Workgroup: RATS Working Group
Internet-Draft:
draft-ietf-rats-tpm-based-network-device-attest-11
Published: 29 January 2022
Intended Status: Informational
Expires: 2 August 2022
Authors: G.C. Fedorkow, Ed. E. Voit
Juniper Networks, Inc. Cisco
J. Fitzgerald-McKay
National Security Agency

TPM-based Network Device Remote Integrity Verification

Abstract

This document describes a workflow for remote attestation of the integrity of firmware and software installed on network devices that contain Trusted Platform Modules [TPM1.2], [TPM2.0], as defined by the Trusted Computing Group (TCG).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There are many aspects to consider in fielding a trusted computing device, from operating systems to applications. Mechanisms to prove that a device installed at a customer's site is authentic (i.e., not counterfeit) and has been configured with authorized software, all as part of a trusted supply chain, are just a few of the many aspects which need to be considered concurrently to have confidence that a device is truly trustworthy.

A generic architecture for remote attestation has been defined in [I-D.ietf-rats-architecture]. Additionally, the use cases for remotely attesting networking devices are discussed within Section 6 of [I-D.richardson-rats-usecases]. However, these documents do not provide sufficient guidance for network equipment vendors and operators to design, build, and deploy interoperable devices.

The intent of this document is to provide such guidance. It does this by outlining the Remote Integrity Verification (RIV) problem, and then identifies elements that are necessary to get the complete, scalable attestation procedure working with commercial networking products such as routers, switches and firewalls. An underlying assumption will be the availability within the device of a Trusted Platform Module [TPM1.2], [TPM2.0] compliant cryptoprocessor to enable the trustworthy remote assessment of the device's software and hardware.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

A number of terms are reused from [I-D.ietf-rats-architecture]. These include: Appraisal Policy for Evidence, Attestation Result, Attester, Evidence, Reference Value, Relying Party, Verifier, and Verifier Owner.

Additionally, this document defines the following term:

Attestation: the process of generating, conveying and appraising claims, backed by evidence, about device trustworthiness characteristics, including supply chain trust, identity, device provenance, software configuration, device composition, compliance to test suites, functional and assurance evaluations, etc.

The goal of attestation is simply to assure an administrator or auditor that the device configuration and software that was launched when the device was last started is authentic and untampered-with. The determination of software authenticity is not prescribed in this document, but it's typically taken to mean a software image generated by an authority trusted by the administrator, such as the device manufacturer.

Within the Trusted Computing Group (TCG) context, the scope of attestation is typically narrowed to describe the process by which an independent Verifier can obtain cryptographic proof as to the identity of the device in question, and evidence of the integrity of software loaded on that device when it started up, and then verify that what's there matches the intended configuration. For network equipment, a Verifier capability can be embedded in a Network Management Station (NMS), a posture collection server, or other network analytics tool (such as a software asset management solution, or a threat detection and mitigation tool, etc.). While informally referred to as attestation, this document focuses on a specific subset of attestation tasks, defined here as Remote Integrity Verification (RIV). RIV takes a network equipment centric perspective that includes a set of protocols and procedures for determining whether a particular device was launched with authentic software, starting from Roots of Trust. While there are many ways to accomplish attestation, RIV sets out a specific set of protocols and tools that work in environments commonly found in network equipment. RIV does not cover other device characteristics that could be attested (e.g., geographic location, connectivity; see [I-D.richardson-rats-usecases]), although it does provide evidence of a secure infrastructure to increase the level of trust in other device characteristics attested by other means (e.g., by Entity Attestation Tokens [I-D.ietf-rats-eat]).

In line with [I-D.ietf-rats-architecture] definitions, this document uses the term Endorser to refer to the role that signs identity and attestation certificates used by the Attester, while Reference Values are signed by a Reference Value Provider. Typically, the manufacturer of a network device would be accepted as both the Endorser and Reference Value Provider, although the choice is ultimately up to the Verifier Owner.

1.3. Document Organization

The remainder of this document is organized into several sections:

* The remainder of this section covers goals and requirements, plus a top-level description of RIV.

* The Solution Overview section outlines how Remote Integrity Verification works.

* The Standards Components section links components of RIV to normative standards.

* Privacy and Security shows how specific features of RIV contribute to the trustworthiness of the Attestation Result.

* Supporting material is in an appendix at the end.

1.4. Goals

Network operators benefit from a trustworthy attestation mechanism that provides assurance that their network comprises authentic equipment, and has loaded software free of known vulnerabilities and unauthorized tampering. In line with the overall goal of assuring integrity, attestation can be used to assist in asset management, vulnerability and compliance assessment, plus configuration management.

The RIV attestation workflow outlined in this document is intended to meet the following high-level goals:

* Provable Device Identity - This specification requires that an Attester (i.e., the attesting device) includes a cryptographic identifier unique to each device. Effectively this means that the device's TPM must be so provisioned during the manufacturing cycle.

* Software Inventory - A key goal is to identify the software release(s) installed on the Attester, and to provide evidence that the software stored within hasn't been altered without authorization.

* Verifiability - Verification of software and configuration of the device shows that the software that the administrator authorized for use was actually launched.

In addition, RIV is designed to operate either in a centralized environment, such as with a central authority that manages and configures a number of network devices, or 'peer-to-peer', where network devices independently verify one another to establish a trust relationship. (See Section 3.3 below)

1.5. Description of Remote Integrity Verification (RIV)

Attestation requires two interlocking mechanisms between the Attester network device and the Verifier:

* Device Identity, the mechanism providing trusted identity, can reassure network managers that the specific devices they ordered from authorized manufacturers for attachment to their network are those that were installed, and that they continue to be present in their network. As part of the mechanism for Device Identity, cryptographic proof of the identity of the manufacturer is also provided.

* Software Measurement is the mechanism that reports the state of mutable software components on the device, and can assure administrators that they have known, authentic software configured to run in their network.

Using these two interlocking mechanisms, RIV is a component in a chain of procedures that can assure a network operator that the equipment in their network can be reliably identified, and that authentic software of a known version is installed on each device. Equipment in the network includes devices that make up the network itself, such as routers, switches and firewalls.

Software used to boot a device can be described as a chain of measurements, anchored at the start by a Root of Trust for Measurement (see Section 9.2), each measuring the next stage and recording the result in tamper-resistant storage, normally ending when the system software is fully loaded. A measurement signifies the identity, integrity and version of each software component registered with an Attester's TPM [TPM1.2], [TPM2.0], so that a subsequent verification stage can determine if the software installed is authentic, up-to-date, and free of tampering.

RIV includes several major processes, split between the Attester and Verifier:

1. Generation of Evidence is the process whereby an Attester generates cryptographic proof (Evidence) of claims about device properties. In particular, the device identity and its software configuration are both of critical importance.

2. Device Identification refers to the mechanism assuring the Relying Party (ultimately, a network administrator) of the identity of devices that make up their network, and that their manufacturers are known.

3. Conveyance of Evidence reliably transports the collected Evidence from Attester to a Verifier to allow a management station to perform a meaningful appraisal in Step 4. The transport is typically carried out via a management network. The channel must provide integrity and authenticity, and, in some use cases, may also require confidentiality.

4. Finally, Appraisal of Evidence occurs. This is the process of verifying the Evidence received by a Verifier from the Attester, and using an Appraisal Policy to develop an Attestation Result, used to inform decision making. In practice, this means comparing the Attester's measurements reported as Evidence with the device configuration expected by the Verifier. Subsequently the Appraisal Policy for Evidence might match Evidence found against Reference Values (aka Golden Measurements), which represent the intended configured state of the connected device.

All implementations supporting this RIV specification require the support of the following three technologies:

1. Identity: Device identity in RIV is based on IEEE 802.1AR Device Identity (DevID) [IEEE-802-1AR], coupled with careful supply-chain management by the manufacturer. The Initial DevID (IDevID) certificate contains a statement by the manufacturer that establishes the identity of the device as it left the factory. Some applications with a more-complex post-manufacture supply chain (e.g., Value Added Resellers), or with different privacy concerns, may want to use alternative mechanisms for platform authentication (for example, TCG Platform Certificates [Platform-Certificates], or post-manufacture installation of Local Device ID (LDevID)).

2. Platform Attestation provides evidence of configuration of software elements present in the device. This form of attestation can be implemented with TPM Platform Configuration Registers (PCRs), Quote and Log mechanisms, which provide cryptographically authenticated evidence to report what software was started on the device through the boot cycle. Successful attestation requires an unbroken chain from a boot-time root of trust through all layers of software needed to bring the device to an operational state, in which each stage computes the hash of components of the next stage, then updates the attestation log and the TPM. The TPM can then report the hashes of all the measured hashes as signed evidence called a Quote (see Section 9.1 for an overview of TPM operation, or [TPM1.2] and [TPM2.0] for many more details).

