution.
- In Step 2, the RoT verifies a module of code.
- In Step 3, the RoT stores a record of the verification within itself.
- In Step 4, it transfers execution to the newly validated module.
- In Step 5, the newly validated module begins execution.
- In Step 6, the newly validated module verifies yet another module.
- In Step 7, it records the verification in a shielded location.
- In Step 8, it transfers execution to the newly validated module.
- In Step 9, the newly validated module begins execution. The succession of recorded verification of modules forms a Chain of Trust for Verification.

At any point, any code on the platform, whether the code is within a Chain of Trust for Verification or not, can inspect the verification records for this Chain of Trust to gain introspection into the integrity of the validated modules at initial execution time throughout the chain. The green arrows indicate that the RoT is also a Root of Trust for Reporting

5.3 Bootstrapped Root of Trust Anchors Chain of Trust

Figure 5-3: Example of a Bootstrapped Root of Trust and a Chain of Trust Anchored by It



- Step 1 in Figure 5-3 begins execution of an iRoTc.
- In Step 2, the iRoTc verifies the eRoTc.
- Once the iRoTc is satisfied with the verification of the eRoTc, in Step 3 it passes execution to it.
- In Step 4, execution begins within the eRoTc. At this point, the iRoTc and the eRoTc form a complete Bootstrapped Root of Trust for Verification, possibly other services as well, such as Integrity and Reporting.
- In Step 5, that Root of Trust verifies its first module, which contains code for verification of future modules.
- In Step 6, it memorializes the verification in a shielded location within the Bootstrapped RoT, taking advantage of its Integrity Service (i.e. the Bootstrapped RoT protects the integrity of the record of the verification of the first module).
- In Step 7, it passes execution to the newly validated module.
- In Step 8, execution of the newly validated module begins, extending the Chain of Trust for Verification from the Bootstrapped RoT to the first validated module.
- In Step 9, the newly validated module verifies the next module in the execution chain.
- In Step 10, the first validated module memorializes the verification in a protection location within the Bootstrapped RoT, taking advantage again of its Integrity Service.
- In Step 11, once the first validated module is satisfied with the verification of the second module, it passes execution to it.

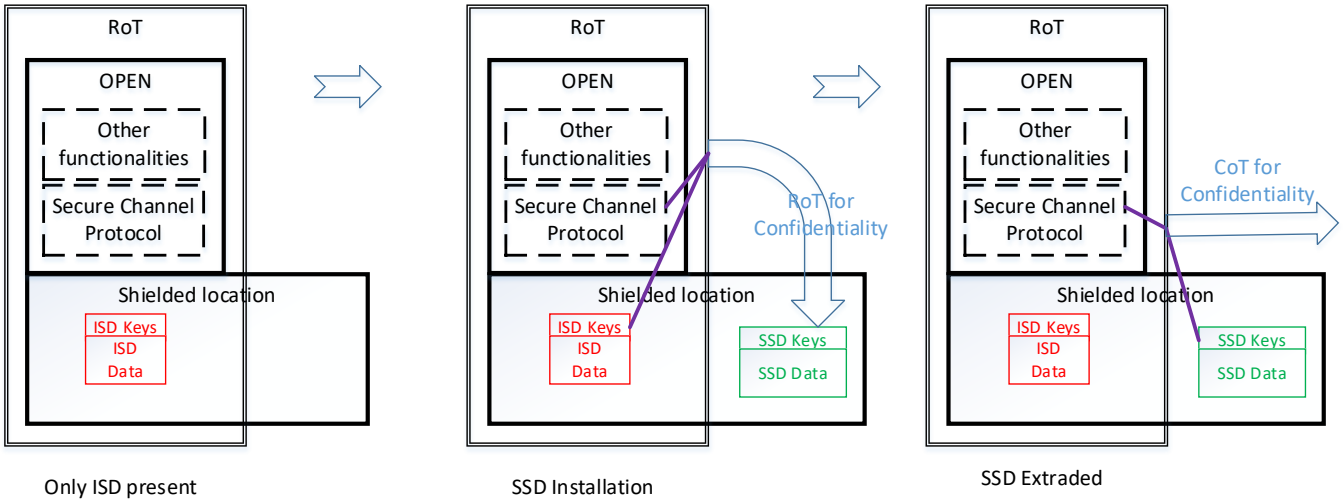
- In Step 12, execution of the second validated module begins. If the second validated module contains more verification code, then we say that we have extended the Chain of Trust for Verification to the second validated module. The green arrows indicate that the Bootstrapped RoT also serves as a Root of Trust for Reporting for each of the validated modules.

5.4 Chain of Trust for Confidentiality

In the GlobalPlatform SE context, the Secure Channel Protocol service is provided by Root of Trust Services (see section 8.2). The Issuer Security Domain Key(s) and software to provide the Secure Channel Protocol are part of the Root of Trust, and they are provisioned during the manufacturing process (see Only ISD present in Figure 5-4). Later, a new Security Domain can be installed under authorization of the owner of the Issuer Security Domain. Thanks to the RoT Confidentiality Service, the key of the new Security Domain can be loaded and stored securely under the authorization of the Issuer Security Domain (see SSD Installation in Figure 5-4). At that time, the newly installed Security Domain may provide Secure Channel Protocol service to applications (applets) by using its own key(s) but with the Secure Channel Protocol code of the Issuer Security Domain (see SSD Extradited in Figure 5-4). When an application uses the Secure Channel Protocol of a Security Domain (different than the Issuer Security Domain), it relies on a Chain of Trust built by the extension of the RoT of Confidentiality Service.

Figure 5-4 illustrates an example of the lifecycle of a Chain of Trust that extends the Confidentiality Service. The figure shows the evolution in time of one RoT that installs a new key through a RoT service for confidentiality; this new key is then reused with the same code of the RoT service for confidentiality to perform operations (i.e. cipher some information).

Figure 5-4: Example of a Chain of Trust that Extends the Confidentiality Service



This document does not list all the possibilities, although Chapter 7 explains Chains of Trust for a few selected services.

The Trusted Computing community is more familiar with the Chain of Trust due to the PC and TPM nature of the devices they have worked with. The embedded and mobile device communities are more familiar with the Bootstrapped Root of Trust model found deployed in secure boot chains. Mapping both models in this document is intended to clarify the concepts for those two key communities and improve communication between the parties.

A Validated Module is one type of module that may make use of a Chain of Trust. It must have been correctly authorized and verified by a security service, and it must have an accessible record of this action.

A Measured Module is another type of module that may make use of a Chain of Trust. It must have been measured by a security service, and it must have an accessible record of this action.

5.5 Intrinsic Trust

In some implementations, a platform that anchors a RoT (pRoT) cannot verify/measure the next executable code or keep a record of this verification. In these cases, a second platform that anchors a RoT (sRoT) can provide these RoT Services. In such implementations, the pRoT has to trust the RoT services provided by the sRoT. We call this relationship an Intrinsic Trust. Additionally, platform code outside a Chain of Trust may also rely on RoT services provided by an sRoT (i.e. a RoT anchored in a different platform on the same device)

The diagram in Figure 5-5 illustrates one example of a relationship between a Primary RoT and a Secondary RoT.

By virtue of living on the same device, the pRoT intrinsically trusts the sRoT, especially if it does not check or verify the integrity of the sRoT before using its security services.

Figure 5-5 provides an example of the Chain of Trust as well as Intrinsic Trust between Primary and Secondary Roots of Trust.

Figure 5-5: Intrinsic Trust between Two Platforms

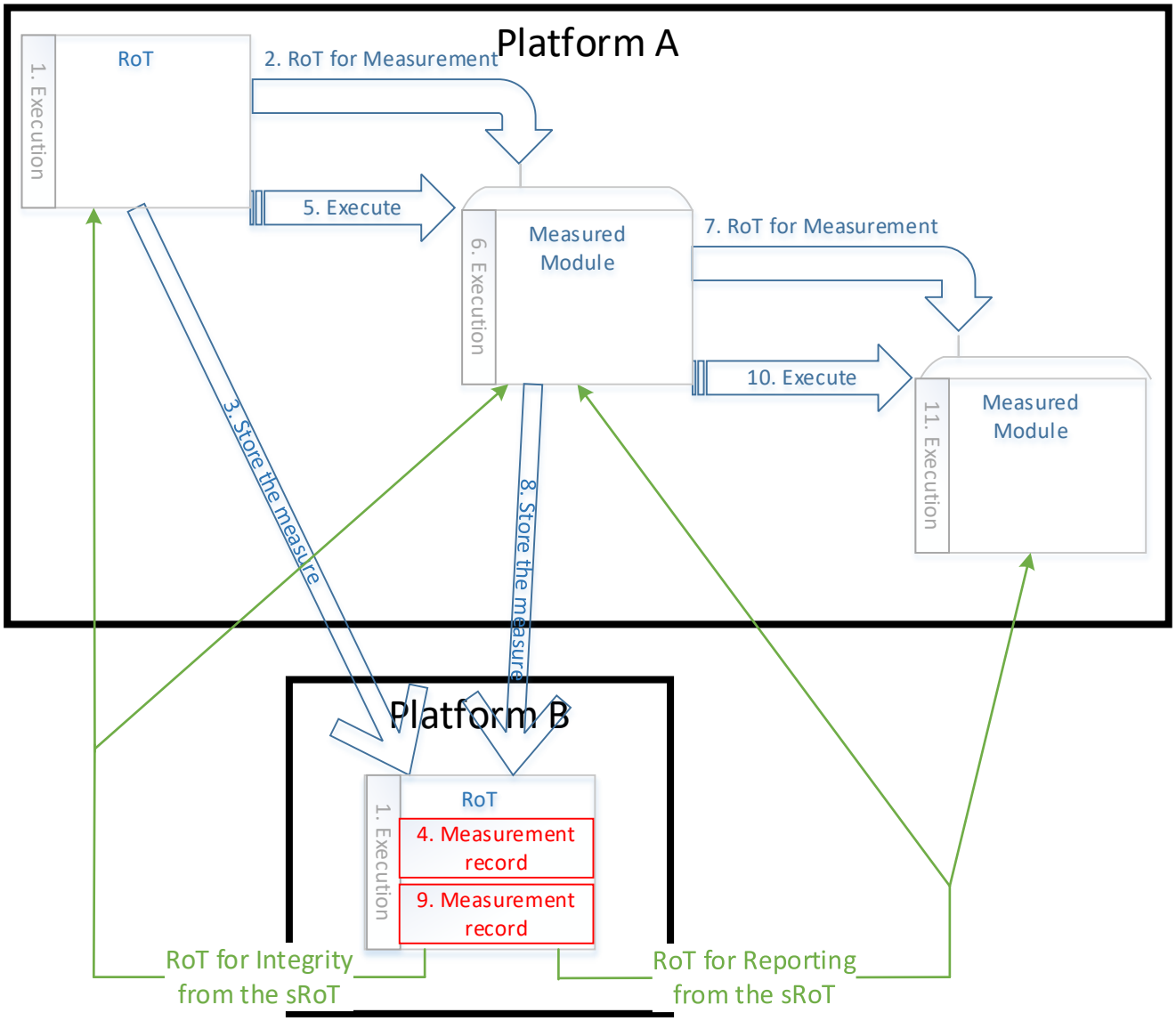


Figure 5-5 illustrates a possible scenario of a device which contains a main application processing unit (Platform A) and an auxiliary cryptographic processing unit (Platform B).

