TCG Roots of Trust Specification

Family "1.0"
Level 00 Revision 0.20
July 9, 2018
Draft

Contact: admin@trustedcomputinggroup.org

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

TCG Public Review

TCG

Disclaimers, Notices, and License Terms

Table of Contents

Table of Figures

Table of Figures

1. Scope

This specification defines the requirements and security properties of Roots of Trust (RoTs) which may be used in the context of other TCG Specifications. This specification provides options which are considered viable for TCG technologies and platforms. The options provided by this specification are selected by platform work groups based on the suitability of the options for platform work group use cases and platform classes. It is permissible for platform workgroups to disallow options from this specification, but not to define options outside of this specification. This specification does not define requirements for other platform components (which may or may not be RoTs) that are updateable and need a trusted update process, such as platform firmware, as this information is well-defined elsewhere. In particular, the requirements for the update process for components which may not be defined as roots of trust is well defined in NIST's publications SP800-147, SP800-147B and SP800-193. These reference documents also define update mechanisms for RoTs, but the requirements may be looser than TCG deems necessary. Where these documents may be leveraged, they are referenced by this specification. Where a referenced requirement or portion thereof, appropriate for another platform component is insufficient or inadvisable for a RoT, this specification explicitly states "see SP800-xxx section A, clause n, except". The "except" statement identifies requirements TCG considers inadvisable for a RoT.

This specification does not define requirements for virtual platforms nor for virtual roots of trust.

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. You will encounter 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, you can consider it normative statements

For 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

This is the first paragraph of one or more paragraphs (and/or sections) containing the text of statements.

To understand the TCG specification the user MUST read the specification. (This use of MUST indicates a keyword usage and requires an action).

2. Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

3. Glossary

Acronym	Description
RoT	Root of Trust
RoTU	Root of Trust for Update
Update Source Authority	The entity who designs, manufactures, and signs an RoT update, may be a component vendor or a platform vendor
Update Image	Is the Update Image for a (mutable) RoT that is integrity-protected and signed by the Update Source Authority

4. Roots of Trust Summary

4.1 Introduction

Start of informative comment

As defined in the TCG Glossary, “a Root of Trust (RoT) is a component that performs one or more security-specific functions, such as measurement, storage, reporting, verification, and/or update. A RoT is trusted always to behave in the expected manner, because its misbehavior cannot be detected (such as by measurement) by attestation or observation”. The TCG Roots of Trust Specification (“this specification”) is intended to supplement the TCG Glossary definition of a Root of Trust by further defining and refining the properties of a RoT, which is a component of trustworthy computing. In order to establish trustworthiness, one must consider not only the RoT, but the trust chain emanating from and the policies applied to the Platform in which the RoT resides. Further details analysis of the chain of trust and establishing a policy of trustworthiness is outside the scope of this specification.

This specification identifies two implementations of a RoT: an immutable RoT and a mutable RoT. Previously, TCG specifications have conflated the term immutable with something that could be changed under certain circumstances. This is no longer the case: in future TCG specifications, immutable indicates that the RoT is not updateable (in the context of a defined threat model). A mutable RoT can be updated, and requirements and implementation guidance for the update procedure are provided in this specification.

End of informative comment

4.2 Types and Trust Models for Roots of Trust

4.2.1 Description of the Architectural Types

There are two architectural types of Roots of Trust: Immutable and Mutable. They differ in implementation, maintainability and trust properties.

An Immutable RoT is unmodifiable by owners, administrators, or users after delivery by RoT manufacturer.

A Mutable RoT is changeable by an authorized entity. The purpose of deploying a Mutable RoT includes bug and vulnerability fixes and the addition of enhanced features.

There may be multiple authorized entities for a Mutable RoT. This specification defines authorized entitles as the RoT Vendor and the RoT Owner. There is, however, no requirement in this specification to support the Owner as an authorized entity. That is left to platform workgroups and implementers to define.