3. Signed Reference Values (aka Reference Integrity Measurements) must be conveyed from the Reference Value Provider (the entity accepted as the software authority, often the manufacturer of the network device) to the Verifier.

1.6. Solution Requirements

Remote Integrity Verification must address the "Lying Endpoint" problem, in which malicious software on an endpoint may subvert the intended function, and also prevent the endpoint from reporting its compromised status. (See Section 5 for further Security Considerations.)

RIV attestation is designed to be simple to deploy at scale. RIV should work "out of the box" as far as possible, that is, with the fewest possible provisioning steps or configuration databases needed at the end-user's site. Network equipment is often required to "self-configure", to reliably reach out without manual intervention to prove its identity and operating posture, then download its own configuration, a process which precludes pre-installation configuration. See [RFC8572] for an example of Secure Zero Touch Provisioning.

1.7. Scope

The need for assurance of software integrity, addressed by Remote Attestation, is a very general problem that could apply to most network-connected computing devices. However, this document includes several assumptions that limit the scope to network equipment (e.g., routers, switches and firewalls):

* This solution is for use in non-privacy-preserving applications (for example, networking, Industrial IoT), avoiding the need for a Privacy Certificate Authority for attestation keys [AK-Enrollment] or TCG Platform Certificates [Platform-Certificates].

* This document assumes network protocols that are common in network equipment such as YANG [RFC7950] and NETCONF [RFC6241], but not generally used in other applications.

* The approach outlined in this document mandates the use of a compliant TPM [TPM1.2], [TPM2.0].

1.7.1. Out of Scope

* Run-Time Attestation: The Linux Integrity Measurement Architecture [IMA] attests each process launched after a device is started (and is in scope for RIV), but continuous run-time attestation of Linux or other multi-threaded operating system processes after they've started considerably expands the scope of

the problem. Many researchers are working on that problem, but this document defers the problem of continuous, in-memory run-time attestation.

*Multi-Vendor Embedded Systems: Additional coordination would be needed for devices that themselves comprise hardware and software from multiple vendors, integrated by the end user. Although out of scope for this document, these issues are accommodated in [I-D.ietf-rats-architecture].

*Processor Sleep Modes: Network equipment typically does not "sleep", so sleep and hibernate modes are not considered. Although out of scope for RIV, Trusted Computing Group specifications do encompass sleep and hibernate states.

*Virtualization and Containerization: In a non-virtualized system, the host OS is responsible for measuring each User Space file or process, but that's the end of the boot process. For virtualized systems, the host OS must verify the hypervisor, which then manages its own chain of trust through the virtual machine. Virtualization and containerization technologies are increasingly used in network equipment, but are not considered in this document.

2. Solution Overview

2.1. RIV Software Configuration Attestation using TPM

RIV Attestation is a process which can be used to determine the identity of software running on a specifically-identified device. The Remote Attestation steps of Section 1.5 are broken into two phases, shown in Figure 1:

*During system startup, or boot phase, each distinct software object is "measured" by the Attester. The object's identity, hash (i.e., cryptographic digest) and version information are recorded in a log. Hashes are also extended into the TPM (see Section 9.1 for more on 'extending hashes'), in a way that can be used to validate the log entries. The measurement process generally follows the layered chain-of-trust model used in Measured Boot, where each stage of the system measures the next one, and extends its measurement into the TPM, before launching it. See [I-D.ietf-rats-architecture], section "Layered Attestation Environments," for an architectural definition of this model.

*Once the device is running and has operational network connectivity, verification can take place. A separate Verifier, running in its own trusted environment, will interrogate the network device to retrieve the logs and a copy of the digests collected by hashing each software object, signed by an attestation private key secured by, but never released by, the TPM. The YANG model described in [I-D.ietf-rats-yang-tpm-charra] facilitates this operation.

The result is that the Verifier can verify the device's identity by checking the subject attribute of the distinguished name and signature of the certificate containing the TPM's attestation public key, and can validate the software that was launched by verifying the correctness of the logs by comparing with the signed digests from the TPM, and comparing digests in the log with Reference Values.

It should be noted that attestation and identity are inextricably linked; signed Evidence that a particular version of software was loaded is of little value without cryptographic proof of the identity of the Attester producing the Evidence.

```
+-------------------------------------------------------+
| +---------+    +--------+   +--------+    +---------+ |
| |UEFI BIOS|--->| Loader |-->| Kernel |--->|Userland | |
| +---------+    +--------+   +--------+    +---------+ |
|     |            |           |                        |
|     |            |           |                        |
|     +------------+-----------+-+                      |
|                   Boot Phase   |                      |
|                                V                      |
|                            +--------+                 |
|                            |  TPM   |                 |
|                            +--------+                 |
|   Router                       |                      |
+--------------------------------|----------------------+
                                 |
                                 |  Verification Phase
                                 |    +-----------+
                                 +--->| Verifier  |
                                      +-----------+

Reset---------------flow-of-time-during-boot--...------->
```

Figure 1: Layered RIV Attestation Model

In the Boot phase, measurements are "extended", or hashed, into the TPM as processes start, with the result that the TPM ends up containing hashes of all the measured hashes. Later, once the system is operational, during the Verification phase, signed digests are retrieved from the TPM for off-box analysis.

2.1.1. What Does RIV Attest?

TPM attestation is focused on Platform Configuration Registers (PCRs), but those registers are only vehicles for certifying accompanying Evidence, conveyed in log entries. It is the hashes in log entries that are extended into PCRs, where the final PCR values can be retrieved in the form of a structure called a Quote, signed by an Attestation key known only to the TPM. The use of multiple PCRs serves only to provide some independence between different classes of object, so that one class of objects can be updated without changing the extended hash for other classes. Although PCRs can be used for any purpose, this section outlines the objects within the scope of this document which may be extended into the TPM.

In general, assignment of measurements to PCRs is a policy choice made by the device manufacturer, selected to independently attest three classes of object:

* Code, (i.e., instructions) to be executed by a CPU.

* Configuration - Many devices offer numerous options controlled by non-volatile configuration variables which can impact the device's security posture. These settings may have vendor defaults, but often can be changed by administrators, who may want to verify via attestation that the operational state of the settings match their intended state.

* Credentials - Administrators may wish to verify via attestation that public keys (and other credentials) outside the Root of Trust have not been subject to unauthorized tampering. (By definition, keys protecting the root of trust can't be verified independently.)

The TCG PC Client Platform Firmware Profile Specification [PC-Client-BIOS-TPM-2.0] gives considerable detail on what is to be measured during the boot phase of platform startup using a UEFI BIOS (www.uefi.org), but the goal is simply to measure every bit of code executed in the process of starting the device, along with any configuration information related to security posture, leaving no gap for unmeasured code to remain undetected, potentially subverting the chain.

For devices using a UEFI BIOS, [PC-Client-BIOS-TPM-2.0] and [PC-Client-EFI-TPM-1.2] give detailed normative requirements for PCR usage. For other platform architectures, where TCG normative requirements currently do not exist, the table in Figure 2 gives non-normative guidance for PCR assignment that generalizes the specific details of [PC-Client-BIOS-TPM-2.0].

By convention, most PCRs are assigned in pairs, which the even-numbered PCR used to measure executable code, and the odd-numbered PCR used to measure whatever data and configuration are associated with that code. It is important to note that each PCR may contain results from dozens (or even thousands) of individual measurements.

Function	Assigned PCR # Code	Assigned PCR # Configuration
Firmware Static Root of Trust, (i.e., initial boot firmware and drivers)	0	1
Drivers and initialization for optional or add-in devices	2	3
OS Loader code and configuration, (i.e., the code launched by firmware) to load an operating system kernel. These PCRs record each boot attempt, and an identifier for where the loader was found	4	5
Vendor Specific Measurements during boot	6	6
Secure Boot Policy. This PCR records keys and configuration used to validate the OS loader		7
Measurements made by the OS Loader (e.g GRUB2 for Linux)	8	9
Measurements made by OS (e.g., Linux IMA)	10	10

Figure 2: Attested Objects

2.1.2. Notes on PCR Allocations

It is important to recognize that PCR[0] is critical. The first measurement into PCR[0] is taken by the Root of Trust for Measurement, code which, by definition, cannot be verified by measurement. This measurement establishes the chain of trust for all subsequent measurements. If the PCR[0] measurement cannot be trusted, the validity of the entire chain is put into question.