- In Step 1 of Platform A, the RoT begins execution. Independently in Platform B, the RoT also begin execution.
- In Step 2, Platform A's RoT acts as a Root of Trust for Measurement and measures the next module in a chain of execution.
- In Step 3, Platform A's RoT stores the measurement in Platform B's RoT thanks to the RoT for Integrity provided by Platform B. In this case, Platform A's RoT has limited capabilities, and relies on Platform B's RoT for certain services. Here we call Platform A's RoT the Primary RoT (pRoT) and Platform B's RoT the Secondary RoT (sRoT).
- In Step 4, Platform B's RoT securely stores the measurement from step 2 in a shielded location.
- In Step 5, Platform A transfers execution from the RoT to the first Measured Module.
- In Step 6, the first Measured Module begins execution.
- In Step 7, Platform A's newly measured module measures the next executable in the chain.
- In Step 8, the first Measured Module sends the measurement of the second Measured Module to Platform B's RoT.
- In Step 9, Platform B stores the second Measured Module's measurement in a shielded location.
- In Step 10, the first Measured Module transfers execution to the second Measured Module.
- In Step 11, the second Measured Module begins execution. At any point following this step, applications on Platform A may request to inspect the measurements stored in Platform B to gain confidence in the integrity of the Measured Modules at the time each first initiated execution. The green arrows on the left indicate that Platform B serves as the RoT for Integrity for Platform A. The green arrows on the right indicate that the RoT in Platform B is a RoT for Reporting for the modules in Platform A. Although Platform A did not measure or verify the integrity of the executables in Platform B, it trusts Platform B to provide RoT Services for Integrity and Reporting.

5.6 Requirements

Requirement 5-1: Origin for Bootstrapped Root of Trust

A Bootstrapped Root of Trust SHALL always start with an Initial Root of Trust Component (iRoTc).

***Note**: Italicized paragraphs throughout this chapter are informative only.*

If a platform implements a Bootstrapped Root of Trust, it is initialized by the iRoTc. A Bootstrapped Root of Trust may contain eRoTcs as code modules.

A platform that does not contain an iRoTc cannot implement a Chain of Trust.

Requirement 5-2: Origin for Chain of Trust

A Chain of Trust SHALL always start with a Root of Trust (RoT).

If a platform implements a Chain of Trust, it is initialized by a RoT. However, it is not necessary that the RoT be a pRoT. An sRoT may also serve as the Root for a Chain of Trust.

Requirement 5-3: Chain of Trust Extension

If a Root of Trust extends a security service by passing execution to a new module, it SHALL measure or verify the integrity of the new module, and record that measurement or verification before passing execution to it.

A second party can verify that the integrity measurement or check (in the case of a verification) took place after its execution.

6 Root of Trust in GlobalPlatform Context

The GlobalPlatform technology discussed in this document is specified in the Card Specification [GPCS] for the Secure Element and the TEE Internal Core API Specification [TEE Core] for the Trusted Execution Environment.

The security architecture of GlobalPlatform Secure Components (SE or TEE) is based on:

- Component management through a hierarchy of Security Domains, where each Security Domain on the platform represents an actor involved in the management of applications
- Strongly isolated execution of applications inside those components

The Security Domain has multiple responsibilities:

- For actors: Store management rules and access rights required to the platform data of the Security Domain owner.
- For service providers: Install applications in a secure and controlled manner, allowing the set-up of services (such as banking applications for a bank).
- For service provider aggregators: Provide a management entity that enables service providers to form relationships over multiple classes of devices. An aggregator (aka TAM or TSM) may exist on many devices and offers this channel to service providers.

The GlobalPlatform secure chip specifications require that a Security Domain exists before any service is provided. The presence of this Security Domain implies the presence of a Root of Trust (RoT) as defined in this document.

When loading the key(s) during the platform manufacturing process, the manufacturer shall:

- Ensure the correctness and authenticity of computing engine and code
- Ensure the secrecy of the keys

The security level of the manufacturing process determines the trust level that you can have in the platform.

For the Secure Element, this is done in a secure and controlled facility, where the facility has strong physical restricted access and the manufacturing process is audited.

Multiple certification schemes (such as Common Criteria or EMVCo) describe the way to evaluate and certify the computing engine, code, and manufacturing process and determine the level of assurance that one can have in the RoT and the first Security Domain of the platform.

A part of the code of the Issuer or root Security Domain code and its data and keys is a Root of Trust, as defined in Chapter 3.

The Issuer or root Security Domain and the Supplementary Security Domain provide security services (extensions) as defined in Chapter 4 for the actors (Card Issuer, Application Provider, or an independent entity responsible for enforcing specific off-card and on-card security policies).

For example, a Security Domain provides secure channel services that protect messages between the platform and an off-platform entity:

- **Entity authentication** – In which the platform or the off-platform entity proves its authenticity to the other entity by means of a challenge-response protocol.
- **Message integrity and authentication** – In which the receiving entity (the platform or off-platform entity) ensures that the data being received from the sending entity (respectively the off-platform entity or platform) actually came from an authenticated entity in the correct sequence and has not been altered.
- **Confidentiality** – In which data being transmitted from the sending entity (the off-platform entity or platform) to the receiving entity (respectively the platform or off-platform entity) is not viewable by an unauthenticated entity.

Suppose vendors build SEs and TEEs to satisfy RoT requirements. Since SE and TEE specifications require them to validate all SE and TEE applications, respectively, these applications become Validated Modules. As long as the GlobalPlatform Secure Components provide reportable assertions of the validations, Validated Modules can become part of a Chain of Trust for the security services they provide, if any.

The owner of a Security Domain may authorize the installation of a new Security Domain on the Secure Element or the Trusted Execution Environment for another actor. GlobalPlatform offers the possibility to create a child Security Domain and include it in the Security Domain Chain of Trust by service extension from the RoT of the GlobalPlatform Secure Component.

The RoT of the GlobalPlatform Secure Component is composed of the first SD, plus additional specific parts of code, data, and keys.

The secure component administration is based on a Chain of Trust that indicates, at each node of the hierarchy, which actors are allowed to perform what actions in the environment.

The Chain of Trust is controlled during the creation of the child Security Domain, and specific rules may be stored that will be enforced during all content updates of the secure component.

To support dynamic ecosystems, it's possible to create different independent hierarchies inside the secure component. This allows different ecosystems to manage in parallel a part of the secure component. In such a multi-tenant environment and to avoid a monopolistic approach, a specific SD or RoT has been created to allow the confidential loading of keys of any actors in the platform. In [GPCS] we refer to this by the term *Controlling Authority SD*. Such an SD performs only confidential key loading services to other SDs.

GlobalPlatform technologies need to fulfill the following requirements to achieve the more general RoT requirements defined in Chapters 3 and 4.

6.1 Requirements for GlobalPlatform Secure Components

Requirement 6-1: Manufacturing Process

The manufacturing process SHALL implement secure procedures to protect the chip design, code, and keys at all stages of the fabrication.

***Note**: Italicized paragraphs throughout this chapter are informative only.*

Example: An SE is assembled in a secure environment where the symmetric keys are injected during the manufacturing process. A Hardware Security Module (HSM) stores master keys and the keys are diversified per chip. Procedures and access controls are in place to prevent key access and software manipulation during the manufacturing process.

Example: For a TEE, the TEE boot chain signing private keys are stored in an HSM. Factory installed SD and TA related master keys are generated on the platform or in an HSM in a diversified manner. Procedures and access controls are in place to prevent unauthorized key and software modification or addition during the manufacturing process.

Requirement 6-2: Manufacturing Process Certification

The platform manufacturer SHALL be certified to prove that the manufacturing process to assemble in the platform the code, data, and keys required by the Roots of Trust fits the customer security requirements. The Certification Report SHALL explain how the certification is accomplished.

This could be realized by applying a certification scheme.

Requirement 6-3: Device Certification

The manufacturer SHALL certify the GlobalPlatform TEE (in the end-user device) or the GlobalPlatform SE according to customer requirements for tamper-resistance to protect the Root of Trust's code, data, and keys.

For example, a TEE will typically be targeted to be certified against [TEE PP], though for some markets additional certifications may be required, such as [FIPS 140-2].

Requirement 6-4: GlobalPlatform Multi-tenant (CASD)

If a Controlling Authority Security Domain (CASD), which is a Root of Trust for Confidentiality, Authentication, and Integrity for GlobalPlatform Secure Elements, is present, it SHALL be loaded during the manufacturing process.

Requirement 6-5: GlobalPlatform Secure Component Services

All GlobalPlatform Secure Components SHALL implement services for Confidentiality, Authentication, Identification, Authorization, and Verification rooted in a Root of Trust.

GlobalPlatform Components MAY implement services for Reporting and/or Update.

Requirement 6-6: GlobalPlatform Secure Components Extend Confidentiality Service

GlobalPlatform Secure Components SHALL extend the Confidentiality service to the Security Domain.

All Security Domains on a GlobalPlatform Secure Component use the same code to perform remote management actions on a particular platform. Authorization of the management actions is performed by validation of key relationships between the authorizing Security Domain and entities outside of the platform. Depending on the platform, this validation may be manifested by the key relationship allowing establishment of an SE Secure Channel or a TEE Security Layer, or through TEE validation of an authorization token. A Security Domain extends the RoT for Confidentiality and (optionally) the RoT for Authorization by reusing the same code but adding keys loaded under the protection of the authorizing Security Domain.

Requirement 6-7: GlobalPlatform Verification RoT

The Initial Root of Trust Component SHALL verify the integrity and presence of key and data sets. If the verification fails, the RoT SHALL forbid any interaction with any interface in the platform enabled by that key or data set.

Requirement 6-8: GlobalPlatform Security Domain Chains of Trust

Each component in a Security Domain Chain of Trust SHALL have a unique identifiable owner.

Requirement 6-9: GlobalPlatform-unique RoT Identification

Each GlobalPlatform Secure Component SHALL have a RoT Identification number, which SHALL be unique among all GlobalPlatform RoTs.

7 Detailed TEE Use Cases for Roots of Trust

The Trusted Execution Environment is an embedded platform found on computing devices, offering an isolated execution space for validated, authorized software (Trusted Applications – TAs) away from the generic execution environment (the REE). It may be integrated into the device SoC, or provided as a separate physical component mounted to the main device PCB.

GlobalPlatform specifications for TEEs enable:

- Trusted storage of TA assets including keys and general data, in a manner such that only the storing TA may use, retrieve, or modify this data
- General Trusted OS functionality such as key manipulation and creation
- TA access to a simple user interface, enabling TAs to verify user identity in certified isolation, and display confidential information without risk of any other device resident code intercepting or interfering with that information
- TA communication with suitably connected SEs on the device, using SCP if required
- TA communication with remote servers over internet protocols, using TLS if required
- TA communication with applications in the REE

TEEs can provide SD and TA OTA management structures in much the same way as can be found in the Secure Element. In the TEE, these allow:

- Management isolation through the creation of root SDs
  - A root SD is a Security Domain with no managing parent SD.
- Management delegation through the creation of child SDs
  - A child SD is a Security Domain with a designated parent SD, which can provide some or all management operations if required.

Implementations can choose to base the OTA management security on either secret key or PKI technology, and have the choice of using Security Layers (similar to SE SCP) or protected authorization token based interaction. Implementations can also choose to allow the installation of root SDs, child SDs, and TAs either in the factory or in the field.

TEE OTA management capability includes potential rights to:

- Install further SDs, update existing SDs, remove SDs, provision SD data
- Install further TAs, update existing TAs, remove TAs, provision a subset of TA data
- Perform general TEE management operations such as Factory Reset

Whether a specific TEE SD may perform any or all of these operations depends on its assigned rights.

The GlobalPlatform specification for OTA management of the TEE is called the TEE Management Framework [TMF].