A change to a Mutable RoT authorized by the Vendor requires authorization provided by the RoT Vendor. Application of a vendor authorized update is determined by the platform Owner.

4.2.2 Trust Model Differences

4.2.2.1 Immutable RoT

The trust properties for an Immutable RoT are predicated by the RoT hardware and unchangeable code deployed by the vendor's manufacturing process. An Immutable RoT is expected to remain identical across all devices within a set of a device models based on a defined threat model. It is also expected not to change across time and, therefore, will behave the same during each device's lifespan. An interested party may evaluate a sample of devices to verify correct behavior then choose to trust the manufacturer not to change the device's manufacturing process.

4.2.2.2 Vendor Controlled Mutable RoT

The trust in a Vendor Controlled Mutable RoT is predicated in the RoT manufacturer's development and authentication process for modifications to the RoT and protection of the manufacturer's authorization mechanism.

4.3 Immutability and Vendor Controlled Mutability

The following table provides a comparison between an Immutable RoT and a Mutable RoT

Table 1 Comparison of Immutable and Mutable RoTs

Characteristic/Consequence	Immutable RoT	Mutable RoT
Updateability	Unchangeable (The RoT uses hardware and software mechanisms to prevent changes)	Changeable **Note:** The RoT uses hardware and software mechanisms to ensure that changes are authenticated and authorized.
Management Entity	None. **Note**: The immutability in the face of hardware attacks may be defined in a Protection Profile.	RoT Vendor (Update Source Authority) **Note:** Resistance to unauthorized modifications (attacks) may be defined in a Protection Profile.
Implications for Serviceability	Implementation issues cannot be addressed post manufacturing	Changeable under the ultimate control of an Update Source Authority

The security of either implementation depends on the Strength of Function as designed and manufactured by the RoT Vendor.

5. Protecting a Root of Trust

Start of informative comment

It is critical that Roots of Trust are protected from unauthorized modification or run-time interference. RoTs can be categorized into those that perform their function at boot time (e.g. a BIOS-based Root of Trust for Measurement), and those that provide service during normal platform operation (e.g. a TPM.) This section contains informative examples and guidance for building Roots of Trust that function securely and reliably.

End of informative comment

5.1 Considerations for a Root of Trust that Performs its Function at Boot Time

Start of informative comment

There are several considerations to protecting a RoT that performs its function at boot time.

First, the execution engine (processor complex, or other) must provide the RoT with a “safe place to stand”. Reset puts the RoT into a known, initial state and environment, and is foundational for RoTs that perform their function at boot-time.

Platform reset prepares an execution environment in which a simple boot-program can execute reliably. This will usually involve setting internal processor registers and state to defined initial values. A Root of Trust that executes at boot time must carefully validate inputs that are not inherently protected and may be under the control of an adversary and should favor using simple data structures for inputs that are easy to validate.

Second, the platform must have mechanisms to protect the stored image of the RoT when it is no longer able to protect itself because it has performed its boot-time function and has passed control to non-RoT code. If the RoT program is stored in ROM, then the program is inherently protected. If the RoT is updatable, then the platform must provide a protection mechanism that the RoT can use to protect its image before it passes control to non-RoT code. As an example, a common protection mechanism in storage is a “protection latch” that can be used by a RoT to write-protect a region of storage in such a way that write-access can only be regained on platform reset, when the RoT regains control.

Third, a boot-time RoT must be protected from other autonomous processors on the platform that can interfere with its operation. This may involve explicit platform protections or may be because other processors are also reset and perform no function until they are explicitly initialized.

End of informative comment

5.2 Considerations for RoTs that Operate at Run Time

Start of informative comment

A RoT that performs any function at run-time needs protection for its stored image. TCG defines and specifies a wide range of RoT-based technologies that incorporate varying run time execution environments. These environments employ protections that may include, for example, separate physical packages, integrated secure execution environments, and environments protected using processor privilege levels or security modes.

