Platform Security Model 1.0

Document number:	ARM DEN 0128
Release Quality:	Release
Issue Number:	0
Confidentiality:	Non-Confidential
Date of Issue:	27/05/2021

© Copyright Arm Limited 2021. All rights reserved.

Contents

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

About this document

This document is one of a set of resources provided by Arm that can help organisations develop products that meet the security requirements of PSA Certified on Arm-based platforms. The PSA Certified scheme provides a framework and methodology that helps silicon manufacturers, system software providers and OEMs to develop more secure products. Arm resources that support PSA Certified range from threat models, standard architectures that simplify development and increase portability, and open-source partnerships that provide ready-to-use software. You can read more about PSA Certified here: www.psacertified.org and find more Arm resources here: developer.arm.com/platform-security-resources.

This Security Model motivates the set of platform security architecture specifications that target internet and other forms of connected compute-centric devices. This document underlies the PSA Certified™ framework and certification scheme.

Key goals for designing devices based on essential security properties are described in this security model. These essential properties are used to define a Platform Root of Trust for a platform on which end-products can be built. This Platform Root of Trust covers the set of hardware and firmware necessary to support non-secure and secure processing.

A connected device needs to operate within an ecosystem. This document outlines how such a device and some of the essential security features might fit within an ecosystem. It does this by illustrating typical entities, capabilities and processes required to securely deploy connected services.

Release Information

The change history table lists the changes that have been made to this document.

Date	Version	Confidentiality	Change
May 2021	1.0 REL 0	Non-Confidential	Arm version of DEN 0079 PSA Certified 1.0 Release 0

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Platform Security Model 1.0

Copyright ©2021 Arm Limited or its affiliates. All rights reserved. The copyright statement reflects the fact that some draft issues of this document have been released, to a limited circulation.

Confidential Proprietary Notice

This Licence is a legal agreement between you and Arm Limited ("**Arm**") for the use of Arm's intellectual property (including, without limitation, any copyright) embodied in the document accompanying this Licence ("**Document**"). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this Licence. By using or copying the Document you indicate that you agree to be bound by the terms of this Licence.
"**Subsidiary**" means any company the majority of whose voting shares is now or hereafter owner or controlled, directly or indirectly, by you. A company shall be a Subsidiary only for the period during which such control exists.

This Document is **NON-CONFIDENTIAL** and any use by you and your Subsidiaries ("Licensee") is subject to the terms of this Licence between you and Arm.

Subject to the terms and conditions of this Licence, Arm hereby grants to Licensee under the intellectual property in the Document owned or controlled by Arm, a non-exclusive, non-transferable, non-sub-licensable, royalty-free, worldwide licence to:

(i) use and copy the Document for the purpose of designing and having designed products that comply with the Document;
(ii) manufacture and have manufactured products which have been created under the license granted in (i) above; and
(iii) sell, supply and distribute products which have been created under the license granted in (i) above.

Licensee hereby agrees that the licences granted above shall not extend to any portion or function of a product that is not itself compliant with part of the Document.

Except as expressly licensed above, Licensee acquires no right, title or interest in any Arm technology or any intellectual property embodied therein.

THE DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. Arm may make changes to the Document at any time and without notice. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PETMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENCE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE'S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF THE DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENCE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

This Licence shall remain in force until terminated by Licensee or by Arm. Without prejudice to any of its other rights, if Licensee is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to Licensee. Licensee may terminate this Licence at any time. Upon termination of this Licence by Licensee or by Arm, Licensee shall stop using the Document and destroy all copies of the Document in its possession. Upon termination of this Licence, all terms shall survive except for the licence grants.

Any breach of this Licence by a Subsidiary shall entitle Arm to terminate this Licence as if you were the party in breach. Any termination of this Licence shall be effective in respect of all Subsidiaries. Any rights granted to any Subsidiary hereunder shall automatically terminate upon such Subsidiary ceasing to be a Subsidiary.

The Document consists solely of commercial items. Licensee shall be responsible for ensuring that any use, duplication or disclosure of the Document complies fully with any relevant export laws and regulations to assure that the Document or any portion thereof is not exported, directly or indirectly, in violation of such export laws.

This Licence may be translated into other languages for convenience, and Licensee agrees that if there is any conflict between the English version of this Licence and any translation, the terms of the English version of this Licence shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. No licence, express, implied or otherwise, is granted to Licensee under this Licence, to use the Arm trade marks in connection with the Document or any products based thereon. Visit Arm's website at https://www.arm.com/company/policies/trademarks for more information about Arm's trademarks.

The validity, construction and performance of this Licence shall be governed by English Law.

Copyright © [2021] Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Arm document reference: LES-PRE-21585 version 4.0

Potential for change

The contents of this specification are subject to change.

In particular, the following may change:

- Feature addition, modification, or removal
- Parameter addition, modification, or removal
- Numerical values, encodings, bit maps

First draft, major changes and re-writes to be expected.

Feedback

Arm welcomes feedback on its documentation.

Feedback on this book

If you have comments on the content of this book, send an e-mail to arm.psa-feedback@arm.com Give:

- The title (Platform Security Model).
- The number and issue (DEN 0128 1.0 Beta 0)
- The page numbers to which your comments apply.
- The rule identifiers to which your comments apply, if applicable.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

1 Platform Security Model Overview

This Platform Security Model is one of the documents in a holistic set of deliverables that includes Threat Models and Security Analyses documentation, documents specifying security requirements and Application Programming Interfaces (API), all architecture agnostic. Together with an open source reference implementation and test suites, this enables consistent design-in at the right level of security.

This framework builds upon best practice from across the industry and is aimed at different entities throughout the supply chain, from chip designers, software vendors and device developers to cloud and network infrastructure providers. Though the focus is on local network or internet connected devices, many aspects are relevant for non-connected devices. It is the top-level document for the other platform security specifications and defines a common language, high-level robustness rules, and models.

Products may go through a security evaluation, such as PSA Certified, to provide a measure of the robustness of the implementation.

The set of core security goals given below provide a high-level, abstract, way to think about the essential features that are necessary to secure and establish trust. Abstraction allows these goals to be applied as required, for example, to an end user connected device, a hardware component, a software component, or a service. In describing the goals, the term *device* is used to represent any entity at any level that must be secure and trustworthy.

Goal 1: Devices are uniquely identifiable.

In order to interact with a specific device instance, that instance must be uniquely identifiable. The identity must be attestable and that attestation verifiable as a means of proving the device identity, see Goal 3.

Goal 2: Devices support a security lifecycle.

The security state of a device within its security lifecycle depends on software versions, run-time measurements, hardware configuration, status of debug ports, and on the product lifecycle phase. Product lifecycle phases include, for example, development, deployment, returns, and end-of-life. Each security state defines the security properties of the device. The security state must be attestable, see Goal 3, and may impact access to bound data, see Goal 9.

Goal 3: Devices are securely attestable.

The trustworthiness of a device must be established. This requires that its identity, see Goal 1, and security state, see Goal 2, are proven through attestation. To have validity, device identification and attestation data must be part of a device governance regime.

Goal 4: Devices ensure that only authorized software can be executed.

Secure boot and secure loading processes are necessary to ensure that only authorized software can be executed on the device. See also Goal 6. Allowing unauthorized software is acceptable only if such software cannot compromise the security of the device.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Goal 5: Devices support secure update.

Device software, credentials, programmable hardware configuration, must be updateable to resolve security issues or to provide feature updates. Updates must not compromise the device security. Authentication of an update is required. However, execution of any updated software must be authorized in accordance with Goal 4.

Goal 6: Devices prevent unauthorized rollback of updates.

Updates are necessary to resolve known security issues, or provide feature updates, see Goal 5. Preventing rollback, known as anti-rollback, to a previous version with a known (and subsequently fixed) vulnerability is essential. However, authorized rollback for recovery purposes may be allowed.

Goal 7: Devices support isolation.

Isolation of a trustworthy service from less trusted or untrusted services is essential to protect the integrity of that service. More generally, isolation boundaries aim to prevent one service from compromising other services, for example, between any on-device services and between on-device services and the connected world.

Goal 8: Devices support interaction over isolation boundaries.

Interaction over isolation boundaries, see Goal 7, is essential if isolated services are to serve a purpose. Any such interaction must not be able to compromise the interacting services or device. This will require validation of exchanged data. It may also be necessary to ensure the confidentiality and integrity of any data exchanged.

Goal 9: Devices support unique binding of sensitive data to a device.

Sensitive data, for example, user or service credentials, or secret keys, must be bound to a device to prevent disclosure outside of the device. It may also be required to bind such data to prevent disclosure beyond its owner. Inherently secure storage or confidentiality and integrity assured storage may be used. Where binding relies on cryptography and keys, see Goal 10, the keys are sensitive data and so must be bound to the device or the data owner. It may also be necessary to bind the data to the security state, for example, to deny access during debug, see Goal 2.

Goal 10: Devices support a minimal set of trusted services and cryptographic operations that are necessary for the device to support the other goals.

Trusted services may include configuration of the hardware to support security lifecycle (see Goal 2), isolation (see Goal 7), and cryptographic services that may use bound secrets (for example, keys) used to support attestation (see Goal 3), secure boot and secure loading (see Goal 4), and binding of data (see Goal 9). The trusted services must be kept as small as possible to enable analysis and reduce the likelihood flaws.

These goals inform and are embodied in the platform security architecture specifications that are designed to help in the development and deployment of secure products. It is recommended that all features are implemented. However, the features supported to secure a device are determined by, for example, the intended application domain and cost, by any applicable national standards, by ecosystem operators, and by any certification scheme. However, implementing the security features to increase the security level and provide product differentiation is a motivation behind this document.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

These goals inform and are embodied in the platform security architecture specifications that are designed to help in the development and deployment of secure products. It is recommended that all features are implemented. However, the features supported to secure a device are determined by, for example, the intended application domain and cost, by any applicable national standards, by ecosystem operators, and by any certification scheme. However, implementing the security features to increase the security level and provide product differentiation is a motivation behind this document.

