Use cases for Remote Attestation common encodings
draft-richardson-rats-usecases-06

Abstract

This document details mechanisms created for performing Remote Attestation that have been used in a number of industries. The document initially focuses on existing industry verticals, mapping terminology used in those specifications to the more abstract terminology used by the IETF RATS Working Group.

The document aspires to describe possible future use cases that would be enabled by common formats.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................... 3
   2.1. Static attestations ....................................... 4
   2.2. Session attestations ..................................... 4
   2.3. Statements ................................................ 4
   2.4. Hardware Root Of Trust ................................... 4
   2.5. Template for Use cases ................................... 5
3. Requirements Language .......................................... 5
4. Overview of Sources of Use Cases .............................. 6
5. Use case summaries ............................................. 6
   5.1. Device Capabilities/Firmware Attestation .................. 6
   5.1.1. Relying on an (third-party) Attestation Server ....... 7
   5.1.2. Autonomous Relying Party ............................. 7
   5.1.3. Proxy Root of Trust ................................. 8
   5.1.4. network scaling - small ............................. 8
   5.1.5. network scaling - medium ............................. 9
   5.1.6. network scaling - large ............................. 9
   5.2. Hardware resiliency / watchdogs .......................... 10
   5.3. IETF TEEP WG use case .................................. 10
   5.4. Confidential Machine Learning (ML) model ................ 11
   5.5. Critical infrastructure ................................ 11
   5.5.1. Computation characteristics .......................... 12
   5.6. Virtualized multi-tenant hosts ........................... 13
   5.7. Cryptographic Key Attestation .......................... 13
   5.7.1. Device Type Attestation .............................. 14
   5.7.2. Key storage attestation ................................ 14
   5.7.3. End user authorization ................................ 15
   5.8. Geographic attestation .................................. 15
   5.8.1. I am here ............................................. 16
   5.8.2. I am near ............................................. 16
   5.8.3. You are here .......................................... 16
   5.9. Connectivity attestation ................................ 16
   5.10. Component connectivity attestation ...................... 17
   5.11. Device provenance attestation ........................... 17
   5.12. DNS privacy policy ...................................... 18
   5.13. Time base unidirectional attestation .................... 19
6. Technology users for RATSnonce ............................... 19
   6.1. Trusted Computing Group Remove Integrity Verification
       (TCG-RIV) .................................................. 19
1. Introduction

The recently chartered IETF RATS WG intends to create a system of attestations that can be shared across a multitude of different users.

This document exists as place to collect use cases for the common RATS technologies in support of the IETF RATS charter point 1. This document is not expected to be published as an RFC, but remain open as a working document. It could become an appendix to provide motivation for a protocol standards document.

End-user use cases that would either directly leverage RATS technology, or would serve to inform technology choices are welcome, however.

2. Terminology

Critical to dealing with and contrasting different technologies is to collect terms which are compatible, to distinguish those terms which are similar but used in different ways.

This section will grow to include forward and external references to terms which have been seen. When terms need to be disambiguated they will be prefixed with their source, such as "TCG(claim)" or "FIDO(relying party)"
Platform attestations generally come in two categories. This document will attempt to indicate for a particular attestation technology falls into this.

2.1. Static attestations

A static attestation says something about the platform on which the code is running.

2.2. Session attestations

A session attestation says something about how a session key used in a connection such as TLS connection was created. It is usually the result of evaluating attestations that are attached to the certificates used to create such a session.

2.3. Statements

The term "statement" is used as the generic term for the semantic content which is being attested to.

2.4. Hardware Root Of Trust

[SP800-155] offers the following definition for root of trust.

"Roots of Trust are components (software, hardware, or hybrid) and computing engines that constitute a set of unconditionally trusted functions. Reliable and trustworthy BIOS integrity measurement and reporting depend upon software agents; each software agent relies upon Roots of Trust, and the level of trustworthiness in each agent depends on its Roots of Trust. BIOS integrity measurement requires the coordination of a Measurement Agent to harvest measurements, a Storage Agent to protect the measurements from modification until they can be reported, and a Reporting Agent to reliably report the measurements. Each of these agents has a corresponding Root of Trust (Root of Trust for Measurement, etc.) These Roots of Trust must act in concert and build on each other to enable reliable and trustworthy measurement, reporting, and verification of BIOS integrity measurements."

SP800-155 uses the terms RoT for Reporting, Storage and Measurement, but not RoT for Verification - it uses "Verification Agent". Though it is assumed the verifier is trustworthy.

However, [tcgglossary] (page 9) includes a RoT for Verification (RTV) as well.

The TCG Glossary also offers a general definition for Root of Trust "A component that performs one or more security-specific functions, such as measurement, storage, reporting, verification, and/or update.
It is trusted always to behave in the expected manner, because its misbehavior cannot be detected (such as by measurement) under normal operation.

[SP800-147B] defines RoT for Update (RoTU) and RoTU verification (RoTU-v).

The TCG definition seems more concise than the NIST, but gets to the same point.

For the purpose of this document, a hardware root of trust refers to security functionality that is trusted to behave in the expected manner, because its misbehavior cannot be detected under normal operation and resists soft exploits by encapsulating the functionality in hardware.

2.5. Template for Use cases

Each use case will consist of a table with a number of constant fields, as illustrated below. The claim names will be loosely synchronized with the EAT draft. The role workflow (formerly "attestation type") will be described in the architecture draft. It will describe two classes of workflow: the passport type (Attestee sends evidence to Attester, receives signed statement, which is sent to relying party), or the background check type (Attestee sends measurements to Relying party, Relying Party checks with Attester).

Use case name: Twelve Monkeys

Who will use it: Army of the Twelve Monkeys SDO

Attester: James Cole

Relying Party: Dr. Kathryn Reilly

Message Flow: Passport

Claims used as evidence: OEM Identity, Age Claim, Location Claim, ptime Claim

Description: James Cole must convince Dr. Reilly he is from the future, and not insane.

3. Requirements Language

This document is not a standards track document and does not make any normative protocol requirements using terminology described in [RFC2119].
4. Overview of Sources of Use Cases

The following specifications have been covered in this document:

- The Trusted Computing Group "Network Device Attestation Workflow" [I-D.fedorkow-rats-network-device-attestation]
- Android Keystore
- Fast Identity Online (FIDO) Alliance attestation,

This document will be expanded to include summaries from:

- Trusted Computing Group (TCG) Trusted Platform Module (TPM)/Trusted Software Stack (TSS)
- ARM "Platform Security Architecture" [I-D.tschofenig-rats- PSA-token]
- Intel SGX attestation [intelsgx]
- Windows Defender System Guard attestation [windowsdefender]
- Windows Device Health Attestation [windowshealth]
- Azure Sphere Attestation [azureattestation]:
- IETF NEA WG [RFC5209]

Additional sources are welcome and requested.

5. Use case summaries

This section lists a series of cases where an attestation is done.

5.1. Device Capabilities/Firmware Attestation

This is a category of claims

Use case name: Device Identity

Who will use it: Network Operators

Attester: varies

Message Flow: varies
Relying Party: varies

Claims used as evidence: TBD

Description: Network operators want a trustworthy report of identity and version of information of the hardware and software on the machines attached to their network. The process starts with some kind of Root of Trust that provides device identity and protected storage for measurements. The mechanism performs a series of measurements, and expresses this with an attestation as to the hardware and firmware/software which is running.

This is a general description for which there are many specific use cases, including [I-D.fedorkow-rats-network-device-attestation] section 1.2, "Software Inventory"

5.1.1. Relying on an (third-party) Attestation Server

Use case name: Third Party Attestation Server

Who will use it: Network Operators

Message Flow: background check

Attester: manufacturer of OS or hardware system

Relying Party: network access control systems

Claims used as evidence: TBD

Description: The measurements from a heterogenous network of devices are provided to device-specific attestation servers. The attestation servers know what the "golden" measurements are, and perform the appropriate evaluations, resulting in attestations that the relying parties can depend upon.

5.1.2. Autonomous Relying Party

Use case name: Autonomous

Who will use it: network operators

Message Flow: passport

Attester: manufacturer of OS or hardware system

Relying Party: peer systems
Claims used as evidence: TBD

Description: The signed measurements are sent to a relying party which must validate them directly. They are not sent to a third party. (It may do so with the help of a signed list of golden values, or some other process). The relying party needs to validate the signed statements directly.

This may occur because the network is not connected, or even because it can not be connected until the equipment is validated.

5.1.3. Proxy Root of Trust

Use case name: Proxy Root of Trust

Who will use it: network operators

Message Flow: passport

Attester: manufacturer of OS or hardware system

Relying Party: peer systems

Claims used as evidence: TBD

Description: A variety of devices provide measurements via their Root of Trust. A proxy server collects these measurements, and (having applied a local policy) then creates a device agnostic attestation. The relying party can validate the claims in a standard format.

5.1.4. network scaling - small

Use case name: Network scaled - small

Who will use it: enterprises

Message Flow: background check

Attester: manufacturer of OS or hardware system

Relying Party: network equipment

Claims used as evidence: TBD

Description: An entire network of systems needs to be validated (such as all the desktops in an enterprise’s building). The infrastructure is in the control of a single operator and is
already trusted. The network can be partitioned so that machines that do not pass attestation can be quarantined. A 1:1 relationship between the device and the relying party can be used to maintain freshness of the attestation.

5.1.5. network scaling - medium

Use case name: Network scaled - medium

Who will use it: larger enterprises, including network operators

Message Flow: passport

Attester: manufacturer of OS or hardware system

Relying Party: network equipment

Claims used as evidence: TBD

Description: An entire network of systems needs to be validated: such as all the desktops in an enterprise’s building, or all the routers at an ISP. The infrastructure is not necessarily trusted: it could be subverted, and it must also attest. The devices may be under a variety of operators, and may be mutually suspicious: each device may therefore need to process attestations from every other device. An NxM mesh of attestations may be untenable, but a system of N:1:M relationships can be setup via proxy attestations.

5.1.6. network scaling - large

Use case name: Network scaled - large

Who will use it: telco/LTE operators

Message Flow: passport

Attester: manufacturer of OS or hardware system

Relying Party: malware auditing systems

Claims used as evidence: TBD

Description: An entire network of systems need to be continuously attested. This could be all of the smartphones on an LTE network, or every desktop system in a worldwide enterprise. The network operator wishes to do this in order to maintain identities of connected devices more than to validate correct firmware, but both situations are reasonable.
5.2. Hardware resiliency / watchdogs

Use case name: Hardware watchdog

Who will use it: individual system designers

Message Flow: passport

Attester: manufacturer of OS or hardware system

Relying Party: bootloader or service processor

Claims used as evidence: TBD

Description: One significant problem is malware that holds a device hostage and does not allow it to reboot to prevent updates to be applied. This is a significant problem, because it allows a fleet of devices to be held hostage for ransom. Within CyRes the TCG is defining hardware Attention Triggers that force a periodical reboot in hardware.