A TEE can support a number of customized security services in a host device such as a mobile phone or tablet. In particular, it can provide:

- Security service for entity authentication
- Security service for user authentication
- Security service for data integrity
- Security service for data confidentiality
- Security service key provisioning

These security services can also be implemented without using a TEE. Using a TEE opens up the possibility of a level of security assurance for the aforementioned services that might be out of reach of solutions that do not involve a TEE. The reasons for this are as follows:

- Protocol simplicity: Given that an application has to communicate with a range of protocols in relation to security and non-security issues, separating the security-critical protocol usage into a TA should simplify the security design and hence enable easier design time validation of those security-related protocols.
- Software simplicity: Given that an application has to perform a range of functionality, both security and non-security related, separating the security-critical functionality into a TA should simplify the security design and therefore enable easier design time validation of that security-related functionality.
- Physical and logical protection of keys: In order to offer the security services mentioned above, a TEE has to perform authorization and validation through keys, in an isolated manner. TEE technology has been engineered with exactly that goal in mind.
- Multi-stakeholder key management: In a multi-application environment, there may be multiple stakeholders, each with their own cryptographic keys. A TEE that complies with the GlobalPlatform specifications allows different stakeholders to manage their keys independently of each other.

The provision of RoT Security Services is one of the reasons that TEEs bring value to devices. A GlobalPlatform TEE offers security services to its TAs. Depending on how the GlobalPlatform TEE boots, those security services may be from a Non-Bootstrapped RoT—e.g. TEE Run Time OS in ROM—or from a Bootstrapped RoT—e.g. TEE Run Time OS validated as authorized by ROM (Non-Bootstrapped RoT or other Bootstrapped RoT) or based on a RoT with components outside the TEE (sRoT). Since these security services are rooted in RoTs, users can have confidence in the security services offered by the GlobalPlatform TEE and its TAs.

Implementing functionality in a TA inside the TEE environment brings the above advantages. In addition to this, similar to the way that a TCG TPM is a component of the Root of Trust offered by an x86 PC, when the TEE (or a TA) is present, it may be treated as a component of a Root of Trust by other platforms in the device.

The following sections discuss the TEE as defined in [TEE PP] and TEE System Architecture [TEE Sys Arch] and how the architecture relates to the concept of Root of Trust.

7.1 Factory Creation, Installation, and Life Cycle of TEE Root of Trust

This section describes events in the device life cycle of a Root of Trust as used in the GlobalPlatform TEE Protection Profile [TEE PP].

[TEE PP] defines the device life cycle of the TEE as follows:

Figure 7-1: Life Cycle of TEE-enabled Device

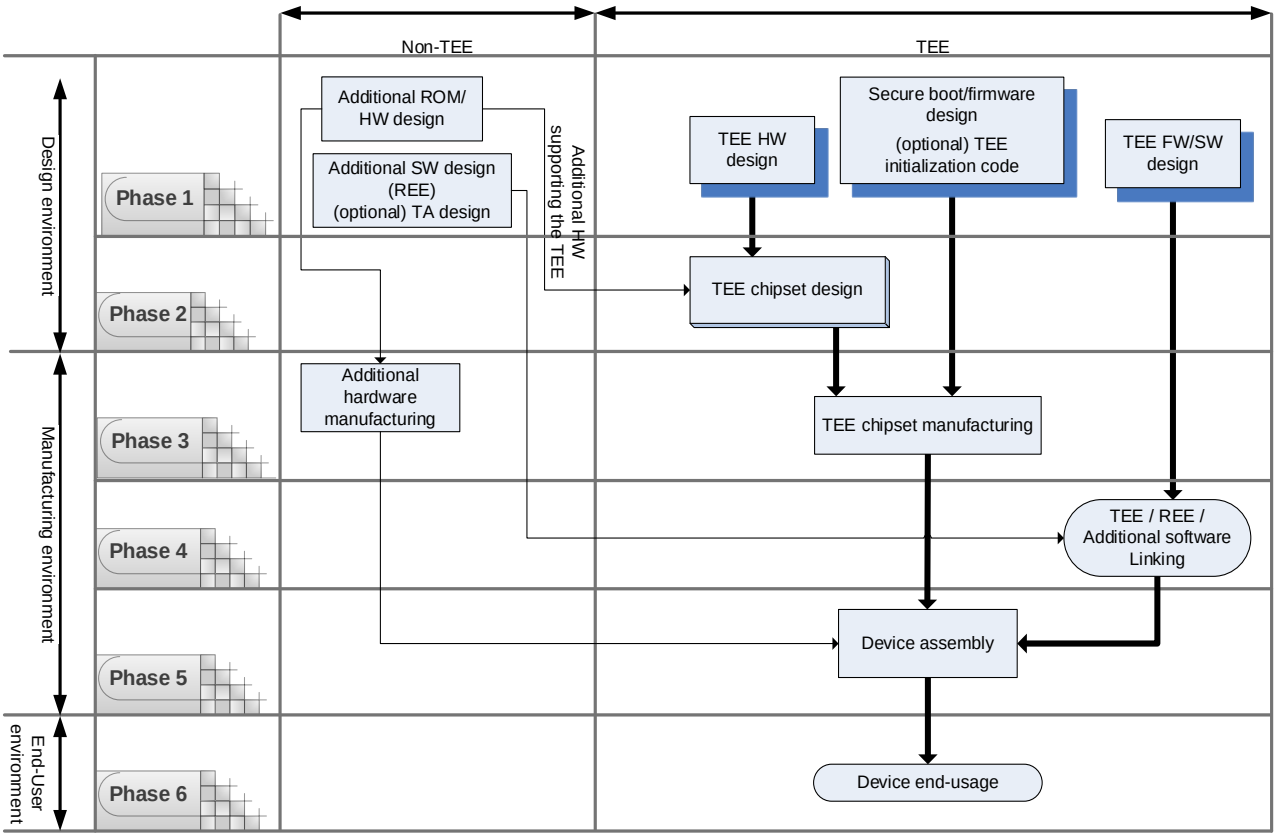
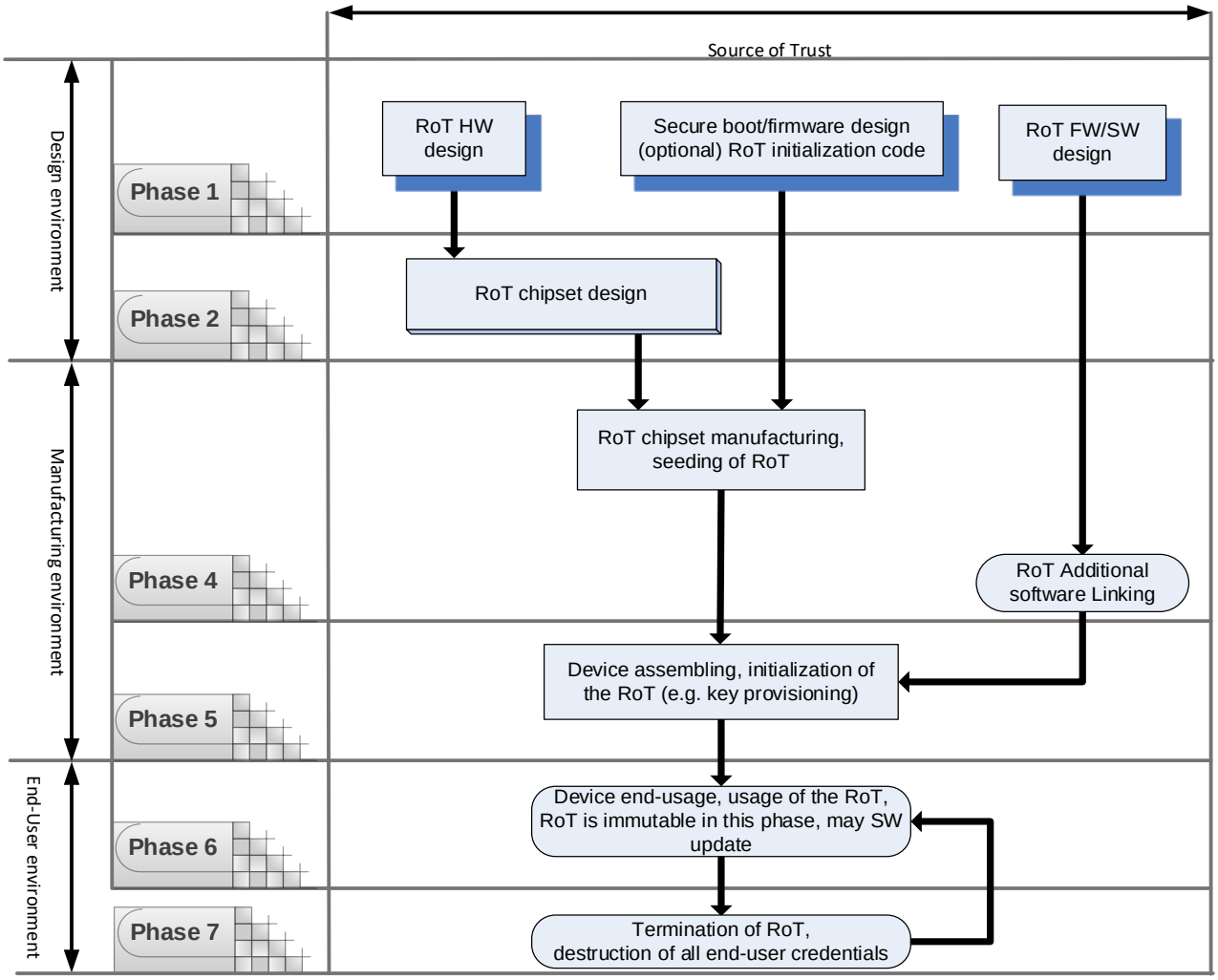


Table 7-1: Life Cycle of TEE-enabled Device

Phases	Actors
1 & 2: Firmware / Software / Hardware design	The TEE software developer: • Is in charge of TEE software development and testing compliant with GlobalPlatform specifications • May also develop the TEE initialization code that instantiates/initializes the TEE (e.g. part of the secure boot code) • Specifies the TEE software linking requirements The device manufacturer may design additional REE software that will be linked with the TEE in phase 4 to provide REE-controlled resources. The device manufacturer may also design Trusted Applications that will be integrated in phase 4. The TEE hardware designer is in charge of designing (part of) the processor(s) where the TEE software runs and designing (part of) the hardware security resources used by the TEE. The silicon vendor designs the ROM code and the secure portion of the TEE chipset. If the silicon vendor is not designing the full TEE hardware, the silicon vendor integrates (and potentially augments) the TEE hardware designed by the TEE hardware designer(s).
3: Silicon/chip manufacturing	The silicon vendor produces the chipset for a device which may contain many platforms. If the platform is a TEE, it enables, sets, or seeds the Root of Trust of the TEE. It may perform similar operations for the other platforms.
4: Software manufacturing	The device manufacturer is responsible for the integration, validation, and preparation of the software to load in the product that will include the TEE, any pre-installed Trusted Applications, and additional software required to use the product (e.g. REE, Client Applications).
5: Device manufacturing	The device manufacturer is responsible for the device assembly and initialization and any other operation on the device (including loading or installing Trusted Applications) before delivery to the end user.
6: End-usage phase	The end user gets a device ready for use. The Trusted Applications Manager is responsible for the loading, installation, and removal of Trusted Applications post-issuance.
7: End-usage termination	The end user terminates their relationship to allow device resale by performing a factory reset of the TEE.

Based on the life cycle of the TEE-enabled device, the life cycle of a RoT may be defined as follows:

Figure 7-2: Life Cycle of Root of Trust



7.2 Roots of Trust and TEE

There are four levels of concern about RoTs in relation to a TEE in a device.

1	RoT used during initialization of a TEE	How did the TEE RTE instantiate itself into a run time state? This is of concern to those who wish to trust the TEE itself.
2	RoT Security Services offered to application software on the TEE platform	What does the TEE OS software offer to application software? This is of concern to those who wish to make use of the trusted environment to instigate further services.
3	RoT Security Services offered to remote entities (off device) by the TEE platform	What does the TEE OS software offer to remote software management? This is of concern to those who wish to make use of the trusted environment to remotely manage TEE services.
4	RoT Security Services that applications in the platform may offer to other TEE platforms	How can a Trusted Application (TA) act as a component of a Root of Trust for another platform?

The following sections address those four areas.