1.1 Connected Devices Within an Ecosystem

Devices are expected to be deployed within the context of an ecosystem. The ecosystem provider is expected to define the requirements for a device based on technical, commercial, and regulatory requirements, and the security processes of the provider. These devices are expected to comply with a variety of functional and security requirements, depending on the use cases of the deployment ecosystem. A generic ecosystem is outlined in Figure 1, and the essential platform security elements are described in the following text.

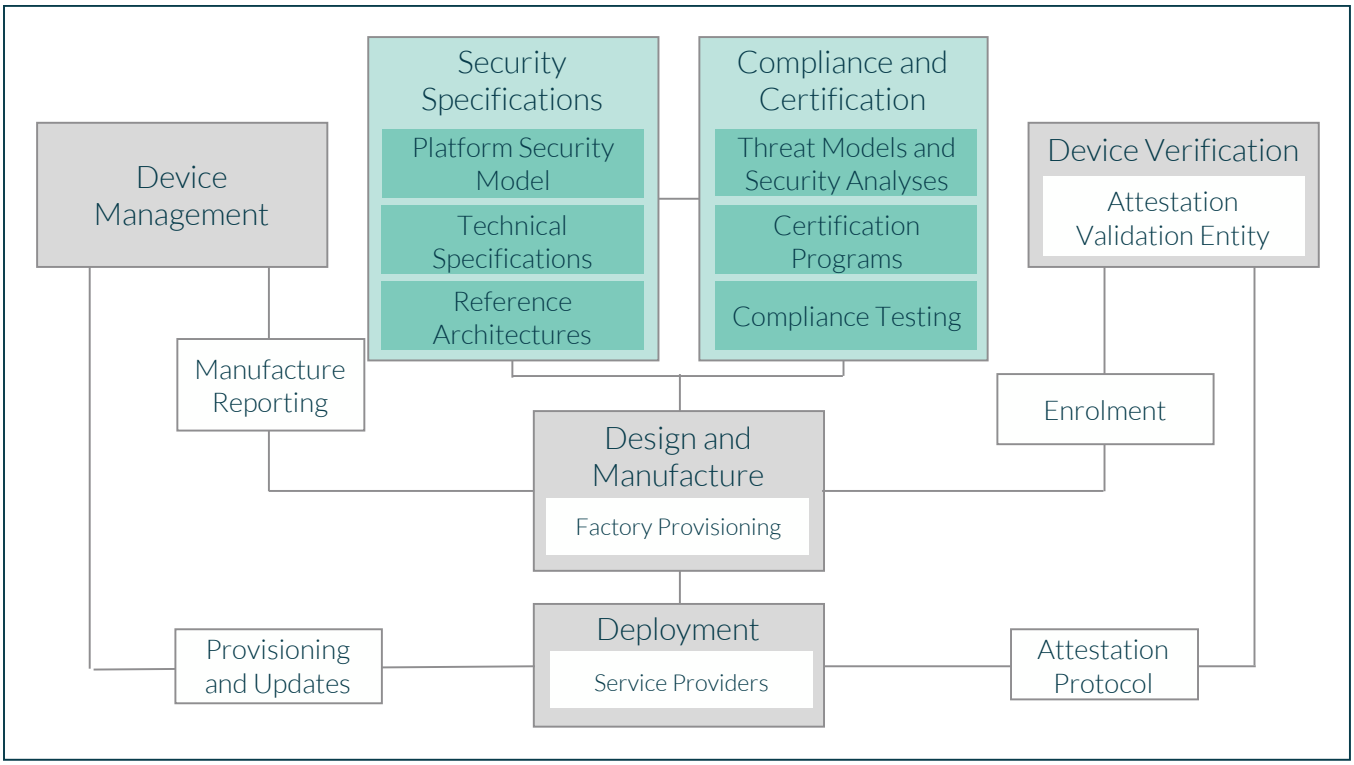

Figure 1: Generic Ecosystem

- Security specifications enable the design and deployment of compliant devices that are compatible with the ecosystem requirements:
    - The Platform Security Model, this document, defines the top-level security concepts, identifies and defines the Platform Root of Trust and generic Root of Trust services
    - Technical specifications define the security requirements, outline solution architectures and provide the necessary mitigations identified by Threat Model and Security Analysis (TMSA). This includes generic material and references other standards that can be applied to any implementation

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

    - Reference architectures are any other applicable specifications that define, for example, hardware and software functional and robustness requirements, and standardized functional interfaces
- Certification and compliance show that a deployed device is compliant and compatible with the ecosystem requirements
    - Threat Models and Security Analyses (TMSA) identify use case-specific security threats and motivate use case-specific security functional requirements
    - Compliance testing, for example PSA Functional API Certification, deals with interfaces, functional behavior, and interoperability
    - Certification programs, for example PSA Certified, assess the implementation of the security functional requirements of a compliant device for vulnerability against the identified threats. The robustness that is required is based on use case, cost and security trade-off and the assessment scope. Certification schemes are an assessment and not a guarantee that a device is free from vulnerabilities

Figure 1 also illustrates the following elements that are typical in an ecosystem, but not defined by in this security model:

- Design and manufacture: Secure-by-design devices should be designed and then manufactured based on security specifications with the aim of achieving robustness certification and functional compliance. Device manufacture includes the provisioning of root secrets and other sensitive information. See section 3.
- Deployment: Service providers manage and support deployed devices through:
    - Device management: Device manufacturers provide device manufacture data, provisioning, firmware update services, and other support functions. Device management considers devices throughout their lifetime, from factory provisioning through deployment, any re-deployment, in-field analysis, repair, to end-of-life. Data specific to the device management system must be defined by that system. The storage of such data may make use of the Platform Root of Trust services
    - Device verification: Devices are enrolled with a device verification system, including attestation verification. Depending on ecosystem requirements, device and attestation verification services are expected to be deployed by manufacturers, service providers, or by industry consortia

1.2 Device Model

The generic device model, shown in Figure 2, defines a Secure Processing Environment (SPE) and a Non-secure Processing Environment (NSPE). This security model requires the SPE to be isolated from the NSPE, see section 2.3. Such isolation is also a requirement of PSA Certified.

The SPE hosts the Platform Root of Trust, the Platform Root of Trust Services and any Application Root of Trust Service(s).

- The Platform Root of Trust (PRoT), which is discussed in section 2, consists of:
    - The Immutable Platform Root of Trust, which is inherently trusted and never changes on a production device
    - The Updateable Platform Root of Trust, which is trusted only through verification that is anchored to the Immutable Platform Root of Trust. See section 4

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

- A security subsystem that the PRoT relies on for protection of its assets or that implement some of its services is called a Trusted Subsystem. Examples include secure elements, TPMs and SIMs, secure peripherals

- Application Root of Trust (ARoT) service(s) provide application-specific trusted services and are not defined in this document. They are expected to use interfaces provided by the updateable PRoT (but have no direct access to immutable PRoT components). Application Root of Trust services execute in the Secure Processing Environment and are expected to be in Secure Partitions (see section 2.3.2)

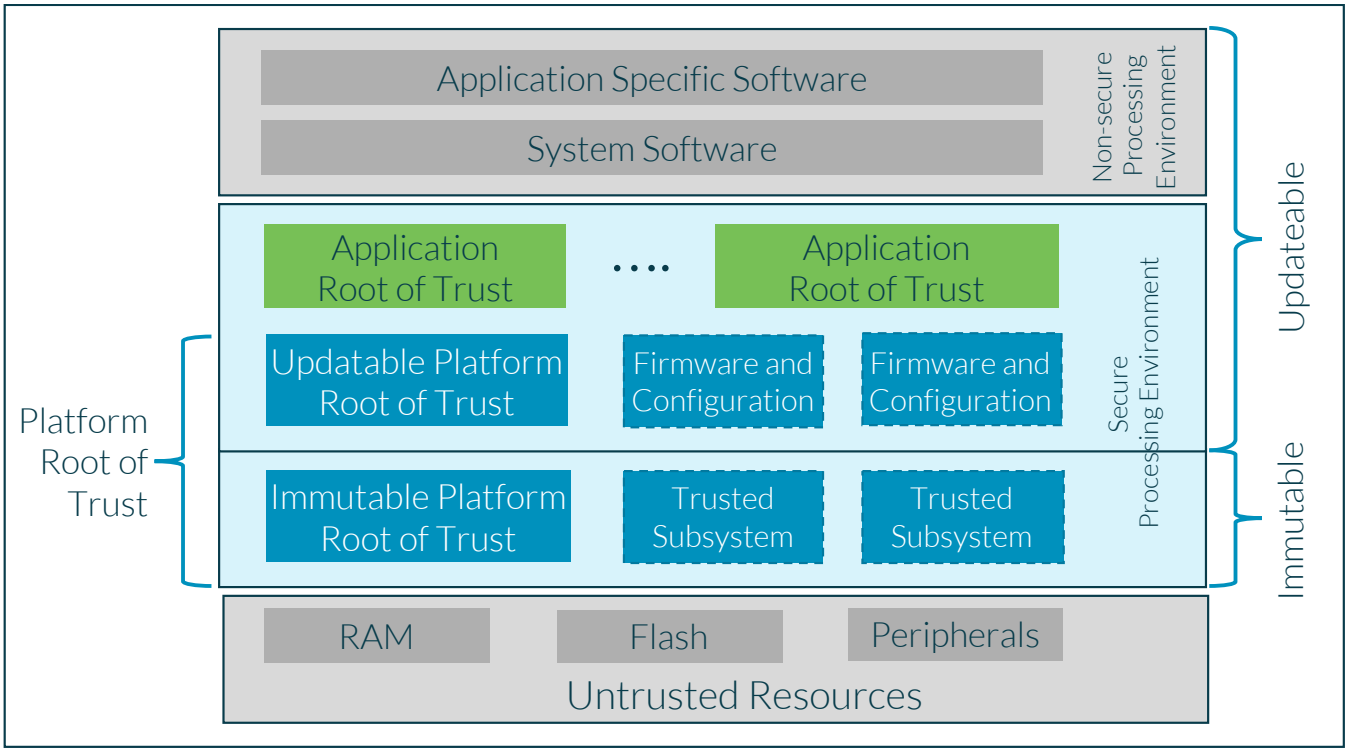

Figure 2: Device model

The Platform Root of Trust includes the hardware and software components that implement the services that are covered in this document. Both the following properties are required:

