SPECIFICATION

TCG Mobile Reference Architecture

Version	2
Revision	45
March 6, 2023	

Contact:
admin@trustedcomputinggroup.org

Work in Progress

PUBLIC REVIEW

DISCLAIMERS, NOTICES, AND LICENSE TERMS

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

This document is copyrighted by Trusted Computing Group (TCG), and no license, express or implied, is granted herein other than as follows: You may not copy or reproduce the document or distribute it to others without written permission from TCG, except that you may freely do so for the purposes of (a) examining or implementing TCG specifications or (b) developing, testing, or promoting information technology standards and best practices, so long as you distribute the document with these disclaimers, notices, and license terms.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on patent licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
2/1	August 23, 2018	• Initial Release of Version 1.00.
2/2	October 2, 2018	• Updated, more granular comments
2/3	October 30, 2018	• Added comments from 10/30 MPWG Review
2/4	November 13, 2018	• Added comments from 11/13 MPWG Review
2/5	November 27, 2018	• Added comments from 11/27 MPWG Review
2/6	June 5, 2019	• Implemented a number of minor fixes that we noted (and accepted changes on the mundane ones for clarity), through discussion of boot sequence • Began moving informative sections to the Annex
2/7	June 5, 2019	• Added comments from 6/24 MPWG Review • Restored background sections in the body of the document, global update of TPM to TPM Mobile, moved architecture section earlier, and updates to the Mobile Device Architecture and Boot Sequence sections
2/8	August 31, 2019	• Added comments from 7/15/19 MPWG Review • Accepted some changes • Moved some DICE/MARS references up to the scope section • Started drafting the architecture section; included TCG technologies
2/9	November 25, 2019	• Added comments from 9/16/19 MPWG Review • Removed MARS reference (but keep tracking that change if there's a publication to reference in the future) • More work in the trusted computing architecture section, pulling in example graphics and more work on the language
2/10	January 2[illegible], 2020	• Added comments from 11/25/19 MPWG Review • Accepted some changes • Update footer date • Updates to the architecture section to paraphrase and reference other TCG specs • Update to the architecture section graphics
2/11	March 1, 2020	• Added comments from 1/27/2020 • Revised the TCG part of the Trusted Computing Architecture section again. Removed TSS figure and referenced it in the TSS doc. Same with TNC and SED. • Began describing relationship to other orgs specifications
2/12	April 18, 2020	• Cleaning up the unfinished architecture section, adding comments for future clarity and moving to another section for now. • Minor updates to the Device Boot Section and comments for review • Comments for review to a Protected Environment Section
2/13	May 16, 2020	• Updates based on 4/27/2020 MPWG review • Made Secure Boot section normative and updated language accordingly • Added some figures from ARM documentation for reference only • Removed SBPS Section and any references to it • Removed TPM Identity & Ownership Section
2/14	June 14	• Updates based on 6/X/20 MPWG review • Accepted many changes for clarity. • Update 1.4 Document Structure, and Section 3 • Remove 1 example from Section 4
2/15	July 18, 2020	• Updates based on 6/17 MM Review • Removed graphics in the Annexes • Re-organized sections • Make the introduction less TPM-centric • Review and comment RoT section (in its new place)

2/16	August 13, 2020	• Updates based on 7/20/20 MM Review • Accept changes • Make Scope less TPM-specific • Begin re-write of RoT section
2/17	October 4, 2020	• Accepted changes • Revised draft of RoT section and removed RoT discussion from appendix • Updates to the Trusted Computing Architecture section
2/18	October 26, 2020	• Moved several sections to the Appendix • Accepted a lot of changes • Moved unused terms out of the Glossary • Added sentences to the RoT • Comments from 10/26 review
2/19	December 29, 2020	• Accepted changes (esp in RoT section) • Updates to PE section • Notes on classification of PE reqs and section structure for discussion • Modifications to Appendix structure and content
2/20	January 28, 2021	• Updates to the Mobile Device Boot Sequence section and new proposed outline.
2/21	March 18, 2021	• Boot Sequence section updates: o Moving normative sentence to PE section o Moved section to the Appendix o Drafted new text for secure/measured boot technologies o Accepted changes to clean it up
2/22	March 27, 2021	• Protected Environment updates o Revamped Introductory text o Reorganized requirements o Accepted changes to clean it up
2/23	April 11, 2021	• Protected Environment updates o Continued re-organization (heading, moving a few around) o Revise and cut some requirements o Finish first pass • Change from device to mobile platform
2/24	April 18, 2021	• Replace trusted computing standards appendix with API appendix o Fix document structure • Draft PE communication/API requirements • Added PE CIA Table
2/25	May 3, 2021	• Re-write of Trusted Application Implementation section
2/26	May 17, 2021	• Re-write of Annex C Trusted Computing Interfaces. • Re-write of Annex D PE Implementation Examples • Cleaning up some References
2/27	June 12, 2021	• Update key word language • Addressing minor comments throughout • Fix cross-references • Update references
2/28	August 19, 2021	• Updating references in Glossary • Adding comments/updates from a beginning to end read-through • Update Roots of Trusts and references • Add trusted computing APIs
2/29	August 27, 2021	• Accepted a lot of changes • Added acknowledgments • Updated RoT definitions and references • Updated Trusted Computing Interfaces • Updated Requirements references
2/30	October 8, 2021	• Updates from end-to-end read through of this and MCP
2/31	November 13, 2021	• Clean up and acknowledgements
2/32	December 10, 2021	• Correct document version (change from 3 to 2), and accepted changes.
2/33	December 27, 2021	• Merge versions with recent comments and document version update

2/34	December 29, 2021	• Addressing recent straightforward comments (not the “more significant” comments)
2/35	March 13, 2022	• Addressing several “more significant” in acknowledgements, Roots of Trust, and PE Requirements sections
2/36	April 4, 2022	• Revisions to Protected Environment Requirements based on MPWG review • Change Mobile Platform Architectures to Mobile Platform Structures • Re-organize Mobile Platform Boot Phases Appendix
2/37	August 21, 2022	• Finished revisions to RoT Section. • Drafted Chains of Trust text • Revised Boot Sequence section • Protected Environment Requirements Updates in progress
2/38	September 11, 2022	• Accepting changes and cleaning up document • Adding some informative labels • Adding some references to RIP, NIST Supply Chain, etc • Re-ordering references
2/39	September 18, 2022	• Very minor tweaks caught during final review.
2/40	September 18, 2022	Accepted all changes in document.
2/41	November 2, 2022	• Addressing comments from pre-tech committee review. • Significant: Added Mobile Platform Requirements Section before Protected Environment Requirements Section and moved several requirements into it. • Included cross-references between Annex A and D • Accepted many changes for clarity.
2/42	November 16, 2022	• Minor tweaks from MPWG 11/14 Review
2/43	November 16, 2022	• Accepted all changes in document.
2/44	February 20, 2023	• Addressing all Technical Committee Comments
2/45	March 6, 2023	• Accepted all changes in document.

ACKNOWLEDGEMENTS

The TCG wishes to thank all those who contributed to this specification. This document builds on work done over the course of 10 years in various TCG work groups. Special thanks to the following current and former members of the TCG who contributed to this document:

Name	Affiliation
Ronald Aigner	Microsoft
Henk Birkholz	Fraunhofer SIT
Bo Bjerrum	Intel Corporation
Alec Brusilovsky (MPWG co-chair)	InterDigital Communications, LLC
Michael Chan	Samsung Semiconductor Inc.
Cedric Colnot	NXP Semiconductors
Carlin Covey	Freescale Semiconductors
Paul England	Microsoft
Jan-Erik Ekberg	Nokia Corporation
Andreas Fuchs	Fraunhofer SIT
Jon Geater	Trustonic
Wael Ibrahim	American Express
Dean Liberty	Advanced Micro Devices, Inc.
Andrew Martin	University of Oxford
Ira McDonald (co-editor)	High North Inc.
Kathleen McGill (MPWG co-chair)	Johns Hopkins University, Applied Physics Lab
John Mersh	ARM Ltd
Hadi Nahari	NVIDIA
Amy Nelson	Dell, Inc.
Ken Nicolson	Panasonic
Carlton Northern	MITRE Corp.
Jeremy O’Donoghue	Qualcomm Inc.
Niall O'Donoghue	Nokia
John Padgette	Accenture Federal
Michael Peck	MITRE Corp.
Gilles Peskine	Trustonic
Nicolas Ponsini	Trusted Logic
Graeme Proudler	Independent
Emily Ratliff	Advanced Micro Devices, Inc.
Andre Rein	Huawei
Joshua Schiffman	HP Inc.
Ariel Segall	MITRE Corp.
Atul Shah	Microsoft
Herve Sibert	STMicroelectronics
Rob Spiger	Microsoft
Marcus Streets	ARM Ltd.
Janne Uuselehto	Nokia
Paul Waller	National Cyber Security Centre (UK)
David Wooten	Microsoft
Stuart Yoder	ARM Ltd.

CONTENTS

1 SCOPE

This specification defines a reference architecture for the implementation of Trusted Computing technologies in modern mobile platforms. The cornerstone of this architecture is a Protected Environment that provides isolated resources for Trusted Applications. This architecture allows any possible implementation of a Protected Environment that meets the security requirements defined in this specification. Several example implementations are described in Annex D. The examples in this document are not mandatory.

The target audience of this specification includes designers, developers, and implementers of trusted computing technologies in mobile platforms.

This specification supports the use cases defined in [1].

1.1 Key Words

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

1.2 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statements.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

In some cases, entire sections are comprised of informative text, in order to provide background information or context for the normative statements in other sections of this specification. These sections are labelled by appending “INFORMATIVE” to the section title.

1.3 References

[1] Trusted Computing Group, Mobile Trusted Module 2.0 Use Cases, March 2011, https://trustedcomputinggroup.org/resource/mobile-trusted-module-2-0-use-cases/.

[2] Trusted Computing Group Glossary, Version 1.1, Revision 1.0, May 2017, https://trustedcomputinggroup.org/resource/tcg-glossary/.

[3] Trusted Computing Group, Hardware Requirements for Device Identifier Composition Engine Level 00, Revision 78, March 2018, https://trustedcomputinggroup.org/resource/hardware-requirements-for-a-device-identifier-composition-engine/.

[4] NIST Special Publication 800-164: Guidelines on Hardware Rooted Security in Mobile Devices (Draft), October 2012, https://csrc.nist.gov/publications/detail/sp/800-164/draft.

[5] GlobalPlatform, Root of Trust Definitions and Requirements V1.1, June 2018, https://globalplatform.org/specs-library/globalplatform-root-of-trust-definitions-and-requirements/.

[6] ARM Ltd., Platform Security Architecture Security Model 1.0, December 2019, https://developer.arm.com/documentation/den0128/latest.

[7] Trusted Computing Group, Trusted Platform Module Library Parts 1-4, Specification Version 2.0, Revision 1.59, November 2019, https://trustedcomputinggroup.org/work-groups/trusted-platform-module/.

[8] Trusted Computing Group, TPM 2.0 Mobile Common Profile, Version 1.0, Revision 31, December 2015, https://trustedcomputinggroup.org/resource/tcg-tpm-2-0-mobile-common-profile/.

[9] Unified Extensible Firmware Interface Forum, The UEFI Specification Version 2.9, March 2021, https://uefi.org/sites/default/files/resources/UEFI_Spec_2_9_2021_03_18.pdf.

[10] NIST Special Publication 800-161r1: Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations, May 2022, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161r1.pdf.

[11] ARM Ltd., ARM TrustZone Technology, https://developer.arm.com/ip-products/security-ip/trustzone.

[12] Intel Corporation, Intel Software Guard Extensions, https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html.

[13] David Kaplan, Jeremy Powell, and Tom Woller, AMD memory encryption, https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf, April 2016.

[14] Committee on National Security Systems, Committee on National Security Systems Glossary, April 2015, https://rmf.org/wp-content/uploads/2017/10/CNSSI-4009.pdf.

[15] GlobalPlatform, TEE Secure Element API Version 1.1.2, February 2021, https://globalplatform.org/specs-library/tee-secure-element-api/.

[16] NIST SP 800-66: Revision 1, An Introductory Resource Guide for Implementing the Health Insurance Portability and Accountability Act (HIPAA) Security Rule, October 2008, https://csrc.nist.gov/publications/detail/sp/800-66/rev-1/final.

[17] NIST SP 800-57 Part 1 Rev. 5: Recommendation for Key Management: Part 1 - General, May 2020, https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final.

[18] NIST SP 800-131A Rev. 2: Transitioning the Use of Cryptographic Algorithms and Key Lengths, March 2019, https://csrc.nist.gov/publications/detail/sp/800-131a/rev-2/final.

[19] Internet Engineering Task Force, RFC 4086: Randomness Requirements for Security June 2005, https://datatracker.ietf.org/doc/html/rfc4086.

[20] International Organization for Standardization/International Electrotechnical Commission, ISO/IEC 18031:2011 Information technology – Security techniques – Random bit generation Edition 2, November 2011, https://www.iso.org/standard/54945.html.

[21] Internet Engineering Task Force, RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms specification June 2010, https://datatracker.ietf.org/doc/html/rfc5905.

[22] Internet Engineering Task Force, RFC 8633: Network Time Protocol Best Current Practices July 2019, https://datatracker.ietf.org/doc/html/rfc8633.

[23] GlobalPlatform, TEE Protection Profile Version 1.3, December 2016, https://globalplatform.org/specs-library/tee-protection-profile-v1-3/.

[24] International Organization for Standardization/International Electrotechnical Commission, ISO/IEC 19770-2: Information technology — IT asset management — Part 2: Software identification tag, October 2015, https://www.iso.org/standard/65666.html.

[25] Trusted Computing Group, Runtime Integrity Preservation in Mobile Devices, Version 1.0, Revision 106, November 2019, https://trustedcomputinggroup.org/resource/tcg-runtime-integrity-preservation-in-mobile-devices/.

[26] Trusted Computing Group, Virtualized Trusted Platform Architecture Specification, Version 1.0, Revision 26, September 2011, https://trustedcomputinggroup.org/resource/virtualized-trusted-platform-architecture-specification/.

[27] DENX Software Engineering, Das U-Boot – the Universal Boot Loader, May 2021, https://www.denx.de/wiki/U-Boot/WebHome.

[28] LWN, Verified U-Boot, October 2013, https://lwn.net/Articles/571031/.

[29] Trusted Computing Group, MARS Use Cases and Considerations Version 1, Revision 27, January 2021, https://trustedcomputinggroup.org/resource/mars-use-cases-and-considerations/.

[30] Trusted Computing Group, TPM Software Stack Specifications (TSS2), October 2019, https://trustedcomputinggroup.org/work-groups/software-stack/.

[31] Trusted Computing Group, TPM 2.0 Mobile Command Response Buffer Version 2, Revision 12, November 2019, https://trustedcomputinggroup.org/resource/tpm-2-0-mobile-command-response-buffer-interface-specification/.

[32] GlobalPlatform, TEE Internal Core API Specification Version 1.3, February 2021, https://globalplatform.org/specs-library/tee-internal-core-api-specification/.

[33] GlobalPlatform, TEE Client API Specification Version 1.0, July 2010, https://globalplatform.org/specs-library/tee-client-api-specification/.

[34] GlobalPlatform, TEE Sockets API Specification Version 1.0.1, January 2017, https://globalplatform.org/specs-library/tee-sockets-api-specification/.

[35] Internet Engineering Task Force, TEE Provisioning Architecture, https://datatracker.ietf.org/wg/teep/about/.

[36] Internet Engineering Task Force, TEE Provisioning Protocol, https://datatracker.ietf.org/doc/draft-ietf-teep-protocol/.

[37] GlobalPlatform, TEE Management Framework including ASN.1 Profile v1.1.2, November 2020, https://globalplatform.org/specs-library/tee-management-framework-including-asn1-profile-1-1-2/.

[38] GlobalPlatform, Open Mobile API Specification v3.3, August 2018, https://globalplatform.org/specs-library/open-mobile-api-specification-v3-3/.

[39] GlobalPlatform, Trusted Platform Services Client API Version 0.0.0.16, Member Review #2, April 2022.

[40] GlobalPlatform, TEE System Architecture Version 1.2, November 2018, https://globalplatform.org/specs-library/tee-system-architecture-v1-2/.

[41] International Organization for Standardization/International Electrotechnical Commission, ISO/IEC 7816-3:1997 Identification cards - Integrated circuit(s) cards with contacts - Part 3: Electronic signals and transmission protocols, https://www.iso.org/standard/14735.html.

[42] ETSI TS 102 226 (Release 6) Smart cards: Remote APDU structure for UICC based applications, European Telecommunications Standards Institute Project Smart Card Platform (EP SCP), 2004, https://www.etsi.org/deliver/etsi_ts/102200_102299/102226/06.18.00_60/ts_102226v061800p.pdf.

[43] GlobalPlatform, Card Specification Version 2.3.1, March 2018, https://globalplatform.org/specs-library/card-specification-v2-3-1/.

[44] European Smart Card Industry Association, Protection Profile Smart Card Integrated Circuit with Embedded Software Version 2.0, June 1999, https://www.commoncriteriaportal.org/files/ppfiles/PP9911.pdf.

[45] GlobalPlatform, Functional Certification, http://globalplatform.org/compliance.asp.

[46] GlobalPlatform, Java Card API and Export File for Card Specification v2.2.1 v1.5, https://globalplatform.org/specs-library/globalplatform-card-api-org-globalplatform/.

1.4 Document Structure

This document is divided and organized into the following Sections:

- Section 2 contains a reference glossary defining the terms and definitions used throughout this specification.
- Section 3 is a brief introduction to the motivation and core concepts of this specification.
- Section 4 provides an introduction to Roots of Trust.
- Section 5 describes the boot sequences used by various mobile platforms.
- Section 6 defines requirements for a Mobile Platform that supports a Protected Environment.
- Section 7 defines requirements for a Protected Environment to host a Trusted Application.
- Section 8 defines requirements for a Trusted Application implementation to operate securely within a Protected Environment.
- Annex A provides an informative overview of typical mobile platform structures.
- Annex B gives example implementations of the boot sequences used by various mobile platforms.
- Annex C introduces Trusted Computing interface APIs that support this Mobile Reference Architecture.
- Annex D describes examples of implementations of Protected Environments.

2 Glossary

GLOSSARY TERM	DESCRIPTION
Application-specific integrated circuit (ASIC)	A microchip that is designed for a specific application (in contrast to a general-purpose chip such as a microprocessor).
Attestation	The process of vouching for the accuracy of information. A platform can attest to its description of platform characteristics that affect the integrity (trustworthiness) of the platform. Attestation requires reliable evidence of the attesting entity (modified from [2])
Device	A shorthand in this document for a Mobile Platform.
DICE	The Device Identifier Composition Engine (DICE) is an immutable, or securely updated, engine that executes very early in the boot cycle and derives a value from a Unique Device Secret (UDS) and the cryptographic identity of the first mutable code on the platform [3].
Fuse	A mechanism of internal switches within the ASIC that can be electrically blown to enable write-once non-volatile storage of information within that ASIC.
Integrity Measurement	A value representing a platform characteristic that affects the integrity of a platform [2].
Measured Boot	A boot process where images are measured (for example by calculating their digests) and the measurements are extended into integrity-protected storage. See Annex B for more details.
Mobile Platform	A physical entity encompassing all the hardware, firmware, software, and data necessary for it to function and provide services to an end user. Also known as a Mobile Device, or just Device in this specification. A laptop, tablet, smartphone, feature phone, basic phone, or other similar device that typically includes cellular connectivity (2G, 3G, LTE, 5G, etc.) and/or Internet connectivity (Ethernet, WiFi, WiMax, etc.) and associated protocol stacks.
Protected Environment	A functional element with its own execution and memory resources that are isolated from other components. See Sections 6 and 7 for details.
Rich Operating System (Rich OS)	An environment optimized for versatility and features where the bulk of device applications are executed. This environment is also called the Rich Execution Environment, Regular Execution Environment, or REE.
Root of Trust (RoT)	Roots of Trust (RoT) provide the foundation of any Trusted Computing architecture. They are hardware, firmware, and software components that perform specific, critical security functions [2], [4], [5], [6], [7].
Root of Trust for Confidentiality (RTC)	A Root of Trust providing confidentiality for secrets stored in secure storage locations (modified from [2]).
Root of Trust for Integrity (RTI)	A Root of Trust providing integrity for data stored in secure storage locations (modified from [2]).
Root of Trust for Measurement (RTM)	A Root of Trust that makes the initial integrity measurement, and adds it to a tamper-resistant storage location (modified from [2]).
Root of Trust for Reporting (RTR)	A Root of Trust that provides authenticity and non-repudiation services for the purposes of attesting to the origin and integrity of platform characteristics [2].
Root of Trust for Storage (RTS)	The combination of the RTC and RTI [2].
Root of Trust for Update (RTU)	A Root of Trust for Verification that verifies the integrity and authenticity of an update payload before initiating the update process [2].
Root of Trust for Verification (RTV)	A Root of Trust that verifies an integrity measurement against a policy or expected value [modified from 2].

Secure Boot	A boot process where images are validated before execution (for example by calculating their digests and comparing them to expected values). See Annex B for more details.
Trusted Platform Module (TPM)	An implementation of the functions defined in the TCG Trusted Platform Module Specification; the set of Roots of Trust with Shielded Locations and Protected Capabilities [2].
TPM Mobile	An implementation that conforms to the TCG TPM 2.0 Mobile Common Profile [8].
Trusted Application (TA)	An application that runs as an isolated process within a Protected Environment.
Unified Extensible Firmware Interface (UEFI)	A software interface between an operating system and platform firmware. See [9] for details.

3 Introduction

Start of informative comment

The evolution of mobile computing has led to smart devices that are in almost constant personal and business use. Users routinely access email, social media, application stores, financial institutions, and a variety of other services from their mobile platforms. Manufacturers, system integrators, and developers of secure applications can incorporate Trusted Computing mechanisms to provide security for and trust in these mobile use cases. Trusted Computing mechanisms include techniques to ensure platform integrity, resource isolation, and protected storage.

This specification adapts Trusted Computing concepts to apply them to mobile platforms. The diversity of the mobile market place and underlying mobile hardware structures (see Annex A) require a sound security model on which to build trusted services. That diversity also subjects mobile platforms to significant supply chain risk. This specification does not address mobile supply chain risk management directly, but many organizations, such as NIST, have provided guidance on the topic [10].

Central to this architecture is a Protected Environment. This specification defines a set of requirements for a Protected Environment to provide sufficient isolation of memory and execution resources for Trusted Applications. This specification places no limitations on how a Protected Environment is implemented, as long as it meets the requirements. Some examples of Protected Environment implementations are:

- Environments isolated by processor features [11][12][13]
- A virtual environment hosted by a hypervisor
- A physically isolated Application-Specific Integrated Circuit (ASIC) [14]

To improve the understanding of the architecture, example implementation models of a Protected Environment are described in informative Annex D.

End of Informative Comment

4 Roots of Trust

Start of informative comment

Roots of Trust (RoT) are the foundation of any Trusted Computing architecture. The RoT are a prerequisite for implementing a Protected Environment capable of hosting Trusted Applications. The RoT can be implemented as part of a Protected Environment, as dedicated hardware, or as other components, depending on the architecture of a mobile platform. A detailed discussion of RoT is beyond the scope of this specification, but several organizations have defined RoT, including the TCG [2], [3], NIST [4], GlobalPlatform [5], and ARM [6].

An RoT is a component that enables discovery of a platform's trustworthiness. These are the primary RoT recognized in all public standards:

- Root of Trust for Measurement (RTM): An RoT that makes the initial integrity measurement and records it in a tamper-resistant storage location.
- Root of Trust for Reporting (RTR): An RoT that provides authenticity and non-repudiation services for the purposes of attesting to the origin and integrity of platform characteristics.
- Root of Trust for Storage (RTS): An RoT providing confidentiality and integrity for data stored in secure storage locations.

The RTS can be decomposed, notionally or functionally, by the properties of confidentiality and integrity that it provides. In those cases, the following RoT definitions apply:

- Root of Trust for Confidentiality (RTC): An RoT providing confidentiality for secrets stored in secure storage locations.
- Root of Trust for Integrity (RTI): An RoT providing integrity for data stored in secure storage locations.
- Root of Trust for Storage (RTS): The combination of the RTC and RTI.

Many standards bodies have expanded the definition of RoT to encompass components that perform one or more security-specific functions that are critical to enforcement of trusted platforms [4]. In order to maintain consistency with the standards in the mobile ecosystem, these RoT are included in the set of RoT of this specification:

- Root of Trust for Verification (RTV): An RoT that verifies an integrity measurement against a policy or expected value.
- Root of Trust for Update (RTU): An RTV that verifies the integrity and authenticity of an update payload before initiating the update process [illegible].

Finally, in some specifications, an RoT is trusted to behave in the expected manner because its misbehaviour cannot be detected [2], [4]. However, in typical mobile platform architectures, many low-level components that provide RoT functionality are verifiable during Secure Boot. In this specification, verifiable components that provide security capabilities for the platform are referred to as an RoT, despite the fact that their misbehaviour can be detected.

4.1 Chains of Trust

A trustworthy platform can establish a chain of trust from an RoT to other components, either executables or data, on the platform. Through this process, the RoTs establish the trustworthiness of a component. Trust in that component is then used to establish the trustworthiness of the next component in the chain.

Typically, a chain of trust extends a security function, or service, from an RoT to other components. Examples common to mobile platforms include the following:

- A Chain of Trust for Verification extends the verification service from the RTV to other components. The RTV verifies a component that includes a verification service similar to the RTV. Afterwards, the verified component can verify other components on the host mobile platform.
- A Chain of Trust for Confidentiality expands confidentially protections to storage external to the secure storage locations. The RTC can protect keys that encrypt data stored in flash memory of a mobile platform.

- A Chain of Trust for Integrity extends integrity protections from the RTI to other components. For example, the RTI can include a key or other data that is used by the RTV to validate a component. This validation adds the component to a Chain of Trust for Integrity. Thereafter, modules that are validated based on data protected by the RTI or components of its Chain of Trust for Integrity are added to the Chain of Trust for Integrity themselves.

End of Informative Comment

5 Mobile Platform Boot Sequences

Start of informative comment

This section describes the various stages that can comprise the boot sequence and establish a Protected Environment, with particular emphasis on typical mobile platform architectures.

5.1 Boot ROM

Once the hardware is initialized, it starts to execute the Boot ROM. The Boot ROM is software stored in ROM that is built into the ASIC. Boot ROM integrity is inherent because the cost of modifying on-chip ROM falls outside the bounds of most commercial threat models. The software in the Boot ROM performs further hardware initialization before initiating Secure Boot.

The Boot ROM serves as the RTV and forms the root of the Chain of Trust for Verification. The software in the Boot ROM locates the first boot image, typically in some form of non-volatile storage. The Boot ROM then measures the located image and verifies the measurement. The expected value of the measurement is based on keys and other data that are stored or derived from values in write-once memory. These values are either part of the RTI or a Chain of Trust for Integrity. If all checks pass, the platform executes the loaded software.

5.2 Secure Boot

The most widely implemented boot mechanism on mobile platforms is Secure Boot. In Secure Boot, each module of code acts as a link in both a transitive Chain of Trust for Verification tracing back to the RTV and a transitive Chain of Trust for Integrity tracing back to the RTI.

During Secure Boot, any module of code (or a data file in some cases) is measured and validated before it is loaded. The validity of the measurement is determined based on information stored either in write-once memory or in previously verified code modules. If any of these measurements fail validation, the software enters a remediation mode.

5.3 Measured Boot

Measured Boot is the process followed once an integrity-protected storage mechanism is available for recording new data during the boot phase. During Measured Boot, any module of code (or a data file in some cases) is measured before it is loaded. The measurement is recorded in integrity-protected storage and is available for use in binding or attestation schemes thereafter. Examples of suitable storage mechanisms include a TPM Mobile, a Device Identifier Composition Engine (DICE) [4], or a Secure Element (SE) [15]. The code module is then executed or the data is used, regardless of the value of the measurement.

5.4 Overlapping Secure Boot and Measured Boot

The Secure Boot and Measured Boot Phases can overlap during the boot sequence. As soon as an integrity-protected storage mechanism is available for recording new data, the measurement can be recorded. However, the measurements may still be validated by the RTV or a module on the Chain of Trust for Verification in order to proceed with the boot sequence.

There are potential benefits to overlapping Secure Boot and Measured Boot in this way. Secure Boot enforces control over the platform runtime environment and can implant a cryptographic key on the platform. Practically, that key will be unique to the platform, and not updated with platform software updates. Measured Boot provides measurement and attestation of the platform software over the course of software updates and uses the cryptographic key to attest to the measurements. This approach allows external entities to establish trust in the platform without the need to change platform keys with every software update.

End of Informative Comment

6 Mobile Platform Requirements

Start of Informative Comment

This section describes the requirements for a mobile platform to support a Protected Environment. A Protected Environment provides execution and memory resources that are isolated from other components of a platform. The isolated execution property of a Protected Environment distinguishes a Protected Environment from the Regular Operating System (OS) environment. A mobile platform can satisfy these requirements using any mechanisms. Annex A describes examples of high-level structural components of contemporary mobile platforms.

End of Informative Comment

6.1 Mobile Platform Fundamental Requirements

MF1. A mobile platform SHALL implement a Protected Environment using mechanisms that enable each Protected Environment to satisfy the requirements in Section 7.

MF2. A mobile platform SHALL implement every cryptographic algorithm of this specification with a cryptographic strength [17] of at least 112 bits [18].

MF3. A mobile platform SHALL provide a Protected Environment with access to an entropy source that complies with IETF RFC 4086 [19] and ISO/IEC 18031:2011 [20].

MF4. A mobile platform SHALL provide a Protected Environment with a trusted source of time–of-day information [21] [22].

Start of Informative Comment

Physical protections (such as the use of Read-Only Memory (ROM), fuses, physical separation, or some processor isolation mechanisms) are adequate substitutes for cryptographic protections as long as the physical protections are sufficient to defeat general software attacks and at least simple hardware attacks, comparable to [23].

End of Informative Comment

6.2 Mobile Platform Secure Boot Requirements

MB1. A mobile platform SHALL implement the Boot ROM phase.

MB2. A mobile platform SHALL implement the Secure Boot phase.

MB3. A mobile platform SHALL cryptographically verify the integrity of all Protected Environment capabilities and extend a Chain of Trust for Integrity from the hardware Roots of Trust.

6.3 Mobile Platform Measured Boot Requirements

MM1. A mobile platform SHALL implement the Measured Boot phase.

MM2. A mobile platform SHALL generate evidence that unambiguously identifies every platform firmware and software component that implements a Protected Environment and is used by a Protected Environment [24].

MM3. A mobile platform SHALL record evidence that unambiguously identifies every platform firmware and software component that implements a Protected Environment and is used by a Protected Environment, in integrity-protected storage [24].

MM4. A mobile platform SHALL provide evidence in requirements MM2 and MM3 that is sufficient to identify the manufacturer and the version of each component.

6.4 Mobile Platform Update Requirements

MU1. A mobile platform SHALL provide a means to update a Protected Environment or any software on which a Protected Environment depends.

MU2. A mobile platform SHALL cryptographically verify the integrity and authenticity of updates to a Protected Environment or any software on which a Protected Environment depends, before the updates are installed.

MU3. A mobile platform SHALL prevent rollback of software implementing a Protected Environment to older versions.

6.5 Mobile Platform Runtime Requirements

MR1. A mobile platform SHALL isolate a Protected Environment from all other execution environments.

MR2. A mobile platform SHALL protect communications between a Protected Environment and all other execution environments.

MR3. A mobile platform SHALL implement communication mechanisms between a Protected Environment and other execution environments, based on published standards.

6.6 Mobile Platform Storage Requirements

MS1. A mobile platform SHALL provide a Protected Environment with exclusive access to non-volatile storage locations.

MS2. A mobile platform SHALL prevent the disclosure of Protected Environment information stored either in run time memory or in non-volatile storage locations, through debug facilities of a mobile platform.

MS3. A mobile platform SHALL isolate the execution resources of cryptographic services or capabilities used by a Protected Environment from entities external to that Protected Environment.

7 Protected Environment Requirements

Start of Informative Comment

A Protected Environment executes Trusted Applications and protects them from interference from other Trusted Applications and from code executing outside the Protected Environment. All implementations of trusted services in a Protected Environment, such as storage, measurement, and verification, are founded on the underlying RoT. When a mobile platform does not have dedicated hardware for a Protected Environment, the mobile platform establishes a Protected Environment through transitive chains of trust from a hardware RoT during Secure Boot. Annex D describes Example Implementations of a Protected Environment.

End of Informative Comment

7.1 Protected Environment Fundamental Requirements

PF1. A Protected Environment SHALL provide Trusted Applications with access to an entropy source that complies with IETF RFC 4086 [19] and ISO/IEC 18031:2011 [20].

PF2. A Protected Environment SHALL provide Trusted Applications with a trusted source of time–of-day information [21] [22].

7.2 Protected Environment Secure Boot Requirements

PB1. A Protected Environment SHALL cryptographically verify the integrity of Trusted Applications and extend a Chain of Trust for Integrity from the hardware Roots of Trust.

7.3 Protected Environment Update Requirements

PU1. A Protected Environment SHALL provide a means to update Trusted Applications.

PU2. A Protected Environment SHALL cryptographically verify the integrity and authenticity of updates to Trusted Applications before the updates are installed.

PU3. A Protected Environment SHALL prevent rollback of software implementing Trusted Applications to older versions.

7.4 Protected Environment Runtime Requirements

PR1. A Protected Environment SHALL isolate its execution resources used by its Trusted Applications from other Trusted Applications within that Protected Environment.

PR2. A Protected Environment SHALL protect communications between Trusted Applications within that Protected Environment.

PR3. A Protected Environment SHOULD implement communication mechanisms between Trusted Applications within that Protected Environment, based on published standards.

Start of Informative Comment

See Annex C for examples of published communication standards.

The requirements in section 7.4 do not apply if a Protected Environment does not support hosting of multiple Trusted Applications.

See TCG Runtime Integrity Preservation in Mobile Devices [25] for a more thorough discussion of runtime protections.

End of Informative Comment

7.5 Protected Environment Storage Requirements

PS1. A Protected Environment SHALL protect the integrity and confidentiality of information in its non-volatile storage locations.

PS2. A Protected Environment SHALL provide Trusted Applications with access to that Protected Environment's non-volatile storage locations.

PS3. A Protected Environment SHALL prevent replay or rollback of that Protected Environment's non-volatile storage locations.

PS4. A Protected Environment SHALL prevent the inappropriate disclosure of information stored in non-volatile storage locations, such that only its Trusted Applications and that Protected Environment can access the information.

8 Trusted Application Requirements

Start of Informative Comment

A Trusted Application is an application that runs within a Protected Environment with protected execution and storage resources (see Section 7). While Section 7 describes the requirements of a Protected Environment to secure Trusted Applications, this section defines requirements and recommendations of the Trusted Applications themselves.

These requirements support a TPM Mobile implemented as a Trusted Application, but there are numerous other use cases for Trusted Applications throughout the mobile ecosystem, such as mobile banking or secure messaging applications.

End of Informative Comment

8.1 Trusted Application Constraints

T1. A Trusted Application SHALL use the non-volatile storage mechanism provided by its host Protected Environment to store the Trusted Application's non-volatile data.

T2. A Trusted Application SHALL use cryptographic mechanisms provided by its host Protected Environment.

T3. A Trusted Application SHALL NOT use a cryptographic mechanism provided by a source other than its host Protected Environment when a suitable Protected Environment mechanism is available.

T4. A Trusted Application SHALL NOT implement (or advertise) cryptographic protections that exceed the cryptographic protections of its host Protected Environment (see Section 7).

T5. A Trusted Application SHALL use the entropy source provided by its host Protected Environment for random number generation.

T6. A Trusted Application SHALL use the time-of-day information provided by its host Protected Environment.

Start of Informative Comment

Requirement T3 does not preclude the implementation of proprietary algorithms or higher layer protocols in a Trusted Application.

End of Informative Comment

A. Mobile Platform Structure

Start of Informative Comment

This section is an informative introduction to the high-level structural components of a contemporary mobile platform. The focus is on features implemented in hardware or closely related to hardware. Although the reference setting is mobile phones, the same spectrum of hardware features can be present in other mobile platforms, such as music players and tablets. This TCG Mobile Reference Architecture supports all of these structures for a Protected Environment, and all structures can support the implementation of a Trusted Application.

A.1 Protected Environment Isolated by Processor Features

Figure 1 illustrates a structure that is present in many mobile platforms and is suitable for an implementation of a Protected Environment. Processor features isolate a Protected Environment from the code executing in the Rich OS within a mobile platform.

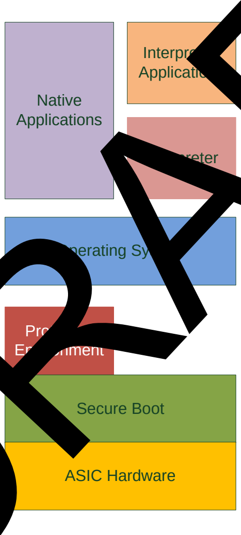

Figure 1 – Structure of a Mobile Platform with a Protected Environment Isolated by Processor Features

If a Protected Environment executes solely in on-chip memories, or if secure virtual memory techniques are used, the strength of the constructed isolation boundary can resemble that of a hardware boundary. If mobile platform secrets are protected by these mechanisms, their integrity and correctness is no longer contingent upon the integrity and correctness of any software component booted into the Rich OS domain. Annex D.1 provides an example of a Protected Environment implemented using this method of isolation.

A.2 Protected Environment Isolated by a Hypervisor

Figure 2 illustrates a hypervisor-enabled structure that is useful for adding another operating system to a mobile platform running on single-core hardware. The hypervisor reduces the exposure of secrets implemented in the hardware. Early in Secure Boot, the processor memory management unit (MMU) can be configured to give the hypervisor sole access to platform secrets.

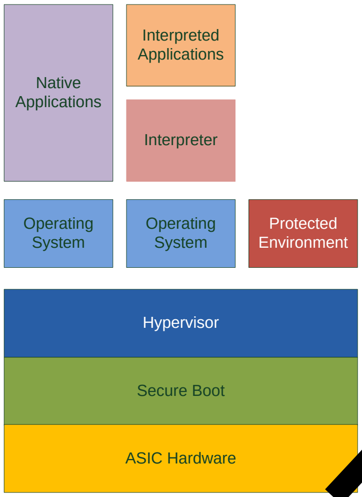

Figure 2 – Example Structure of a Mobile Platform with a Protected Environment Isolated by a Hypervisor

Hypervisor structures can provide isolation to implement the Protected Environment requirements defined in Section 7. A Trusted Application can run in such a Protected Environment and provide services for a single guest OS or for the hypervisor itself.

This specification does not define the complete structure for hypervisor-enabled platforms, but the TCG has published a Virtualized Trusted Platform Architecture [26].

A.3 Protected Environment Isolated by Processors or Cores

Figure 3 shows three models using multiple processor cores:

1. The code running on each processor core is individually securely booted and associated with a core-specific Protected Environment.

2. One or more processor cores provide a secure, isolated environment for a Protected Environment while the other cores run the Rich OS.

3. A Protected Environment is in a separate ASIC to achieve isolation.

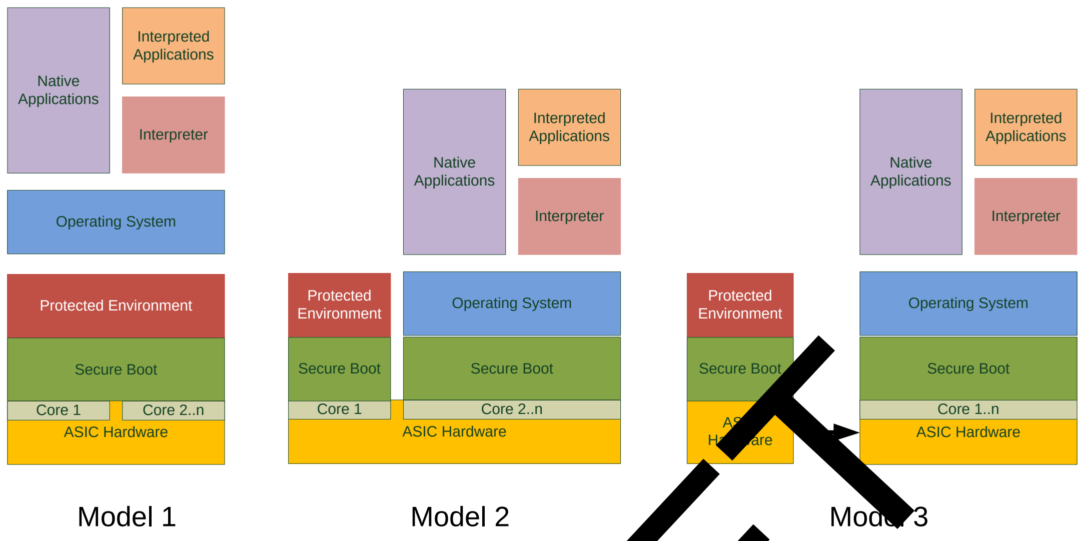

Figure 3 – Examples of Multi-core and Multiprocessor Structures

Annex D.1 provides an example of a Protected Environment implemented using Model 3 for isolation.

End of Informative Comment

B. Mobile Platform Boot Sequence Examples

Start of Informative Comment

B.1 Secure Boot Examples

B.1.1 Unified Extensible Firmware Interface (UEFI) Secure Boot

Many platforms use a mechanism known as UEFI [9] to perform their boot sequence. UEFI is a specification that defines a software interface between platform firmware and the operating system. UEFI includes a Secure Boot mechanism that launches only drivers and images that are signed by a valid key. If signature verification of any image fails, the UEFI firmware initiates OEM-specific recovery to restore trusted firmware. This mechanism is very similar to the Secure Boot mechanism described in this document.

B.1.2 U-Boot

U-Boot is an open-source bootloader widely used in embedded devices [27][28]. It includes a Secure Boot mechanism called verified boot that uses digital signatures to verify software images [28]. The approach in verified boot requires a trusted public key to be present and protected on a mobile platform to verify the integrity and authenticity of software images. From there, a Chain of Trust for Verification can be established from the loaded images back to the Boot ROM. Verified boot also allows system software updates, as long as updates are signed by the trusted authority whose public key is on the platform. More flexible U-Boot solutions are also possible, in which the user provides a public key, for example, by inserting an SD card.

B.2 Measured Boot Examples

B.2.1 TPM Mobile

The TPM Mobile [8] is a profile of the TCG TPM 2.0 [7] that provides Shielded Locations and Protected Capabilities for a mobile platform. A Shielded Location is a suitable storage location for integrity measurements. Protected Capabilities are the set of commands with exclusive permissions to access Shielded Locations. When a platform with a TPM Mobile boots, it is capable of Measured Boot once the TPM Mobile is available and module measurements can be extended into its Shielded Locations. These measurements are then available for the functions described in the TPM 2.0 Library Specification [7].

B.2.2 Measurement and Attestation Roots (MARS)

In some embedded systems, the inclusion of a TPM profile may be untenable due to power, space, and cost limitations. The purpose of Measurement and Attestation RootS (MARS) is to specify how microcontrollers can implement the RoT required for measurement and attestation with acceptably low overhead [29]. These MARS implementations include measurement, recording, and reporting capabilities that enable measured boot on low-end embedded systems.

B.2.3 Device Identifier Composition Engine (DICE)

A Device Identifier Composition Engine (DICE) is a TCG solution to enable hardware-based cryptography and attestation with minimal hardware requirements [3]. A DICE is an immutable, or securely updated, engine that executes very early in the boot cycle and derives a value from a Unique Device Secret (UDS) and the cryptographic identity (hash) of the first mutable code on the platform. This value is called the Compound Device Identifier (CDI). The CDI is used to identify that first mutable code and the system executing it. To use DICE safely, the CDI must not be revealed to software that might impersonate the system that is executing the first mutable code. The DICE specification outlines requirements for a platform that protects the UDS and the CDI, to attest to all mutable code without the need for a TPM or other hardware security module.

End of Informative Comment

C. Trusted Computing Communication Interfaces

Start of Informative Comment

The TCG has developed mobile specifications that are compatible with specifications published by Standards Development Organizations (SDOs) within the mobile ecosystem. Among these, there are several communication and programming interfaces that support the requirements of Protected Environment and Trusted Application implementations. The following is a list of example interfaces, but any standardized interfaces that meet the requirements of this specification are candidates for this Mobile Reference Architecture. For more details about the interfaces on this list, see the referenced specifications.

- The TCG TPM 2.0 Trusted Software Stack contains multiple layers of application- and system-level interfaces to isolate TPM 2.0 client applications from low-level details of interfacing to the TPM. Application developers can use this software specification to develop interoperable client applications [30].
- The TCG TPM 2.0 Mobile Command Response Buffer (CRB) Interface is a kernel interface to a TPM that is intended to work with all architectures. The CRB Interface makes it possible to write a driver that can interact with a TPM, whether the TPM is a discrete component on a peripheral bus or a Trusted Application in a Protected Environment [31].
- The GlobalPlatform TEE Internal Core API [32] is a set of C APIs exposed to Trusted Applications running in the TEE. The APIs provide a variety of functions, such as memory management, communications with client applications in the Regular Execution Environment (REE), trusted storage, cryptographic facilities, time management, and peripheral interfaces.
- The GlobalPlatform TEE Client API [33] defines communications between applications running in the REE and applications running in the TEE. This API serves as a base layer on which developers can implement higher level protocols.
- The GlobalPlatform TEE Sockets API is a generic C interface that allows Trusted Applications in a TEE to establish socket-style network communications with a remote server [34]. The TEE Sockets API specification does not require particular protocols or data structures, but it does include guidance on implementing some protocols such as TCP and UDP.
- The Internet Engineering Task Force (IETF) TEE Provisioning (TEEP) Working Group is developing TEEP Architecture [35] and TEEP Protocol [36] specifications in order to provide TEEs with lifecycle management and security domain management of Trusted Applications.
- The GlobalPlatform TEE Management Framework describes a security model for the administration of TEEs, TAs, and their corresponding Security Domains (SDs). The specification addresses the various roles and responsibilities of multiple stakeholders involved in the administration of TEEs and TAs, as well as the mechanisms of administration [37].
- The GlobalPlatform TEE Secure Element API [15] is a thin interface to support communication to an SE connected to the device that implements a TEE. This API defines a transport interface based on the Open Mobile API Specification [38]. The TEE SE API is particularly pertinent to those implementing use cases involving both a TEE and an SE.
- The GlobalPlatform Trusted Platform Services (TPS) Client API [39] is a draft specification that enables access to platform services offered by standardized secure components. GlobalPlatform and Trusted Computing Group members are collaborating to keep specifications compatible, such that the TPS Client API can be used to access a SE, a TEE, or a TPM.

End of Informative Comment

D. Protected Environment Implementation Examples

Start of Informative Comment

D.1 Protected Environment Isolated by Processor Features

In this example, a Protected Environment and the Rich OS share the same processor(s). The separation of a Protected Environment from the Rich OS is achieved by a processor mode or extension [11], [12], [13] similar to the mobile platform structure described in Annex A.1. Typically, a trusted operating system runs within a Protected Environment and hosts Trusted Applications.

During the boot sequence, the system initializes a Protected Environment first, so that the Protected Environment can complete its boot sequence without interference from code executing in the Rich OS. The Protected Environment also reserves the resources it requires, leaving the remainder for use by the Rich OS. When Protected Environment initialization is completed, the Rich OS boot process starts.

An example implementation of this Protected Environment model is a GlobalPlatform TEE [40]. A GlobalPlatform compliant TEE that includes anti-rollback features should be capable of meeting all Protected Environment requirements.

D.2 Protected Environment in a separate ASIC

In this model, a Protected Environment is not located in the same ASIC as the application processors. This example is similar to the mobile platform structure described in Annex A.3, in particular Model 3 of Figure 3. A communications link is necessary between the Applications ASIC and a Protected Environment ASIC.

This model involves a different boot process because the two ASICs are independent. If a Protected Environment ASIC does not support a Rich OS, a simple boot sequence is sufficient. For maximum security, in this case, a Protected Environment ASIC could boot from internal non-volatile memory. The Application ASIC follows a standard boot sequence as defined in Section 3. The Application ASIC can connect to a Protected Environment ASIC after a Protected Environment is established, in order to store measurements securely during Measured Boot.

An SE is an example of an external ASIC that could host a Protected Environment. SEs are highly secure ASICs designed to protect high value assets (such as financial secrets in credit cards or access secrets for Mobile Networks) in mobile platforms, from both hardware and software attacks. The interfaces and security mechanisms of SEs are defined by a number of ISO [41], ETSI [42] and GlobalPlatform [43] standards. Common Criteria certification [44] and compliance testing [45] are required for commercial products.

There is a set of GlobalPlatform standards [46] that define an execution environment within an SE that supports programs written in a subset of Java. SEs implementing these standards are known as JavaCards. The capabilities and protection level offered by JavaCards exceed the requirements for a Protected Environment, although the amount of secure storage provided is limited.

End of Informative Comment