7.2.1 #1 Roots of Trust Security Services for TEE RTE Instantiation

The term “boot time” refers to the time frame from the reset/power-up of the underlying hardware to the time an operating system has completed its initialization and loading and is offering services to generic applications. Based on this definition, boot time software also includes any firmware/ROM code that takes over the control of execution after the device is reset.

Only some TEEs need a boot time. TEEs resident in FLASH-based execution environments (like in an SE) do not need to go through a distinct boot time involving a multi-step load/validate/execute boot as (like in an SE) their code has been validated on installation, is executed in the validated location, and is guarded by hardware against integrity failures.

As indicated in [OMTP ATE TR1], it is assumed that the integrity of the initial trusted boot code of a TEE is intrinsically guaranteed. Furthermore, OMTP Flexible Trusted Boot requirements and OEM-dependent boot operations require that, during boot time, some services or operations need to be performed in a TEE. Therefore, a minimal set of the TEE capabilities must exist during the general device boot time and, to enable some of these services, a Trusted OS (or some simplified version thereof) may also exist. Such a minimal set of TEE capabilities is referred to here as a Boot time TEE (BTEE).

It is not the current intention of GlobalPlatform to define the capabilities of a BTEE, but it should be clear that all elements involved in that boot action form parts of the TEE from the point of view of its instantiation and protection.

7.2.1.1 Typical Boot Sequence Involving a TEE

Figure 7-3 illustrates two simplified implementation examples of a device starting a TEE platform alongside an REE platform. These are both examples and many variants are possible.

In both examples, the TEE platform boots from the embedded mask boot ROM code inside the SoC. This ROM code will be simple and generally enough to enable moving to the next stage, and in good designs perform a minimal POST.

This ROM code, combined with the computing engine and associated data, form the Initial Root of Trust Component on this example platform.

Single Stage Boot

Single stage boot will then continue execution of the TEE RTE from the on-SoC FLASH; the same process is found in an SE.

Multi Stage Boot

Alternatively, the TEE code in the boot ROM may then load more complex boot stage software components from non-volatile storage, verify them using the enabling TEE assets (for example, hashes stored in the boot ROM or one-time programmable (OTP) fuses), before finally executing them in faster RAM memory.

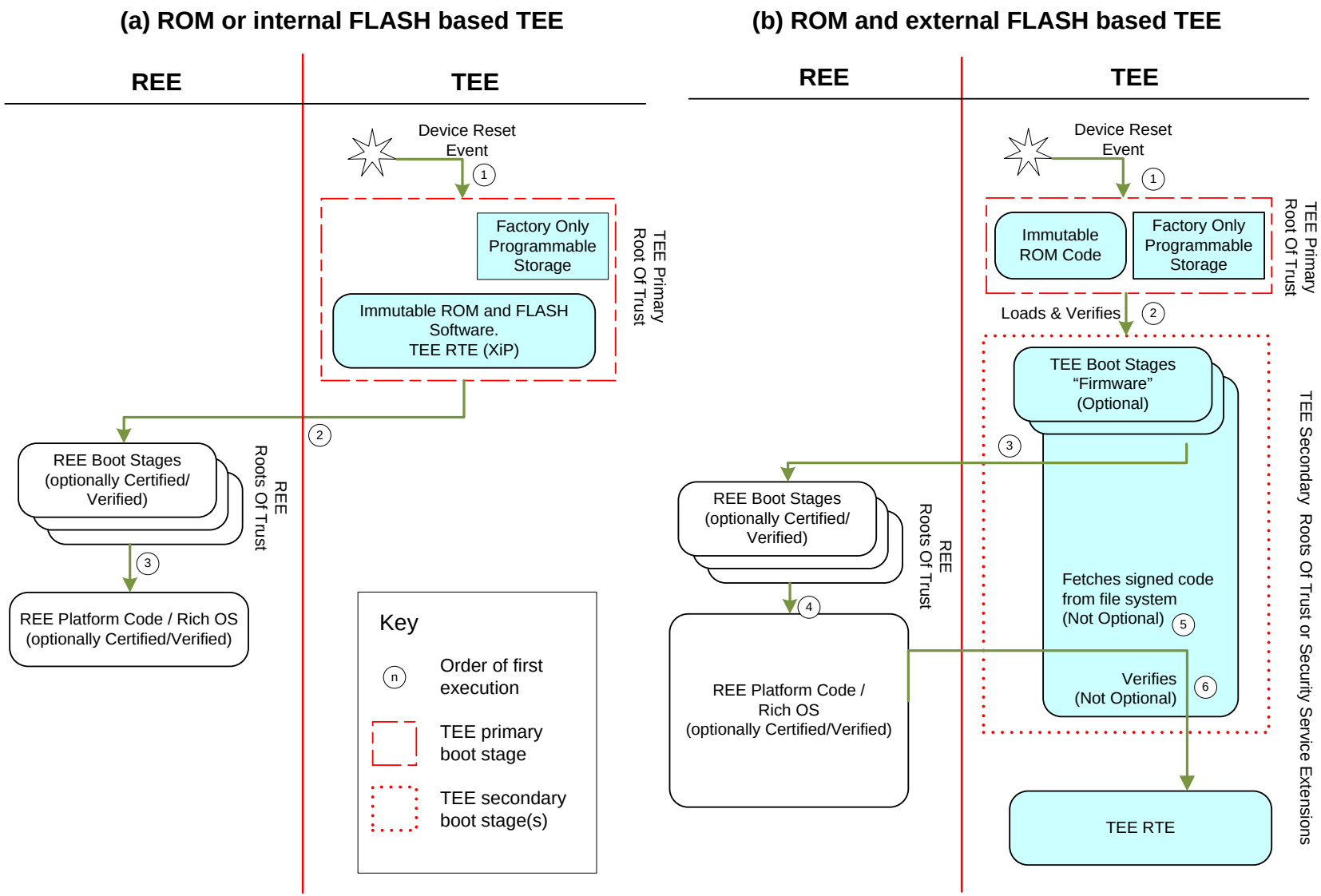
When to Boot Relative to the Rest of the Device

During boot, some TEEs will configure the general host device's isolation settings (e.g. internal data firewalls) and so some TEEs need to boot before other platforms on the SoC, to protect their assets. This "first to boot" is not necessary in TEE platforms that have their own dedicated resources, but even then, it is often desirable to boot the TEE before the REE, as this allows the TEE to provide security services to an REE, enabling REE Secure Boot. While implementations may choose to do otherwise, a GlobalPlatform TEE is only required to offer its services once the REE has completed its own boot sequence.

Failure in TEE Secure Boot

Typically, if any loaded software component verification fails during boot, the normal boot process halts and the device may reboot with a possible error report/indication. In many devices, it will then seek out externally-sourced flash images to fix the problem, while other devices will default to a simplified TEE in ROM, enabling emergency service functionality.

Figure 7-3: Example of Device Boot with TEE Platform



As shown in Figure 7-3 and discussed in the following sections, a TEE MAY go through various phases while it prepares to offer GlobalPlatform compliant functionality. This section covers RoT functionality in relation to this boot sequence and does not cover the RoT functionality exposed by the RTE.

While there are many possible variants, we will cover only the two based on systems shown in Figure 7-3.

7.2.1.2 TEE 1 of Figure 7-3: TEE RTE XiP in ROM or FLASH

This is a system where the TEE RTE and other software are executed from a non-volatile store without blocks being preloaded into some other execution space, such as RAM. This is a sensible design when the processor bus speed matches the access speed of non-volatile store memory, which is common in technology used in SE systems.

If TEE software is running XiP in either of the following:

- ROM code and data used for XiP is intrinsically trusted, or
- FLASH code and data used for XiP is intrinsically trusted once it has passed verification during the process of being written to the FLASH

...then there is no cold boot verification before the execution of any code, as there is no access available for unauthorized modification of the code base. Mechanisms may be provided to perform runtime verification (e.g. automatic memory checksums and more advanced techniques).

As the XiP code is the only code in such a platform on cold boot, it is the first code executed on cold boot. No entity exists before the XiP code's first execution that can provide measurement or verification (with or without record).

In SoC design and code execution methodology terms, this TEE variant may be considered identical to that of an SE, with the only difference being in the runtime environment offered to the applications and the communication connection port, so all the SE application RoT material from Chapter 8 equally applies here.

From the point of view of remote management, the RoT service principles are those of the TEE management system and are described in section 7.2.4.

7.2.1.3 TEE 2 of Figure 7-3: TEE RTE Software Loaded through TEE Secure Boot

From the point of view of "trust", the exact time the GlobalPlatform compliant part of the TEE RTE is loaded makes little difference.

What is critical is that:

- The first code executed on the TEE platform is the first stage of a Chain of Trust that can be shown to validate that the TEE RTE is the intended and correct code to provide the RTE.
- The first stage of the boot chain is intrinsically trusted.
  - As the intrinsically trusted first stage in a boot chain, the use of On-SoC ROM prevents interference with the code reaching the computing engine of the TEE.
- Any additional stages in a boot chain sequence must be validated before execution.
  - These form Extended RoT Components if no record of validation is available and if additional service or a modification of the service is provided by these stages.
  - These may form security service extensions to a RoT, if a record of validation is available from that RoT.

  Because they may be either service extensions or RoTs, they will be referred to below as "Boot Phases".

Boot Phase Services

Each phase of a TEE secure boot chain must provide the following services to the next phase. These primitives are not necessarily exposed as services to other code, but are useful to consider in terms of explaining the functionality of each phase. Only for the iRoTc must these be RoT Security Services rather than service extensions.

The following sections describe how the Root of Trust in the TEE implements the services associated with TEE secure boot. The services are not exposed as explicit services, but are integrated into the functionality offered by the boot phases.

- Confidentiality service
  - Each boot phase will use a confidentiality service rooted in a RoT for Confidentiality in which it stores the secret materials used by the following phase, but prevents access to secrets which later phases are not authorized to access.

    For example, TEEs typically use a hardware unique key (HUK) to device bind and protect non-volatile data. This may have been derived from a common "root" HUK from which other device secrets are derived. It is essential that the root HUK is not exposed to unauthorized entities as this may enable them to derive the HUKs of other entities.
- Identification service
  - Each boot phase must contribute to a RoT for identification that will be exposed through the Trusted OS when the system is in its run time state.

    How a boot phase contributes is implementation dependent. This material is used to generate the `gpd.tee.firmware.*` TEE property set to uniquely identify the boot code used to establish the TEE RTE.

    For example:

    - *A boot phase may provide a temporary service to the next layer to pass on the identity of the parent RoT.*

    or:

    - *A boot phase may provide some information through the platform software build system, which is only available to the TEE RTE. Such information will therefore not be available unless the authentication process has validated boot phases to a point to instigate the TEE RTE.*
- Verification Service
  - Each boot phase must use a service to validate the code and data that is rooted in a RoT for Verification to instigate the following phase.

- Reporting and Identification Service
  - The RoT will expose a reporting service through the TEE RTE.
  - This service will allow discovery of identification of the platform capabilities including security services, along with the unique version relating to the separate elements of the boot chain that lead to the availability of this service.
  - Typically, the reporting will take the form of validated information rather than the digest material of the validation itself.

    For example:

    - *It will report the version of the boot phases rather than hashes of those boot phases.*
      - *See the `gpd.tee.firmware.*` TEE property set.*
    - *It will report the version of the TEE RTE rather than hashes of that code.*
      - *See the `gpd.tee.trustedos.*` TEE property set.*
    - *It will report the availability of applications by responding to access requests for those applications (rather than by returning the hashes of those applications).*
      - *See TEE Client API Specification [TEE Client].*

Each phase of a TEE secure boot chain MAY provide the following service:

- Update Service
  - This may enable the boot phase to update itself.
    - Updates are authorized against key materials held by the boot phase.
  - This may enable the boot phase to install later boot phases when it finds that those phases are currently invalid or unavailable.
  - This may allow a boot phase to update its authorizations such that a change of ownership has occurred and it may then allow different owners to present later boot phases.
    - This would be done be updating signature validation material held by the boot phase being updated.