- Transaction accessibility, which means that access to transactions between the components is restricted to only those components that need access, for example:
  - When on the same die, access could be permitted through physical topology or secure configuration that is set and protected by the PRoT, or
  - Where physical access cannot be denied, cryptography can be used to ensure that the transactions are protected in integrity, confidentiality, and against replay
- Inseparability, which means that the unauthorized substitution of a component is not possible, for example:
  - When on the same die, the hardware components are physically inseparable, but
  - Where on more than one die, more generally where inseparability cannot be denied, then separation results in failure in a safe and secure way, for example, using cryptography to ensure that the transactions are protected in integrity, confidentiality, and against replay

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

The Non-secure Processing Environment (NSPE), shown in Figure 2, hosts all the non-secure system software and application-specific software components that provide the required device functionality. Typically, the system software may comprise an operating system or some run-time executive, together with any middleware, standard stacks and libraries, chip-specific device drivers, etc. It may make use of the interfaces that are provided by the PRoT services. The application-specific software may use the interfaces provided by any associated Application RoT service and the interfaces that are provided by the PRoT services. The functionality of the NSPE is not covered by this security model.

Figure 2 also shows untrusted resources, these are not part of the PRoT nor of the Application RoT, and may include random access memory, flash and peripherals.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2 Platform Root of Trust

The Platform Root of Trust is the primary root of trust on a device. Figure 3 illustrates the minimal set of required services, which are described in this document, and a typical split between immutable and updateable implementations. This split is assumed in section 2.1 and section 2.2. There is no intentional mapping in Figure 3 to any specific implementation or application programming interfaces. If they do not compromise the integrity, security, or functionality of the services that are defined in this document, additional application-specific services may be provided.

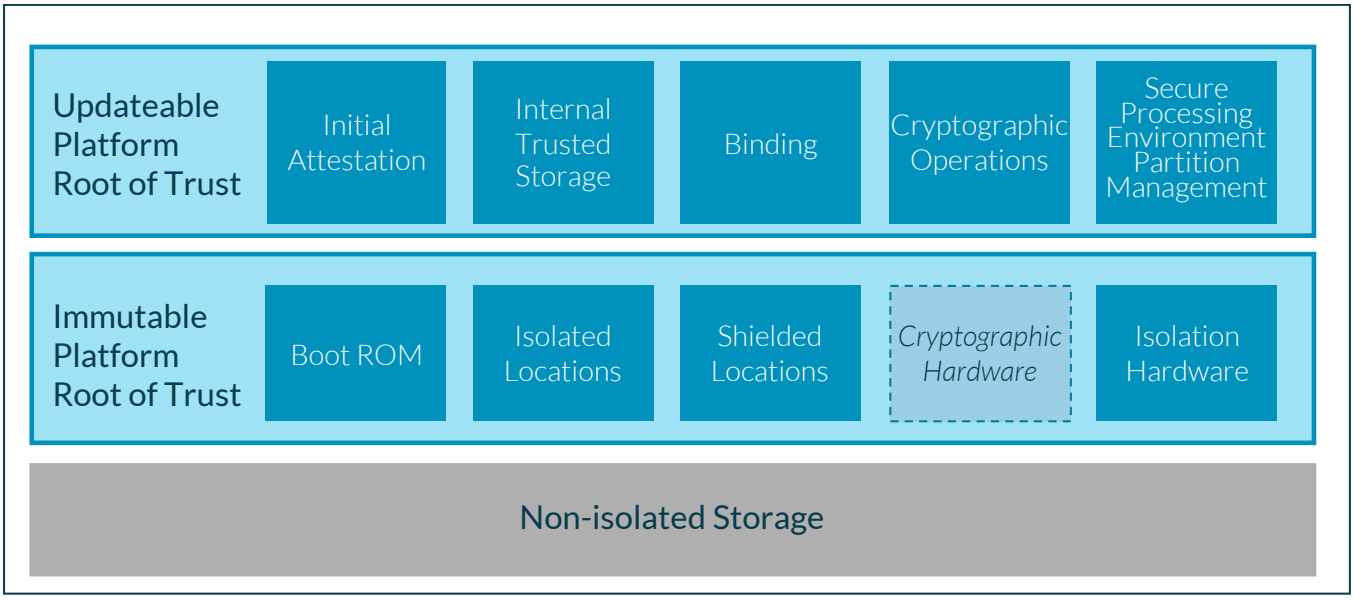

Figure 3: Minimal set of platform security services

2.1 Immutable Elements

The Immutable Platform Root of Trust includes the following elements:

- The Boot ROM contains the first code to execute after release from reset. This means the Boot ROM is the ultimate root of trust of the platform and device. The Boot ROM must be on-chip, and is typically implemented as Mask ROM, locked OTP, or locked on-chip flash. See section 4
- Isolated Locations, which contain sensitive data and are non-volatile memory, typically applies only to very limited areas of on-chip storage but may apply to data that is held in, for example, external secure element devices. Some stored data may be updateable
    - Access should only be through the Internal Trusted Storage service. See section 5.1
    - There should be no access via debug interfaces, including JTAG, and any scan chains
- Shielded Locations are tamper-resistant Isolated Locations. Shielded Locations are intended to hold provisioned secrets, including PRoT secret parameters. See section 2.4. However, the use may depend on certification requirements, deployment requirements, or to mitigate threats that are identified during threat analysis. Tamper resistance may include mechanisms to make active probing difficult, for example, physical disassembly for access to internal buses and interfaces. Side-channel analysis, for example, power and timing, may also form part of the certification requirements

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

- Isolation Hardware is the hardware that is necessary to implement the isolation requirements detailed in section 2.3. The extent of the isolation depends on any certification scheme, deployment requirements or to mitigate threats identified during threat analysis. Control and configuration of the hardware that isolates the SPE should be by the PRoT

Any implementation interfaces to these elements should be available only to the Updateable PRoT implementation.

2.2 Updateable Elements

The Updateable Platform Root of Trust includes the following elements:

- Secure Processing Environment Partition Management, which enforces SPE partition level isolation-based access control policies. See section 2.3.2
- Internal Trusted Storage (ITS), which provides secure partition-based access control to Isolated Locations and Shielded Locations. See section 5.1
- Binding, which allows keys that are used by the Crypto Operations to be bound to an owning secure partition, a device instance, and the security state of the device, see section 5.2. This allows a variety of secure storage services to be to be implemented at the application level, see section 6
- Cryptographic Service, which provides generic cryptographic operations, supports key hiding, and restricts the use of specific keys to specific PRoT services and to owning secure partitions, Application Root of Trust services and the Non-secure Processing Environment. See section 5.3
- Initial Attestation Service allows a validation entity, as shown in Figure 1, to validate the trustworthiness of the Platform Root of Trust, its implementation, and updateable components loaded during secure boot. See section 5.4

Any implementation interfaces to these elements should be available to Application RoT(s) but may also be available to the Non-secure Processing Environment.

2.3 Isolation

Isolation ensures that less trusted software cannot compromise more trusted software. More generally, this means that software in one component cannot compromise the code, run-time state, and secrets of any other component. Figure 4 shows the device isolation model.

2.3.1 **Temporal Isolation**

Code running before a temporal isolation boundary must complete its task and then handover to the stage after the temporal boundary. It may leave state for use by the code that executes after the temporal boundary. This state should be stored on-chip because external storage exposes the state to attack. Code that executes after a temporal boundary can access only the state that is left by the previous stage. Note that re-use of code to reduce the overall footprint may be acceptable, if it does not affect the state and does not expose any sensitive data.

Temporal isolation is used only in secure boot, as shown in Figure 4. Temporal isolation protects sensitive device assets that are used during the secure boot process, see section 4, from access by later stages. This assumes that secure boot and the later stages are performed on the same processor. Where separate processors on the same chip are used, then the objectives of temporal isolation may be met by, for example, physical isolation. For convenience, the term temporal isolation is used in this document regardless of the implementation.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