End of informative comment

5.3 Updateable Roots of Trust and the Root of Trust for Update

Start of informative comment

The considerations in this section apply to all Roots of Trust, including the Root of Trust for Update for a Mutable RoT.

Note however, that the RoTU for a RoT possesses special privileges to update its associated Roots of Trust. In particular, the RoT is *not* protected from its associated RoTU.

It is beyond the scope of this document to describe acceptable architectures that may be used to distinguish authorized RoTU servicing operations from adversaries, but commonly used techniques include (a) dedicated electrical signals and traces that connect the RoTU to the RoT that it is servicing, (b) a combined/integrated RoT/RoTU, and/or (c) an RoTU that boots first and starts or services its subservient RoT.

End of informative comment

6. Actors and Elements of a Mutable Root

Start of informative comment

The update of a Mutable RoT is dependent on a number of elements and actors. This section defines these elements and actors and the associated requirements.

The Update Image for a Mutable RoT is defined as the package containing the changes to the Mutable RoT. The Update Image may be a patch to the RoT or may replace the entire RoT image.

The Update Image for a Mutable RoT is defined as a package authorized by the Update Source Authority. This includes a signature of the Update Image supplied by the Update Source Authority. This signature provides proof of origin and integrity protection for the Update Image. The Update Source Authority is defined to be the entity or organization that provides, signs and warrants the Update Image.

The Root of Trust for Update is defined to be the component that has ultimate authority over whether an Update Image is an authorized update for a RoT (see below).

An Update may require additional authorization: Update Authorizing Agents are defined as additional entities that authorize the application of the update. Note this may be a platform owner or IT administrator.

End of informative comment.

6.1 Root of Trust for Update

Start of informative comment

The Root of Trust for Update (RoTU) is the ultimate authority for the update of a Mutable RoT. A RoTU consists of:

- An Update Component: the component that actually applies the Update Image,
- An Enforcement Component: the component that validates that the Update Image is valid, and
- Secure Storage to protect the RoTU engine and any configuration data (e.g. keys and certificates) from unauthorized modification.

The RoTU is the only component in the platform that is privileged to update its related RoT or RoTs. The RoT that is being updated will be protected at all times against modification by non-RoTU components.

End of informative comment

6.2 Lifecycle of a Vendor Controlled Mutable RoT

Start of informative comment

The following figure describes the lifecycle of a Mutable RoT from the point at which an Update Image is created to the actual update of the RoT. The figure separates this lifecycle into 3

domains: the Update Source Authority domain, the platform Owner domain, and the RoTU domain.

The Update Source Authority is responsible for updating the source code, and building the Update Image and signing it, thus producing the Protected Update Image. The Update Source Authority then makes the Update Image available to customers.

The platform Owner or their designated representative, i.e. an Admin, makes the decision to update the RoTs in platforms in their environment, or to not update the RoTs. The platform Owner receives the Update Image from the Update Source Authority and applies the Update Image in their environment. Note: this can happen via a variety of channels. The delivery of Update Images is not discussed in this specification.

The RoTU takes control of the lifecycle at this point and is in ultimate control of whether an update is authorized. The RoTU verifies the authenticity and integrity of the Update Image, by verifying its signature and applying the update. If the RoTU cannot verify the Protected Update Image, then the RoT is not updated. If the RoTU cannot apply the update for other reasons, the RoT remains unchanged.