Distinctions Between PCR[0], PCR[2], PCR[4] and PCR[8] are summarized below:

* PCR[0] typically represents a consistent view of rarely-changed Host Platform boot components, allowing Attestation policies to be defined using the less changeable components of the transitive trust chain. This PCR typically provides a consistent view of the platform regardless of user selected options.

* PCR[2] is intended to represent a "user configurable" environment where the user has the ability to alter the components that are measured into PCR[2]. This is typically done by adding adapter cards, etc., into user-accessible PCI or other slots. In UEFI systems these devices may be configured by Option ROMs measured into PCR[2] and executed by the UEFI BIOS.

* PCR[4] is intended to represent the software that manages the transition between the platform's Pre-Operating System start and the state of a system with the Operating System present. This PCR, along with PCR[5], identifies the initial operating system loader (e.g., GRUB for Linux).

* PCR[8] is used by the OS loader (e.g. GRUB) to record measurements of the various components of the operating system.

Although the TCG PC Client document specifies the use of the first eight PCRs very carefully to ensure interoperability among multiple UEFI BIOS vendors, it should be noted that embedded software vendors may have considerably more flexibility. Verifiers typically need to know which log entries are consequential and which are not (possibly controlled by local policies) but the Verifier may not need to know what each log entry means or why it was assigned to a particular PCR. Designers must recognize that some PCRs may cover log entries that a particular Verifier considers critical and other log entries that are not considered important, so differing PCR values may not on their own constitute a check for authenticity. For example, in a UEFI system, some administrators may consider booting an image from a removable drive, something recorded in a PCR, to be a security violation, while others might consider that operation an authorized recovery procedure.

Designers may allocate particular events to specific PCRs in order to achieve a particular objective with local attestation, (e.g., allowing a procedure to execute, or releasing a particular decryption key, only if a given PCR is in a given state). It may also be important to designers to consider whether streaming notification of PCR updates is required (see [I-D.birkholz-rats-network-device-subscription]). Specific log entries can only be validated if the Verifier receives every log entry affecting the relevant PCR, so (for example) a designer might want to separate rare, high-value events such as configuration changes, from high-volume, routine measurements such as IMA [IMA] logs.

2.2. RIV Keying

RIV attestation relies on two credentials:

* An identity key pair and matching certificate is required to certify the identity of the Attester itself. RIV specifies the use of an IEEE 802.1AR Device Identity (DevID) [IEEE-802-1AR], signed by the device manufacturer, containing the device serial number. This requirement goes slightly beyond 802.1AR; see Section 2.4 for notes.

* An Attestation key pair and matching certificate is required to sign the Quote generated by the TPM to report evidence of software configuration.

In a TPM application, both the Attestation private key and the DevID private key MUST be protected by the TPM. Depending on other TPM configuration procedures, the two keys are likely to be different; some of the considerations are outlined in TCG "TPM 2.0 Keys for Device Identity and Attestation" [Platform-DevID-TPM-2.0].

The TCG TPM 2.0 Keys document [Platform-DevID-TPM-2.0] specifies further conventions for these keys:

* When separate Identity and Attestation keys are used, the Attestation Key (AK) and its X.509 certificate should parallel the DevID, with the same device ID information as the DevID certificate (that is, the same subject and subjectAltName (if present), even though the key pairs are different). This allows a quote from the device, signed by an AK, to be linked directly to the device that provided it, by examining the corresponding AK certificate. If the subject in the AK certificate doesn't match the corresponding DevID certificate, or they're signed by differing authorities the Verifier may signal the detection of an Asokan-style person-in-the-middle attack (see Section 5.2).

* Network devices that are expected to use secure zero touch provisioning as specified in [RFC8572] MUST be shipped by the manufacturer with pre-provisioned keys (Initial DevID and Initial AK, called IDevID and IAK). IDevID and IAK certificates MUST both be signed by the Endorser (typically the device manufacturer). Inclusion of an IDevID and IAK by a vendor does not preclude a mechanism whereby an administrator can define Local Identity and Attestation Keys (LDevID and LAK) if desired.

2.3. RIV Information Flow

RIV workflow for network equipment is organized around a simple use case where a network operator wishes to verify the integrity of software installed in specific, fielded devices. A normative taxonomy of terms is given in [I-D.ietf-rats-architecture], but as a reminder, this use case implies several roles and objects:

1. The Attester, the device which the network operator wants to examine.

2. A Verifier (which might be a network management station) somewhere separate from the Device that will retrieve the signed evidence and measurement logs, and analyze them to pass judgment on the security posture of the device.

3. A Relying Party, which can act on Attestation Results. Interaction between the Relying Party and the Verifier is considered out of scope for RIV.

4. Signed Reference Integrity Manifests (RIMs), containing Reference Values, can either be created by the device manufacturer and shipped along with the device as part of its software image, or alternatively, could be obtained several other ways (direct to the Verifier from the manufacturer, from a third party, from the owner's observation of what's thought to be a "known good system", etc.). Retrieving RIMs from the device itself allows attestation to be done in systems that may not have access to the public internet, or by other devices that are not management stations per se (e.g., a peer device; see Section 3.1.3). If Reference Values are obtained from multiple sources, the Verifier may need to evaluate the relative level of trust to be placed in each source in case of a discrepancy.

These components are illustrated in Figure 3.

```
+----------------+        +-------------+        +---------+--------+
|Reference Value |        | Attester    | Step 1 | Verifier|        |
|Provider        |        | (Device     |<-------| (Network| Relying|
|(Device         |        | under       |------->| Mngmt   | Party  |
|Manufacturer    |        | attestation)| Step 2 | Station)|        |
|or other        |        |             |        |         |        |
|authority)      |        |             |        |         |        |
+----------------+        +-------------+        +---------+--------+
       |                                             /\
       |                  Step 0                      |
       -----------------------------------------------
```

Figure 3: RIV Reference Configuration for Network Equipment

* In Step 0, The Reference Value Provider (the device manufacturer or other authority) makes one or more Reference Integrity Manifests (RIMs), corresponding to the software image expected to be found on the device, signed by the Reference Value Provider, available to the Verifier (see Section 3.1.3 for "in-band" and "out of band" ways to make this happen).

* In Step 1, the Verifier (Network Management Station), on behalf of a Relying Party, requests Identity, Measurement Values, and possibly RIMs, from the Attester.

* In Step 2, the Attester responds to the request by providing a DevID, quotes (measured values, signed by the Attester), and optionally RIMs.

Use of the following standards components allows for interoperability:

1. TPM Keys MUST be configured according to [Platform-DevID-TPM-2.0], or [Platform-ID-TPM-1.2].

2. For devices using UEFI and Linux, measurements of firmware and bootable modules MUST be taken according to TCG PC Client [PC-Client-EFI-TPM-1.2] or [PC-Client-BIOS-TPM-2.0], and Linux IMA [IMA]

3. Device Identity MUST be managed as specified in IEEE 802.1AR Device Identity certificates [IEEE-802-1AR], with keys protected by TPMs.

4. Attestation logs from Linux-based systems MUST be formatted according to the Canonical Event Log format [Canonical-Event-Log]. UEFI-based systems MUST use the TCG UEFI BIOS event log [PC-Client-EFI-TPM-1.2] for TPM1.2 systems, and TCG PC Client Platform Firmware Profile [PC-Client-BIOS-TPM-2.0] for TPM2.0.

5. Quotes MUST be retrieved from the TPM according to TCG TAP Information Model [TAP] and the CHARRA YANG model [I-D.ietf-rats-yang-tpm-charra]. While the TAP IM gives a protocol-independent description of the data elements involved, it's important to note that quotes from the TPM are signed inside the TPM, and MUST be retrieved in a way that does not invalidate the signature, to preserve the trust model. The [I-D.ietf-rats-yang-tpm-charra] can be used for this purpose. (See Section 5 Security Considerations).

6. Reference Values MUST be encoded as defined in the TCG RIM document [RIM], typically using SWID [SWID], [NIST-IR-8060] or CoSWID tags [I-D.ietf-sacm-coswid].

2.4. RIV Simplifying Assumptions

This document makes the following simplifying assumptions to reduce complexity:

* The product to be attested MUST be shipped by the equipment vendor with both an IEEE 802.1AR Device Identity and an Initial Attestation Key (IAK) with certificate in place. The IAK certificate MUST contain the same identity information as the DevID (specifically, the same subject and subjectAltName (if used), signed by the manufacturer), but it's a type of key that can be used to sign a TPM Quote, but not other objects (i.e., it's marked as a TCG "Restricted" key; this convention is described in "TPM 2.0 Keys for Device Identity and Attestation" [Platform-DevID-TPM-2.0]). For network equipment, which is generally non-privacy-sensitive, shipping a device with both an IDevID and an IAK already provisioned substantially simplifies initial startup.

* IEEE 802.1AR does not require a product serial number as part of the subject, but RIV-compliant devices MUST include their serial numbers in the DevID/IAK certificates to simplify tracking logistics for network equipment users. All other optional 802.1AR fields remain optional in RIV

* The product MUST be equipped with a Root of Trust for Measurement (RTM), Root of Trust for Storage and Root of Trust for Reporting (as defined in [SP800-155]) which together are capable of conforming to TCG Trusted Attestation Protocol Information Model [TAP].

* The authorized software supplier MUST make available Reference Values in the form of signed SWID or CoSWID tags.

2.4.1. Reference Integrity Manifests (RIMs)

[I-D.ietf-rats-yang-tpm-charra] focuses on collecting and transmitting evidence in the form of PCR measurements and attestation logs. But the critical part of the process is enabling the Verifier to decide whether the measurements are "the right ones" or not.

While it must be up to network administrators to decide what they want on their networks, the software supplier should supply the Reference Values, in signed Reference Integrity Manifests, that may be used by a Verifier to determine if evidence shows known good, known bad or unknown software configurations.

In general, there are two kinds of reference measurements:

1. Measurements of early system startup (e.g., BIOS, boot loader, OS kernel) are essentially single-threaded, and executed exactly once, in a known sequence, before any results could be reported. In this case, while the method for computing the hash and extending relevant PCRs may be complicated, the net result is that the software (more likely, firmware) vendor will have one known good PCR value that "should" be present in the relevant PCRs after the box has booted. In this case, the signed reference measurement could simply list the expected hashes for the given version. However, a RIM that contains the intermediate hashes can be useful in debugging cases where the expected final hash is not the one reported.

2. Measurements taken later in operation of the system, once an OS has started (for example, Linux IMA [IMA]), may be more complex, with unpredictable "final" PCR values. In this case, the Verifier must have enough information to reconstruct the expected PCR values from logs and signed reference measurements from a trusted authority.

In both cases, the expected values can be expressed as signed SWID or CoSWID tags, but the SWID structure in the second case is somewhat more complex, as reconstruction of the extended hash in a PCR may involve thousands of files and other objects.

TCG has published an information model defining elements of Reference Integrity Manifests under the title TCG Reference Integrity Manifest Information Model [RIM]. This information model outlines how SWID tags should be structured to allow attestation, and defines "bundles" of SWID tags that may be needed to describe a complete software release. The RIM contains metadata relating to the software release it belongs to, plus hashes for each individual file or other object that could be attested.

Many network equipment vendors use a UEFI BIOS to launch their network operating system. These vendors may want to also use the TCG PC Client Reference Integrity Measurement specification [PC-Client-RIM], which focuses specifically on a SWID-compatible format suitable for expressing measurement values expected from a UEFI BIOS.

2.4.2. Attestation Logs

Quotes from a TPM can provide evidence of the state of a device up to the time the evidence was recorded, but to make sense of the quote in most cases an event log that identifies which software modules contributed which values to the quote during startup MUST also be provided. The log MUST contain enough information to demonstrate its integrity by allowing exact reconstruction of the digest conveyed in the signed quote (that is, calculating the hash of all the hashes in the log should produce the same values as contained in the PCRs; if they don't match, the log may have been tampered with. See Section 9.1).

There are multiple event log formats which may be supported as viable formats of Evidence between the Attester and Verifier, but to simplify interoperability, RIV focuses on just three:

* TCG UEFI BIOS event log for TPM 2.0 (TCG PC Client Platform Firmware Profile) [PC-Client-BIOS-TPM-2.0]

* TCG UEFI BIOS event log for TPM 1.2 (TCG EFI Platform Specification for TPM Family 1.1 or 1.2, Section 7) [PC-Client-EFI-TPM-1.2]

* TCG Canonical Event Log [Canonical-Event-Log]

3. Standards Components

3.1. Prerequisites for RIV

The Reference Interaction Model for Challenge-Response-based Remote Attestation ([I-D.birkholz-rats-reference-interaction-model]) is based on the standard roles defined in [I-D.ietf-rats-architecture]. However additional prerequisites have been established to allow for interoperable RIV use case implementations. These prerequisites are intended to provide sufficient context information so that the Verifier can acquire and evaluate measurements collected by the Attester.

3.1.1. Unique Device Identity

A secure Device Identity (DevID) in the form of an IEEE 802.1AR DevID certificate [IEEE-802-1AR] MUST be provisioned in the Attester's TPMs.

3.1.2. Keys

The Attestation Key (AK) and certificate MUST also be provisioned on the Attester according to [Platform-DevID-TPM-2.0], or [Platform-ID-TPM-1.2].

It MUST be possible for the Verifier to determine that the Attester's Attestation keys are resident in the same TPM as its DevID keys (see Section 2.2 and Section 5 Security Considerations).

3.1.3. Appraisal Policy for Evidence

As noted in Section 2.3, the Verifier may obtain Reference Values from several sources. In addition, administrators may make authorized, site-specific changes (e.g. keys in key databases) that could impact attestation results. As such, there could be conflicts, omissions or ambiguities between some Reference Values and collected Evidence.

The Verifier MUST have an Appraisal Policy for Evidence to evaluate the significance of any discrepancies between different reference sources, or between reference values and evidence from logs and quotes. While there must be an Appraisal Policy, this document does not specify the format or mechanism to convey the intended policy, nor does RIV specify mechanisms by which the results of applying the policy are communicated to the Relying Party.

3.2. Reference Model for Challenge-Response

Once the prerequisites for RIV are met, a Verifier is able to acquire Evidence from an Attester. The following diagram illustrates a RIV information flow between a Verifier and an Attester, derived from Section 7.1 of [I-D.birkholz-rats-reference-interaction-model]. In this diagram, each event with its input and output parameters is shown as "Event(input-params)=>(outputs)". Event times shown correspond to the time types described within Appendix A of [I-D.ietf-rats-architecture]:

```
.----------.                           .-----------------------.
| Attester |                          | Relying Party/Verifier |
'----------'                          '------------------------'
  time(VG)                                                     |
generateClaims(attestingEnvironment)                           |
   | => claims, eventLogs                                      |
   |                                                           |
   |                                                     time(NS)
   | <-- requestAttestation(handle, authSecIDs, claimSelection) |
   |                                                           |
 time(EG)                                                      |
collectClaims(claims, claimSelection)                          |
   | => collectedClaims                                        |
   |                                                           |
generateEvidence(handle, authSecIDs, collectedClaims)          |
   | => evidence                                               |
   |                                                   time(RG,RA)
   | evidence, eventLogs -------------------------------------> |
   |                                                           |
   |               appraiseEvidence(evidence, eventLogs, refValues)
   |                                        attestationResult <= |
   |                                                           |
   ~                                                           ~
   |                                                    time(RX)
```

Figure 4: IETF Attestation Information Flow

*Step 1 (time(VG)): One or more Attesting Network Device PCRs are extended with measurements. RIV provides no direct link between the time at which the event takes place and the time that it's attested, although streaming attestation as in [I-D.birkholz-rats-network-device-subscription] could.

*Step 2 (time(NS)): The Verifier generates a unique random nonce ("number used once"), and makes a request for one or more PCRs from an Attester. For interoperability, this MUST be accomplished via an interface that implements the YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs [I-D.ietf-rats-yang-tpm-charra]. TPM1.2 and TPM2.0 both allow nonces as large as the operative digest size (i.e., 20 or 32 bytes; see [TPM1.2] Part 2, Section 5.5 and [TPM2.0] Part 2, Section 10.4.4).

*Step 3 (time(EG)): On the Attester, measured values are retrieved from the Attester's TPM. This requested PCR evidence, along with the Verifier's nonce, called a Quote, is signed by the Attestation Key (AK) associated with the DevID. Quotes are retrieved according to CHARRA YANG model [I-D.ietf-rats-yang-tpm-charra]. At the same time, the Attester collects log evidence showing the values have been extended into that PCR. Section 9.1 gives more detail on how this works, including references to the structure and contents of quotes in TPM documents.

* Step 4: Collected Evidence is passed from the Attester to the Verifier

* Step 5 (time(RG,RA)): The Verifier reviews the Evidence and takes action as needed. As the interaction between Relying Party and Verifier is out of scope for RIV, this can be described as one step.

  - If the signature covering TPM Evidence is not correct, the device SHOULD NOT be trusted.

  - If the nonce in the response doesn't match the Verifier's nonce, the response may be a replay, and device SHOULD NOT be trusted.

  - If the signed PCR values do not match the set of log entries which have extended a particular PCR, the device SHOULD NOT be trusted.

  - If the log entries that the Verifier considers important do not match known good values, the device SHOULD NOT be trusted. We note that the process of collecting and analyzing the log can be omitted if the value in the relevant PCR is already a known-good value.

  - If the set of log entries are not seen as acceptable by the Appraisal Policy for Evidence, the device SHOULD NOT be trusted.

  - If time(RG)-time(NS) is greater than the Appraisal Policy for Evidence's threshold for assessing freshness, the Evidence is considered stale and SHOULD NOT be trusted.

3.2.1. Transport and Encoding

Network Management systems MUST retrieve signed PCR based Evidence using [I-D.ietf-rats-yang-tpm-charra] with NETCONF or RESTCONF.

Implementations that use NETCONF MUST do so over a TLS or SSH secure tunnel. Implementations that use RESTCONF transport MUST do so over a TLS or SSH secure tunnel.

Log Evidence MUST be retrieved via log interfaces specified in [I-D.ietf-rats-yang-tpm-charra].

3.3. Centralized vs Peer-to-Peer

Figure 4 above assumes that the Verifier is trusted, while the Attester is not. In a Peer-to-Peer application such as two routers negotiating a trust relationship, the two peers can each ask the other to prove software integrity. In this application, the information flow is the same, but each side plays a role both as an Attester and a Verifier. Each device issues a challenge, and each device responds to the other's challenge, as shown in Figure 5. Peer-to-peer challenges, particularly if used to establish a trust relationship between routers, require devices to carry their own signed reference measurements (RIMs). Devices may also have to carry Appraisal Policy for Evidence for each possible peer device so that each device has everything needed for remote attestation, without having to resort to a central authority.

```
+---------------+                            +---------------+
| RefVal        |                            | RefVal        |
| Provider A    |                            | Provider B    |
| Firmware      |                            | Firmware      |
| Configuration |                            | Configuration |
| Authority     |                            | Authority     |
|               |                            |               |
+---------------+                            +---------------+
      |                                              |
      |       +------------+        +------------+  |
      |       |            | Step 1 |            |  |   \
      |       | Attester   |<------>| Verifier   |  |    |
      |       |            |<------>|            |  |    |  Router B
      +------>|            | Step 2 |            |  |    |- Challenges
       Step 0A|            |        |            |  |    |  Router A
              |            |------->|            |  |    |
              |- Router A -| Step 3 |- Router B -|  |   /
              |            |        |            |  |
              |            |        |            |  |
              |            | Step 1 |            |  |   \
              | Verifier   |<------>| Attester   |<-+    |  Router A
              |            |<------>|            |       |- Challenges
              |            | Step 2 |            |       |  Router B
              |            |        |            |       |
              |            |<-------|            |       |
              +------------+ Step 3 +------------+      /
```

Figure 5: Peer-to-Peer Attestation Information Flow

In this application, each device may need to be equipped with signed RIMs to act as an Attester, and also an Appraisal Policy for

Evidence and a selection of trusted X.509 root certificates, to allow the device to act as a Verifier. An existing link layer protocol such as 802.1X [IEEE-802.1X] or 802.1AE [IEEE-802.1AE], with Evidence being enclosed over a variant of EAP [RFC3748] or LLDP [LLDP] are suitable methods for such an exchange.

4. Privacy Considerations

Network equipment, such as routers, switches and firewalls, has a key role to play in guarding the privacy of individuals using the network. Network equipment generally adheres to several rules to protect privacy:

* Packets passing through the device must not be sent to unauthorized destinations. For example:

  - Routers often act as Policy Enforcement Points, where individual subscribers may be checked for authorization to access a network. Subscriber login information must not be released to unauthorized parties.

  - Network equipment is often called upon to block access to protected resources from unauthorized users.

* Routing information, such as the identity of a router's peers, must not be leaked to unauthorized neighbors.

* If configured, encryption and decryption of traffic must be carried out reliably, while protecting keys and credentials.

Functions that protect privacy are implemented as part of each layer of hardware and software that makes up the networking device. In light of these requirements for protecting the privacy of users of the network, the network equipment must identify itself, and its boot configuration and measured device state (for example, PCR values), to the equipment's administrator, so there's no uncertainty as to what function each device and configuration is configured to carry out. Attestation is a component that allows the administrator to ensure that the network provides individual and peer privacy guarantees, even though the device itself may not have a right to keep its identity secret.

See [NetEq] for more context on privacy in networking devices.

While attestation information from network devices is not likely to contain privacy-sensitive content regarding network users, administrators may want to keep attestation records confidential to avoid disclosing versions of software loaded on the device, information which could facilitate attacks against known vulnerabilities.

5. Security Considerations

Attestation Evidence from the RIV procedure are subject to a number of attacks:

* Keys may be compromised.

* A counterfeit device may attempt to impersonate (spoof) a known authentic device.

* Person-in-the-middle attacks may be used by a compromised device to attempt to deliver responses that originate in an authentic device.

* Replay attacks may be attempted by a compromised device.

5.1. Keys Used in RIV

Trustworthiness of RIV attestation depends strongly on the validity of keys used for identity and attestation reports. RIV takes full advantage of TPM capabilities to ensure that evidence can be trusted.

Two sets of key-pairs are relevant to RIV attestation:

* A DevID key-pair is used to certify the identity of the device in which the TPM is installed.

* An Attestation Key-pair (AK) key is used to certify attestation Evidence (called 'quotes' in TCG documents), used to provide evidence for integrity of the software on the device

TPM practices usually require that these keys be different, as a way of ensuring that a general-purpose signing key cannot be used to spoof an attestation quote.

In each case, the private half of the key is known only to the TPM, and cannot be retrieved externally, even by a trusted party. To ensure that's the case, specification-compliant private/public key-pairs are generated inside the TPM, where they are never exposed, and cannot be extracted (See [Platform-DevID-TPM-2.0]).

Keeping keys safe is a critical enabler of trustworthiness, but it's just part of attestation security; knowing which keys are bound to the device in question is just as important in an environment where private keys are never exposed.

While there are many ways to manage keys in a TPM (see [Platform-DevID-TPM-2.0]), RIV includes support for "zero touch" provisioning (also known as zero-touch onboarding) of fielded devices (e.g., Secure ZTP, [RFC8572]), where keys which have predictable trust properties are provisioned by the device vendor.

Device identity in RIV is based on IEEE 802.1AR Device Identity (DevID). This specification provides several elements:

* A DevID requires a unique key pair for each device, accompanied by an X.509 certificate,

* The private portion of the DevID key is to be stored in the device, in a manner that provides confidentiality (Section 6.2.5 [IEEE-802-1AR])

The X.509 certificate contains several components:

* The public part of the unique DevID key assigned to that device allows a challenge of identity.

* An identifying string that's unique to the manufacturer of the device. This is normally the serial number of the unit, which might also be printed on a label on the device.

* The certificate must be signed by a key traceable to the manufacturer's root key.

With these elements, the device's manufacturer and serial number can be identified by analyzing the DevID certificate plus the chain of intermediate certificates leading back to the manufacturer's root certificate. As is conventional in TLS or SSH connections, a random nonce must be signed by the device in response to a challenge, proving possession of its DevID private key.

RIV uses the DevID to validate a TLS or SSH connection to the device as the attestation session begins. Security of this process derives from TLS or SSH security, with the DevID providing proof that the session terminates on the intended device. See [RFC8446], [RFC4253].

Evidence of software integrity is delivered in the form of a quote signed by the TPM itself. Because the contents of the quote are signed inside the TPM, any external modification (including reformatting to a different data format) after measurements have been taken will be detected as tampering. An unbroken chain of trust is essential to ensuring that blocks of code that are taking measurements have been verified before execution (see Figure 1).

Requiring measurements of the operating software to be signed by a key known only to the TPM also removes the need to trust the device's operating software (beyond the first measurement in the RTM; see below); any changes to the quote, generated and signed by the TPM itself, made by malicious device software, or in the path back to the Verifier, will invalidate the signature on the quote.

A critical feature of the YANG model described in [I-D.ietf-rats-yang-tpm-charra] is the ability to carry TPM data structures in their native format, without requiring any changes to the structures as they were signed and delivered by the TPM. While alternate methods of conveying TPM quotes could compress out redundant information, or add an additional layer of signing using external keys, the implementation MUST preserve the TPM signing, so that tampering anywhere in the path between the TPM itself and the Verifier can be detected.

5.2. Prevention of Spoofing and Person-in-the-Middle Attacks

Prevention of spoofing attacks against attestation systems is also important. There are two cases to consider:

* The entire device could be spoofed. If the Verifier goes to appraise a specific Attester, it might be redirected to a different Attester. Use of the 802.1AR Device Identity (DevID) in the TPM ensures that the Verifier's TLS or SSH session is in fact terminating on the right device.

* A device with a compromised OS could return a fabricated quote providing spoofed attestation Evidence.

Protection against spoofed quotes from a device with valid identity is a bit more complex. An identity key must be available to sign any kind of nonce or hash offered by the Verifier, and consequently, could be used to sign a fabricated quote. To block a spoofed Attestation Result, the quote generated inside the TPM must be signed by a key that's different from the DevID, called an Attestation Key (AK).

Given separate Attestation and DevID keys, the binding between the AK and the same device must also be proven to prevent a person-in-the-middle attack (e.g., the 'Asokan Attack' [RFC6813]).

This is accomplished in RIV through use of an AK certificate with the same elements as the DevID (same manufacturer's serial number, signed by the same manufacturer's key), but containing the device's unique AK public key instead of the DevID public key.

The TCG document TPM 2.0 Keys for Device Identity and Attestation [Platform-DevID-TPM-2.0] specifies OIDs for Attestation Certificates that allow the CA to mark a key as specifically known to be an Attestation key.

These two key-pairs and certificates are used together:

* The DevID is used to validate a TLS connection terminating on the device with a known serial number.

* The AK is used to sign attestation quotes, providing proof that the attestation evidence comes from the same device.

5.3. Replay Attacks

Replay attacks, where results of a previous attestation are submitted in response to subsequent requests, are usually prevented by inclusion of a random nonce in the request to the TPM for a quote. Each request from the Verifier includes a new random number (a nonce). The resulting quote signed by the TPM contains the same nonce, allowing the Verifier to determine freshness, (i.e., that the resulting quote was generated in response to the Verifier's specific request). Time-Based Uni-directional Attestation [I-D.birkholz-rats-tuda] provides an alternate mechanism to verify freshness without requiring a request/response cycle.

5.4. Owner-Signed Keys

Although device manufacturers MUST pre-provision devices with easily verified DevID and AK certificates if zero-touch provisioning such as described in [RFC8572] is to be supported, use of those credentials is not mandatory. IEEE 802.1AR incorporates the idea of an Initial Device ID (IDevID), provisioned by the manufacturer, and a Local Device ID (LDevID) provisioned by the owner of the device. RIV and [Platform-DevID-TPM-2.0] extends that concept by defining an Initial Attestation Key (IAK) and Local Attestation Key (LAK) with the same properties.

Device owners can use any method to provision the Local credentials.

* TCG document [Platform-DevID-TPM-2.0] shows how the initial Attestation keys can be used to certify LDevID and LAK keys. Use of the LDevID and LAK allows the device owner to use a uniform identity structure across device types from multiple manufacturers (in the same way that an "Asset Tag" is used by many enterprises to identify devices they own). TCG document [Provisioning-TPM-2.0] also contains guidance on provisioning Initial and Local identity keys in TPM 2.0.

* Device owners, however, can use any other mechanism they want to assure themselves that local identity certificates are inserted into the intended device, including physical inspection and programming in a secure location, if they prefer to avoid placing trust in the manufacturer-provided keys.

Clearly, local keys can't be used for secure Zero Touch provisioning; installation of the local keys can only be done by some process that runs before the device is installed for network operation.

On the other end of the device life cycle, provision should be made to wipe local keys when a device is decommissioned, to indicate that the device is no longer owned by the enterprise. The manufacturer's Initial identity keys must be preserved, as they contain no information that's not already printed on the device's serial number plate.

5.5. Other Factors for Trustworthy Operation

In addition to trustworthy provisioning of keys, RIV depends on a number of other factors for trustworthy operation.

* Secure identity depends on mechanisms to prevent per-device secret keys from being compromised. The TPM provides this capability as a Root of Trust for Storage.

* Attestation depends on an unbroken chain of measurements, starting from the very first measurement. See Section 9.1 for background on TPM practices.

* That first measurement is made by code called the Root of Trust for Measurement, typically done by trusted firmware stored in boot flash. Mechanisms for maintaining the trustworthiness of the RTM are out of scope for RIV, but could include immutable firmware, signed updates, or a vendor-specific hardware verification technique. See Section 9.2 for background on roots of trust.

* The device owner SHOULD provide some level of physical defense for the device. If a TPM that has already been programmed with an authentic DevID is stolen and inserted into a counterfeit device, attestation of that counterfeit device may become indistinguishable from an authentic device.

RIV also depends on reliable Reference Values, as expressed by the RIM [RIM]. The definition of trust procedures for RIMs is out of scope for RIV, and the device owner is free to use any policy to validate a set of reference measurements. RIMs may be conveyed out-of-band or in-band, as part of the attestation process (see Section 3.1.3). But for network devices, where software is usually shipped as a self-contained package, RIMs signed by the manufacturer and delivered in-band may be more convenient for the device owner.

The validity of RIV attestation results is also influenced by procedures used to create Reference Values:

* While the RIM itself is signed, supply-chains SHOULD be carefully scrutinized to ensure that the values are not subject to unexpected manipulation prior to signing. Insider-attacks against code bases and build chains are particularly hard to spot.

* Designers SHOULD guard against hash collision attacks. Reference Integrity Manifests often give hashes for large objects of indeterminate size; if one of the measured objects can be replaced with an implant engineered to produce the same hash, RIV will be unable to detect the substitution. TPM1.2 uses SHA-1 hashes only, which have been shown to be susceptible to collision attack. TPM2.0 will produce quotes with SHA-256, which so far has resisted such attacks. Consequently, RIV implementations SHOULD use TPM2.0.

6. IANA Considerations

This document has no IANA actions.

7. Conclusion

TCG technologies can play an important part in the implementation of Remote Integrity Verification. Standards for many of the components needed for implementation of RIV already exist:

* Platform identity can be based on IEEE 802.1AR Device Identity, coupled with careful supply-chain management by the manufacturer.

* Complex supply chains can be certified using TCG Platform Certificates [Platform-Certificates].

* The TCG TAP mechanism coupled with [I-D.ietf-rats-yang-tpm-charra] can be used to retrieve attestation evidence.

* Reference Values must be conveyed from the software authority (e.g., the manufacturer) in Reference Integrity Manifests, to the system in which verification will take place. IETF and TCG SWID and CoSWID work ([I-D.ietf-sacm-coswid], [RIM]) forms the basis for this function.

8. Acknowledgements

The authors wish to thank numerous reviewers for generous assistance, including William Bellingrath, Mark Baushke, Ned Smith, Henk Birkholz, Tom Laffey, Dave Thaler, Wei Pan, Michael Eckel, Thomas Hardjono, Bill Sulzen, Willard (Monty) Wiseman, Kathleen Moriarty, Nancy Cam-Winget and Shwetha Bhandari

9. Appendix

9.1. Using a TPM for Attestation

The Trusted Platform Module and surrounding ecosystem provide three interlocking capabilities to enable secure collection of evidence from a remote device, Platform Configuration Registers (PCRs), a Quote mechanism, and a standardized Event Log.

Each TPM has at least eight and at most twenty-four PCRs (depending on the profile and vendor choices), each one large enough to hold one hash value (SHA-1, SHA-256, and other hash algorithms can be used, depending on TPM version). PCRs can't be accessed directly from outside the chip, but the TPM interface provides a way to "extend" a new security measurement hash into any PCR, a process by which the existing value in the PCR is hashed with the new security measurement hash, and the result placed back into the same PCR. The result is a composite fingerprint comprising the hash of all the security measurements extended into each PCR since the system was reset.

Every time a PCR is extended, an entry should be added to the corresponding Event Log. Logs contain the security measurement hash plus informative fields offering hints as to which event generated the security measurement. The Event Log itself is protected against accidental manipulation, but it is implicitly tamper-evident - any verification process can read the security measurement hash from the log events, compute the composite value and compare that to what ended up in the PCR. If there's no discrepancy, the logs do provide an accurate view of what was placed into the PCR.

Note that the composite hash-of-hashes recorded in PCRs is order-dependent, resulting in different PCR values for different ordering of the same set of events (e.g. Event A followed by Event B yields a different PCR value than B followed by A). For single-threaded code, where both the events and their order are fixed, a Verifier may validate a single PCR value, and use the log only to diagnose a mismatch from Reference Values. However, operating system code is usually non-deterministic, meaning that there may never be a single "known good" PCR value. In this case, the Verifier may have to verify that the log is correct, and then analyze each item in the log to determine if it represents an authorized event.

In a conventional TPM Attestation environment, the first measurement must be made and extended into the TPM by trusted device code (called the Root of Trust for Measurement, RTM). That first measurement should cover the segment of code that is run immediately after the RTM, which then measures the next code segment before running it, and so on, forming an unbroken chain of trust. See [TCGRoT] for more on Mutable vs Immutable roots of trust.

The TPM provides another mechanism called a Quote that can read the current value of the PCRs and package them, along with the Verifier's nonce, into a TPM-specific data structure signed by an Attestation private key, known only to the TPM.

As noted above in Section 5 Security Considerations, it's important to note that the Quote data structure is signed inside the TPM. The trust model is preserved by retrieving the Quote in a way that does not invalidate the signature, as specified in [I-D.ietf-rats-yang-tpm-charra]. The structure of the command and response for a quote, including its signature, as generated by the TPM, can be seen in [TPM1.2] Part 3, Section 16.5, and [TPM2.0] Section 18.4.2.

The Verifier uses the Quote and Log together. The Quote contains the composite hash of the complete sequence of security measurement hashes, signed by the TPM's private Attestation Key. The Log contains a record of each measurement extended into the TPM's PCRs. By computing the composite hash of all the measurements, the Verifier can verify the integrity of the Event Log, even though the Event Log itself is not signed. Each hash in the validated Event Log can then be compared to corresponding expected values in the set of Reference Values to validate overall system integrity.

A summary of information exchanged in obtaining quotes from TPM1.2 and TPM2.0 can be found in [TAP], Section 4. Detailed information about PCRs and Quote data structures can be found in [TPM1.2], [TPM2.0]. Recommended log formats include [PC-Client-BIOS-TPM-2.0], and [Canonical-Event-Log].

9.2. Root of Trust for Measurement

The measurements needed for attestation require that the device being attested is equipped with a Root of Trust for Measurement, that is, some trustworthy mechanism that can compute the first measurement in the chain of trust required to attest that each stage of system startup is verified, a Root of Trust for Storage (i.e., the TPM PCRs) to record the results, and a Root of Trust for Reporting to report the results [TCGRoT], [SP800-155], [SP800-193].

While there are many complex aspects of a Root of Trust, two aspects that are important in the case of attestation are:

* The first measurement computed by the Root of Trust for Measurement, and stored in the TPM's Root of Trust for Storage, must be assumed to be correct.

* There must not be a way to reset the Root of Trust for Storage without re-entering the Root of Trust for Measurement code.

The first measurement must be computed by code that is implicitly trusted; if that first measurement can be subverted, none of the remaining measurements can be trusted. (See [SP800-155])

It's important to note that the trustworthiness of the RTM code cannot be assured by the TPM or TPM supplier - code or procedures external to the TPM must guarantee the security of the RTM.

9.3. Layering Model for Network Equipment Attester and Verifier

Retrieval of identity and attestation state uses one protocol stack, while retrieval of Reference Values uses a different set of protocols. Figure 5 shows the components involved.

```
+-----------------------+              +-------------------------+
|                       |              |                         |
|       Attester        |<-------------|        Verifier         |
|       (Device)        |------------->|   (Management Station)  |
|                       |      |       |                         |
+-----------------------+      |       +-------------------------+
                               |
           -------------------- --------------------
           |                                       |
-------------------------------    ---------------------------------
|Reference Values             |    |         Attestation           |
-------------------------------    ---------------------------------

********************************************************************
*         IETF Attestation Reference Interaction Diagram           *
********************************************************************

    .......................           .......................
    . Reference Integrity .           .  TAP (PTS2.0) Info  .
    .      Manifest       .           . Model and Canonical .
    .                     .           .     Log Format      .
    .......................           .......................

    *************************              **********************
    * YANG SWID Module      *              * YANG Attestation   *
    * I-D.ietf-sacm-coswid  *              * Module             *
    *                       *              * I-D.ietf-rats-     *
    *                       *              * yang-tpm-charra    *
    *************************              **********************

    *************************  ************ ************************
    * XML, JSON, CBOR (etc) *  *    UDP   * * XML, JSON, CBOR (etc)*
    *************************  ************ ************************

    *************************              ************************
    *   RESTCONF/NETCONF    *              *   RESTCONF/NETCONF   *
    *************************              ************************

    *************************              ************************
    *       TLS, SSH        *              *       TLS, SSH       *
    *************************              ************************
```

Figure 6: RIV Protocol Stacks

IETF documents are captured in boxes surrounded by asterisks. TCG documents are shown in boxes surrounded by dots.

9.4. Implementation Notes

Figure 7 summarizes many of the actions needed to complete an Attestation system, with links to relevant documents. While documents are controlled by several standards organizations, the implied actions required for implementation are all the responsibility of the manufacturer of the device, unless otherwise noted.

As noted, SWID tags can be generated many ways, but one possible tool is [SWID-Gen]

Component	Controlling Specification
Make a Secure execution environment o Attestation depends on a secure root of trust for measurement outside the TPM, as well as roots for storage and reporting inside the TPM. o Refer to TCG Root of Trust for Measurement. o NIST SP 800-193 also provides guidelines on Roots of Trust	TCG RoT UEFI.org
Provision the TPM as described in TCG documents.	[Platform-DevID-TPM-2.0] TCG Platform Certificate
Put a DevID or Platform Cert in the TPM o Install an Initial Attestation Key at the same time so that Attestation can work out of the box o Equipment suppliers and owners may want to implement Local Device ID as well as Initial Device ID	TCG TPM DevID TCG Platform Certificate IEEE 802.1AR
Connect the TPM to the TLS stack o Use the DevID in the TPM to authenticate TAP connections, identifying the device	Vendor TLS stack (This action is simply configuring TLS to use the DevID as its client certificate)
Make CoSWID tags for BIOS/Loader/Kernel objects o Add reference measurements into SWID tags o Manufacturer should sign the SWID tags o The TCG RIM-IM identifies further procedures to create signed RIM documents that provide the necessary reference information	IETF CoSWID ISO/IEC 19770-2 NIST IR 8060
Package the SWID tags with a vendor software release o A tag-generator plugin such as [SWID-Gen] can be used	Retrieve tags with I-D.ietf-sacm-coswid TCG PC Client RIM

```
|  Use PC Client measurement definitions          | TCG PC Client  |
|  to define the use of PCRs                      | BIOS           |
|  (although Windows  OS is rare on Networking    |                |
|  Equipment, UEFI BIOS is not)                   |                |
--------------------------------------------------------------------
|  Use TAP to retrieve measurements               |                |
|    o  Map to YANG                               | YANG Module for|
|  Use Canonical Log Format                       |   Basic        |
|                                                 |   Attestation  |
|                                                 | TCG Canonical  |
|                                                 |   Log Format   |
--------------------------------------------------------------------
| Posture Collection Server (as described in IETF |                |
|  SACMs ECP) should request the                  |                |
|  attestation and analyze the result             |                |
| The Management application might be broken down |                |
|  to several more components:                    |                |
|    o  A Posture Manager Server                  |                |
|       which collects reports and stores them in |                |
|       a database                                |                |
|    o  One or more Analyzers that can look at the|                |
|       results and figure out what it means.     |                |
--------------------------------------------------------------------
```

Figure 7: Component Status

10. References

10.1. Normative References

[Canonical-Event-Log] Trusted Computing Group, "DRAFT Canonical Event Log Format Version: 1.0, Revision: .30", December 2020, <https://www.trustedcomputinggroup.org/wp-content/uploads/TCG_IWG_CEL_v1_r0p30_13feb2021.pdf>.

[I-D.ietf-rats-yang-tpm-charra] Birkholz, H., Eckel, M., Bhandari, S., Voit, E., Sulzen, B., (Frank), L. X., Laffey, T., and G. C. Fedorkow, "A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs", Work in Progress, Internet-Draft, draft-ietf-rats-yang-tpm-charra-12, 14 January 2022, <https://www.ietf.org/archive/id/draft-ietf-rats-yang-tpm-charra-12.txt>.

[I-D.ietf-sacm-coswid] Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", Work in Progress, Internet-Draft, draft-ietf-sacm-coswid-20, 26 January 2022, <https://www.ietf.org/archive/id/draft-ietf-sacm-coswid-20.txt>.

[IEEE-802-1AR] Seaman, M., "802.1AR-2018 - IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity, IEEE Computer Society", August 2018.

[IMA] dsafford, kds_etu, mzohar, reinersailer, and serge_hallyn, "Integrity Measurement Architecture", June 2019, <https://sourceforge.net/p/linux-ima/wiki/Home/>.

[PC-Client-BIOS-TPM-2.0] Trusted Computing Group, "PC Client Specific Platform Firmware Profile Specification Family "2.0", Level 00 Revision 1.05", May 2021, <https://trustedcomputinggroup.org/wp-content/uploads/TCG_PCClient_PFP_r1p05_v23_pub.pdf>.

[PC-Client-EFI-TPM-1.2] Trusted Computing Group, "TCG EFI Platform Specification for TPM Family 1.1 or 1.2, Specification Version 1.22, Revision 15", January 2014, <https://trustedcomputinggroup.org/resource/tcg-efi-platform-specification/>.

[PC-Client-RIM] Trusted Computing Group, "TCG PC Client Reference Integrity Manifest Specification, v1.04", December 2019, <https://trustedcomputinggroup.org/wp-content/uploads/TCG_PC_Client_RIM_r1p04_pub.pdf>.

[Platform-DevID-TPM-2.0] Trusted Computing Group, "TPM 2.0 Keys for Device Identity and Attestation, Specification Version 1.0, Revision 2", September 2020, <https://trustedcomputinggroup.org/resource/tpm-2-0-keys-for-device-identity-and-attestation/>.

[Platform-ID-TPM-1.2] Trusted Computing Group, "TPM Keys for Platform Identity for TPM 1.2, Specification Version 1.0, Revision 3", August 2015, <https://trustedcomputinggroup.org/resource/tpm-keys-for-platform-identity-for-tpm-1-2-2/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/rfc8572>.

[RIM] Trusted Computing Group, "TCG Reference Integrity Manifest (RIM) Information Model, v1.0, r0.16", June 2019, <https://trustedcomputinggroup.org/wp-content/uploads/TCG_RIM_Model_v1p01_r0p16_pub.pdf>.

[SWID] The International Organization for Standardization/ International Electrotechnical Commission, "Information Technology Software Asset Management Part 2: Software Identification Tag, ISO/IEC 19770-2", October 2015, <https://www.iso.org/standard/65666.html>.

[TAP] Trusted Computing Group, "TCG Trusted Attestation Protocol (TAP) Information Model for TPM Families 1.2 and 2.0 and DICE Family 1.0, Version 1.0, Revision 0.36", October 2018, <https://trustedcomputinggroup.org/resource/tcg-tap-information-model/>.

10.2. Informative References

[AK-Enrollment] Trusted Computing Group, "TCG Infrastructure Working Group - A CMC Profile for AIK Certificate Enrollment Version 1.0, Revision 7", March 2011, <https://trustedcomputinggroup.org/resource/tcg-infrastructure-working-group-a-cmc-profile-for-aik-certificate-enrollment/>.

[I-D.birkholz-rats-network-device-subscription] Birkholz, H., Voit, E., and W. Pan, "Attestation Event Stream Subscription", Work in Progress, Internet-Draft, draft-birkholz-rats-network-device-subscription-03, 17 August 2021, <https://www.ietf.org/archive/id/draft-birkholz-rats-network-device-subscription-03.txt>.

[I-D.birkholz-rats-reference-interaction-model] Birkholz, H., Eckel, M., Newton, C., and L. Chen, "Reference Interaction Models for Remote Attestation Procedures", Work in Progress, Internet-Draft, draft-birkholz-rats-reference-interaction-model-03, 7 July 2020, <https://www.ietf.org/archive/id/draft-birkholz-rats-reference-interaction-model-03.txt>.

[I-D.birkholz-rats-tuda] Fuchs, A., Birkholz, H., McDonald, I. E., and C. Bormann, "Time-Based Uni-Directional Attestation", Work in Progress, Internet-Draft, draft-birkholz-rats-tuda-06, 12 January 2022, <https://www.ietf.org/archive/id/draft-birkholz-rats-tuda-06.txt>.

[I-D.ietf-rats-architecture] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-architecture-14, 9 December 2021, <https://www.ietf.org/archive/id/draft-ietf-rats-architecture-14.txt>.

[I-D.ietf-rats-eat]
Lundblade, L., Mandyam, G., and J. O'Donoghue, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-11, 24 October 2021, <https://www.ietf.org/archive/id/draft-ietf-rats-eat-11.txt>.

[I-D.richardson-rats-usecases] Richardson, M., Wallace, C., and W. Pan, "Use cases for Remote Attestation common encodings", Work in Progress, Internet-Draft, draft-richardson-rats-usecases-08, 2 November 2020, <https://www.ietf.org/archive/id/draft-richardson-rats-usecases-08.txt>.

[IEEE-802.1AE] Seaman, M., "802.1AE MAC Security (MACsec)", 2018, <https://1.ieee802.org/security/802-1ae/>.

[IEEE-802.1X] IEEE Computer Society, "802.1X-2020 - IEEE Standard for Local and Metropolitan Area Networks--Port-Based Network Access Control", February 2020, <https://standards.ieee.org/standard/802_1X-2020.html>.

[LLDP] IEEE Computer Society, "802.1AB-2016 - IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery", March 2016, <https://standards.ieee.org/standard/802_1AB-2016.html>.

[NetEq] Trusted Computing Group, "TCG Guidance for Securing Network Equipment, Version 1.0, Revision 29", January 2018, <https://trustedcomputinggroup.org/resource/tcg-guidance-securing-network-equipment/>.

[NIST-IR-8060] National Institute for Standards and Technology, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", April 2016, <https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8060.pdf>.

[Platform-Certificates] Trusted Computing Group, "TCG Platform Attribute Credential Profile, Specification Version 1.0, Revision 16", January 2018, <https://trustedcomputinggroup.org/resource/tcg-platform-attribute-credential-profile/>.

[Provisioning-TPM-2.0] Trusted Computing Group, "TCG TPM v2.0 Provisioning Guidance, Version 1.0, Revision 1.0", March 2015, <https://trustedcomputinggroup.org/wp-content/uploads/TCG-TPM-v2.0-Provisioning-Guidance-Published-v1r1.pdf>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol

(EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC6813] Salowey, J. and S. Hanna, "The Network Endpoint Assessment (NEA) Asokan Attack Analysis", RFC 6813, DOI 10.17487/RFC6813, December 2012, <https://www.rfc-editor.org/info/rfc6813>.

[SP800-155] National Institute of Standards and Technology, "BIOS Integrity Measurement Guidelines (Draft)", December 2011, <https://csrc.nist.gov/csrc/media/publications/sp/800-155/draft/documents/draft-sp800-155_dec2011.pdf>.

[SP800-193] National Institute for Standards and Technology, "NIST Special Publication 800-193: Platform Firmware Resiliency Guidelines", April 2018, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-193.pdf>.

[SWID-Gen] Labs64, Munich, Germany, "SoftWare IDentification (SWID) Tags Generator (Maven Plugin)", n.d., <https://github.com/Labs64/swid-maven-plugin>.

[TCGRoT] Trusted Computing Group, "DRAFT: TCG Roots of Trust Specification", October 2018, <https://trustedcomputinggroup.org/wp-content/uploads/TCG_Roots_of_Trust_Specification_v0p20_PUBLIC_REVIEW.pdf>.

[TPM1.2] Trusted Computing Group, "TPM Main Specification Level 2 Version 1.2, Revision 116", March 2011, <https://trustedcomputinggroup.org/resource/tpm-main-specification/>.

[TPM2.0] Trusted Computing Group, "Trusted Platform Module Library Specification, Family "2.0", Level 00, Revision 01.59", November 2019, <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.

Authors' Addresses

Guy Fedorkow (editor)
Juniper Networks, Inc.
10 Technology Park Drive
Westford, Massachusetts 01886
United States of America

Email: gfedorkow@juniper.net

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Jessica Fitzgerald-McKay
National Security Agency
9800 Savage Road
Ft. Meade, Maryland 20755
United States of America

Email: jmfitz2@nsa.gov