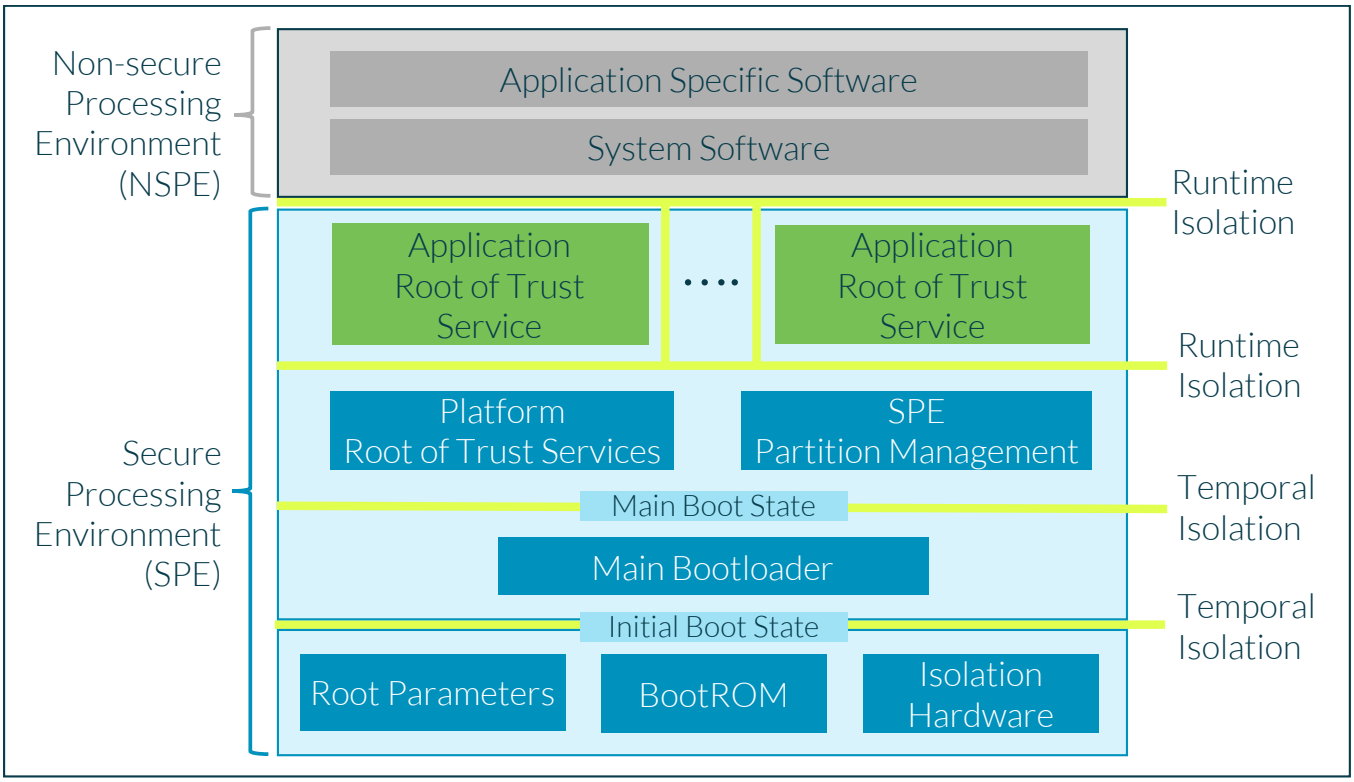

Figure 4: Isolation model

2.3.2 Partition Isolation

A partition is defined as the logical boundary of a software entity, where intended interaction across the boundary is only via defined interfaces. Software within a partition is not necessarily isolated from software in other partitions, however, software should be designed assuming that all isolation boundaries are enforced. This encourages the development of robust and portable code, even when not all isolation boundaries are enforced.

Secure partitions are applicable to the Secure Processing Environment (SPE) and is required in this security model. Isolation between partitions within the Non-secure Processing Environment is not required by this security model, though is recommended where supported.

SPE partition-based run-time isolation applies to components that execute following secure boot and initialization of the SPE Partition Management function. Figure 4 shows the following three run-time isolation boundaries:

- Secure Processing Environment (SPE) isolation. The PRoT and all Application RoT service(s) are run-time isolated from the Non-secure Processing Environment (NSPE). This is a mandatory requirement for this security model and for PSA Certified
- PRoT isolation. The PRoT is isolated from the Application RoT service(s), in addition to the SPE isolation. Support for this isolation boundary depends on the device requirements and on any applicable PSA Certified assurance levels
- Application RoT services are isolated from each other, in addition to SPE isolation and PRoT isolation. Support for this isolation boundary depends on the device requirements and on any applicable PSA Certified assurance levels

A secure partition is expected to host one or more related root of trust services, that is, services that share underlying functionality or data. It is recommended that unrelated services are in separate secure partitions. A

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

secure partition must only host services that are either completely within the Platform Root of Trust, or completely within an Application Root of Trust. A secure partition cannot include anything within the Non-secure Processing Environment.

This security model requires enforcement of these boundaries through hardware. Control of the isolation hardware, which is implementation specific, is the task of the SPE Partition Management function.

2.4 Platform Parameters

A device platform requires at least the immutable parameters, or equivalent, listed below and in Table 1.

- Implementation ID; public data that uniquely identifies the implementation. Typically, this will allow identification of the device manufacturer, the model and version number, and any other data necessary to uniquely identify the Immutable PRoT
- Hardware Unique Key; a secret key unique to each device instance that is used to derive other device unique secrets and to bind data to that instance (see section 5.2)
- Boot Validation Key; is used for authentication during secure boot, see section 4
- Initial Attestation Key; this should be the private part of an asymmetric[1] key-pair accessible only to the Initial Attestation service, see section 5.4. To prevent cloning or spoofing, this key must be unique to each device instance or to a collection of identical devices. They should not be shared between different versions of a device, between different devices, or between manufacturers
- Instance ID: a public value that identifies the Initial Attestation Key

Table 1: Platform parameters

Parameter	Initial param	Updateable param	Security class	Recommended provisioning
Implementation ID	Yes	Yes	Public	Isolated Location
Hardware Unique Key	Yes	No	Secret	Shielded/Isolated Location
Boot Validation Key	Yes	No	see section 4	Isolated Location or Boot ROM
Initial Attestation Key	Yes	Yes	Secret	Derived or Shielded/Isolated Location
Instance ID	No	Yes	Public	Derived or Isolated Location

As a consequence of temporal isolation boundaries for secure boot, see sections 2.3.1 and 4.4, the parameters are categorized as follows, based on whether each is secret or public. Secret means that the data that must not be accessible to unauthorized agents, and public means that the data may be shared inside and outside the device.

- Initial parameters are those used by the Immutable Root of Trust. Depending on the certification profile, initial parameters are stored in Isolated Locations or Shielded Locations. They should only be directly

[1] The ecosystem may permit symmetric cryptography if the device cannot support asymmetric operations.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

accessible or useable by the Immutable Root of Trust code, see section 2.1, and can be copied to, or used as seeds for the derivation, of Initial Boot State data, see section 4.4.1

- Updateable parameters are those that are used by the Updateable Root of Trust services. They should only be directly accessible by the Updateable Root of Trust code, and can be copied to or derived and stored in the Main Boot State, as required, see section 4.4.2.

Direct use of a Shielded Location or Isolated Location, or using derivation, is implementation specific. In general, derivation can be more flexible and future proof. Derivation can support additional strategies for protecting secret parameters and may reduce the risk of exposure of secret parameters during device manufacture. However, derivation may require additional computational resources at boot.

2.5 Trusted Subsystems

A trusted subsystem is any configurable or updateable hardware security function, possibly containing a processing element, that the PRoT relies on for correct operation. All configuration and updates must be performed by the PRoT.

A trusted subsystem may have its own root of trust and its own security lifecycle. However, a trusted subsystem must always be subordinate to the Platform Root of Trust. This means that its configuration and state must always be attestable by the Platform Root of Trust. See section 5.4.

2.6 Non-isolated Storage

Non-isolated storage, as shown in Figure 3, is storage where access control either logically or physically (if off-chip or removable) cannot be enforced by the PRoT. non-isolated storage is useful for application specific data. Section 6 outlines how to protect any sensitive data through the use of binding, see section 5.2 and the cryptographic service, see section 5.3.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3 PRoT Security Lifecycle

A lifecycle defines the states of an object through its lifetime. Each security state in the security lifecycle of a device defines the security properties in that state. Security state can depend on:

- Software versions
- Run-time status such as data measurements, hardware configuration, and status of debug ports
- Product lifecycle phase, for example, development, deployment, returned to manufacturer, or end-of-life

The generic security lifecycle that is shown in Figure 5 is intended to capture the minimum set of lifecycle states and transitions for the Platform Root of Trust. This section is primarily concerned with how platform security data and secrets may be compromised when debug is enabled, and how to ensure valid attestation only in trustworthy states. The states and their properties are defined in this section.

Other components in a system may have their own lifecycles, for example, trusted subsystems, the system software, and application software. The overall product built from the components may also have its own lifecycle. Such lifecycles, and any associated data, are implementation or application specific and out of scope of this document. Note that such lifecycles must never be in a state that conflicts with or compromises the security requirements of the Platform Root of Trust security lifecycle.

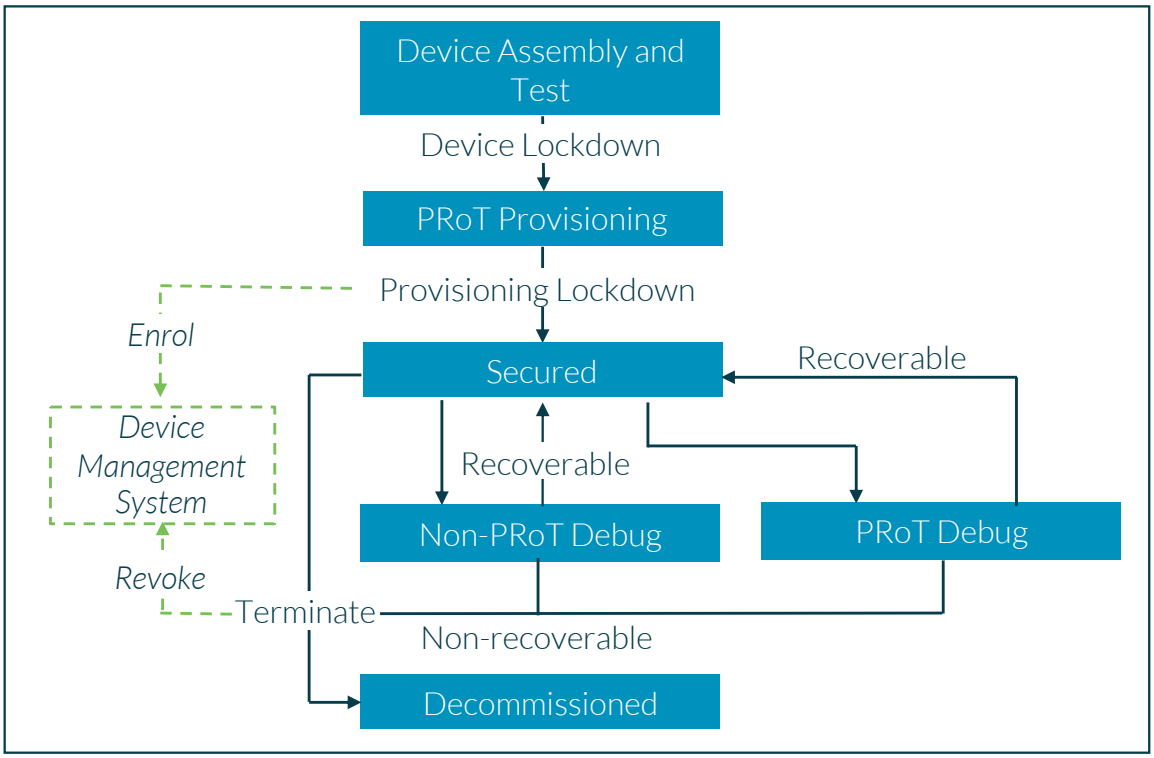

Figure 5 : Generic Platform Root of Trust security lifecycle

- Device assembly and test: The device will be running manufacture and diagnostic software. This is not a secure state and therefore is not an attestable state
    - Manufacture and diagnostic software should not be signed by the same authority as final production software