End of informative comment

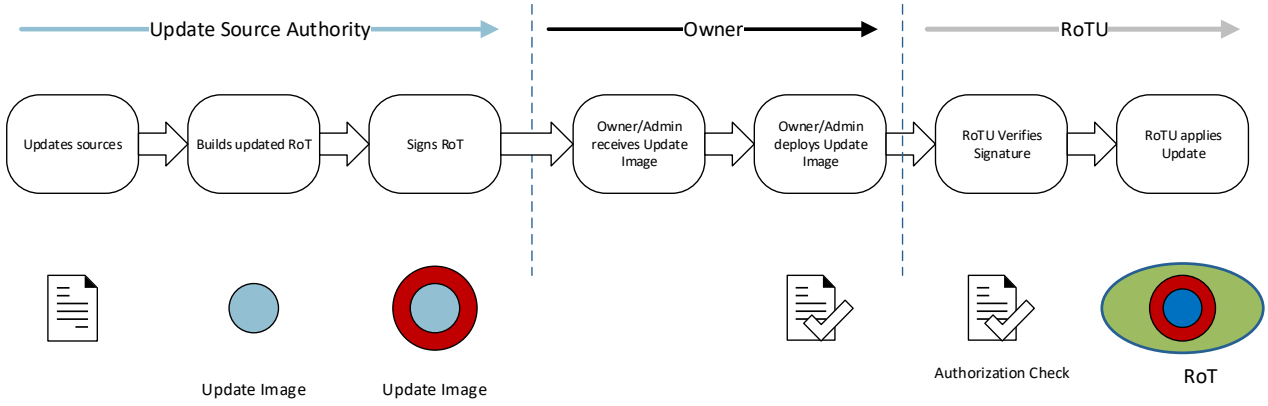

Figure 1 Lifecycle of a Mutable RoT

7. Immutable RoT Requirements

Start of informative comment

In the context of TCG specifications, immutability indicates that a component is designed to be non-updateable. For example, a RoT that is implemented entirely in mask ROM would be considered to be immutable.

Note however, that as defined by TCG, immutability does not imply there are no means to change the RoT; hardware attacks, broken cryptosystems, and implementation bugs may allow immutable RoTs to be subverted or replaced. Immutability is in the context of a particular set of threats.

TCG may publish protection profiles and dictate minimum strength-of-function to specify the threats that compliant components should withstand.

End of Informative comment

1. An Immutable RoT is unmodifiable by owners, administrators, or users after delivery by RoT manufacturer
2. Unmodifiable by attackers to the extent specified by a threat model as defined by TCG platform workgroups.
3. An Immutable RoT may provide an identifier allowing firmware and/or software to query it.

8. Vendor Controlled Mutable RoTU Requirements

8.1 RoTU requirements

Start of informative comment

All of the requirements in this section are predicated on the threat model identified by the platform workgroup which specifies the usage of the RoTU for their platform. The RoTU is a particular type of RoT. It may or may not be a mutable RoT, but it services other RoTs which are mutable. Some of the requirements in this section are requirements for the RoTU because it is a special type of RoT.

End of informative comment

The Root of Trust for Update for a Mutable Root of Trust SHALL comply with the requirements in this section.

1. The RoTU SHALL:
    a. Employ a cryptographically Authenticated Update Mechanism for its associated Root of Trust that meets the requirements set forth in [800-193] Section 4.2.1 "Authenticated Update Mechanism," *apart from* the Secure Local Update mechanism defined in [800-193] Section 3.5.3, which is *prohibited* by this specification (requirement 2.a, below).
    b. Be implemented such that another entity cannot interfere with the behavior of the RoTU, including update of a RoT.
    i. Be implemented to use hardware protection mechanisms to protect RoTU when stored, e.g. "secure storage".
    ii. Use hardware protection mechanisms to protect the runtime execution environment of the RoTU.
    c. Be Time of check/Time of update secure.
    d. Protect the Update Source Authority public key against unauthorized modification.
    e. The mechanism to update the Update Source Authority public key MUST be at least as strong as the security of the RoTU.
    f. Update the RoT transactionally, i.e. apply the entire Update Image or reject the update and retain the original image, see Section 8.2 (Update Image Requirements).

Start of informative comment

The RoTU of a RoT enjoys a uniquely privileged function in devices, and RoTUs that are compliant with this specification must be under the sole administrative control of the device vendor.

Note that the Root of Trust may grant administrative device control to device owners, administrators and users, but the RoT itself is not owner-replaceable.

The cryptographic validation mechanism of requirement 1a requires that a Protected Update Image be verified against a certificate (or other protected data structure.) Vendors must

ensure that the candidate Update Image is protected from interference during verification, and until the update is applied and protected.

End of informative comment

2. The RoTU MUST NOT
    a. Allow override of signature checks of update packages, e.g. physically present user must not be able to install arbitrary code.
    b. Allow any debug connection which would allow manipulation of the RoT or RoTU unless the authentication of the privileged debug entity is at least as strong as the authentication mechanisms of the RoTU.

Start of informative comment

SP800-193 Section 3.5.3 allows entities demonstrating unambiguous physical presence to update device firmware. Devices compliant with this specification must only be updatable through cryptographically authenticated update (requirements 1.a in this specification, and [800-193] Section 4.2.1.1.

Some devices have recovery modes or debug connections that can be used for low-level servicing. These mechanisms are allowed as long as they do not weaken the protection afforded by the main RoTU.

End of informative comment

3. The RoTU SHOULD:
    a. Be smaller and better protected than the RoT it is servicing.
    b. Only process simple inputs, e.g. be resilient to fuzz testing of inputs.
    c. Relinquish privileges that are not needed as soon as possible.

Start of informative comment

Roots of Trust may perform complex functions, and complexity in the face of adversaries can result in compromise. Vendors can reduce the risk of non-field reparable compromise by limiting update privileges to the smallest and best-protected modules. For RoTUs that perform updates at boot-time, this can be achieved by the RoTUs relinquishing update privileges before passing control to the remainder of the RoT. RoTUs that perform their function during normal operation (i.e. not at boot time) should use appropriate hardware mechanisms to protect the update process from unauthorized entities.

Firmware and hardware security defects are usually benign unless the defect is exploited with a malicious input. The chance of a defect being exploited can be vastly reduced if inputs to the RoT are simple, and the input validation code is subjected to the highest levels of scrutiny.

Systems that are complex or have been running for a long time are more likely to be compromised.

Platforms grant Roots of Trust for Update the highest level of privilege, including the privilege to update the RoTU itself. During normal operation – e.g. during normal boot sequences– not all privileges are needed. Resiliency is improved if the Root of Trust relinquishes unnecessary privileges as soon as possible. (The privilege is restored when the platform next resets.)

End of informative comment

2. The RoTU MAY
    a. Generate log events.
    b. Allow opt-out control to users, admins.
    c. Prohibit "downgrade" to old versions.
    d. Allow itself to be updated.
    i. If the RoTU can be updated, it SHOULD follow the requirements in this section.

8.2 Update Image Requirements

Start of informative comment

The following requirements apply to the Update Image.

End of informative comment

1. The Update Image SHALL have a different version identifier from previously applied Update Images.
2. The Update Image SHALL be cryptographically signed by the Update Source Authority.
3. The Update Image MAY be the complete image for the RoT or MAY be a patch to the existing RoT.

8.3 Mutable RoT Requirements

1. A Mutable RoT SHALL use hardware protection mechanism when stored (e.g. secure storage) such that only the RoTU can update the stored RoT image.
2. A Mutable RoT SHALL protect itself when running.
3. A Mutable RoT SHOULD:
    a. Be simple,
    b. Only process simple inputs (e.g. be "fuzz testing" resilient)
    c. Relinquish privilege as soon as possible.
4. A Mutable RoT SHOULD retain the same Update Source Authority following an update.
    a. If the Update Source Authority is to be changed, the current Update Source Authority MUST authorize the change to the new Update Source Authority.

Start of informative comment

In the event that the vendor of a Mutable RoT is acquired by another company or sells its product portfolio (bankruptcy, going out of business or divestiture), it may be necessary or

desirable to transfer control of the Mutable RoT to a new entity. In that case, the original vendor of the RoT must authorize the acquiring vendor to update the Mutable RoT, thus resulting in a new Update Source Authority.

End of informative comment

5. A RoTU that is updatable SHALL comply with the requirements in section (8.3).