Note: Code in the TEE RTE or the TEE boot phases may not be updated without reapplying for compliance and certification. Such updates are normally only done to add additional compliant functionality. For example, as no hardware changes may be required, a TEE RTE may be updated to add support for GlobalPlatform TEE Sockets API [TEE Sockets] even though it was originally released without this functionality.

7.2.2 Boot of TEE RTE

A TEE using XiP in ROM and FLASH is identical to an SE and so inherits the same RoT principles as an SE (see Chapter 8).

The following principles only apply to a TEE that boots by loading code from non-volatile storage to be executed in RAM.

iRoTc Initialization

The Initial Root of Trust Component is a computing engine, code, (optionally) keys to validate later boot phases, (optionally) keys to validate remote entities, and (optionally) chip-unique root secrets, that shall be installed with all its static data configured, during the manufacturing process.

Manufacturing Process

The manufacturing process shall be protected with the necessary material to protect the chip design, code, and keys during the fabrication.

Example: The TEE RTE code signing private key is stored in an HSM and it is not possible to copy or replace the private key during or after the manufacturing process.

TEE RTE

TEE RTE shall boot through a chain of RoT verifications rooted in components in a Chain of Trust for Verification or rooted directly in a Root of Trust for Verification. This RoT will provide the services listed in the following principles.

Code Used During the Boot Phases

Each boot phase shall provide a verification service[3] and an authorization service through which it shall perform verification and authorization checking of the next phase code. If verification and authorization is successful, the current boot phase shall allow the execution of the next phase code.

That is, to start a following boot phase, that boot phase will require various code to execute. The instigating phase must verify the integrity and authenticity of that code and also check that it has been authorized (typically by comparing that code's cryptographic signature against keys held in the current module).

Data Used During Boot Phase Initialization

Each boot phase shall provide a verification service[3] and an authorization service through which it shall perform verification and authorization checking of the next phase initialization data before moving to that next phase.

That is, to start a following boot phase, that boot phase will require various material. The instigating phase must verify the integrity and authenticity of that material.

In this case, data includes keys, digests, hashes, checksums, etc. – that is, all material that is not code.

[3] If the previous phase doesn't provide measurement, this phase will be a RoT for Verification.

Failure of Verification or Authorization of Next Boot Phase Material

If the verification or authorization of material for the next boot phase fails:

- The platform SHALL go into a mode that does not offer TEE services from that following phase.
- The current boot phase may still offer services from itself and previous phases, and may seek alternate fallback boot phases (that must themselves be correctly verified) if the initial boot sequence has failed.

Data Used During Boot Phase but Not During Initialization

Each boot phase may instigate a verification service[4] and authorization service through which it may perform verification or authorization checking of data before use.

Once a boot phase has started, it fetches further materials to enable services it offers. That material may be verified and/or checked for authorization before use.

Boot Phase and Confidentiality Service

Each boot phase shall provide a confidentiality service[4] for secret data provided by itself or its parents and used in the following phases.

Boot Phase and Identification

Each boot phase shall provide (directly or implicitly) an information service[4] for identification of the overall boot material.

RoT and Update

Any RoT may offer RoT Security Service for Update. Any component in a Bootstrapped Root of Trust may offer RoT Security Service for Update.

[4] If the previous phase doesn't provide measurement, this phase will be a RoT for Verification.

7.2.3 #2 RoT Security Services Offered to a Trusted Application on the TEE Platform

A TEE that has booted its RTE will offer Roots of Trust services to other software on the device. These services may be offered from Non-Bootstrapped RoT or Bootstrapped Root of Trust Components depending on how the TEE was implemented (see section 7.2.1).

That “other software” may be resident in the REE, or may be Trusted Applications, drivers, etc., executing in the TEE itself.

The TEE RTE is only required to offer fully GlobalPlatform-compliant services once the whole system has reached its run time state (see [TEE Sys Arch]); however, subsets of these services and proprietary services may be offered before that point in the device boot cycle.

The following services are not exposed as explicit services, but are integrated into the functionality offered by the Trusted OS.

7.2.3.1 Authentication Service

The TEE RTE offers limited levels of authentication guarantee of the software entity communicating to the TA from the REE.

The TEE RTE offers authentication guarantee of the software entity communicating to the TA from the same TEE.

In addition, where the appropriate GlobalPlatform API specifications have been implemented, a TA may use the TEE to hold key material in shielded locations for the purpose of authentication. This material, in combination with security protocols provided by the TEE RTE, may be used to authenticate software entities on other platforms.

- TLS protocols are available to authenticate entities available over TCP/IP and UDP/IP.
- GlobalPlatform SCP protocols are available to authenticate entities in Secure Elements connected to the device and linked to the TEE.

7.2.3.2 Confidentiality Service

The Confidentiality Service maintains shielded locations in volatile and non-volatile memory. The protection of the shielded locations is vendor-implementation-specific. Confidentiality and access control are provided through isolation of the applications in the runtime environment. For example:

- The firewall of the TEE RTE prevents one TA from accessing the objects of another TA.
- The firewall of the TEE RTE prevents anything outside the TEE from accessing assets inside the TEE, and the firewalls managed by the Trusted OS prevent one TA from accessing objects that are assets of another TA.

7.2.3.3 Identification Service

The Identification Service maintains a shielded location that prevents modification.

This service provides a set of device specific certificates that may be verified by an off-card entity communicating with the TA.

It is exposed through the GlobalPlatform TEE properties:

`gpd.tee.trustedos.implementation.binaryversion`

`gpd.tee.firmware.implementation.binaryversion`

In addition, the Root of Trust for Identification maintains the integrity of a platform-unique ID number (`gpd.tee.deviceID`). This ID number provides an index that may be used by a remote entity to retrieve material to enable a per platform secure channel or other similar relationships to be established.

7.2.3.4 Authorization Service

In the TEE, there may be multiple Trusted Applications, each one with its own keys and data. As a general rule, the Trusted OS ensures that a Trusted Application can only access its own stored keys and data and not those of another application. Access to stored Trusted Application keys and data is therefore subject to authorization in that only the owning application can access the data.

In addition, there are authorization rules regarding the interaction of different instances of the same application to stored application keys and data.

The Authorization Service maintains a shielded location to store the rules regarding access to the restricted feature as described above. It guarantees that the rules cannot be changed without authorization and verifies that the rule is applied when an application tries to access a restricted resource.

7.2.3.5 Verification Service

All items extracted from Trusted Storage are verified upon extraction.

All TA code and data (including keys) are stored in Trusted Storage while not in use.

7.2.4 #3 RoT Security Services Offered to Remote Entities by the TEE Platform

A TEE provides a remotely managed environment to control the resident TAs.

The remote management entity (TAM or TSM) acts through local endpoints in the TEE platform called Security Domains.

Each Security Domain has an assigned owner.

- Commands from that owner are authenticated, validated (and optionally decrypted) against keys installed in the SD.
- Replies to those commands are signed (and optionally encrypted) by keys held in the SD.

There are three general sets of operations that may be available to an SD if it has the appropriate rights.

- The SD may store secrets (data or keys) in itself or a child SD or TA.
- The SD may install, remove, or update a child TA code or SD presence.
- The SD may perform certain TEE management operations.

Only correctly authorized and authenticated remote entities may perform particular management operations. Those operations are further restricted by a permission hierarchy structure; e.g. an SD that cannot perform TEE management operations cannot create another SD that can perform TEE management operations.

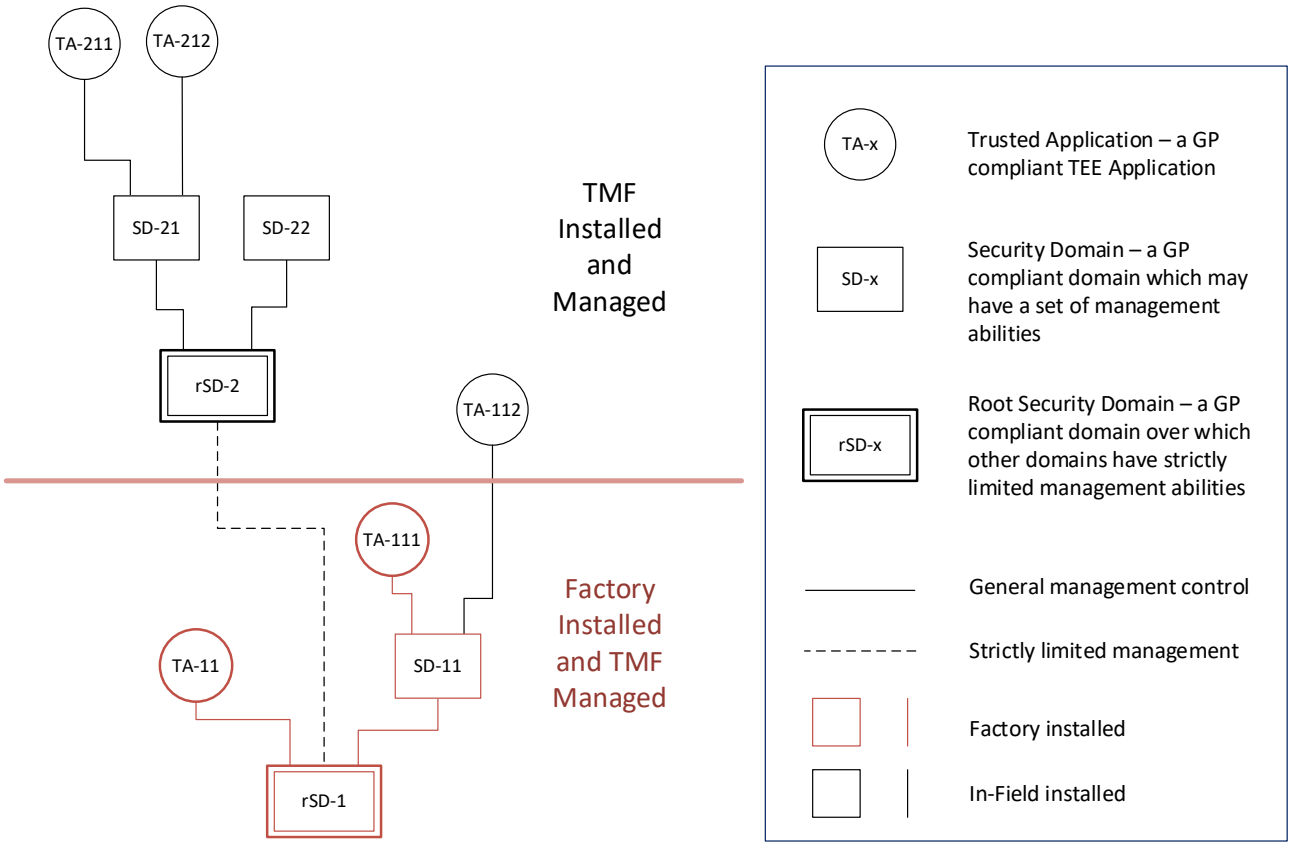
From the external view of a managing remote entity (such as a management server), a TEE platform appears as one of two classes of management nodes.

- The first class is known as a root SD (rSD).
  - A root SD is a domain that restricts the operations necessary for the management of TEE services (creation, update, blocking, removal, etc.) mainly to its owner remote entity and to any remote entity responsible for one of the root SD's ordinary sub-domains.
  - A TEE may have many rSDs and they do not all have to be installed in the factory.
- The second class is an ordinary SD.
  - This has a hierarchy of parent SDs that may potentially perform management operations on it or its children.
  - An ordinary SD can have another SD as its parent and they may be installed in the factory.

Note: This structure is NOT identical to that found in an SE but is very similar in many aspects.

The following diagram shows an example of management hierarchy in a TEE.