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

    - Hardware debug interfaces may be open, secure boot may not be enabled
    - Test secrets and identities may be present. Production platform security parameters must not be present
- PRoT Provisioning: The device is programmed with production platform security parameters, see Table 1[1], and is configured so it can enter the Secured state. This, though, is not a secure state and so not an attestable state
    - Production platform security parameters are provisioned and locked to prevent modification. Random values that are generated on the device should use a random source with sufficient entropy. Where parameters are provisioned separately, for example on a secure element, then it should not be possible for any device software to access or perform any operations with them until the device has reached this state
    - Hardware debug ports should be disabled, or an access control mechanism activated
    - Secure boot should be enabled
    - Only signed production software should be present. Manufacture and diagnostic software must not be present
    - Once the device has been provisioned it may be enrolled with a device management system. This will declare the existence of the device to the ecosystem
- Secured: The PRoT is fully operational and secured. This is the primary security state for most of the life of the device. Only in this state do the platform security parameters become available to any device software. Provisioning of any application-specific data that is to be protected using PRoT services can now be performed. This is expected to require manufacture reporting for tracking and identification of fully secured devices. This is a secure state and is an attestable state
- Debug: Figure 5 shows non-PRoT debug and PRoT debug states. If entering either debug state is supported once a device is in the Secured state, then a system reset is required. This is because the device includes production secrets and the act of reset requires the removal of, or denial of access to, any previous data and derivation of Secure state sensitive data. Only if the device can still be considered trustworthy after debug can it be returned to the Secured state; this is called recoverable debug. Otherwise, debug is considered as non-recoverable, see section 3.1
- Decommissioned: Entering this state requires a system reset. This is because the device includes production secrets and the act of reset requires the removal of any previous data. This is not a secure state and so not an attestable state
    - Production platform security parameters secrets should be permanently inaccessible, denying attestation and access to any bound data. The overall product lifecycle may require non-secret platform identities to remain accessible
    - It is not possible for the device to leave the Decommissioned state unless a full factory reset is possible, and the device becomes indistinguishable from a new device. This means that the device can be securely re-provisioned with a new set of platform security parameters stored in Isolated Locations or Shielded Locations, or in the Boot ROM, effectively resulting in a new device instance
    - Revocation of the device by the device management system can be used to ensure that the device is no longer operational. This is necessary even if the device can be factory reset

[1] This may include the Immutable PRoT firmware if not a design-time mask ROM.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Additional states and sub-states may exist in any real process. Any sub-state must retain the properties of its parent. Both parent and sub-states must always be attestable once a device enters an attestable state. The security state is included in the PRoT attestation, see section 5.4. Therefore, it is not permitted for the security state to change without a device reset if either the current state or the new state is attestable.

3.1 Debug

It may be necessary to debug a device that has reached the Secured state in its lifecycle, see Figure 5.

When the device is in Secured state, logging of events or reporting of diagnostics in a way that does not expose sensitive data and does not affect the trustworthy operation of the device is permitted. The ecosystem may require that such logs and reports are only accessible by a device management system, and are integrity and confidentiality protected. This is called non-intrusive debug.

Debug that is more intrusive than event logging depends on the access level that is granted to the debugging agent. For example, it is possible that device-sensitive data can be compromised or that secure boot may be circumvented, meaning that the device is no longer trustworthy. Table 2 identifies the scope of various forms of intrusive debug. Depending on the hardware implementation, not all will be supported, which limits the debug options that are available to the manufacturer.

Table 2: Intrusive debug

Debug State	Scope	Secure and Attestable State	Security Implications
Non-PRoT Debug	Non-secure Processing Environment	Yes	Compromises only NSPE operation and data. Application RoT and PRoT remain intact and trustworthy.
	Application RoT	Yes	Compromises only Application RoT operation and data. PRoT remains intact and trustworthy.
PRoT Debug	Non-recoverable	No	Compromises PRoT operation and parameters, which should be made permanently inaccessible.
	Recoverable	No	May compromise the PRoT operation but cannot compromise the platform security secret parameters that are stored in Isolated Locations or Shielded Locations.

Intrusive forms of debug should be restricted to authorized debug agents and require a secure authorization process, with the debug rights granted by an appropriate device management service. This process is beyond the scope of this document. However, other platform security specifications provide guidance. Non-PRoT debug is beyond the scope of this document.

Debug of the PRoT is the most intrusive. When the PRoT debug state is entered it must not be possible to issue an attestation token that is signed using the Initial Attestation Key, see 5.4.1. This is because the device is no longer in a trustworthy state. Also, it must not be possible to derive production binding keys in order to prevent access to securely stored data, see section 5.2.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Recoverable PRoT debug applies to devices that can deny access to production platform security secret parameters when in PRoT debug state and can be restored to a fully trustworthy and attestable state on exit, provided the debug activity has not compromised the Secured state.

Non-recoverable PRoT debug applies to devices that are not able to protect the platform security parameters and are not able to ensure that the debug has not compromised the Secured state. This type of device must make the platform security parameters permanently inaccessible on entry to debug. The only valid next state is Decommissioned.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

4 PRoT Secure Boot and Firmware Update

All devices must support a secure boot flow to ensure only authorized software can be executed on the device. Secure boot, sometimes called verified boot, uses cryptography to verify the next stage code and any metadata. Execution of the next stage proceeds only if any validation checks on the verified metadata pass, for example, version comparison, see section 4.3. The secure boot flow must apply to all the firmware and software in the SPE. It should also apply to the first NSPE image loaded. Note that in some contexts, the term Measured Boot is used, however, this involves measuring the code, typically for attestation, but without any verification and validity checks.

The secure boot flow must start with an inherently trusted Boot ROM in the Immutable PRoT; this is the trust anchor for the boot validation chain. Both the following are recommended, as shown in Figure 6:

- The Boot ROM is small, simple, and verifiable. This minimizes the risk of a vulnerability that cannot be corrected once on the chip, and
- The complex steps are handled by a main bootloader, which is subject to validity check by the Boot ROM, because it can be corrected through a secure firmware update process, see section 4.5

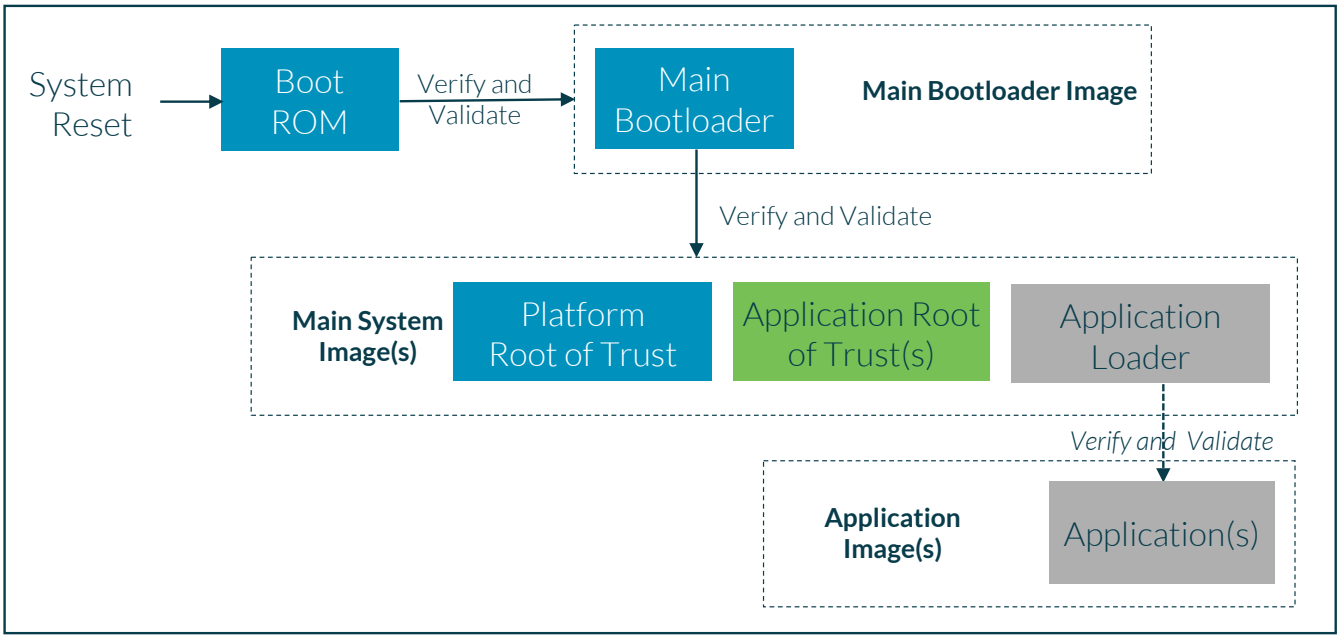

Figure 6: Example Secure Boot Flow

The example in Figure 6 shows the Main Bootloader verifying and validating the Platform and Application Root of Trust codes. Figure 6 also shows that the NSPE First Stage Loader is also verified and validated, and that it verifies and validates each of the NSPE images. Actual implementations may include more boot stages, or support multiple images to allow for incremental updates, or to support supply chains with different signing entities. Any such variation must meet the following security properties to establish a secure boot chain:

1. A device must always boot from a fixed address in the Boot ROM following system reset. This is because it acts as the trust anchor for the secure boot chain. The term system reset is used to describe a complete system reset, including any trusted subsystems, see section 2.5. No run-time state from before the reset should be retained or used, except where necessary if suspend or hibernate are supported, see section 4.6

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2. The Boot ROM verifies and validates all images that are associated with the next stage before executing any next stage code. The immutable Boot Validation Key (BVK), see Table 1, is used. The BVK is the public part of an asymmetric key pair. Despite being a public key, it is good practice to consider a BVK as secret. The Boot ROM may use a stored hash value for the second and subsequent verification of the same image, see section 4.1
3. The next stage verifies and validates all data and images that are associated with the following stage before executing any of the following stage code. This process is repeated until all the chained images have been verified and validated. The keys required and any management of anti-rollback, see section 4.3, are not defined by the Platform Root of Trust. However, following the principles that are outlined in this document is recommended

4.1 Image Signing, Validation and Encryption

All images should be signed using asymmetric cryptography and must be verified before use. Each signature must cover at least the image content, for example code and data, any critical parameters, for example, entry point for the code, and the version number, see section 4.2.

Asymmetric cryptography during boot can be too time-consuming for some applications. Securely storing the computed cryptographic hash value of the images verified during an initial secure boot asymmetric check allows the stored value to be used to reduce the time on subsequent checks. Using a saved hash value in this way is called Hash Locking. For the hash to be secure, the stored value must be accessible only by the Boot ROM. To prevent the execution of a different image, the stored hash value should be integrity protected. To prevent the use of an old image, the integrity-protected stored hash should also be replay protected. To support update, the stored hash value must be updateable.

The private counterpart of the on-device Boot Validation Key, see Table 1, must be held securely by the operator or device manufacturer, and should never be stored on the device. This applies also to the private counterparts of any other on-device keys that are used in the validation of the boot chain.

Symmetric signing is not recommended. This is because any disclosure of the key from a compromised device allows any image to be signed. If symmetric signing is unavoidable then the key must be unique per device, thus compromising one device does not allow images for another to be signed.

Boot images should be encrypted if they contain sensitive data or code. This requires a secret Boot Decryption Key that should be available only to the Immutable PRoT. That key may be stored in an Isolated or Shielded Location, or derived from another Isolated or Shielded Location. It is normal for the image to be signed then encrypted, thus the device decrypts the image before verification and validation. However, if the content is considered confidential and should not be visible to the signing entity then encryption followed by signing is necessary.

4.2 Verified and Version Validated Components

Regarding the example in Figure 6, the verified components column of Table 3 shows the minimal set of verified components. If a component comprises multiple sub-images, each image should be verified separately. For the version checked components columns, see section 4.3.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3: Verification and version check requirements

Component	Verified components		Version checked components	
	Secure boot verification	Hash recorded for attestation	Anti-rollback check	Version recorded for attestation
Boot ROM	N/A	N/A	N/A	Yes
Main Bootloader	Yes	Yes	Yes	Yes
Platform Root of Trust				
Trusted Subsystems				
Application Root of Trust				
Application Loader	Yes	Yes	Yes	Yes

4.3 Anti-rollback

Anti-rollback is used to reject earlier versions of the firmware, software or data that may contain known errors or vulnerabilities. Secure boot must only allow components that have the same or newer (typically higher) version number than the reference version number for that component to be executed. To ensure that the component version number is valid, it must be included in the signature of the component and verified before use. The verified version number of each software component must be compared against a reference version number as part of a secure boot process. To ensure the integrity of the reference version number, it is, typically, stored in an updateable Isolated Location.

The Boot ROM must include an anti-rollback check on all images that it verifies on devices in the Secured lifecycle state, see section 3. Policies for updating the reference number used by the Boot ROM are covered in section 4.3.1 and section 4.3.2. Subsequent stages in the secure boot chain should include an anti-rollback check on the images that are validated by that stage. The principles that are discussed in section 4.3.1 and section 4.3.2 can be applied.

Regarding the example in Figure 6, the minimal set of version checked components is given in the Version Checked Components columns of Table 3.

For the purpose of attestation, see section 5.4, the version of the Boot ROM must also be recorded. If components are not individually versioned, they should be recorded with the version of the overall image.

4.3.1 Anti-rollback Reference Version Update

The reference version number must be updated only within a secure boot process. Operational requirements will determine when this is done. For the version checks that are performed by the Boot ROM, the following methods are recognized:

- Automatic update on reset: The Boot ROM updates the anti-rollback reference version when it has successfully loaded a newer version[1]. Success means that the image has passed all secure boot checks

[1] Where the Boot ROM can differentiate an intentional reset from a failure-induced reset, it may choose not to update the reference counter in the case of a failure reset and so fall back to the last known good version.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

- Update on command. The anti-rollback reference version is updated in response to a secure message from an external device management service. This means that the existing version is revoked only after the device management service signals that the newer version has no identified faults

The first option does not support a legitimate rollback if the new image boots correctly, but some aspect of its function is later found to be broken and the service provider decides that it is necessary to rollback to an earlier version.

The second option means that the service provider decides when to revoke the previous version after rolling out a newer version to the device. This option may also signal which version, that is newer than the current anti-rollback reference counter, to load if more than one version is available. A secure messaging service is required to ensure that the messages are from a trusted command issuer and are replay protected. This ability to legitimately rollback leaves devices susceptible to illegitimate rollback for longer. This ability also makes devices susceptible to denial of service attacks that block the update message.

To ensure that a device remains bootable, the device should never set the reference version counter to a value that is higher than the value of the newest image that is available.

4.3.2 **Anti-rollback Reference Number Reset**

Where the chip uses a technology that allows the reference version number to be reset, the device can support a factory reset mode or can be signalled from a device management service to reset the reference number. This requires a secure messaging service[1] to ensure that messages are from a trusted command issuer and are replay protected.

4.4 Boot State

Boot state refers to the data intentionally left at a temporal isolation boundary (section 2.3.1) in the secure boot flow. There are two isolation boundaries defined, as shown in Figure 6, leading to an Initial Boot State and a Main Boot State.

4.4.1 Initial Boot State

The initial boot state is the set of data that is provided by the Boot ROM for use by the Main Bootloader. Initial boot state includes the following:

- A random boot seed that is generated on each system reset. This can be used by later services, for example, to allow an attestation validation entity to ensure that attestation reports for different Attestation End Points, see section 5.4, were generated in the same boot session
- For each component that is validated by the Boot ROM, see Table 3, at least the following component information must be captured:
    - Version
    - Signer Identity[2]
    - The measurement that is made by the Boot ROM

[1] It must not be easier to subvert the messaging protocol than to subvert the secure boot. This means that authentication of the command issuer should have at least the same cryptographic properties as the ones that are used for image signing.

[2] The Signer Identity, if present, would typically be in image metadata.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

- Any Initial Parameters, see Table 1, that are required by later stages must be copied from Isolated Locations or Shielded locations, or be derived as appropriate. This is to comply with the parameter visibility rules. See section 2.4.

Initial boot state must be only directly accessible to the Main Bootloader, and should not be modified once set by the Boot ROM.

4.4.2 **Main Boot State**

The main boot state is the set of data provided by the Main Bootloader for use by the Platform Root of Trust Services. The main boot state includes the following:

- The initial boot state data, see section 4.4.1
- All platform security parameters, see Table 1, must be copied to the main boot state, to comply with the parameter visibility rules. see section 2.4
- For each component that is validated by the Main Bootloader, see Table 3, at least the following component information must be captured: version and signer ID[5] and the measurement that is made by the Main Bootloader code

Main Boot State must be only directly accessible to the Platform Root of Trust services, and to ensure that the data are trusted, should not be modified once set by the Main Bootloader.

4.5 Firmware Update

Update of firmware is crucial for fixing security vulnerabilities and enhancing the features of devices that are already deployed. It is essential that the update mechanism cannot be used to compromise the device with unauthorized software.

The selection and download of updates depend on the specific ecosystem and is not defined here. Ideally:

- Downloads are over a secure connection from an authorized repository. This helps to prevent networked attackers from sending arbitrary images, though this may make it impractical for future operators to use a different repository. The download may be encrypted
- Downloads are authenticated and integrity checked at the time of download, though this may be impractical in some constrained systems. The checks should follow the concepts that are covered in the other parts of section 4

Where resources permit, a banking scheme is recommended. This ensures that a bootable image is always available until that image is revoked through increment of the anti-rollback reference counter, see section 4.3.1. This requires enough non-volatile memory to store at least two copies of the images and any dependencies.

In all cases, a new image can only be executed after it has been validated by the secure boot process as covered in this section 4.

4.6 System Suspend and Hibernation

Suspension and hibernation are low power modes that allow a device to resume from a known point more rapidly than a full system start (a cold boot). Support for either is optional.