This can be implemented by forcing a reboot unless attestation to an Attestation Server succeeds within the period interval, and having a reboot do remediation by bringing a device into compliance, including installation of patches as needed.

This is unlike the previous section on Device Attestation in that the attestation comes from a network operator, as to the device’s need to continue operating, and is evaluated by trusted firmware (the relying party), which resets a watchdog timer.

5.3. IETF TEEP WG use case

Use case name: TAM validation

Who will use it: The TAM server

Message Flow: background check

Attester: Trusted Execution Environment (TEE)

Relying Party: end-application

Claims used as evidence: TBD

Description: The "Trusted Application Manager (TAM)" server wants to verify the state of a TEE, or applications in the TEE, of a device. The TEE attests to the TAM, which can then decide whether
to install sensitive data in the TEE, or whether the TEE is out of compliance and the TAM needs to install updated code in the TEE to bring it back into compliance with the TAM’s policy.

5.4. Confidential Machine Learning (ML) model

Use case name: Machine Learning protection

Who will use it: Machine Learning systems

Message Flow: TBD

Attester: hardware TEE

Relying Party: machine learning model owner

Claims used as evidence: TBD

Description: An example use case is where a device manufacturer wants to protect its intellectual property in terms of the ML model it developed and that runs in the devices that its customers purchased, and it wants to prevent attackers, potentially including the customer themselves, from seeing the details of the model. This works by having some protected environment (e.g., a hardware TEE) in the device attest to some manufacturer’s service, which if attestation succeeds, then the manufacturer service releases the model, or a key to decrypt the model, to the requester. If a hardware TEE is involved, then this use case overlaps with the TEEP use case.

5.5. Critical infrastructure

Use case name: Critical Infrastructure

Who will use it: devices

Message Flow: TBD

Attester: plant controller

Relying Party: actuator

Claims used as evidence: TBD

Description: When a protocol operation can affect some critical system, the device attached to the critical equipment wants some assurance that the requester has not been compromised. As such, attestation can be used to only accept commands from requesters...
that are within policy. Hardware attestation in particular, especially in conjunction with a TEE on the requester side, can provide protection against many types of malware.

5.5.1. Computation characteristics

Use case name: Shared Block Chain Computational claims

Who will use it: Consortia of Computation systems

Message Flow: TBD

Attester: computer system (physical or virtual)

Relying Party: other computer systems

Claims used as evidence: TBD

Description: A group of enterprises organized as a consortium seeks to deploy computing nodes as the basis of their shared blockchain system. Each member of the consortium must forward an equal number of computing nodes to participate in the P2P network of nodes that form the basis of the blockchain system. In order to prevent the various issues (e.g. concentration of hash power, anonymous mining nodes) found in other blockchain systems, each computing node must comply to a predefined allowable manifest of system hardware, software and firmware, as agreed to by the membership of the consortium. Thus, a given computing node must be able to report the (pre-boot) configuration of its system and be able to report at any time the operational status of the various components that make-up its system.

The consortium seeks to have the following things attested: system configuration, group membership, and virtualization status.

This is a peer-to-peer protocol so each device in the consortium is a relying party. The attestation may be requested online by another entity within the consortium, but not by other parties. The attestation needs to be compact and interoperable and may be included in the blockchain itself at the completion of the consensus algorithm.

The attestation will need to start in a hardware RoT in order to validate if the system is running real hardware rather than running a virtual machine.
5.6. Virtualized multi-tenant hosts

Use case name: Multi-tenant hosts

Who will use it: Virtual machine systems

Message Flow: TBD

Attester: virtual machine hypervisor

Relying Party: network operators

Claims used as evidence: TBD

Description: The host system will do verification as per 5.1.

The tenant virtual machines will do verification as per 5.1

The network operator wants to know if the system _as a whole_ is free
of malware, but the network operator is not allowed to know who the
tenants are.

This is contrasted to the Chassis + Line Cards case (To Be Defined: TBD).

Multiple Line Cards, but a small attestation system on the main card
can combine things together. This is a kind of proxy.

5.7. Cryptographic Key Attestation

Use case name: Key Attestation

Who will use it: network authentication systems

Message Flow: TBD

Attester: device platform

Relying Party: internet peers

Claims used as evidence: TBD

Description: The relying party wants to know how secure a private
key that identifies an entity is. Unlike the network attestation,
the relying party is not part of the network infrastructure, nor
do they necessarily have a business relationship (such as
ownership) over the end device.
5.7.1. Device Type Attestation

Use case name: Device Type Attestation

Who will use it: mobile platforms

Message Flow: TBD

Attester: device platform

Relying Party: internet peers

Claims used as evidence: TBD

Description: This use case convinces the relying party of the characteristics of a device. For privacy reasons, it might not identify the actual device itself, but rather the class of device. The relying party can understand from either in-band (claims) or out-of-band (model numbers, which may be expressed as a claim) whether the device has trustworthy features such as a hardware TPM, software TPM via TEE, or software TPM without TEE. Other details such as the availability of finger-print readers or HDMI outputs may also be inferred.

5.7.2. Key storage attestation

Use case name: Key storage Attestation

Who will use it: secure key storage subsystems

Message Flow: TBD

Attester: device platform

Relying Party: internet peers

Claims used as evidence: TBD

Description: This use case convinces the relying party only about the provenance of a private key by providing claims of the storage security of the private key. This can be conceived as a subset of the previous case, but may be apply very specifically to just a keystore. Additional details associated with the private key may be provided as well, including limitations on usage of the key.

Key storage attestations may be consumed by systems provisioning public key certificates for devices or human users. In these cases, attestations may be incorporated into certificate request protocols.
(e.g., EST (#rfc7030), CMP (#rfc4210), ACME (#rfc8555), SCEP [I-D.gutmann-scep], etc.) and processed by registration authorities or certification authorities prior to determining contents for any issued certificate.

5.7.3. End user authorization

Use case name: End User authorization

Who will use it: authorization systems

Message Flow: TBD

Attester: device platform

Relying Party: internet peers

Claims used as evidence: TBD

Description: This use case convinces the relying party that the digital signatures made by the indicated key pair were done with the approval of the end-user operator. This may also be considered possible subset of the device attestation above, but the attestation may be on a case-by-case basis. The nature of the approval by the end-user would be indicated. Examples include: the user unlocked the device, the user viewed some message and acknowledge it inside an app, the message was displayed to the user via out-of-app control mechanism. The acknowledgements could include selecting options on the screen, pushing physical buttons, scanning fingerprints, proximity to other devices (via bluetooth beacons, chargers, etc)

5.8. Geographic attestation

Use case name: Location attestation

Who will use it: geo-fenced systems

Message Flow: passport (probably)

Attester: secure GPS system(s)

Relying Party: internet peers

Claims used as evidence: TBD

Description: The relying party wants to know the physical location (on the planet earth) of the device. This may be provided
directly by a GPS/GLONASS/Galileo module that is incorporated into a TPM. This may also be provided by collecting other proximity messages from other device that the relying party can form a trust relationship with.

5.8.1. I am here

The simplest use case is the claim of some specific coordinates.

5.8.2. I am near

The second use case is the claim that some other devices are nearby. This may be absolute ("I am near device X, which claims to be at location A"), or just relative, ("I am near device X"). This use could use "I am here" or "I am near" claims from a 1:1 basis with device X, or use some other protocol. The nature of how the proximity was established would be part of this claim. In order to defeat a variety of mechanisms that might attempt to proxy ("wormhole") radio communications, highly precise clocks may be required, and there may also have to be attestations as to the precision of those clocks.

An additional example of being near would be for the case where two smartphones can establish that they are together by recording a common random movement, such as both devices being shaken together. Each device may validate the claim from the other (in a disconnected fashion), or a third party may validate the claim as the relying party.

This could be used to establish that a medical professional was in proximity of a patient with implanted devices who needs help.

5.8.3. You are here

A third way to establish location is for a third party to communicate directly with the relying party. The nature of how this trust is established (and whether it is done recursively) is outside of the scope here. What is critical is that the identity of "You" can be communicated through the third party in a way that the relying party can use, but other intermediaries can not view.

5.9. Connectivity attestation

Use case name: Connectivity attestation

Who will use it: entertainment systems

Message Flow: TBD
Attester: hardware-manufacturer/TEE

Relying Party: connected peer

Claims used as evidence: TBD

Description: The relying party wants to know what devices are connected. A typical situation would be a media owner needing to know what TV device is connected via HDMI and if High-bandwidth Digital Content Protection (HDCP) is intact.

5.10. Component connectivity attestation

Use case name: Component connectivity

Who will use it: chassis systems with pluggable components

Message Flow: background check

Attester: line card

Relying Party: management/control plane software

Claims used as evidence: TBD

Description: A management controller or similar hardware component wants to know what peripherals, rack scale device or other dynamically configurable components are currently attached to the platform that is under management controller control. The management controller may serve as attestation verifier over a local bus or backplane but may also aggregate local attestation results and act as a platform attester to a remote verifier.

5.11. Device provenance attestation

Use case name: RIV - Device Provenance

Who will use it: Industrial IoT devices

Message Flow: passport

Attester: network management station

Relying Party: a network entity

Claims used as evidence: TBD
Description: A newly manufactured device needs to be onboarded into a network where many if not all device management duties are performed by the network owner. The device owner wants to verify the device originated from a legitimate vendor. A cryptographic device identity such as an IEEE802.1AR is embedded during manufacturing and a certificate identifying the device is delivered to the owner onboarding agent. The device authenticates using its 802.1AR IDevID to prove it originated from the expected vendor.

The device chain of custody from the original device manufacturer to the new owner may also be verified as part of device provenance attestation. The chain of custody history may be collected by a cloud service or similar capability that the supply chain and owner agree to use.

[I-D.fedorkow-rats-network-device-attestation] section 1.2 refers to this as "Provable Device Identity", and section 2.3 details the parties.

5.12. DNS privacy policy

Use case name: DNS-over-TLS or DNS-over-HTTPS server privacy policy

Who will use it: enterprises and browsers and BYOD operating systems

Message Flow: passport

Attester: review agency

Relying Party: browsers and operating systems

Claims used as evidence: DNS server identity, privinfo (see draft-reddy-dprive-dprive-privacy-policy)

Description: Users want to control how their DNS queries are handled by DNS servers so they can configure their system to use DNS servers that comply with their privacy expectations.

This use case communicates an attestation from a DoH server to a web browser or equivalent in a desktop or mobile operating system. The attester is a third party which has performed some kind of review of the DNS server. This may include significant levels of Device Capability attestation as to what is running and how it is configured (see Section 5.1), in which case this is a form of Proxy Root of Trust (Section 5.1.3).
5.13. Time base unidirectional attestation

Use case name: Time base unidirectional attestation (TUDA)

Who will use it: high security facilities, with network diode: air
gap-ish firewall (information leaves, but never enters). Any
network services that are RESTful can fall into this category!
Clients can GET/ the info, and it must be complete and stand-alone
without interaction. Or it may be pushed via MQTT or CoAP
Observe.