Figure 7-4: Example of Security Domain Structure



7.2.4.1 RoT Principles of TEE Management Framework (TMF) Services of a TEE RTE

Because the [TMF] requirements are a toolbox of operations, with no particular operations required to be supported by a particular TEE, only the authentication and identification services are mandatory for a TMF-supporting TEE to implement. All the other services are optional depending on the TMF operations the TEE supports.

The following services are not exposed as explicit services, but are integrated into the functionality offered by the TEE TMF software or equivalent TEE OTA management software.

Identification Service

A TEE RTE providing TMF facilities shall offer a RoT for Information to external parties. Using the TMF mechanism, this may also be available through unsecured enquiry to an SD.

A remote entity may enquire of, or otherwise apply (for example, as a constraint), the TEE platform identity (`gpd.tee.deviceID`) such that it can then use platform-specific management keys to communicate to the platform.

Reporting Service

A TEE RTE providing TMF facilities may offer a RoT for Reporting to external parties. If available through the TMF mechanism, this shall be available through an SD.

A remote entity can use the TEE Audit operations to discover the available capabilities of a TEE. If this is done through a TMF security layer, then the capability statement will be signed by the security layer key material.

Authentication Service

A TEE RTE providing TMF facilities shall offer services based on a RoT for Authentication to TMF commands from third parties. Using the TMF mechanism, these services shall be available through an SD.

The authorization of all SD commands to the TMF shall be validated. This requires authenticating their relationship with the Security Domain, and may extend to allowing the rights of the Security Domain to be examined through TMF interfaces.

Update Service

A TEE RTE providing TMF facilities may offer update services. If available through the TMF mechanism, this may be available through a correctly privileged SD.

The update service allows an authenticated and authorized SD command to update key material, code, and data stored in shielded locations.

Confidentiality Service

A TEE RTE providing TMF facilities shall provide service based on a RoT for Confidentiality to SDs, to provide the SD guaranteed confidentiality in data it stores and transmits.

The confidentiality service maintains shielded locations for the purpose of storing secret critical security parameters, such as symmetric keys and private asymmetric keys.

Integrity Service

A TEE RTE providing TMF facilities shall provide service based on a RoT for Integrity to SDs, to enable the SD to guarantee integrity in the code and data it stores.

The Integrity Service maintains shielded locations for the purpose of storing and protecting the integrity of non-secret code and data.

7.2.5 #4 RoT Capabilities that Applications in the TEE Platform May Offer to Other Platforms

A Trusted Application or RTE service in a TEE may offer facilities to other platforms to enable their RoTs or other functionality.

For example: A TA may provide the functionality of a TCG TPM. That functionality might be used by the REE in concert with a computing engine and boot code to create a RoT for the REE platform.

There are no specific RoT principles for this section, as these will depend on the component being offered by the TEE.

- GlobalPlatform TEEs offer no specific RoT Security Services to other platforms.
- GlobalPlatform TEEs enable writers to create TAs which may offer a range of RoT Security Services as seen fit by the TA creator.

8 Detailed Secure Element Use Case for Root of Trust

8.1 Factory Creation, Installation, and Life Cycle of Secure Element

This section describes events in Secure Element creation.

This section was extracted from the [Security IC Platform] document.

Figure 8-1: Secure Element Life Cycle

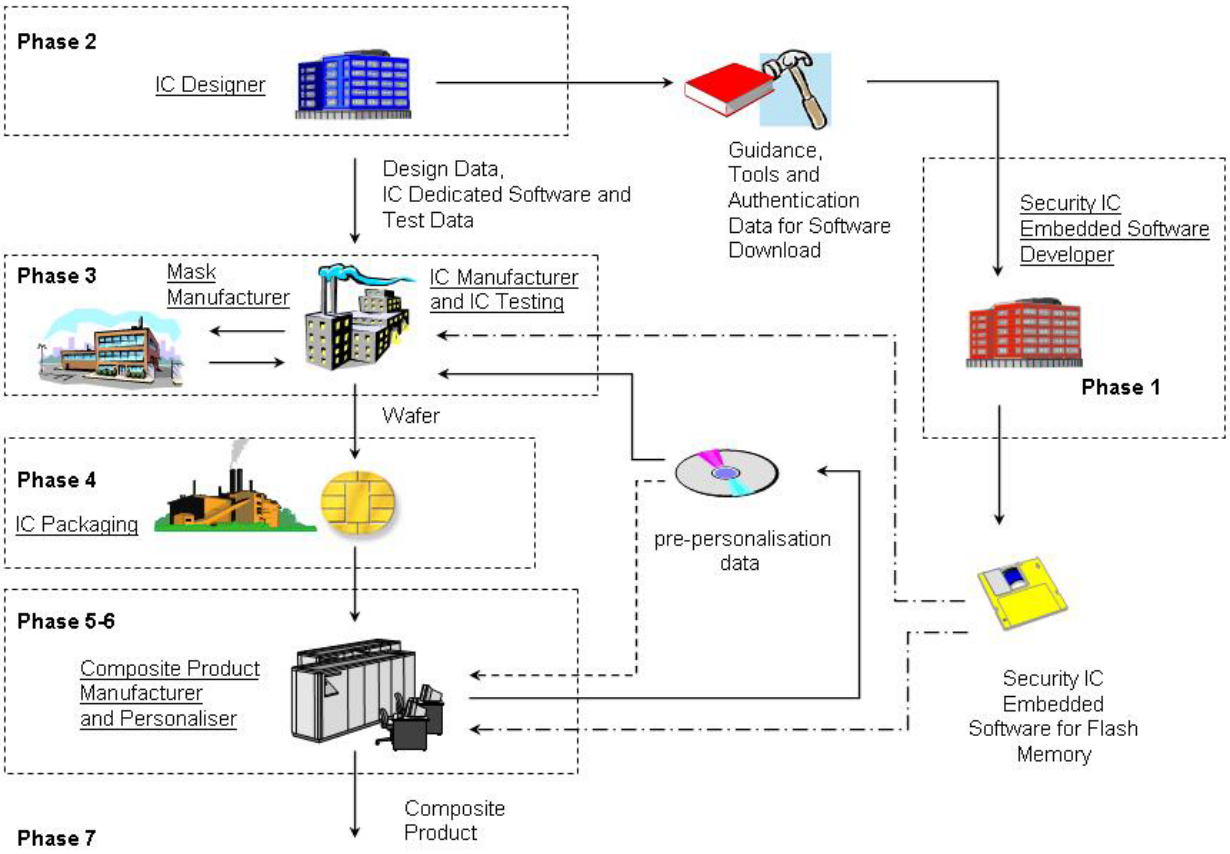


Table 8-1: Life Cycle of SE

Phases	Actors
Phase 1	The **Security IC (or SE) Embedded Software Developer** is in charge of: • The Security IC (or SE) embedded software development, and • The specification of IC (or SE) pre-personalization requirements, though the actual data for IC (or SE) pre-personalization come from Phase 6 (or Phase 4 or 5).
Phase 2	The **IC (or SE) Designer** • Designs the IC (or SE), • Develops IC (or SE) Dedicated Software, • Provides information, software, and tools to the Security IC (or SE) Embedded Software Developer, and • Receives the Security IC (or SE) embedded software from the developer, through trusted delivery and verification procedures. From the IC (or SE) design, IC (or SE) Dedicated Software and Security IC (or SE) Embedded Software, the **IC Designer**: • Constructs the Security IC (or SE) database necessary for the IC (or SE) photomask fabrication.
Phase 3	The **IC Manufacturer** is responsible for: • Producing the IC (or SE) through three main steps: IC (or SE) manufacturing, IC (or SE) testing, and IC (or SE) pre-personalization. The **IC (or SE) Mask Manufacturer**: • Generates the photomasks for the IC (or SE) manufacturing based upon an output from the Security IC (or SE) database.
Phase 4	The **IC (or SE) Packaging Manufacturer** is responsible for: • The IC (or SE) packaging and testing.
Phase 5	The **Composite Product Manufacturer** is responsible for: • The Security IC (or SE) product finishing process and testing.
Phase 6	The **Personalizer** is responsible for: • The Security IC (or SE) personalization and final tests.
Phase 7	The **Security IC (or SE) Issuer** is responsible for: • The Security IC (or SE) product delivery to the Security IC (or SE) consumer, and the end of life process.

If the TOE comprises programmable non-volatile memory, the Security IC (or SE) Embedded Software may be loaded onto the chip in phases 3, 4, 5, or 6.

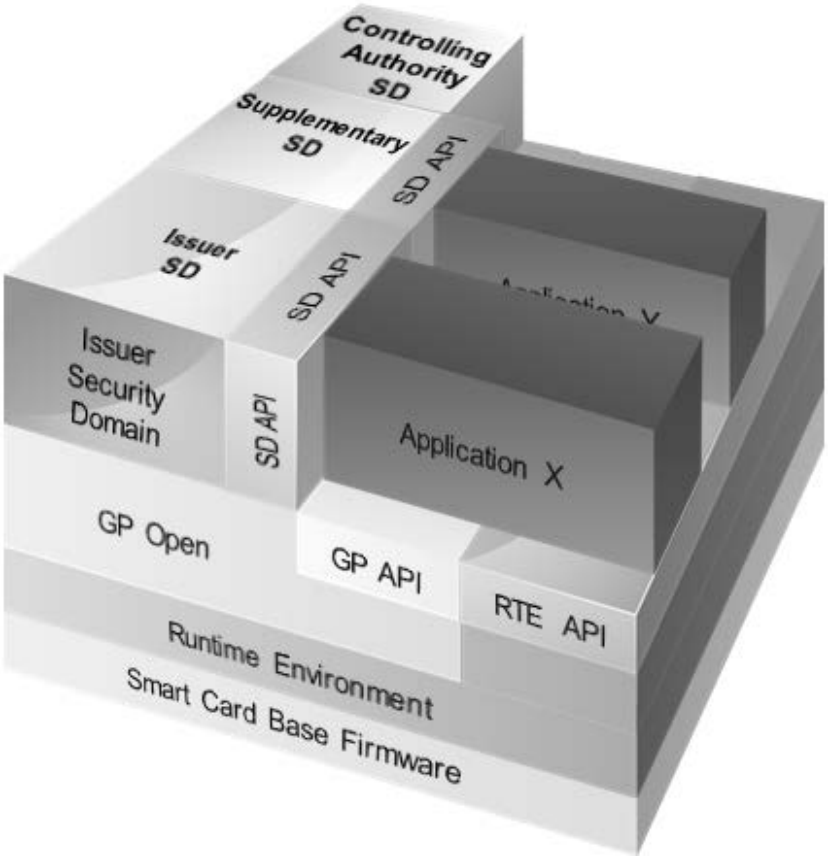
8.2 Root of Trust and Secure Element

The Secure Element as a removable form factor like the UICC technically enforces the business relationship between the end user of the mobile device and the telecom network provider. This is done independently of the mobile device. In this context, it serves as a true root of trust both for the telecom network provider and for the end user of the mobile device and does not require RoT Security Services provided by the mobile device. This modus operandi can be extended to additional secure elements with the same form factor or in the form of a microSD.

On the one hand, third parties such as banks, public transport operators, and government agencies can provide these secure elements and use the root of trust that they own on them for applications such as home banking, ticketing or subscription for public transport, or in a legal/administration ecosystem. On the other hand, the telecom network provider that owns the root of trust of the UICC can offer to serve as root of trust on behalf of third parties that wish to deploy applications on this UICC. The SE owner does this by the installation of an SSD, the writing of its secure channel keys set, and the extradition of the SSD [GPCS]. The stakeholders (SE owner and third party) rely on the SE Initial Root of Trust Component for authorization and confidentiality in these operations. Moreover, the confidentiality of the key set of the SE owner (in the ISD) is a condition for ensuring the confidentiality of the key set of the third party. The SSD is part of a Chain of Trust by extending the security services.

An SE can also be used in a standalone fashion as a banking card, ID card, public transport card, or access token. In that case, it does not serve as a root of trust for a mobile device but rather in the context of applications that involve a terminal or external “reader” such as a payment terminal, ATM, or access gate.

Figure 8-2: Secure Element Architecture Diagram



The main components of the GlobalPlatform SE chip architecture are summarized below.

- System code (or Smart Card Base Firmware) and secure hardware: The system code is the part of the code that corresponds to the lower layer services such as memory management, access to (communication) peripherals, etc. It contributes to the security of the product by providing a mechanism to continuously verify that the code is executing correctly and to confirm the integrity of the parameters, data, and keys being used or processed. If an anomaly is detected, this layer takes appropriate action (e.g. aborting execution and becoming mute or incrementing some ratification counter). In providing these security services, the system code makes use of hardware dedicated to security (e.g. Active Shield, monitoring of environmental parameters, protection against faults, memory scrambling, or runtime execution environment monitoring).
- GlobalPlatform Environment (OPEN):
  - Performs application loading and related Card Content management.
  - Manages the installation of applications loaded to the card.
  - Dispatches APDU commands to the currently selected application, and sets the application referenced in the SELECT command to be the selected application.
  - Manages the internal registry as an information resource. This registry contains information for managing the Card Executable Load Files, applications, and Security Domain associations and privileges.
- Run Time Environment (RTE): Java Card Runtime Environment (JCRE) or MULTOS. The Run Time Environment is responsible for providing firewalls between all applications installed on the SE, implying that an application can by default access only its own data and keys. There are mechanisms to enable an application to make use of services offered by another application on the platform, but this is governed by the OPEN and subject to authorization.
- Secure channel protocol combining dedicated code and the associated key set. It provides cryptographic services such as command authorization and authentication, response authentication, and both command and response encryption, with a dedicated mechanism (and key) for key encryption.
- Issuer Security Domain (ISD) has its Secure Channel Protocol available. The ISD is present on the SE since the beginning of its “life”. It also has a command set allowing it to manage its data and to load and install other SDs or applications.
- Supplementary Security Domain (SSD) has its Secure Channel Protocol. The SSD and its initial Secure Channel key set are installed secured by the Secure Channel Protocol and with the ISD Secure Channel key set. After personalization, it relies on its own Secure Channel key set for installing the remainder of its data. This may include rolling its Secure Channel key set to another one. Its functionality is similar to that of the ISD but it has a different owner and somewhat lesser privileges.
- Cardholder Verification Method (CVM) application (called Global PIN if there is exactly one).

The components listed above contain the code and hardware mechanisms which the SE utilizes to provide RoT Security Services; the following sections describe and provide examples of these services. The code in this set of components is immutable, thus meeting the immutability requirement for Roots of Trust. The data and key set associated with the ISD and the CASD are mutable under authorization (except their identifications) and they are set during the manufacturing process.

- The SSD is in the immutable memory and is set during the manufacturing process. The key sets and data of the SSDs are in the mutable memory and they are provisioned after the manufacturing process. For this reason, SSDs are not included as Roots of Trust for SE. However, through the proper verification of this code, including runtime verification, one can consider them to be in a Chain of Trust.
- Additionally, applet owners can load applets after the manufacturing process by using the security service delivered by the Security Domain.

The Non-Bootstrapped Root of Trust in an SE is composed of a subset of the following components:

- System code and Secure Element hardware protection
- OPEN
- Runtime Environment
- ISD + Secure Channel Protocol + related data (data + keys set)

The Root of Trust implements the following security services:

- Authentication Service
- Confidentiality Service
- Identification Service
- Authorization Service
- Reporting Service
- Update Service
- Verification Service

The following sections describe how the Root of Trust in the Secure Element implements the services.

The security services described in this section are provided only by [GPCS]. This section does not cover additional security services or service adaptations provided by an application/applet (Validated Module) running on the platform.

All the code in the Secure Element that composes the RoT is in an immutable memory. The data (including keys) associated with the RoT may be updated under authorization. Each Security Domain (after extradition) has its own key set but the code that implements the Secure Channel is shared between all Security Domains. The Secure Channel makes use of several keys for several purposes (S-ENC, S-MAC, DEK).

In the remainder of this chapter, the term MAC is used as a generic term to cover all mechanisms implemented by a secure channel; see [GPCS] and Amendments.

Requirement 8-1: SE Tamper Resistant

> The SE SHALL provide a shielded location. This shielded location SHALL be tamper resistant. The implementation is a combination of hardware and software and its implementation is vendor specific.

8.3 Authentication Service

8.3.1 Cardholder Verification Method

The Authentication Service maintains at least one shielded location to be used to protect the reference PIN (CVM or global PIN) to be used for end-user authentication. The Authentication Service maintains an interface to the CVM application that:

- Maintains the integrity of the CVM management data, including the PIN value.
- Protects the PIN value against dictionary attacks.
- Protects the PIN value against disclosure.
- Restricts its use for user authentication only.

8.3.2 Secure Channel Protocol

The Authentication Service maintains a shielded location to be used to protect the key set used for the Secure Channel Protocol for on-card and off-card authentication. The Root of Trust for Authentication maintains an interface to the Secure Channel Protocol that:

- Maintains the integrity of the key set (S-MAC key used to generate the Secure Channel C-MAC sessions key).
- Protects the key values against several types of attack (fault injection, simple and differential power analysis, etc.).
- Protects the key values against disclosure.
- Restricts the use of each key only to applications that have the right to access it.

8.4 Confidentiality Service

8.4.1 Application Firewall

The Confidentiality Service maintains shielded locations in non-volatile memory. The protection of the shielded locations is vendor-implementation-specific. Confidentiality and access control are provided through isolation of the applications in the runtime environment. For example, the firewall of the JCRE prevents each application from accessing the objects of another application.

8.4.2 Secure Channel Protocol

The Confidentiality Service maintains a shielded location for protecting the key set used for the Secure Channel Protocol for on-card and off-card encryption/decryption. The Root of Trust for Confidentiality maintains an interface to the Secure Channel Protocol that:

- Maintains the integrity of the key set (S-ENC key used to generate the Secure Channel encryption session key and the DEK key used to generate the Secure Channel data encryption session key).
- Protects the key values against several types of attacks (fault injection, simple and differential power analysis, etc.).
- Protects the key values against disclosure.
- Restricts the use of each key to only the applications that have the right to access it.

When a cryptographic key must be transferred from an off-card entity to an SE, the key owner (SE owner or third party with an SSD on the SE) can rely on the Root of Trust for Confidentiality to protect and use the Secure Channel Protocol in an authorized fashion, ensuring authorization and integrity by means of encryption and MAC, to securely transfer and store the value in a shielded location maintained by the Root of Trust of Confidentiality. The Root of Trust for Confidentiality maintains an interface for secure channel protocols to store the secure channel keys of an SD that it uses to perform command and response authentication and encryption.

8.5 Identification Service

The Identification Service maintains a shielded location as non-volatile memory. This service provides a signature (a MAC called R-MAC) that may be verified by an off-card entity. The signature is computed over specific data requested by the off-card entity with the key set value of the owner of the Root of Trust or a third party with an SSD on the SE.

The Root of Trust for Identification maintains the integrity of an ID number, storing it in a shielded location and preventing any modification. This ID number is used as common practice to have a unique secret key per chip. This value is used by the off-card entity to find the key set corresponding to a dedicated Secure Element in its HSM. The applications running in the Run Time Environment cannot access this ID number.

8.6 Authorization Service

There are multiple applications (applet instances) in the SE, each with its own keys and data. As a general rule, the OPEN and RTE ensure that an application can only access its own keys and data and not those of another application. Access to application keys and data is hence subject to authorization in that only the owning application can access the data. There are some exceptions to this.

The first exception is the GlobalPlatform API. It allows applications to make use of services such as:

- Cardholder verification using the SE unique PIN
- Secure Channel services, offered by the SD associated with the application
- Card locking and termination
- Contactless parameters (OPEN authorizes only the application with the self-contactless privilege to modify the RF parameter, as it may affect other applications)

The OPEN and RTE ensure that access to these services (and their underlying resources, such as the PIN and the secure channel keys) pass via this API, so this authorization is checking. Moreover, for certain services, the application must have certain privileges (e.g. the right to lock the card) that are maintained by the OPEN and RTE and are attributed when the application is installed by means of a Security Domain. When making use of the API, the OPEN and RTE will verify that the application has the corresponding privilege, which is also part of authorization. Moreover, the application will not be able to decipher the enciphered key value in a PUT KEY command if there is no secure channel session going on AND the PUT KEY command had no valid MAC.

The second exception is the mechanism of shareable interface. This is a mechanism defined at the Java Card level and strictly speaking not a GlobalPlatform concept. However, the GlobalPlatform specifications use it, for instance for the interface to the PIN object (CVM). The shareable interface is a method for an application to access services offered by another application. An applet provider can make use of the shareable interface concept to support offering services to other applications. The OPEN + RTE will ensure that access to such services (and their underlying resources) pass via the shareable interface, so this is authorization. A typical example of such a service is a stored value balance (electronic value) that may be debited or credited via different protocols supported by different applications, but that must be common.

The Authorization Service maintains a shielded location to store the rules regarding access to the restricted feature as described above. It guarantees that rules for the access control of specified resources cannot be changed without authorization and verifies that the appropriate rule is followed before granting an application access to a restricted resource.

8.7 Reporting Service

The Reporting Service maintains a shielded location to protect the key set used for the Secure Channel Protocol between card and issuer (or third party) that provided a signature. The Root of Trust for Reporting maintains an interface to the Secure Channel Protocol that:

- Maintains the integrity of the key set (S-MAC key used to generate the Secure Channel R-MAC session key).
- Protects the key values against several types of attacks (fault injection, simple or differential power analysis, etc.).
- Protects the key value against disclosure.
- Restricts use of each key to only the applications that have the right to access it.
- Limits use of keys to only those suitable for reporting which are available to the using application.

In addition to maintaining the shielded location, the Reporting Service provides a mechanism to sign data. The Issuer Security Domain provides access to its non-secret data and to some information from the internal registry. With the Secure Channel Protocol (with the R-MAC key), it can sign the response that it provides.

8.8 Update Service

The Update Service provides a mechanism to update the contents of the shielded location where a key set is stored. This mechanism may be performed only under an authorization. The authorization is provided by a correct signature (MAC).

This mechanism is used when rolling the Secure Channel Keys in an SD when the ownership of the SD is transferred to another party.

8.9 Verification Service

8.9.1 Loading Information

The Verification Service maintains a shielded location to protect a key set that is used to verify the integrity and the authenticity of the applet code, data, and keys loaded in the Secure Element. This mechanism is provided by the verification of a MAC with the Secure Channel Protocol of an SD with the key set value that is stored in the shielded location.

Whenever the Issuer wants to perform an action that changes the configuration, data, or keys of the SE, it sends a sequence of commands. Such actions include, but are not limited to:

- Loading of applet code
- Installation of an application (applet instance)
- Writing of a key (put key). This can be a secure channel key, but also an application-level key in an application.
- Card life cycle state transitions (locking, unlocking, or terminating the SE)
- Update of SD data elements such as its ID

For all such actions, the SE ensures authorization by means of a secure channel. The commands will have a MAC and the RoT for Verification will verify the MAC before performing the action.

8.9.2 Runtime Verification

The Verification Service provides a mechanism to verify continuously that code is executing correctly and to confirm the integrity of the parameters, data, and keys being used or processed. This is done by a combination of hardware mechanisms (e.g. redundancy in storage, fault detectors, or duplication of execution) and system code including software mechanisms such as repeated execution, etc. As soon as the SE detects an anomaly, it takes appropriate action, such as aborting execution and becoming mute or incrementing some ratification counter. This is not specified in the GlobalPlatform specifications, but may be imposed in Common Criteria documents such as Protection Profiles and specified in Security Targets.

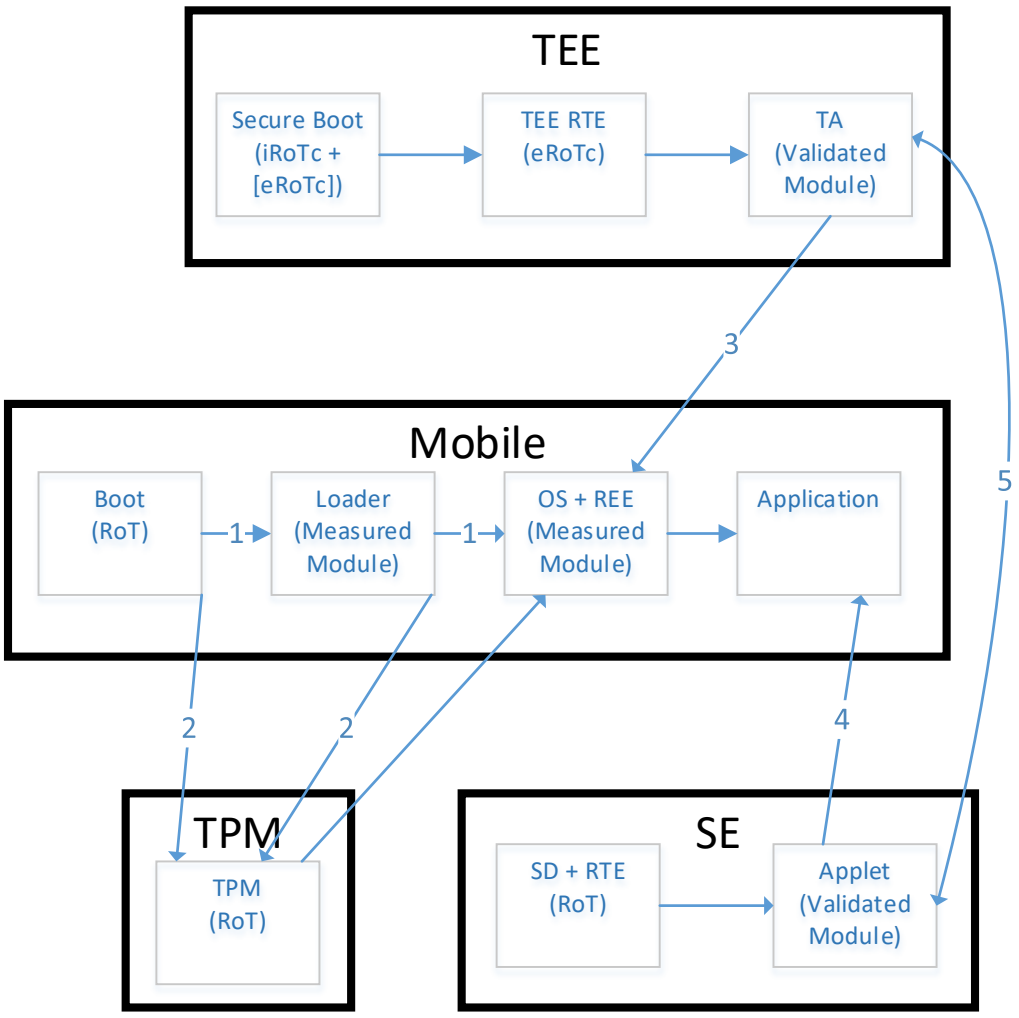
9 Multi-Platform Use Cases for RoTs

A device such as a mobile phone might host several platforms that can provide security services from one to another and vice versa. The implementation of the RoTs and the Security Services provided to another platform are implementation dependent.

The following diagram shows an example of a mobile environment that measured its boot phase and stored the measurements into a TPM component.

- In Step 1, when the OS starts, the boot phase is measured by its platform's RoT.
- In Step 2, the OS can use the RoT for Verification from the TPM platform that acts as an sRoT for the OS.
- In this environment, in Step 3, a TEE is also present and some TA applications may provide security services to the OS or to an application running on the OS.
- In Step 4, the TEE platform will be an sRoT for the OS (or the application). An SE might be equivalent by providing security services to the OS.
- In Step 5, the SE can also provide some security services to the TEE and vice versa (5).

Figure 9-1: Example of Multiple RoTs Platform in Device



9.1 Contactless Transaction with TEE and SE

This use case describes the RoTs behind the interaction between TEE and SE when a TEE TA is providing Trusted User Interface (TUI) services to an SE application for a payment transaction.

9.1.1 Preparation

First, both run time environments must be made operational after power-up of the device.

See the various chapters for how the SE and TEE environments built their Chains of Trust.

Both the Applet and the TA are validated for Authorization and Integrity during the OTA installation phase by the pRoT of the relevant platform.

They are also validated for integrity before or during execution. Again, this is performed by the pRoT of the platform.

They can then be thought of as part of the Chains of Trust for the services they then may offer.

Both the application and the TA are personalized with material such that they can perform mutual identification (for example, they may be personalized with keys to enable them to bring up an SCP link). Such personalization will be done through the GlobalPlatform Security Domain Management mechanisms in both environments, and is therefore bound to the RoT services for that mechanism, as described in previous chapters.

9.1.2 Performing a Contactless Transaction

1. An external NFC source causes the NFC controller event, which triggers an REE application.

   This start transaction event is received by the SE application.

   The SE establishes secure communication with the NFC external source.

   This requires a pRoT in the SE and the External Source.

   Those pRoTs must offer authentication services to establish the secure communication link.

2. The SE receives transaction details.
3. The SE application establishes (via a request through the REE Application) a secure link to the Trusted Application.

   RTEs are providing a number of RoT services to the TA and application at this stage.

   The TA and the application are themselves not RoTs, as their presence is vouched for by the SE or TEE RoT services.

   The secure channel is ultimately based on each platform's RoT, and provides applet and TA endpoint authentication and confidentiality.

4. The SE requests the TA to authenticate the end user and confirm the transaction from that end user.

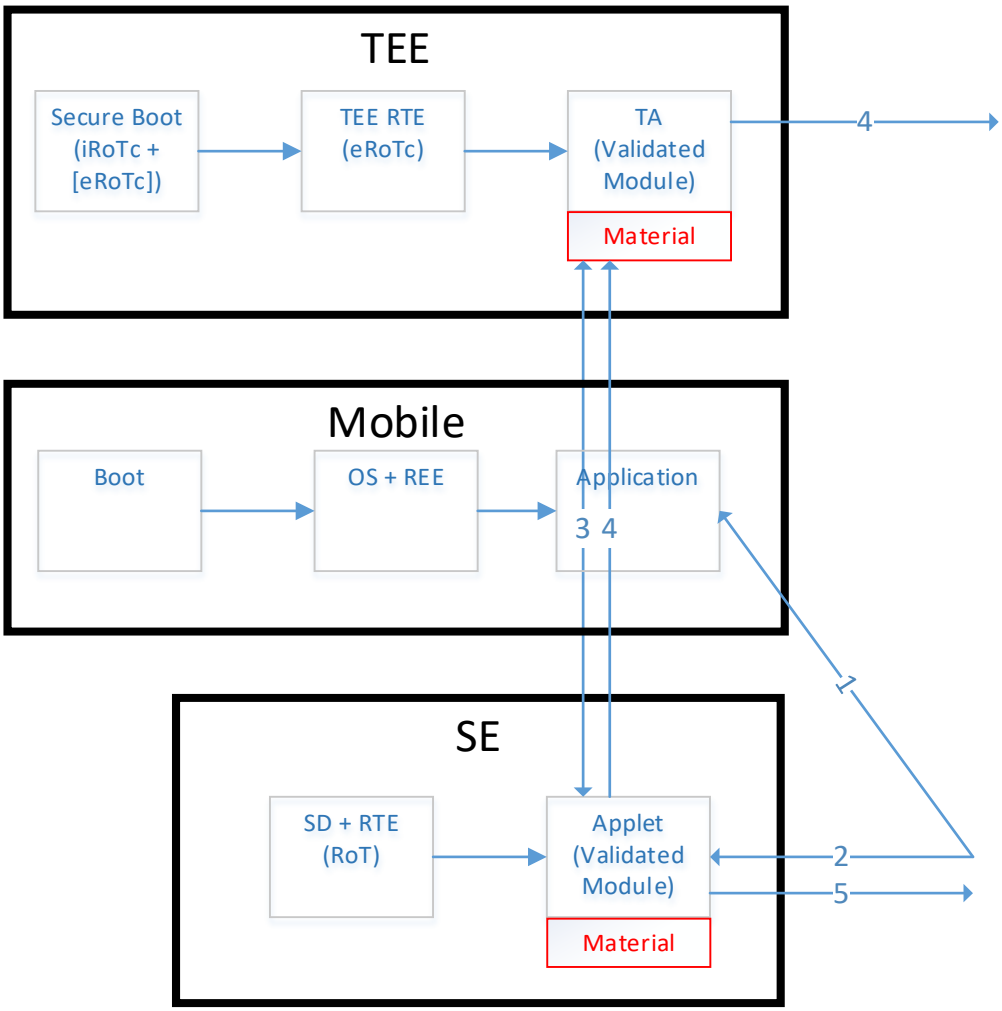
   The TA is passing on material based on a number of RoT services to the application at this stage.

   End-user authentication and any other services are offered by the TEE to the TA whose installation is not reportable through a RoT service.

5. The SE then passes back a reply to the external NFC source.

The following figure illustrates the transaction described above.

Figure 9-2: Contactless Transaction with TEE and SE



9.2 An REE Interacting with a TPM on TEE or SE

This section does not go into depth, as it is assumed that the reader understands the methods by which an REE may make use of a TPM, and the white paper mentioned below explains how a TEE can host a TPM.

In the case of an SE, the same paper can be considered, but with TEE replaced by SE, and Trusted Application replaced by the SE application.

For a description of this use case, see the [TCG Whitepaper].

If a TPM is instantiated as a TA or Applet, then that TPM will be a Validated Module.

Annex A Detailed HSM Use Cases for Roots of Trust

As explained in Requirement 6-1, GlobalPlatform Secure Components are required to protect the keys and their injection during the manufacturing process. A common practice is to use a Hardware Security Module (HSM) to protect the keys. HSMs are dedicated systems that physically and logically secure cryptographic keys and cryptographic processing.

Functions supported by HSMs include:

- Life cycle management of cryptographic keys used to lock and unlock access to digitized information. Remember that the privacy strength of encrypted information is determined by the sophistication of the encryption algorithm and the security of the cryptographic keys. The most sophisticated encryption algorithm is compromised by weak cryptographic key security. Life cycle management of cryptographic keys includes generation, distribution, rotation, storage, termination, and archive.
- Cryptographic processing, which produces the dual benefits of isolating and offloading cryptographic processing from application servers.

In use since the early 1990s, HSMs are available in two forms:

- Standalone network-attached appliances
- Hardware cards that plug into existing network-attached systems

HSMs are fully contained solutions for generating high quality random data, cryptographic processing, key generation, and key storage. As purpose-built appliances, they automatically include the hardware and firmware necessary for these functions in an integrated package. Physical and logical protection of the appliance is supported by a tamper resistant/evident shell; protection from logical threats, depending on the vendor's products, is supported by integrated firewall and intrusion prevention defenses. Some HSM vendors also include integrated support for two-factor authentication. Security certification is typically pursued by HSM vendors and positioned as a product feature.

Cryptography is a resource intensive process that will introduce latency to any application that depends on it. Depending on the application involved and the organization, an objective could be to minimize the latency introduced by cryptography. HSMs have an advantage over software, as they are designed to optimize the efficiency of cryptographic processing. Compared to software running on general purpose servers, HSMs will typically accelerate security-critical cryptographic processing; this is an outcome of being purpose-built.