Suspension and hibernation both require the saving of enough state before going into the low power mode to allow resumption when the power is re-applied. Suspension typically means that some of that state is held in an always-on-power domain on the chip, and any dynamic RAM is placed in self-refresh mode. Hibernation typically means that all the state is written to some persistent storage before the power is removed. Some of this storage must be on-chip to deny substitution and ensure integrity and freshness.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

The state that is necessary for resumption may be accessible when the device is suspended or hibernated. This introduces the risk that it can be modified, breaching the principle that a system that has been suspended or hibernated should be indistinguishable from one that has not. Therefore, that state must be subject to integrity and replay checks on resumption. It may also need to be subject to privacy protection too, for example, secrets may need to be erased or encrypted.

In general, deciding whether to suspend, hibernate, or shut down can be performed by application code. For suspension or hibernation, the creation and signing of the required resume state, and the process of entering the low power mode, must be performed by trusted code. On resume, validation of the saved resume state must be performed by trusted code anchored in the Boot ROM.

If the integrity of any of the saved resume state is invalid, or replay is detected, then the Boot ROM must proceed with a full system restart.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

5 PRoT Elements

This section covers the Platform Root of Trust elements identified in section 2.2 and shown in Figure 3.

5.1 Internal Trusted Storage

Internal trusted storage provides access to data that is stored in Isolated Locations and Shielded Locations, see section 2.1. Individual data objects have an owning secure partition. Only the owner of a secure partition can request access to or modification of its data.

Access to sensitive data may depend on the security lifecycle state, see section 3. For example, it may be necessary to deny access to debugging agents when in debug state. Access for other lifecycle states is application dependent.

Implementation may rely on the physical access properties of Isolated Locations or Protected Locations, or by cryptographic binding, see section 5.2, that is bound to the security lifecycle state of the device.

5.2 Binding

The following types of binding are applicable when sensitive data are stored in Non-isolated Storage, see section 2.6:

- Device Binding, which binds to the owning device. Other devices cannot directly access the data
- Partition Binding, which binds to the owning secure partition. Other secure partitions cannot directly access the data
- Lifecycle State Binding: Access is based on security lifecycle state

Binding relies on secret keys and cryptography. The generation and storage of the required keys depends on the security states and the supported debug modes. It is recommended using run-time-derived keys to support lifecycle binding, see section 3. Derivation is essential if recoverable debug is supported. Entry into a non-recoverable debug state means that key re-derivation is not necessary.

Section 5.2.1 outlines the derivation and properties of keys to support device binding for the secure and debug states, see section 3. Derivation for device binding is essential if recoverable debug is supported. Section 5.2.2 outlines a run-time key derivation scheme to support Partition Binding.

Binding may also be used to protect Internal Trusted Storage. See section 5.1.

5.2.1 **Device-binding Root Keys**

Two binding root keys are defined. Both are security lifecycle-state dependent, and so should be derived as a result of a system reset:

- Secure BRK (SBRK): The same key can only be derived when the device is in a secure security lifecycle state, see section 3. This ensures that the data that is bound to this key cannot be accessed when the device is in any debug mode
- Non-PRoT Debug BRK (DBRK): The same key can only be derived when the device is in a secure security lifecycle state, or in any debug mode that does not compromise the PRoT, see section 3). This key should only be used when it is acceptable for any data that is bound to it can be accessed when the device is in a debug mode

This means that any PRoT data that is cryptographically secured should be bound only to SBRK.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Derivation of the binding root keys must be through a cryptographic one-way derivation from a hardware unique secret key, for example, the Hardware Unique Key, see Table 1. Derivation ensures that any derived BRK is unique to the device. The derivation must result in SBRK and DBRK being different.

The derived SBRK or DBRK should be used only to derive a Partition Specific Binding Key (PSBK), see section 5.2.2.

5.2.2 Partition-specific Binding Keys

Partition Specific Binding Keys (PBSKs) are required where binding to a secure partition is required. The derived PSBK will be unique to that secure partition.

Each PSBK is a cryptographic one-way derivation either from one of the binding root keys, see section 5.2.1, or from an existing derived PSBK that is owned by that secure partition. This means that all PSBKs are ultimately derived from either SBRK or DBRK. When a PSBK is derived, the following data are used.

- Explicit parameters provided by the calling secure partition:
    - Seed: Allows the derivation of different PSBKs for different secure partition specific use cases
    - Use Policy for the derived PSBK: One and only one of:
        - Encryption/decryption
        - Signing/verification
        - PSBK derivation: The derived PSBK can only be used to derive other PSBKs
    - Debug Policy, with reference to section 5.2.1, one of:
        - Secure: The PSBK derivation must be anchored at SBRK
        - Non-PRoT Debug: The PSBK must be anchored at DBRK
- The identity of the calling secure partition, which is called the Partition ID, must be an implicit parameter that is provided on behalf of the calling secure partition. This binds the derived PSBK to that secure partition and guarantees the uniqueness of the derived PBSK

The Use Policy does not include export, i.e. output in the clear, of the derived PSBK. Thus, a PSBK cannot be used where shared knowledge of the key is required.

5.3 Cryptographic Service

The PRoT Cryptographic Service performs cryptographic operations using given input, or derived, or persistent keys that are stored in Isolated Locations or Shielded Locations, see section 2.1. Implementations must meet the following essential security properties:

1. Isolation: Keys and other sensitive data is isolated within the cryptographic service, so that this type of material is not exposed to less trusted software[1]
2. Access control: Keys and other sensitive data must have an owning secure partition and can be used only by that partition. However, see the next point: Policy
3. Policy: This property restricts how individual keys and secrets that are owned by a specific secure partition can be used by that partition, preventing misuse. See section 5.3.1

[1] This does not prevent the use of hardware solutions where the cryptographic service can only request the hardware to use a key.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

5.3.1 Cryptographic Material Policies

The minimum set of policy options is as follows:

- Usage, including allowing or denying the use of specific algorithms or modes to a specific key:
    - encryption and decryption
    - derivation
    - signing and verification
- Export: None, Clear, Wrapped or Delegate. Delegation allows a secure partition to make a key that it owns available, with specified Usage and Export policies, to another specified secure partition or the Non-secure Processing Environment. The usage and export policies of a delegate key may be more restrictive than the source key, but the policies cannot be relaxed

5.3.2 Cryptographic Algorithms and Key Sizes

The use case, any certification profile, or applicable regional or application regulations will determine the required cryptographic algorithms, modes, and key sizes[1].

5.3.3 Random Numbers

A source of random data is required, typically for establishing a secure communication channel. A True Random Number Generator (TRNG) or a suitably seeded Deterministic Random Bit Generator (DRBG) should be used to provide such random data.

5.4 Initial Attestation

An Attestation End Point (AEP) provides evidence in an attestation report that can be checked by an attestation validation entity (VE), see Figure 7. It is essential that an attestation report correctly describes the state of the Attested End Point and the underlying Root of Trust, including whether a debug state has been entered. An Attestation End Point is typically implemented as an Application Root of Trust service.

This security model defines the minimal generic security features to build an attestation protocol on top of the Platform Root of Trust, see Figure 7, but does not define any specific attestation protocol.

- At manufacture, the device is provisioned with the immutable platform security parameters, see Table 1, that are sufficient to uniquely and securely identify the device instance. The manufacturing process will record the issued and manufactured identities for use by the ecosystem, for example, in providing endorsements to the validation entity
- The Platform Root of Trust provides an Initial Attestation service, see 5.4.1.
- The Attested End Point is expected to implement any application-specific attestation protocol
- A validation entity can use the Initial Attestation Token to verify:
    - The implementation of the Platform Root of Trust
    - The identity of the device
    - The security state of the Platform Root of Trust
    - The updateable components that are currently loaded

[1] For example, https://csrc.nist.gov/Projects/Cryptographic-Standards-and-Guidelines.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

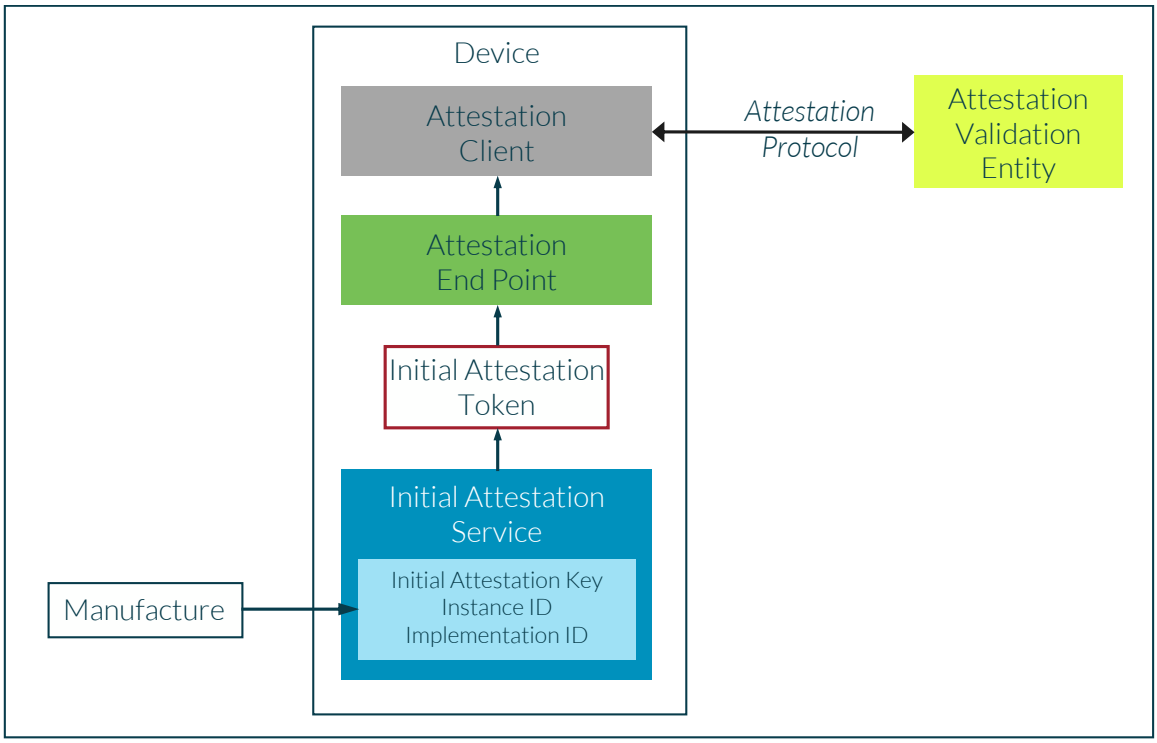

Figure 7: Attestation context

5.4.1 **Initial Attestation**

Figure 8 illustrates a generic initial attestation sequence. The validation entity (VE) challenges the Attestation End Point that is providing metadata in the object record VE_{OR}. The content and use of VE_{OR} depends on the chosen attestation scheme. However, use of a nonce in each challenge to ensure freshness of the data is recommended.