Message Flow: passport

Attester: web services and other sources of status/sensor
information

Relying Party: open

Claims used as evidence: the beginning and ending time as endorsed
by a Time Stamp Authority, represented by a time stamp token. The
real time clock of the system itself. A Root of Trust for time;
the TPM has a relative time from startup.

Description: The output of TUDA are typically a secure audit log,
where freshness is determined by synchronization to an source of
external time.

The freshness is preserved in the evidence, allowing past states of
the device can be determined.

6. Technology users for RATSnonce.

6.1. Trusted Computing Group Remove Integrity Verification (TCG-RIV)

The TCG RIV Reference Document addresses the problem of knowing if a
networking device should be part of a network, if it belongs to the
operator, and if it is running appropriate software. The work covers
most of the use cases in Section 5.1.

This proposal is available as
[I-D.fedorkow-rats-network-device-attestation]. The goal is to be
multi-vendor, scalable and extensible. The proposal intentionally
limits itself to:

- "non-privacy-preserving applications (i.e., networking, Industrial
  IoT )",
- the firmware is provided by the device manufacturer
Service providers and enterprises deploy hundreds of routers, many of them in remote locations where they're difficult to access or secure. The point of remote attestation is to:

- identify a remote box in a way that’s hard to spoof
- report the inventory of software was launched on the box in a way that cannot be spoofed, that is undetectably altered by a "Lying Endpoint"

The use case described is to be able to monitor the authenticity of software versions and configurations running on each device. This allows owners and auditors to detect deviation from approved software and firmware versions and configurations, potentially identifying infected devices. [RFC5209]

Attestation may be performed by network management systems. Networking Equipment is often highly interconnected, so it’s also possible that attestation could be performed by neighboring devices.

Specifically listed to be out of scope for the first generation includes: Linux processes, composite assemblies of hardware/software created by end-customers, and equipment that uses Sleep or Hibernate modes. There is an intention to cover some of these in future versions of the documents.

The TCG-RIV Attestation leverages the TPM to make a series of measurements during the boot process, and to have the TPM sign those measurements. The resulting "PCR" hashes are then available to an external verifier.

A critical component of the RIV is compatibility with existing TPM practice for attestation procedures, as spelled out in the TCG TAP Informational Model [tapinfomodel] and TPM architecture specifications [tpmarchspec].

The TCG uses the following terminology:

- Device Manufacturer
- Attester ("device under attestation")
- Verifier (Network Management Station)
"Explicit Attestation" is the TCG term for a static (platform) attestation

"Implicit Attestation" is the TCG term for a session attestation

Reference Integrity Measurements (RIM), which are signed by device manufacturer and integrated into firmware.

Quotes: measured values (having been signed), and RIMs

Reference Integrity Values (RIV)

devices have a Initial Attestation Key (IAK), which is provisioned at the same time as the IDevID [ieee802-1AR]

PCR - Platform Configuration Registry (deals with hash chains)

The TCG document builds upon a number of IETF technologies: SNMP (Attestation MIB), YANG, XML, JSON, CBOR, NETCONF, RESTCONF, CoAP, TLS and SSH. The TCG document leverages the 802.1AR IDevID and LDevID processes.

6.2. Android Keystore system

[keystore] describes a system used in smart phones that run the Android operation system. The system is primarily a software container to contain and control access to cryptographic keys, and therefore provides many of the same functions that a hardware Trusted Platform Module might provide.

The uses described in section Section 5.7 are the primary focus.

On hardware which is supported, the Android Keystore will make use of whatever trusted hardware is available, including use of a Trusted Execution Environment (TEE) or Secure Element (SE). The Keystore therefore abstracts the hardware, and guarantees to applications that the same APIs can be used on both more and less capable devices.

A great deal of focus from the Android Keystore seems to be on providing fine-grained authorization of what keys can be used by which applications.

XXX - clearly there must be additional (intended?) use cases that provide some kind of attestation.

Android 9 on Pixel 2 and 3 can provide protected confirmation messages. This uses hardware access from the TPM/TEE to display a message directly to the user, and receives confirmation directly from
the user. A hash of the contents of the message can provided in an
attestation that the device provides.

In addition, the Android Keystore provides attestation information
about itself for use by FIDO.

QUOTE: Finally, the Verified Boot state is included in key
attestation certificates (provided by Keymaster/Strongbox) in the
deviceLocked and verifiedBootState fields, which can be verified by
apps as well as passed onto backend services to remotely verify boot
integrity

6.3. Fast IDentity Online (FIDO) Alliance

The FIDO Alliance [fido] has a number of specifications aimed
primarily at eliminating the need for passwords for authentication to
online services. The goal is to leverage asymmetric cryptographic
operations in common browser and smart-phone platforms so that users
can easily authentication.

The use cases of Section 5.7 are primary.

FIDO specifications extend to various hardware second factor
authentication devices.

Terminology includes:

- "relying party" validates a claim
- "relying party application" makes FIDO Authn calls
- "browser" provides the Web Authentication JS API
- "platform" is the base system
- "internal authenticator" is some credential built-in to the device
- "external authenticator" may be connected by USB, bluetooth, wifi,
  and may be an stand-alone device, USB connected key, phone or
  watch.

FIDO2 had a Key Attestation Format [fidoattestation], and a Signature
Format [fidosignature], but these have been combined into the W3C
document [fido_w3c] specification.

A FIDO use case involves the relying party receiving a device
attestation about the biometric system that performs the identification
of the human. It is the state of the biometric system that is being attested to, not the identity of the human!

FIDO does provide a transport in the form of the WebAuthn and FIDO CTAP protocols.

According to [fidotechnote] FIDO uses attestation to make claims about the kind of device which is be used to enroll. Keypairs are generated on a per-device _model_ basis, with a certificate having a trust chain that leads back to a well-known root certificate. It is expected that as many as 100,000 devices in a production run would have the same public and private key pair. One assumes that this is stored in a tamper-proof TPM so it is relatively difficult to get this key out. The use of this key attests to the device type, and the kind of protections for keys that the relying party may assume, not to the identity of the end user.

7. Examples of Existing Attestation Formats.

This section provides examples of some existing attestation formats.

7.1. Android Keystore

Android Keystore attestations take the form of X.509 certificates. The examples below package the attestation certificate along with intermediate CA certificates required to validate the attestation as a certificates-only SignedData message [RFC5652]. The trust anchor is available here: [keystore_attestation].

The attestations below were generated using the generateKeyPair method from the DevicePolicyManager class using code similar to the following.
KeyGenParameterSpec.Builder builder = null;
if (hasStrongBox) {
    builder = new KeyGenParameterSpec.Builder(
        m_alias,
        KeyProperties.PURPOSE_SIGN | KeyProperties.PURPOSE_VERIFY | KeyProperties.PURPOSE_ENCRYPT | KeyProperties.PURPOSE_DECRYPT
            .setKeySize(2048)
            .setDigests(KeyProperties.DIGEST_NONE, KeyProperties.DIGEST_SHA256)
            .setBlockModes(KeyProperties.BLOCK_MODE_CBC, KeyProperties.BLOCK_MODE_GCM)
            .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_RSA_PKCS1, KeyProperties.ENCRYPTION_PADDING_RSA_OAEP)
            .setSignaturePaddings(KeyProperties.SIGNATURE_PADDING_RSA_PSS, KeyProperties.SIGNATURE_PADDING_RSA_PKCS1)
            .setUserAuthenticationRequired(false)
            .setIsStrongBoxBacked(true)
            .setUnlockedDeviceRequired(true);
}
else {
    builder = new KeyGenParameterSpec.Builder(
        m_alias,
        KeyProperties.PURPOSE_SIGN | KeyProperties.PURPOSE_VERIFY | KeyProperties.PURPOSE_ENCRYPT | KeyProperties.PURPOSE_DECRYPT
            .setKeySize(2048)
            .setDigests(KeyProperties.DIGEST_NONE, KeyProperties.DIGEST_SHA256, KeyProperties.DIGEST_SHA384, KeyProperties.DIGEST_SHA512)
            .setBlockModes(KeyProperties.BLOCK_MODE_CBC, KeyProperties.BLOCK_MODE_CTR, KeyProperties.BLOCK_MODE_GCM)
            .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_RSA_PKCS1, KeyProperties.ENCRYPTION_PADDING_RSA_OAEP)
            .setSignaturePaddings(KeyProperties.SIGNATURE_PADDING_RSA_PSS, KeyProperties.SIGNATURE_PADDING_RSA_PKCS1)
            .setUserAuthenticationRequired(false)
            .setIsStrongBoxBacked(false)
            .setUnlockedDeviceRequired(true);
}
builder.setAttestationChallenge(challenge_bytes);
KeyGenParameterSpec keySpec = builder.build();
AttestedKeyPair akp = dpm.generateKeyPair(componentName, algorithm, keySpec, idAttestationFlags);

7.1.1. TEE
Annotations included below are delimited by ASN.1 comments, i.e., //.
Annotations should be consistent with structures described here:
[keystore_attestation].

0 1172: SEQUENCE {
   4 764:   SEQUENCE {
      8    3:     [0] {
         10    1:       INTEGER 2
      }
      13    1:     INTEGER 1
      16   13:     SEQUENCE {
         18    9:       OBJECT IDENTIFIER
            sha256WithRSAEncryption (1 2 840 113549 1 1 11)
   }
}

0:       NULL

27:     SEQUENCE {
23:       SET {
20:         SEQUENCE {
17:           OBJECT IDENTIFIER serialNumber (2 5 4 5)
16:             PrintableString 'c6047571d8f0d17c'
}
}

31:     SEQUENCE {
27:     SEQUENCE {
23:       SET {
20:         SEQUENCE {
17:           OBJECT IDENTIFIER commonName (2 5 4 3)
16:             UTF8String 'Android Keystore Key'
}
}
}

42:     SEQUENCE {
39:     SEQUENCE {
36:       SET {
33:       SEQUENCE {
30:       SET {
27:       SEQUENCE {
24:       SEQUENCE {
21:       SEQUENCE {
18:       SEQUENCE {
16:       SEQUENCE {
13:       SEQUENCE {
10:       SEQUENCE {
7:       SEQUENCE {
4:       SEQUENCE {
1:       SEQUENCE {

439 4: OCTET STRING, encapsulates {
441 2: BIT STRING 4 unused bits
 : '1100'B
 : }
445 323: SEQUENCE {
449 10: OBJECT IDENTIFIER '1 3 6 1 4 1 11129 2 1 17'
461 307: OCTET STRING, encapsulates { -- Attestation Extension
465 303: SEQUENCE {
469 1: INTEGER 2 -- attestationVersion (KM3)
472 1: ENUMERATED 1 -- attestationSecurityLevel (TrustedEnv.)
475 1: INTEGER 3 -- keymasterVersion
478 1: ENUMERATED 1 -- keymasterSecurityLevel (TrustedEnv.)
481 9: OCTET STRING ‘challenge’ -- attestationChallenge
492 0: OCTET STRING -- reserved
 : Error: Object has zero length.
494 44: SEQUENCE {
496 8: [701] { -- softwareEnforced
500 6: INTEGER 01 64 47 2A 4B 64
 : }
508 28: [709] {
526 26: OCTET STRING, encapsulates {
544 24: SEQUENCE { -- AttestationApplicationId
546 20: SET {
554 18: SEQUENCE { -- AttestationPackageInfo
562 13: OCTET STRING ‘AndroidSystem’ -- package_name
580 1: INTEGER 1 -- version
 : }
538 0: SET () -- signature_digests
 : }
540 229: SEQUENCE { -- hardwareEnforced
554 14: [1] {
562 12: SET {
570 1: INTEGER 0 -- KeyPurpose.ENCRYPT
578 1: INTEGER 1 -- KeyPurpose.DECRYPT
586 1: INTEGER 2 -- KeyPurpose.SIGN
594 1: INTEGER 3 -- KeyPurposeVERIFY
 : }
547 1: [2] {
555 1: INTEGER 1 -- Algorithm.RSA
 : }
559 3: [3] {
568 4: INTEGER 2048
::

  570 11:  [5] {  -- digest
  572  9:    SET {
  574  1:      INTEGER 4 -- Digest.SHA256
  577  1:      INTEGER 5 -- Digest.SHA384
  580  1:      INTEGER 6 -- Digest.SHA512

  583 14:  [6] {  -- padding
  585 12:    SET {
  587  1:      INTEGER 4 -- PaddingMode.RSA_PKCS1_1_5_ENCRYPT
  590  1:      INTEGER 2 -- PaddingMode.RSA_OAEP
  593  1:      INTEGER 3 -- PaddingMode.RSA_PKCS1_1_5_SIGN
  596  1:      INTEGER 5 -- PaddingMode.RSA_PSS

  599  5:  [200] {  -- rsaPublicExponent
  603  3:    INTEGER 65537

  608  2:  [503] {  -- noAuthRequired
  612  0:    NULL -- documentation indicates this is a Boolean

  614  3:  [702] {  -- origin
  618  1:    INTEGER 0 -- KeyOrigin.GENERATED

  621  2:  [703] {  -- rollbackResistant
  625  0:    NULL -- documentation indicates this is a Boolean

  627 42:  [704] {  -- rootOfTrust
  631 40:    SEQUENCE {  -- verifiedBootKey
  633 32:      OCTET STRING
      19 62 B0 53 85 79 FF CE 9A C9 F5 07 C4 6A FE 3B
      92 05 5B AC 71 46 46 22 83 C8 5C 50 0B E7 8D 82

  667  1:    BOOLEAN TRUE -- deviceLocked
  670  1:    ENUMERATED 0 -- verifiedBootState (verified)

  673  5:  [705] {  -- osVersion
  677  3:    INTEGER 90000 -- Android P

  682  5:  [706] {  -- osPatchLevel
  686  3:    INTEGER 201806 -- June 2018

  691  8:  [710] {  -- attestationIdBrand
  695  6:    OCTET STRING 'google'

  703  9:  [711] {  -- attestationIdDevice
  707  7:    OCTET STRING 'walleye'
[712] { -- attestationIdProduct
  OCTET STRING 'walleye'
}
[713] { -- attestationIdSerial
  OCTET STRING 'HT83K1A03849'
}
[716] { -- attestationIdManufacturer
  OCTET STRING 'Google'
}
[717] { -- attestationIdModel
  OCTET STRING 'Pixel 2'
}

SEQUENCE {
  OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
  NULL
}

BIT STRING
  0 1304: SEQUENCE {
    INTEGER 2
    [0] {
      INTEGER 10 34 53 32 94 08 68 79 38 72
    }
    25 13: SEQUENCE {
      OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
      NULL
    }
    38 27: SEQUENCE {
      SET {

44 23:  SEQUENCE {
46  3:    OBJECT IDENTIFIER serialNumber (2 5 4 5)
51 16:    PrintableString '87f4514475ba0a2b'
 :  }
 :  }
69 30:  SEQUENCE {
71 13:    UTCTime 26/05/2016 17:14:51 GMT
86 13:    UTCTime 24/05/2026 17:14:51 GMT
 :  }
101 27:  SEQUENCE {
103 25:    SET {
105 23:      SEQUENCE {
107  3:        OBJECT IDENTIFIER serialNumber (2 5 4 5)
112 16:        PrintableString 'c6047571d8f0d17c'
 :  }
 :  }
130 418:  SEQUENCE {
134 13:    SEQUENCE {
136  9:      OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
147  0:      NULL
 :  }
149 399:  BIT STRING, encapsulates {
154 394:    SEQUENCE {
158 385:    INTEGER
      :    00 B3 01 0D 78 BC 06 33 25 CA D6 A7 2C EF 49 05
      :    4C C1 77 36 F2 E5 7B E8 4C 0A 87 8F 77 6A 09 45
      :    9B AC E8 72 DA E2 0E 20 3D 68 30 A5 B6 26 14 77
      :    AD 7E 93 F5 1D 38 A9 DB 5B FE B2 B8 1A 7B CD 22
      :    3B 17 98 FC 1F 4F 77 2D 92 E9 DE 5F 6B 02 09 4E
      :    99 86 53 98 1C 5E 23 B6 A4 61 53 A5 FB D1 37 09
      :    DB C0 0A 40 E9 28 E6 BE E2 8E 57 94 A9 F2 13 3A
      :    11 40 D2 34 99 A6 B4 F3 99 F2 5D 4A 5D 6A 6C 4B
      :    [ Another 257 bytes skipped ]
547  3:    INTEGER 65537
      :  }
552 221:  [3] {
555 218:    SEQUENCE {
558 29:      SEQUENCE {
560  3:        OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
565 22:        OCTET STRING, encapsulates {
567 20:        OCTET STRING
          :        7B 7B F8 43 CA 1F 0F 96 27 0F 10 6F 7D 0C 23 14
          :        72 8F 1D 80
          :    }
589 31: SEQUENCE {
591  3:  OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
596  24:  OCTET STRING, encapsulates {
598  22:   SEQUENCE {
600   0: [0] :
601   0:   0E 55 6F 46 F5 3B 77 67 E1 B9 73 DC 55 E6 AE EA
602   2:     B4 FD 27 DD
604   2:   }
606   2: }
608   2: }
609   2: }
612  12: SEQUENCE {
614  3:  OBJECT IDENTIFIER basicConstraints (2 5 29 19)
616  1:  BOOLEAN TRUE
619  2:  OCTET STRING, encapsulates {
621  0:   SEQUENCE {
623   0:   }
625   0:   }
627   0:   }
629  14: SEQUENCE {
631  3:  OBJECT IDENTIFIER keyUsage (2 5 29 15)
634  1:  BOOLEAN TRUE
637  4:  OCTET STRING, encapsulates {
639  2:   BIT STRING 7 unused bits
641  2:     '1'B (bit 0)
643  2:   }
645  2:   }
647  2:   }
652  36: SEQUENCE {
654  3:  OBJECT IDENTIFIER nameConstraints (2 5 29 30)
657  29:  OCTET STRING, encapsulates {
659  27:   SEQUENCE {
661  25:   [0] {
663  23:   SEQUENCE {
665  21:   [2] 'invalid;email:invalid'
667  21:     }
669  21:     }
671  21:     }
673  21:     }
675  21:     }
677  21:     }
680  84: SEQUENCE {
682  3:  OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
685  77:  OCTET STRING, encapsulates {
687  75:   SEQUENCE {
689  73:   SEQUENCE {
691  71:   [0] {
693  69:   [0] {
695  67:   [6] :
697  67:     'https://android.googleapis.com/attestation/crl/1'
699  67:     '0345332940868793872'
SEQUENCE {
    OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
    NULL
}

BIT STRING
  69 13 A7 56 B3 9F E1 2B CE A2 09 89 E5 DC 03 B4
  B6 FF F6 1E 96 C7 62 C2 31 D1 B3 D6 1A 9E 36 CF
  C2 FC 0E 06 FA 0E CF B5 2D F8 19 D6 13 96 0B 56
  B0 EE 86 3B B1 B8 38 70 4E 57 EB D9 60 DC 58 74
  FE C8 EB A5 78 9F B7 19 5C F0 80 CF 29 16 6B 04
  3A 5D 7C 2E 5F 11 12 36 BE 46 29 45 04 41 8F B5
  AB C6 31 5F 23 28 0C F2 7C 48 4A F6 43 AA 50 D0
  53 96 1E AD 7C A3 89 96 BB 8B BF 2D 9A 0C 16 35
  [ Another 384 bytes skipped ]
}

SEQUENCE {
  INTEGER 2
  INTEGER 03 88 26 67 60 65 89 96 85 74
}

SEQUENCE {
  OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
  NULL
}

SEQUENCE {
  INTEGER 03 88 26 67 60 65 89 96 85 74
}

SEQUENCE {
  OBJECT IDENTIFIER serialNumber (2 5 4 5)
  PrintableString 'f92009e853b6b045'
}

UTCTime 26/05/2016 17:01:32 GMT
UTCTime 24/05/2026 17:01:32 GMT
103  25:   SET {
105   23:     SEQUENCE {
107    3:       OBJECT IDENTIFIER serialNumber (2 5 4 5)
110   16:         PrintableString '87f4514475ba0a2b'
112   16:         }
115   15:   }
120  546:     SEQUENCE {
124   13:       SEQUENCE {
126    9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
130   0:         NULL
133  527:       BIT STRING, encapsulates {
137   13:         SEQUENCE {
142    9:           INTEGER 65537
145  522:         }
149  527:       }
153  513:     INTEGER
158  513:     :       00 D2 60 D6 45 85 E3 E2 23 79 5A DA 45 57 A7 D8
163   :       5B AF BD 9A 37 CB FA 97 C0 65 44 9D 3A C6 47 F6
168   :       0D 0B A2 74 12 CA F7 4B B9 5F FB B4 EC 5A 2B D0
173   :       16 01 DE BE E2 FE D2 76 0D 75 C4 B1 6A CB 3A 67
178   :       07 21 E0 D5 19 68 C8 1B 01 A2 24 02 FE AD 40 D6
183   :       A7 98 16 0F A2 98 2E A7 AD 75 34 84 6F F8 CF 8A
188   :       A1 0E 90 33 40 9E D0 86 26 57 71 CE FF CF 52 E1
193   :       F0 F9 2B 7E 68 62 03 D8 FD FD 02 53 03 19 AC 28
198   :       [ Another 385 bytes skipped ]
203   3:       INTEGER 65537
208   :       }
213   :       }
218  182:     [3] {
222  179:       SEQUENCE {
226   29:         SEQUENCE {
230    3:           OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
234   22:           OCTET STRING, encapsulates {
238   20:             OCTET STRING
242   :               0E 55 6F 46 F5 3B 77 67 E1 B9 73 DC 55 E6 AE EA
247   :               B4 FD 27 DD
252   :             }
256   :         }
261  31:       SEQUENCE {
265   3:         OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
268  24:         OCTET STRING, encapsulates {
272   22:         SEQUENCE {
276   20:           [0]
280   :             36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C
285   :             C9 EA 4F 12
290   :           }
294   :         }
299   :         }
304  31:   }
309  31:   }

SEQUENCE {
  OBJECT IDENTIFIER basicConstraints (2 5 29 19)
  BOOLEAN TRUE
  OCTET STRING, encapsulates {
    SEQUENCE {
      BOOLEAN TRUE
    }
  }
}

SEQUENCE {
  OBJECT IDENTIFIER keyUsage (2 5 29 15)
  BOOLEAN TRUE
  OCTET STRING, encapsulates {
    BIT STRING 1 unused bit
    '1100001'B
  }
}

SEQUENCE {
  OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
  OCTET STRING, encapsulates {
    SEQUENCE {
      [0] {
        [0] {
            '8FA196314D2FA18'}
        }
      }
    }
  }
}

SEQUENCE {
  OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
  NULL
}

BIT STRING
  0E 0D 71 4A 88 0A 58 53 B6 31 14 7D DA 22 31 C6
  06 D6 EF 3B 22 4D D7 A5 C0 3F BF C6 B4 64 A3 FB
  92 C2 CC 67 F4 6C 24 25 49 6E F6 CB 08 D6 A8 0D
  94 06 7F 8C 8C 3C B1 77 CD C2 3F C7 5E A3 85 6D
  F7 A5 94 13 CD 5A 5C F3 9B 0A 0D E1 82 42 F4 C9
  3F AD FC FB 7C AA 27 04 CC 1C 12 45 15 EB E6 70
  A0 6C DE 77 77 54 9B 1F 02 05 76 03 A4 FC 6C 07
: F4 CB BB 59 F5 CB ED 58 D8 30 9B 6E 3C F7 76 C1
: [ Another 384 bytes skipped ]
: }
0 1376: SEQUENCE {
4 840:  SEQUENCE {
8  3:   [0] {
10  1:     INTEGER 2
 : }
13  9:     INTEGER 00 E8 FA 19 63 14 D2 FA 18
24  9:     SEQUENCE {
26  9:       OBJECT IDENTIFIER
 :     sha256WithRSAEncryption (1 2 840 113549 1 1 11)
37  0:       NULL
 : }
39 27:  SEQUENCE {
41 25:   SET {
43 23:    SEQUENCE {
45  3:      OBJECT IDENTIFIER serialNumber (2 5 4 5)
50 16:      PrintableString 'f92009e853b6b045'
 : }
 : }
68 30:  SEQUENCE {
70 13:    UTCTime 26/05/2016 16:28:52 GMT
85 13:    UTCTime 24/05/2026 16:28:52 GMT
 : }
100 27:  SEQUENCE {
102 25:   SET {
104 23:    SEQUENCE {
106  3:      OBJECT IDENTIFIER serialNumber (2 5 4 5)
111 16:      PrintableString 'f92009e853b6b045'
 : }
 : }
129 546: SEQUENCE {
133 13:   SEQUENCE {
135  9:      OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
146  0:      NULL
 : }
148 527: BIT STRING, encapsulates {
153 522:  SEQUENCE {
157 513: INTEGER
 : 00 AF B6 C7 82 2B B1 A7 01 EC 2B B4 2E 8B CC 54
 : 16 63 AB EF 98 2F 32 C7 7F 51 03 0C 97 52 4B
 : 1B 5F E8 09 FB C7 2A A9 45 1F 74 3C BD 9A 6F 13
 : 35 74 4A A5 5E 77 F6 B6 AC 35 35 EE 17 C2 5E 63
 : 95 17 DD 9C 92 E6 37 4A 53 CB FE 25 8F 8F FB B6
 : FD 12 93 78 A2 2A 4C A9 9C 45 2D 47 A5 9F 32 01
: F4 41 97 CA 1C CD 7E 76 2F B2 F5 31 51 B6 FE B2
: FF FD 2B 6F E4 FE 5B C6 BD 9E C3 4B FE 08 23 9D
: [ Another 385 bytes skipped ]

674 3: INTEGER 65537
:
:
:

679 166: [3] {
682 163: SEQUENCE {
685 29:     SEQUENCE {
687 3:     OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
692 22:     OCTET STRING, encapsulates {
694 20:     OCTET STRING
:     36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C
:     C9 EA 4F 12
:     }
:     }
:     }
:     }

716 31: SEQUENCE {
718 3: OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
723 24: OCTET STRING, encapsulates {
725 22: SEQUENCE {
727 20: [0]
:     36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C
:     C9 EA 4F 12
:     }
:     }
:     }

749 15: SEQUENCE {
751 3: OBJECT IDENTIFIER basicConstraints (2 5 29 19)
756 1: BOOLEAN TRUE
759 5: OCTET STRING, encapsulates {
761 3: SEQUENCE {
763 1: BOOLEAN TRUE
:     }
:     }
:     }

766 14: SEQUENCE {
768 3: OBJECT IDENTIFIER keyUsage (2 5 29 15)
773 1: BOOLEAN TRUE
776 4: OCTET STRING, encapsulates {
778 2: BIT STRING 1 unused bit
:     ’1100001’B
:     }
:     }

782 64: SEQUENCE {
784 3: OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
789 57: OCTET STRING, encapsulates {
791 55: SEQUENCE {

7.1.2. Secure Element

The structures below are not annotated except where the difference is specific to the difference between the TEE structure shown above and artifacts emitted by StrongBox.

0 5143: SEQUENCE {
  4 9:  OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15 5128:  [0] {
19 5124:  SEQUENCE {
23  1:    INTEGER 1
26  0:    SET {}
28  11:  SEQUENCE {
30  9:    OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
   :    }
41 5100:  [0] {
45 1114:  SEQUENCE {
49  834:    SEQUENCE {

[0] {
  INTEGER 2
}  
INTEGER 1

SEQUENCE {
  OBJECT IDENTIFIER
  sha256WithRSAEncryption (1 2 840 113549 1 1 11)
  NULL
}

SEQUENCE {
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER serialNumber (2 5 4 5)
      PrintableString ’90e8da3cadfc7820’
    }
  }
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER title (2 5 4 12)
      UTF8String ’StrongBox’
    }
  }
  
  SEQUENCE {
    UTCTime 01/01/1970 00:00:00 GMT
    UTCTime 23/05/2028 23:59:59 GMT
  }
  
  SEQUENCE {
    SET {
      SEQUENCE {
        OBJECT IDENTIFIER commonName (2 5 4 3)
        UTF8String ’Android Keystore Key’
      }
    }
  }
  
  SEQUENCE {
    SEQUENCE {
      OBJECT IDENTIFIER
      rsaEncryption (1 2 840 113549 1 1 1)
      NULL
    }
  }
  
  BIT STRING, encapsulates {
    INTEGER
    00 DE 98 94 D5 E5 05 98 E8 FC 73 4D 26 FB 48 6A
    CA 06 A0 24 FA 05 D1 D2 32 10 46 F8 50 DD 3E 0D
    DF 4F 95 53 D2 CB 10 1F 00 B2 62 15 1E 21 7E 05
    C6 10 AC EE 7A D8 69 F1 1F 32 C3 17 CA D7 07 BE
  }
479 3: INTEGER 65537

484 399: [3] {
488 395:   SEQUENCE {
492 14:     SEQUENCE {
494 3:       OBJECT IDENTIFIER keyUsage (2 5 29 15)
499 1:       BOOLEAN TRUE
502 4:       OCTET STRING, encapsulates {
504 2:         BIT STRING 7 unused bits
507 3:           '1'B (bit 0)
509 3:         }
512 10:     }
516 375:     SEQUENCE {
520 10:       OBJECT IDENTIFIER '1 3 6 1 4 1 11129 2 1 17'
524 359:     OCTET STRING, encapsulates {
528 355:       SEQUENCE {
532 1:         INTEGER 3
535 1:         ENUMERATED 2   -- attestationSecurityLevel (StrongBox)
538 1:         INTEGER 4
541 1:         ENUMERATED 2   -- attestationSecurityLevel (StrongBox)
544 9:         OCTET STRING 'challenge'
548 0:         OCTET STRING
552 53:         SEQUENCE {
555 2:           [509] {
558 0:             NULL
561 11:             [701] {
564 9:               INTEGER 00 FF FF FF FF FF E5 99 78
567 28:               [709] {
570 26:               OCTET STRING, encapsulates {
574 24:                 SEQUENCE {
577 20:                   SET {
580 18:                     SEQUENCE {
584 13:                       OCTET STRING 'AndroidSystem'
588 1:                       INTEGER 1
591 1:                     }
594 0:                   SET ()
597 0:                 }
601 1:               }
604 0:             }
607 0:           }
610 0:         }
613 0:

Richardson, et al. Expires May 7, 2020 [Page 38]
SEQUENCE {
  [1] {
    SET {
      INTEGER 0
      INTEGER 1
      INTEGER 2
      INTEGER 3
    }
  }
  [2] {
    INTEGER 1
  }
  [3] {
    INTEGER 2048
  }
  [4] {
    SET {
      INTEGER 2
      INTEGER 32
    }
  }
  [5] {
    SET {
      INTEGER 0
      INTEGER 4
    }
  }
  [6] {
    SET {
      INTEGER 2
      INTEGER 3
      INTEGER 4
      INTEGER 5
    }
  }
  [503] {
    NULL
  }
  [702] {
    INTEGER 0
  }
  [704] {
    SEQUENCE {
      OCTET STRING
      : 61 FD A1 2B 32 ED 84 21 4A 9C F1 3D 1A FF B7 AA
    }
  }
}
80 BD 8A 26 8A 86 1E D4 BB 7A 15 17 0F 1A B0 0C

BOOLEAN TRUE

ENUMERATED 0

OCTET STRING

77 96 C5 3D 0E 09 46 2B BA BB FB 7B 8A 65 F6 8D

EF 5C 46 88 BF 99 C4 1E 88 42 01 4D 1F 01 2D C5

INTEGER 0

INTEGER 201903

OCTET STRING 'google'

OCTET STRING 'blueline'

OCTET STRING 'blueline'

OCTET STRING '8A2X0KLUU'

OCTET STRING 'Google'

OCTET STRING 'Pixel 3'

INTEGER 20180905

INTEGER 201903

OBJECT IDENTIFIER

sha256WithRSAEncryption (1 2 840 113549 1 1 11)
900  0:    NULL
  :  
902  257:   BIT STRING
  :    83 EA 59 BD BE 37 4A D5 C0 FC F8 FB AC 8B 72 1E
  :    A5 C2 3B 0C C0 04 1B C0 5A 1B A5 DF D4 67 1D B9
  :    08 42 4B E2 2C AC 07 0F D8 0E 24 97 56 9E 14 F2
  :    D0 AC DD 1E FC DD 68 20 11 DF 88 B8 B6 22 AD 2B
  :    DB 9C 2E 5C 3F AF 0B 8F 02 68 AA 34 4B 5E C8 75
  :    B1 IA 09 D2 19 41 24 61 65 97 2C 0D A4 78 43 A7
  :    9A 27 B2 4E 24 11 4F FF E2 D8 04 56 39 75 B2 34
  :    D8 18 C7 25 F3 3F C0 6A 37 AB 49 B6 96 51 61 72
  :    [ Another 128 bytes skipped ]
  :  
1163 1181:  SEQUENCE {
1167  645:   SEQUENCE {
1171  3:     [0] {
1173  1:       INTEGER 2
  :     }
1176  10:     INTEGER 17 10 24 68 40 71 02 97 78 50
1188  13:     SEQUENCE {
1190  9:       OBJECT IDENTIFIER
  :         sha256WithRSAEncryption (1 2 840 113549 1 1 11)
1201  0:     NULL
  :   }
1203  47:   SEQUENCE {
1205  25:     SET {
1207  23:       SEQUENCE {
1209  3:         OBJECT IDENTIFIER serialNumber (2 5 4 5)
1214  16:         PrintableString ‘ccd18b9b608d658e’
  :       }
  :   }
1232  18:   SET {
1234  16:     SEQUENCE {
1236  3:       OBJECT IDENTIFIER title (2 5 4 12)
1241  9:       UTF8String ‘StrongBox’
  :     }
  :   }
1252  30:   SEQUENCE {
1254  13:     UTCTime 25/05/2018 23:28:47 GMT
1269  13:     UTCTime 22/05/2028 23:28:47 GMT
  :   }
1284  47:   SEQUENCE {
1286  25:     SET {
1288  23:       SEQUENCE {
1290  3:         OBJECT IDENTIFIER serialNumber (2 5 4 5)
1295  16:         PrintableString ‘90e8da3cadfc7820’
  :     }
  :   }

{{\texttt{RATS}}}
1697 15:  
1699 3:     OBJECT IDENTIFIER basicConstraints (2 5 29 19)
1704 1:     BOOLEAN TRUE
1707 5:     OCTET STRING, encapsulates {
1709 3:       SEQUENCE {
1711 1:         BOOLEAN TRUE
1714 14:       }
1716 3:     }
1719 77:     OCTET STRING, encapsulates {
1721 73:       SEQUENCE {
1723 71:         SEQUENCE {
1725 69:           [0] {
1727 67:             [6]
1729 65:               'https://android.googleapis.com/attestation/crl/1'
1731 63:               '7102468407102977850'
1733 61:             }
1735 59:             }
1737 57:             }
1739 55:             }
1741 53:             }
1743 51:             }
1745 49:             }
1747 47:             }
1749 45:             }
1751 43:             }
1753 41:             }
1755 39:             }
1757 37:             }
1759 35:             }
1761 33:             }
1763 31:             }
1765 29:             }
1767 27:             }
1769 25:             }
1771 23:             }
1773 21:             }
1775 19:             }
1777 17:             }
1779 15:             }
1781 13:             }
1783 11:             }
1785 9:             }
1787 7:             }
1789 5:             }
1791 3:             }
1793 1:             }
1795 0:             }
1801 513:       BIT STRING
1803 499:       13 22 DA F2 92 93 CE C0 9F 70 40 C9 DA 85 6B 61
1805 487:       6F 8F BE E0 A4 04 55 C1 63 84 61 37 F5 4B 71 6D
1807 475:       62 AA 6F BF 6C E8 48 03 AD 28 85 21 9E 3C 1C 91

3022 3: INTEGER 65537

3027 166: [3] {

3030 163: SEQUENCE {
3033 29: SEQUENCE {
3035 3: OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
3040 22: OCTET STRING, encapsulates {
3042 20: OCTET STRING
: 36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C C9 EA 4F 12
: }

3064 31: SEQUENCE {
3066 3: OBJECT IDENTIFIER
: authorityKeyIdentifier (2 5 29 35)
3071 24: OCTET STRING, encapsulates {
3073 22: SEQUENCE {
3075 20: [0]
: 36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C C9 EA 4F 12
: }

3097 15: SEQUENCE {
3099 3: OBJECT IDENTIFIER basicConstraints (2 5 29 19)
3104 1: BOOLEAN TRUE
3107 5: OCTET STRING, encapsulates {
3109 3: SEQUENCE {
3111 1: BOOLEAN TRUE
: }

3114 14: SEQUENCE {
3116 3: OBJECT IDENTIFIER keyUsage (2 5 29 15)
3121 1: BOOLEAN TRUE
3124 4: OCTET STRING, encapsulates {
3126 2: BIT STRING 1 unused bit
: '1100001'B
:                 }
:                 }
3130  64:                SEQUENCE {
3132   3:                 OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
3137   57:                 OCTET STRING, encapsulates {
3139   55:                 SEQUENCE {
3141   53:                   OCTET STRING, encapsulates {
3143   51:                     SEQUENCE {
3145   49:                       SEQUENCE {
3147   47:                         [0] { 6 :
3149   45:                           [0] {
3151   43:                             [6] :
3153   41:                               'https://android.googleapis.com/attestation/crl/' :
3155   39:                                 }
3157   37:                                 }
3159   35:                                 }
3161   33:                                 }
3163   31:                                 }
3165   29:                                 }
3167   27:                                 }
3169   25:                                 }
3171   23:                                 }
3173   21:                                 }
3175   19:                                 }
3177   17:                                 }
3179   15:                                 }
3181   13:                                 }
3183   11:                                 }
3185   9:                                 }
3187   7:                                 }
3189   5:                                 }
3191   3:                                 }
3193   1:                                 }
3195  13:                SEQUENCE {
3198   9:                 OBJECT IDENTIFIER
3200   :                   sha256WithRSAEncryption (1 2 840 113549 1 1 11)
3209   0:                 NULL
3211  513:                BIT STRING
3214   :                   20 C8 C3 8D 4B DC A9 57 1B 46 8C 89 2F FF 72 AA
3216   :                   C6 F8 44 A1 1D 41 A8 F0 73 6C C3 7D 16 D6 42 6D
3218   :                   8E 7E 94 07 04 4C EA 39 E6 8B 07 C1 3D BF 15 03
3220   :                   DD 5C 85 BD AF B2 C0 2D 5F 6C DB 4E FA 81 27 DF
3222   :                   8B 04 F1 82 77 0F C4 E7 74 5B 7F CE AA 87 12 9A
3224   :                   88 01 CE 8E 9B C0 CB 96 37 9B 4D 26 A8 2D 30 FD
3226   :                   9C 2F 8E ED 6D C1 BE 2F 84 B6 89 E4 D9 14 25 8B
3228   :                   14 4B BA E6 24 A1 C7 06 71 13 2E 2F 06 16 A8 84
3230   :                   [ Another 384 bytes skipped ]
3728 1413:                SEQUENCE {
3732  877:                SEQUENCE {
3736   3:                 [0] {
3738   1:                 INTEGER 2
3740   :                 }
3741  10:                INTEGER 03 88 26 67 60 65 89 96 85 99
3753  13:                SEQUENCE {
3755   9:                 OBJECT IDENTIFIER
3757   :                   sha256WithRSAEncryption (1 2 840 113549 1 1 11)
3766   0:                 NULL
3768  27:                SEQUENCE {
SET {
  SEQUENCE {
    OBJECT IDENTIFIER serialNumber (2 5 4 5)
    PrintableString 'f92009e853b6b045'
  }
}

SEQUENCE {
  UTCTime 20/06/2018 22:47:35 GMT
  UTCTime 17/06/2028 22:47:35 GMT
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER serialNumber (2 5 4 5)
    PrintableString 'ccd18b9b608d658e'
  }
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER title (2 5 4 12)
    UTF8String 'StrongBox'
  }
}

SEQUENCE {
  SEQUENCE {
    OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
    NULL
  }
  BIT STRING, encapsulates {
  INTEGER 65537
  }
}
SEQUENCE {
  SEQUENCE {
    OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
    OCTET STRING, encapsulates {
      OCTET STRING
      1B 17 70 C6 97 DC 84 54 75 7C 3C 98 5C E6 1D 1D
      08 59 5D 53
    }
  }
  SEQUENCE {
    OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
    OCTET STRING, encapsulates {
      SEQUENCE {
        [0]
        36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C
        C9 EA 4F 12
      }
    }
    SEQUENCE {
    }
  }
  SEQUENCE {
    OBJECT IDENTIFIER basicConstraints (2 5 29 19)
    BOOLEAN TRUE
    OCTET STRING, encapsulates {
      SEQUENCE {
        BOOLEAN TRUE
      }
    }
  }
  SEQUENCE {
    OBJECT IDENTIFIER keyUsage (2 5 29 15)
    BOOLEAN TRUE
    OCTET STRING, encapsulates {
      BIT STRING 2 unused bits
      '100000'B (bit 5)
    }
  }
  SEQUENCE {
    OBJECT IDENTIFIER crlDistributionPoints (2 5 29 31)
    OCTET STRING, encapsulates {
      SEQUENCE {
        [0] {
          [0] {
            [6]
            'https://android.googleapis.com/attestation/crl/8'
            'F6734C9FA504789'
          }
        }
      }
    }
  }
}

7.2. Windows 10 TPM

The next two sections provide two views of a CSR generated via invocation of the Certificate Enrollment Manager API similar to the below:
CertificateRequestProperties request = new CertificateRequestProperties();
request.FriendlyName = "Self-Signed Device Certificate";
request.KeyAlgorithmName = KeyAlgorithmNames.Rsa;
request.KeyStorageProviderName = "Microsoft Smart Card Key Storage Provider";
request.UseExistingKey = true;
request.Exportable = ExportOption.NotExportable;
request.ContainerName = prj.GetContainerName();
request.Subject = subject_name;
request.KeyUsages = keyUsages;
request.SmartcardReaderName = smartCardReaderName;

string privacyCa = 
"MIIDezCCAmAgIBAgIBATANBgkqhkiG9w0BAQsFADBUMQswCQYDVQQGEwJVUzEYMBAGA1UEChM1TlBl
+";

byte[] privacyCaBytes = Convert.FromBase64String(privacyCa);
IBuffer buffer = privacyCaBytes.AsBuffer();
request.AttestationCredentialCertificate = new Certificate(buffer);
csrToDiscard = await CertificateEnrollmentManager.UserCertificateEnrollmentManager.
CreateRequestAsync(request);

The structure is essentially a Full PKI Request as described in RFC 5272.
This section provides an annotation attestation statement as extracted from an encrypted attestation extension. The structure of the attestation statement is defined here:


600 1256:                         SEQUENCE {
604    9:                           OBJECT IDENTIFIER ’1 3 6 1 4 1 311 21 24’
615 1241:                           SET {
619 1237:                             OCTET STRING
     :  4B 41 53 54 01 00 00 00 02 00 00 00 1C 00 00 00
     :  00 00 00 00 B9 04 00 00 00 00 00 00 4B 41 44 53
     :  02 00 00 00 18 00 00 00 A1 00 00 00 01 00 00
     :  00 03 00 00 FF 54 43 47 80 00 18 00 00 00
     :  AB 8A 0B E9 0B BB 3F 7F E6 B6 77 91 EF A9 15 8A
     :  03 B2 2B 8C BE 3F EC 56 B6 30 BF 82 73 9C 00 14
     :  13 6E 2F 14 DD AF 30 72 A6 E3 89 04 4D BF 7A 54 26
     :  36 2F 10 D6 00 00 00 00 00 51 4F CB E5 AD 8C 60
     :  E6 C2 70 80 00 D4 2C 65 4C 95 ED 95 00 22 00
     :  0B 2B E6 2C AD 8D E8 9A 85 04 D7 F3 7B 87 4C F8
     :  32 CD B4 F1 80 CA A6 35 B9 2C 39 87 B7 96 03 C3
     :  A3 00 22 00 0B 6C 88 60 B2 80 E3 BE 7D 34 F2 85
     :  DC 26 9D 1B 72 A8 0A 17 CF 31 08 F1 55 F2 9B 4E
     :  82 C8 5B 49 7B 1A F1 4B 12 A1 C5 D1 A4 C5 A4 59
     :  C4 0A 97 E0 88 ED 1C D3 B6 38 4A 5D 6C 27 F5 69
     :  7D 17 AD F6 C0 03 27 09 5D 93 B5 13 EA 50 B5 05
     :  27 7B A0 51 4D 1B 17 52 87 7D B8 A6 05 4A 4F 39
     :  CA 36 5C A1 19 19 0B 73 B4 0E 7F D3 91 DA 91 EE
     :  37 C6 CE 78 AF 15 21 5D EB 5E 5F 23 A7 08 E9 85
     :  D4 6B A0 95 6D D7 E0 3A D1 92 72 B7 D4 E5 35 6A
     :  01 B0 7D 35 D0 99 BA A1 77 35 76 75 E3 90 A8 8B
     :  86 27 B8 3D 47 75 2D 98 D0 23 4E 09 D8 26 6B 32
     :  3C AB AC 50 A2 E8 FF 70 21 85 C5 5E B1 F5 9C B9
     :  6E 21 27 C7 2A CD 84 61 02 47 6A A0 E1 9A 9F AF
     :  02 43 08 D8 BF 9F 69 14 C4 8C 80 32 2D 5C A3 60
     :  48 F5 5E 8E 65 6B 5E B5 0E A4 ED B9 8B F9 C3 D9

The format is structured as follows:

typedef struct {
    UINT32 Magic;
    UINT32 Version;
    UINT32 Platform;
    UINT32 HeaderSize;
    UINT32 cbIdBinding;
    UINT32 cbKeyAttestation;
    UINT32 cbAIKOpaque;
    BYTE idBinding[cbIdBinding];
    BYTE keyAttestation[cbKeyAttestation];
    BYTE aikOpaque[cbAIKOpaque];
} KeyAttestationStatement;

4B 41 53 54 - Magic
01 00 00 00 - Version
02 00 00 00 - Platform
1C 00 00 00 - HeaderSize
00 00 00 00 - cbIdBinding
B9 04 00 00 - cbKeyAttestation
00 00 00 00 -- cbAIKOpaque

The remainder is the keyAttestation, which is structured as follows:
typedef struct {
    UINT32 Magic;
    UINT32 Platform;
    UINT32 HeaderSize;
    UINT32 cbKeyAttest;
    UINT32 cbSignature;
    UINT32 cbKeyBlob;
    BYTE keyAttest[cbKeyAttest];
    BYTE signature[cbSignature];
    BYTE keyBlob[cbKeyBlob];
} keyAttestation;

4B 41 44 53 - Magic
02 00 00 00 - Platform
18 00 00 00 - HeaderSize
A1 00 00 00 -- cbKeyAttest (161)
00 01 00 00 -- cbSignature (256)
00 03 00 00 - cbKeyBlob

keyAttest (161 bytes) ~~~~~~~~~~~ FF 54 43 47 80 17 00 22 00 0B 9A FD AB 8A 0B E9 0B BB 3F 7F E6 B6 77 91 EF A9 15 8A 03 B2 2B 8C BE 3F EC 56 B6 30 BF 82 73 9C 00 14 13 6E 2F 10 D6 00 00 00 00 00 51 4F CB E5 AD 8C 8C 60 E6 C2 70 80 00 D4 2C 65 4C 6B 95 ED 95 00 22 00 0B 2B E6 2C AD 8D E8 9A 85 04 D7 F3 7B B7 4C F8 32 CD B4 F1 80 CA A6 35 B9 2C 39 87 B7 96 03 C3 A3 00 22 00 0B 6C 88 60 B2 80 E3 BE 7D 34 F2 85 DC 26 9D 1B 72 A8 0A 17 CF 31 08 F1 55 F2 9B 4E 82 C8 5B 49 7B ~~~~~~~~~~~

The keyAttest field is of type TPMS_ATTEST. The TPMS_ATTEST structure is defined in section 10.11.8 of https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-2-Structures-00.99.pdf. ~~~~~~~~~~~ FF 54 43 47 - magic 80 17 - type (TPM_ST_ATTEST_CERTIFY) 00 22 - name - TPM2B_NAME.size (34 bytes) 00 0B 9A FD AB 8A 0B E9 0B BB - TPM2B_NAME.name 3F 7F E6 B6 77 91 EF A9 15 8A 03 B2 2B 8C BE 3F EC 56 B6 30 BF 82 73 9C 00 14 - extraData - TPM2B_DATA.size (20 bytes) 13 6E 2F 14 DD AF 30 72 A6 E3 - TPM2B_DATA.buffer 89 4D BF 7A 54 26 36 2F 10 D6 00 00 00 00 51 4F CB E5 - clockInfo - TPMS_CLOCK_INFO.clock AD 8C 8C 60 - TPMS_CLOCK_INFO.resetCount E6 C2 70 80 - TPMS_CLOCK_INFO.restartCount 00 - - TPMS_CLOCK_INFO.safe D4 2C 65 4C 6B 95 ED 95 - firmwareVersion 00 22 - attested - TPMS_CERTIFY_INFO.name.size 00 0B 2B E6 2C AD 8D E8 9A 85 - TPMS_CERTIFY_INFO.name 04 D7 F3 7B B7 4C F8 32 CD B4 F1 80 CA A6 35 B9 2C 39 87 B7 96 03 C3 A3

Richardson, et al.         Expires May 7, 2020                 [Page 54]
Signature (256 bytes) - generated using the AIK private key

The remainder is the keyBlob, which is defined here:
https://github.com/Microsoft/TSS.MSR/blob/master/PCPTool.v11/inc/TpmAtt.h.

7.3. Yubikey

As with the Android Keystore attestations, Yubikey attestations take the form of an X.509 certificate. As above, the certificate is presented here packaged along with an intermediate CA certificate as a certificates-only SignedData message.

The attestations below were generated using code similar to that found in the yubico-piv-tool (https://github.com/Yubico/yubico-piv-tool). Details regarding attestations are here: https://developers.yubico.com/PIV/Introduction/PIV_attestation.html

7.3.1. Yubikey 4

0 1576: SEQUENCE {
4  9:  OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15 1561:  [0] {
19 1557:   SEQUENCE {
23  1:     INTEGER 1
26  0:     SET { }
28  11:     SEQUENCE {
30  9:      OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
    :     }
41 1533:  [0] {
45  742:    SEQUENCE {
`53  3:            [0] {  
55  1:                INTEGER 2  
:  
58  9:                INTEGER 00 A4 85 22 AA 34 AF AE 4F  
69 13:            SEQUENCE {  
71  9:                OBJECT IDENTIFIER  
:                    sha256WithRSAEncryption (1 2 840 113549 1 1 11)  
82  0:                NULL  
:  
84 43:            SEQUENCE {  
86 41:                SET {  
88 39:                    SEQUENCE {  
90  3:                        OBJECT IDENTIFIER commonName (2 5 4 3)  
95 32:                        UTF8String ‘Yubico PIV Root CA Serial 263751’  
:  
129 32:            SEQUENCE {  
131 13:                UTCTime 14/03/2016 00:00:00 GMT  
146 15:                GeneralizedTime 17/04/2052 00:00:00 GMT  
:  
163 33:            SEQUENCE {  
165 31:                SET {  
167 29:                    SEQUENCE {  
169  3:                        OBJECT IDENTIFIER commonName (2 5 4 3)  
174 22:                        UTF8String ‘Yubico PIV Attestation’  
:  
198 290:            SEQUENCE {  
202 13:                SEQUENCE {  
204  9:                    OBJECT IDENTIFIER  
:                        rsaEncryption (1 2 840 113549 1 1 1)  
215  0:                    NULL  
:  
217 271:            BIT STRING  
:                30 82 01 0A 02 82 01 01 00 AB A9 0B 16 9B EF 31  
:                CC 3E AC 18 5A 2D 45 80 75 70 C7 58 B0 6C 3F 1B  
:                59 0D 49 B9 89 E8 6F CE BB 27 6F D8 3C 60 3A 85  
:                00 EF 5C BC 40 99 3D 41 EE EA C0 81 7F 76 48 E4  
:                A9 4C BC D5 6B E1 1F 0A 60 93 C6 FE AA D2 8D 8E  
:                E2 B7 CD 8B 2B F7 9B DD 5A AB 2F CF B9 0E 54 CE  
:                EC 8D F5 5E D7 7B 91 C3 A7 56 9C DC C1 06 86 76  
:                36 44 53 FB 08 25 D8 06 B9 06 8C 81 FD 63 67 CA  
:                [ Another 142 bytes skipped ]  
:  
492 21:            [3] {  
494 19:                SEQUENCE {  

SEQUENCE {
  OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 3'
  OCTET STRING 04 03 03
  ...
}

SEQUENCE {
  OBJECT IDENTIFIER
    sha256WithRSAEncryption (1 2 840 113549 1 1 11)
  NULL
  }

BIT STRING
  52 80 5A 6D C3 9E DF 47 A8 F1 B2 A5 9C A3 80 81
  3B 1D 6A EB 6A 12 62 4B 11 FD 8D 30 F1 7B FC 71
  10 C9 B2 08 FC D1 4E 35 7F 45 F2 10 A2 52 B9 D4
  B3 02 1A 01 56 07 6B FA 64 A7 08 F0 03 FB 27 A9
  60 8D 0D D3 AC 5A 10 CF 20 96 4E 82 BC 9D E3 37
  DA C1 4C 50 E1 3D 16 B4 CA F4 1B FF 08 64 C9 74
  4F 2A 3A 43 E0 DE 42 79 F2 13 AE 77 A1 E2 AE 6B
  DF 72 A5 B6 CE D7 4C 90 13 DF DE DB F2 8B 34 45
  [ Another 128 bytes skipped ]
}

SEQUENCE {
  INTEGER 2
  }

INTEGER
  00 FE B9 AF 03 3B 0B A7 79 04 02 F5 67 AE DF 72
  ED

SEQUENCE {
  OBJECT IDENTIFIER
    sha256WithRSAEncryption (1 2 840 113549 1 1 11)
  NULL
  }

SEQUENCE {
  SET {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    UTF8String 'Yubico PIV Attestation'
    }
  }

SEQUENCE {
  UTCTime 14/03/2016 00:00:00 GMT
  GeneralizedTime 17/04/2052 00:00:00 GMT
  }
SEQUENCE {
  SET {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    UTF8String 'YubiKey PIV Attestation 9e'
  }
  SET {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    UTF8String 'YubiKey PIV Attestation 9e'
  }
  SEQUENCE {
    OCTET STRING 04 03 03 -- firmware version
  }
  SEQUENCE {
    OCTET STRING 02 03 4F 9B B5 -- serial number
  }
  SEQUENCE {
    OCTET STRING 01 01 -- PIN and touch policy
  }
  SEQUENCE {
    OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
    NULL
  }
  BIT STRING
7.3.2. Yubikey 5

0 1613: SEQUENCE {
4  9:   OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15 1598:   [0] {
19 1594:     SEQUENCE {
23  1:       INTEGER 1
26  0:       SET {}
28 11:     SEQUENCE {
30  9:       OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
32  7:       [0] {
37  5:         SEQUENCE {
41  3:           SEQUENCE {
45  1:             INTEGER 2
48  0:             SET {}
52  3:             INTEGER 00 86 77 17 E0 1D 19 2B 26
56  9:             SEQUENCE {
60  2:               OBJECT IDENTIFIER
63  9:                 sha256WithRSAEncryption (1 2 840 113549 1 1 11)
67  0:               NULL
70  2:             }
74  43:             SEQUENCE {
78  41:               SET {
82  39:                 SEQUENCE {
86  3:                   OBJECT IDENTIFIER commonName (2 5 4 3)
90  32:                   UTF8String ’Yubico PIV Root CA Serial 263751’
94  32:                   [0] {
98  129:                     [Another 128 bytes skipped] 

131 13:             UTCTime 14/03/2016 00:00:00 GMT
146 15:             GeneralizedTime 17/04/2052 00:00:00 GMT
:             }
163 33:             SEQUENCE {
165 31:             SET {
167 29:             SEQUENCE {
169  3:                 OBJECT IDENTIFIER commonName (2 5 4 3)
174 22:                 UTF8String 'Yubico PIV Attestation'
:                 }
:                 }
198 290:             SEQUENCE {
202 13:             SEQUENCE {
204  9:                 OBJECT IDENTIFIER
215  0:                 rsaEncryption (1 2 840 113549 1 1 1)
217 271:             BIT STRING
:             [ Another 142 bytes skipped ]
492 41:             [3] {
494 39:             SEQUENCE {
496 17:             SEQUENCE {
498 10:                 OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 3'
510  3:                 OCTET STRING 05 01 02
:                 }
515 18:             SEQUENCE {
517  3:                 OBJECT IDENTIFIER basicConstraints (2 5 29 19)
522  1:                 BOOLEAN TRUE
525  8:                 OCTET STRING 30 06 01 01 FF 02 01 00
:                 }
535 13:             SEQUENCE {
537  9:                 OBJECT IDENTIFIER
548  0:                 sha256WithRSAEncryption (1 2 840 113549 1 1 11)
550 257:             BIT STRING
: 05 57 B7 BF 5A 41 74 F9 5F EC 2E D2 B8 78 26 E5
: EF 4F EA BF 5A 64 C9 CF 06 7F CA 8C 0A FC 1A 47
: 1C D6 AC ED C8 5B 54 72 00 9F B8 59 AB 73 25 B2
: D6 02 A3 59 83 31 69 EE C1 5F 3D F2 2B 1B 22 CA
: B6 FC F9 FB 21 32 9E 08 F3 08 54 6D C9 26 10 42
: 08 1D 3C B5 F0 5A B1 98 D4 68 DC 91 F1 D3 91 54
: 7A A0 34 BB F6 65 EB 13 9F 3A 1C BF 43 C5 D1 D0
: 33 23 C6 25 A0 4C E4 E9 AA 59 80 D8 02 1E B0 10
: [ Another 128 bytes skipped ]

811 800: SEQUENCE {
815 520: SEQUENCE {
819  3: [0] {
821  1: INTEGER 2
: }
824  16: INTEGER
: 17 7D 2D F7 D6 6D 97 CC D6 CF 69 33 87 5B F1 5E
842  13: SEQUENCE {
844  9: OBJECT IDENTIFIER
: sha256WithRSAEncryption (1 2 840 113549 1 1 11)
855  0: NULL
: }
857  33: SEQUENCE {
859  31: SET {
861  29: SEQUENCE {
863  3: OBJECT IDENTIFIER commonName (2 5 4 3)
868  22: UTF8String 'Yubico PIV Attestation'
: }
: }
892  32: SEQUENCE {
894  13: UTCTime 14/03/2016 00:00:00 GMT
909  15: GeneralizedTime 17/04/2052 00:00:00 GMT
: }
926  37: SEQUENCE {
928  35: SET {
930  33: SEQUENCE {
932  3: OBJECT IDENTIFIER commonName (2 5 4 3)
937  26: UTF8String 'YubiKey PIV Attestation 9e'
: }
: }
965 290: SEQUENCE {
969  13: SEQUENCE {
971  9: OBJECT IDENTIFIER
: rsaEncryption (1 2 840 113549 1 1 1)
982  0: NULL
: }

984 271: BIT STRING
  30 82 01 0A 02 82 01 01 00 A9 02 2D 7A 4C 0B B1
  0C 02 F9 E5 9C E5 6F 2D D1 8D F9 CE B3 B3 4D 1B
  61 B0 B4 E0 3F 44 19 72 88 8B 8D 9F 86 4A 5E C7
  3B F0 AF C9 28 5C DB A2 80 C9 43 93 2D FA 39 7F
  E9 39 2D 18 A7 A2 76 8F D4 6C D0 75 96 99 0D
  06 37 9D 90 D5 71 00 6E FB 82 D1 5B 2A 7C 3B 62
  9E AB 15 81 B9 AD 7F 3D 30 1C 2B 4B 9D C4 D5 64
  32 9A 54 D6 23 B1 65 92 A3 07 57 E2 62 10 2B 93
  [ Another 142 bytes skipped ]

1259 78: [3] {
1261 76: SEQUENCE {
1263 17: SEQUENCE {
1265 10: OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 3'
1277 3: OCTET STRING 05 01 02 -- firmware version
  }
1282 20: SEQUENCE {
1284 10: OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 7'
1296 6: OCTET STRING 02 04 00 93 6A A0 -- serial number
  }
1304 16: SEQUENCE {
1306 10: OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 8'
1318 2: OCTET STRING 01 01 -- PIN and touch policy
  }
1322 15: SEQUENCE {
1324 10: OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 9'
1336 1: OCTET STRING 02 -- form factor
  }
1339 13: SEQUENCE {
1341 9: OBJECT IDENTIFIER
  sha256WithRSAEncryption (1 2 840 113549 1 1 11)
1352 0: NULL
  }
1354 257: BIT STRING
  9F EB 7A 4C F0 7C 67 11 ED C5 84 07 C8 19 41 B2
  71 42 08 28 D6 CD A8 5F DC AK 79 75 6C F1 E5 4D
  28 95 89 69 9D C0 2E A7 D4 48 51 B0 75 FF 63 FD
  B8 79 93 03 EA BB 8A 67 D8 E7 EC C9 1C 8E 3F AF
  74 30 D4 7E 74 A4 26 50 9F D4 57 AE 23 C0 8A 63
  4E F3 C7 CF 5A AF 91 11 A2 6B 3B 49 24 32 26 88
  DB 4F 6F BE BC F0 2D A9 A2 88 B4 5F 54 AF 42 72
  08 74 64 57 76 5A 02 9A 9D 21 4B FD 7F 44 8F AF
  [ Another 128 bytes skipped ]
  }
8. Privacy Considerations.

TBD

9. Security Considerations

TBD.

10. IANA Considerations

TBD.

11. Acknowledgements

Thomas Hardjono provided the text on blockchain system. Dave Thaler suggested many small variations. Frank Xialiang suggested the scaling scenarios that might preclude a 1:1 protocol between attesters and relying parties. Henk Birkholz provided many reviews. Kathleen Moriarty provided many useful edits. Ned Smith, Anders Rundgren and Steve Hanna provided many useful pointers to TCG terms and concepts. Thomas Fossati and Shawn Willden elucidated the Android Keystore goals and limitations.

12. References

12.1. Normative References


12.2. Informative References


[keystore]

[keystore_attestation]
Google, ., "Verifying hardware-backed key pairs with Key Attestation", n.d.,
<https://developer.android.com/training/articles/security-key-attestation>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen,
"Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210,
DOI 10.17487/RFC4210, September 2005,

[RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo,

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652,
DOI 10.17487/RFC5652, September 2009,

"Enrollment over Secure Transport", RFC 7030,
DOI 10.17487/RFC7030, October 2013,

"Automatic Certificate Management Environment (ACME)", RFC 8555,
DOI 10.17487/RFC8555, March 2019,


[SP800-155] NIST, ., "BIOS Integrity Measurement Guidelines (Draft)",
Appendix A.  Changes

o created new section for target use cases

o added comments from Guy, Jessica, Henk and Ned on TCG description.

Authors’ Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca
Carl Wallace  
Red Hound Software  

Email: carl@redhoundsoftware.com

Wei Pan  
Huawei Technologies  

Email: william.panwei@huawei.com