The AEP requests an Initial Attestation Token (IAT) from the Initial Attestation Service, providing at least the metadata that must be validated by the VE for the chosen scheme. Typically, this will be a cryptographic hash of the AEP-specific data in the object record AEP_{OR}, and VE_{OR}. This is shown as denoted $H(AEP_{OR}, VE_{OR})$ in Figure 8.

The Initial Attestation Service constructs its object record, denoted IAS_{OR}, including:

- Recorded boot state of every updateable component loaded at secure boot. See section 4.4.1 and section 4.4.2
- Boot seed from the Initial Boot State. See section 4.4.1
- Current security lifecycle state of the system. See section 3
- Instance ID, Implementation ID and calling Partition ID

The Initial Attestation Key is used to sign a cryptographic hash of the data from the AEP and IAS_{OR}. Once signed using the IAK, it is returned to the AEP. This is the Initial Attestation Token. Typically, these will be encoded in a standard form, for example the IETF Entity Attestation Token[1].

The challenge is then completed by the AEP returning the signed IAT and its object record AEP_{OR} to the validation entity. The VE can use the returned data to validate:

[1] https://datatracker.ietf.org/doc/draft-tschofenig-rats-psa-token/

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

- The trustworthiness of the Platform Root of Trust and its implementation
- The authenticity of the Object Record AEP_{OR}
- The context of the original validating entity challenge data VE_{OR}

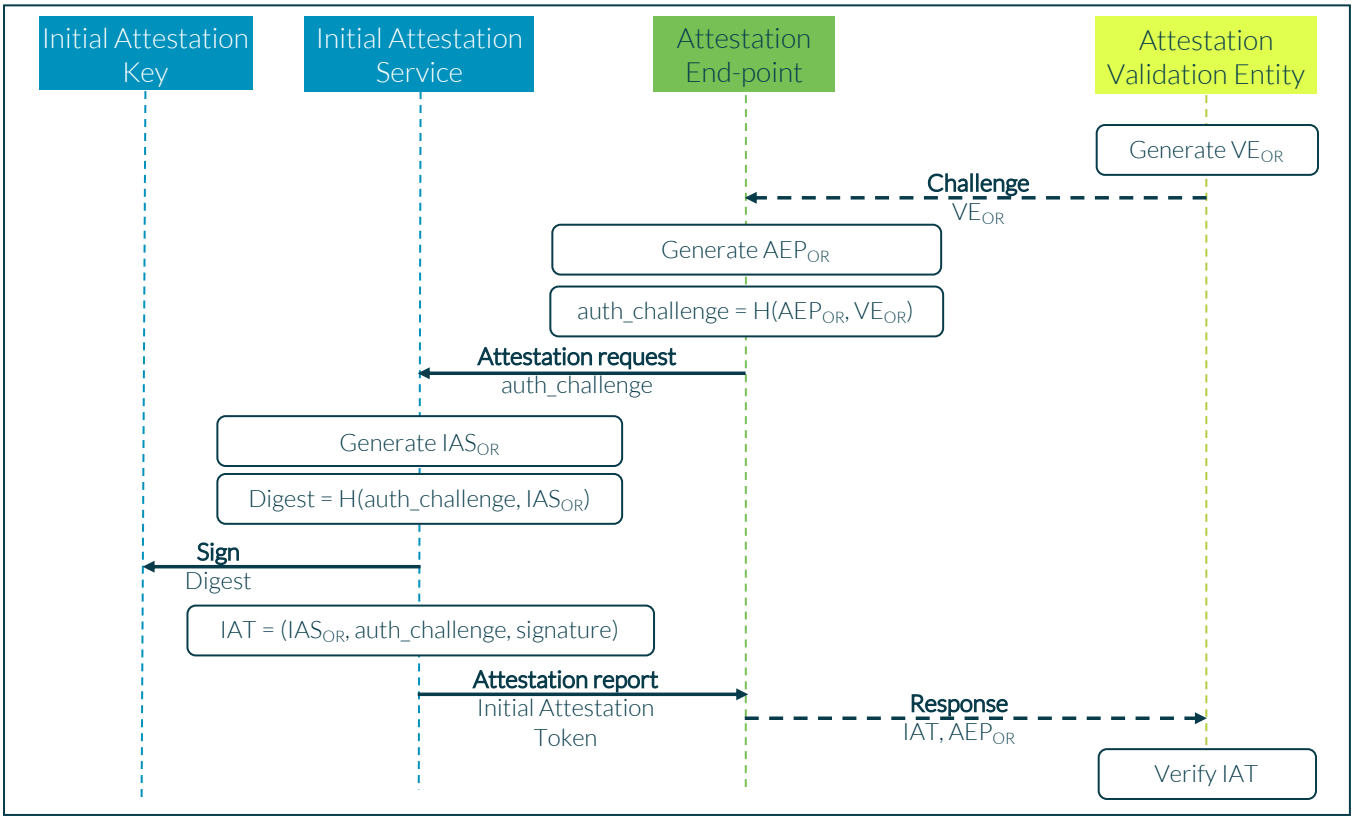

Figure 8: Example initial attestation challenge and response sequence

5.4.2 Delegated Attestation

Though beyond the scope of this document, the outline Initial Attestation Service of 5.4.1 can form the basis of delegated attestation. This allows for proof of possession of the Initial Attestation Key as the root of an application specific attestation system. For example, a Delegated Attestation Service (DAS) generates its own Delegate Attestation Key (DAK), which is included in the Initial Attestation Token by invoking the Initial Attestation Service.

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

6 Secure Storage for Applications

Many devices require secure persistent storage to hold sensitive data. That data could come from the manufacturer, or could be application-generated, service-generated, or user-generated. Typically, such data is stored in flash, which may be on chip or off chip. Such storage called non-isolated storage, see section 2.6.

Application RoT secure storage and application secure storage functions that make use of Non-isolated Storage, as shown in Figure 9, in general, need to have the following properties:

- Defined ownership of all stored data, regardless of the management of data on the storage device
- Privacy and integrity protection to prevent the data from being accessed or modified by an unauthorized agent, including when the device is in a Non-secure state, such as during debug
- Replay protection to prevent a stored set of data from being replaced by a previously stored version of the data set

Depending on implementation requirements and certification profiles, these properties may be enforced by isolation, or by cryptography, or a combination of both, see section 2.3 and section 5.2.

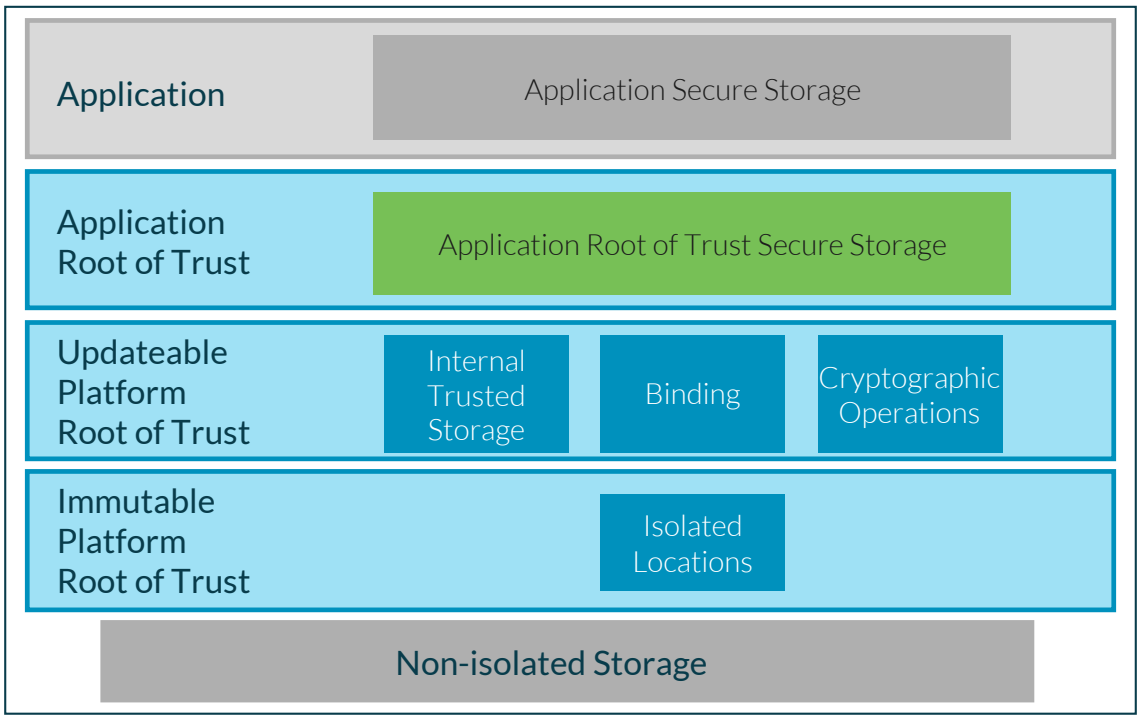

Figure 9: Non-isolated secure persistent storage

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential