

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

H. Birkholz
Fraunhofer SIT
M. Wiseman
GE Global Research
H. Tschofenig
ARM Ltd.
N. Smith
Intel
M. Richardson
Sandelman Software Works
November 04, 2019

Remote Attestation Procedures Architecture
draft-birkholz-rats-architecture-03

Abstract

An entity (a relying party) requires a source of truth and evidence about a remote peer to assess the peer's trustworthiness. The evidence is typically a believable set of claims about its host, software or hardware platform. This document describes an architecture for such remote attestation procedures (RATS).

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
1.1. Motivation	3
1.2. Opportunities	3
1.3. Overview of Document	4
1.4. RATS in a Nutshell	5
1.5. Remote Attestation Workflow	5
1.6. Message Flows	7
1.6.1. Passport Model	7
1.6.2. Background Check	8
2. Terminology	9
3. Reference use cases	10
3.1. Device Capabilities/Firmware Attestation	11
3.2. IETF TEEP WG Use-Case	11
3.3. Safety Critical Systems	12
3.4. Virtualized Multi-Tenant Hosts	12
3.5. Cryptographic Key Attestation	13
3.6. Geographic Evidence	13
3.7. Device Provenance Attestation	14
4. Conceptual Overview	14
4.1. Two Types of Environments	15
4.2. Evidence Creation Prerequisites	16
4.3. Trustworthiness	17
4.4. Workflow	17
4.5. Interoperability between RATS	18
5. RATS Architecture	18
5.1. Goals	18
5.2. Attestation Principles	18
5.3. Attestation Workflow	19
5.3.1. Roles	19
5.3.2. Role Messages	20
5.4. Principals (Entities?) - Containers for the Roles	22
6. Privacy Considerations	23
7. Security Considerations	23
8. Acknowledgements	23
9. References	23
9.1. Normative References	24
9.2. Informative References	24
Authors' Addresses	25

1. Introduction

Remote Attestation provides a way for an entity (the Relying Party) to determine the health and provenance of an endpoint/host (the Attester). Knowledge of the health of the endpoint allows for a determination of trustworthiness of the endpoint.

1.1. Motivation

The IETF has long spent it's time focusing on threats to the communication channel (see [RFC3552] and [DOLEV-YAO]), assuming that endpoints could be trusted and were under the observation of trusted, well-trained professionals. This assumption has not been true since the days of the campus mini-computer. For some time after the desktop PC became ubiquitous, the threat to the endpoints has been dealt with as an internal matter, with generally poor results. Enterprises have done some deployment of Network Endpoint Assessment ([RFC5209]) to assess the security posture about an endpoint, but it has not been ubiquitous.

The movement towards personal mobile devices ("smartphones") and the continuing threat from unmanaged residential desktops has resulted in a renewed interest in standardizing internet-scale endpoint remote attestation. Additionally, the rise of the Internet of Things (IoT) has made this issue even more critical: some skeptics have even renamed it to the Internet of Threats [iothreats] :-). IoT devices have poor or non-existent user interfaces, as such as there are not even good ways to assess the health of the devices manually: a need to determine the health via remote attestation is now critical.

In addition to the health of the device, knowledge of its provenance helps to determine the level of trust, and prevents attacks to the supply chain.

1.2. Opportunities

The Trusted Platform Module (TPM) is now a commonly available peripheral on many commodity compute platforms, both servers and desktops. Smartphones commonly have either an actual TPM, or have the ability to emulate one in software running in a Trusted Execution Environment [I-D.ietf-teep-architecture]. There are now few barriers to creating a standards-based system for remote attestation procedures.

A number of niche solutions have emerged that provide for use-case specific remote attestation, but none have the generality needed to be used across the Internet.

1.3. Overview of Document

The architecture described in this document (along with the accompanying solution and reference documents) enables the use of common formats for communicating Claims about an Attester to a Relying Party. [FIXME Attester? Flows? To what end?]

Existing transports were not designed to carry attestation Claims. It is therefore necessary to design serializations of Claims that fit into a variety of transports, for instance: X.509 certificates, TLS negotiations, YANG modules or EtherNet/IP. There are also new, greenfield uses for remote attestation. Transport and serialization of these can be done without retrofitting. This is (will be) described in [INSERT reference to adopted document on transport].

While it is not anticipated that the existing niche solutions described in the use cases section Section 3 will exchange claims directly, the use of a common format enables common code. As some of the code needs to be in intentionally hard to modify trusted modules, the use of a common formats and transfer protocols significantly reduces the cost of adoption to all parties. This commonality also significantly reduces the incidence of critical bugs.

In some environments the collection of Evidence by the Attester to be provided to the Verifier is part of an existing protocol: this document does not change that, rather embraces those legacy mechanisms as part of the specification. This is an evolutionary path forward, not revolutionary. Yet in other greenfield environments, there is a desire to have a standard for Evidence as well as for Attestation Results, and this architecture outlines how that is done.

This introduction gives an overview of the message flows and roles involved. Following this, is a terminology section that is referenced normatively by other documents and is a key part of this document. There is then a section on use cases and how they leverage the roles and workflows described.

In this document, terms defined within this document are consistently Capitalized [work in progress. please raise issues, if there are Blatant inconsistencies].

Current verticals that use remote attestation include:

- o The Trusted Computing Group "Network Device Attestation Workflow" [I-D.fedorkow-rats-network-device-attestation]
- o Android Keystore [keystore]

- o Fast Identity Online (FIDO) Alliance attestation [fido]
- o A number of Intel SGX niche systems based upon OTRP.

1.4. RATS in a Nutshell

1. Remote Attestation message flows typically convey Claims that contain the trustworthiness properties associated with an Attested Environment (Evidence).
2. A corresponding provisioning message flows conveys Reference trustworthiness claims that can be compared with attestation Evidence. Reference Values typically consist of firmware or software digests and details about what makes the attesting module a trusted source of Evidence.
3. Relying Parties are performing tasks such as managing a resource, controlling access, and/or managing risk. Attestation Results helps Relying Parties determine levels of trust.

1.5. Remote Attestation Workflow

The logical information flow is from Attester to Verifier to Relying Party. There are variations presented below on how this integrates into actual protocols.

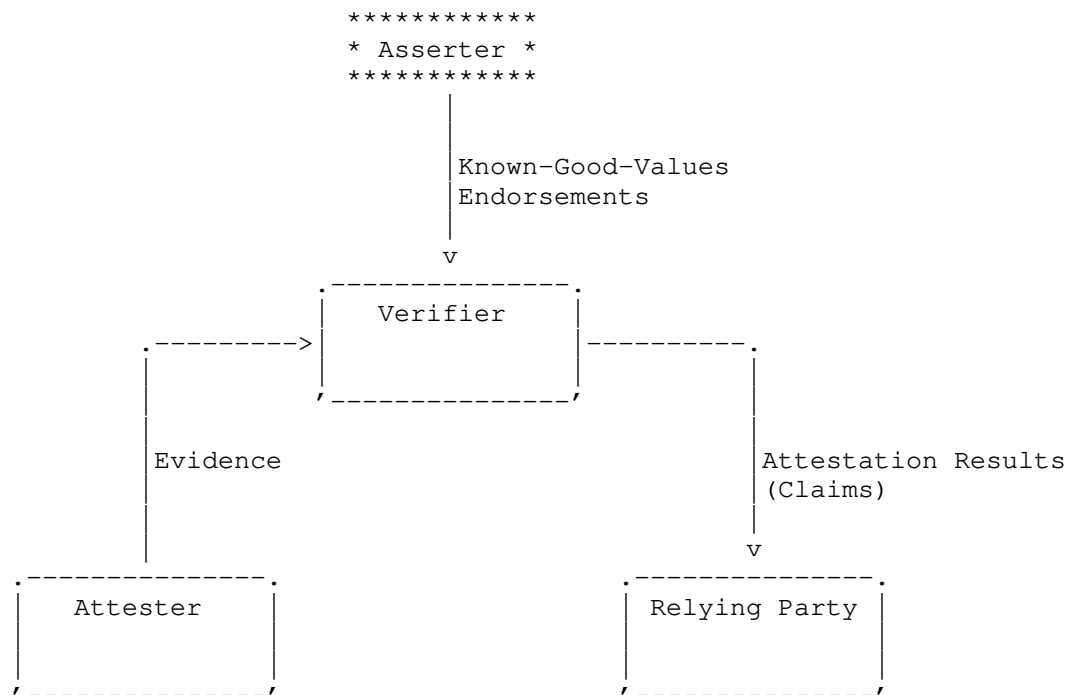


Figure 1: RATS Workflow

In the architecture shown above, specific content items (payload conveyed in message flows) are identified:

- o Evidence is as set of believable Claims about distinguishable Environments made by an Attester.
- o Known-Good-Values are reference Claims used to appraise Evidence by an Verifier.
- o Endorsements are reference Claims about the type of protection that enables an Attester to create believable Evidence. Endorsements enable trust relationships towards system components or environments Evidence cannot be created for by an Attester.
- o Attestation Results are the output from the appraisal of Evidence, Known-Good-Values and Endorsements and are consumed by Relying Parties.

Attestation Results are the output of RATS.

Assessment of Attestation Results is be multi-faceted and out-of-scope for the architecture.

If appropriate Endorsements about the Attester are available, Known-Good-Values about the Attester are available, and if the Attester is capable of creating believable Evidence - then the Verifier is able to create Attestation Results that enable Relying Parties to establish a level of confidence in the trustworthiness of the Attester.

The Asserter role and the format for Known-Good-Values and Endorsements are not subject to standardization at this time. The current verticals already include provisions for encoding and/or distributing these objects.

1.6. Message Flows

Two distinct flows have been identified for passage of Evidence and production of Attestation Results. It is possible that there are additional situations which are not captured by these two flows.

1.6.1. Passport Model

In the Passport Model message flow the Attester provides it's Evidence directly to the Verifier. The Verifier will evaluate the Evidence and then sign an Attestation Result. This Attestation Result is returned to the Attester, and it is up to the Attester to communicate the Attestation Result (potentially including the Evidence, if disclosable) to the Relying Party.

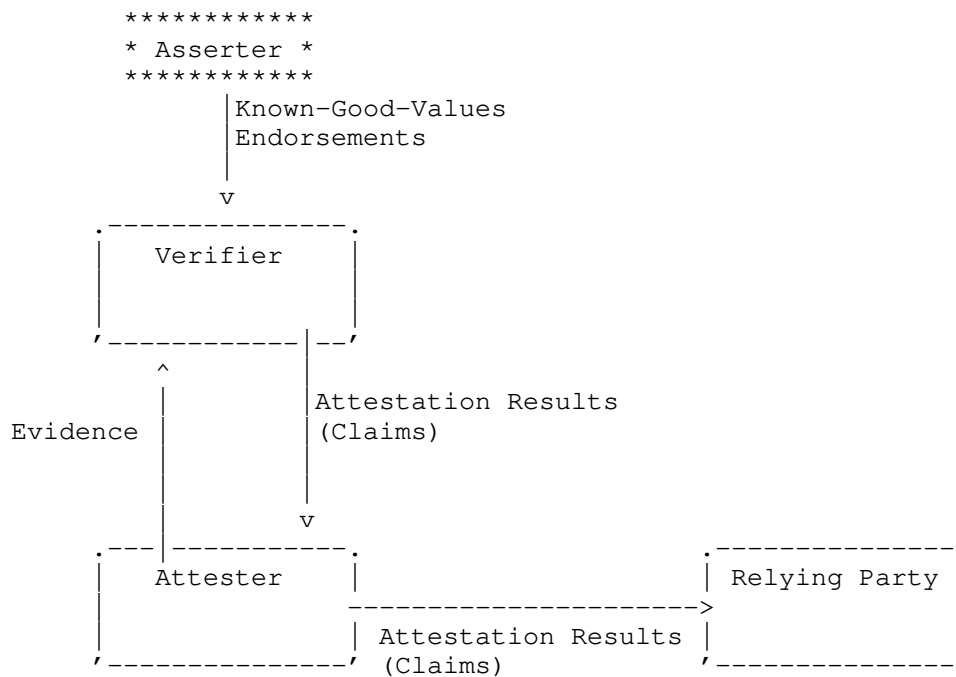


Figure 2: RATS Passport Flow

This flow is named in this way because of the resemblance of how Nations issue Passports to their citizens. The nature of the Evidence that an individual needs to provide to it's local authority is specific to the country involved. The citizen retains control of the resulting document and presents it to other entities when it needs to assert a citizenship or identity claim.

1.6.2. Background Check

In the Background-Check message flow the Attester provides it's Evidence to the Relying Party. The Relying Party sends this evidence to a Verifier of its choice. The Verifier will evaluate the Evidence and then sign an Attestation Result. This Attestation Result is returned to the Relying Party, which processes it directly.

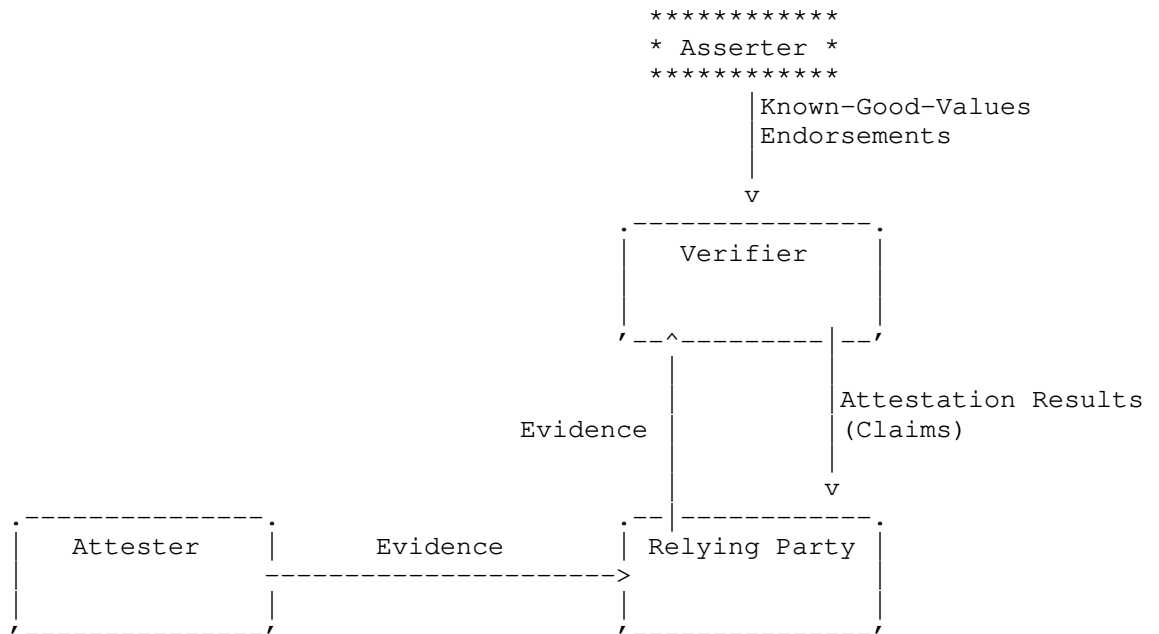


Figure 3: RATS Background Check Flow

This flow is named in this way because of the resemblance of how employers and volunteer organizations perform background checks. When a prospective employee provides claims about education or previous experience, the employer will contact the respective institutions or former employers to validate the claim. Volunteer organizations often perform police background checks on volunteers in order to determine the volunteer's trustworthiness.

2. Terminology

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.

Appraisal: A Verifier process that compares Evidence to Reference values while taking into account Endorsements and produces Attestation Results.

Asserter: See Section 5.3.1.2.

Attester: See Section 5.3.1.1.

Attested Environment: A target environment that is observed or controlled by an Attesting Environment.

Attesting Environment: An environment capable of making trustworthiness Claims about an Attested Environment.

Background-Check Message Flow: An attestation workflow where the Attester provides Evidence to a Relying Party, who consults one or more Verifiers who supply Attestation Results to the Relying Party. See Section 1.6.2.

Claim: A statement about the construction, composition, validation or behavior of an Entity that affects trustworthiness. Evidence, Reference Values and Attestation Results are expressions that consists of one or more Claims.

Conveyance: The process of transferring Evidence, Reference Values and Attestation Results between Entities participating in attestation workflow.

Entity: A device, component (see System Component [RFC4949]), or environment that implements one or more Roles.

Evidence: See Section 5.3.2.1.

Passport Message Flow: An attestation workflow where the Attester provides Evidence to a Verifier who returns Attestation Results that are then forwarded to one or more Relying Parties. See Section 1.6.1.

Reference Values: See Section 5.3.2.2. Also referred to as Known-Good-Values.

Relying Party: See Section 5.3.1.4.

Attestation Results: See Section 5.3.2.3.

Role: A function or process in an attestation workflow, typically described by: Attester, Verifier, Relying Party and Asserter.

Verifier: See Section 5.3.1.3.

3. Reference use cases

This section provides an overview of a number of distinct use cases that benefit from a standardized claim format. In addition to outlining the user, the specific message flow is identified from among the flows detailed in Section 1.6.

3.1. Device Capabilities/Firmware Attestation

This is a large category of claims that includes a number of subcategories, not detailed here.

Use case name: Device Identity

Who will use it: Network Operators, larger enterprises

Attester: varies

Message Flow: sometimes passport and sometimes background check

Relying Party: varies

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"

3.2. 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

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.

3.3. Safety Critical Systems

Use case name: Safety Critical Systems

Who will use it: Power plants and other systems that need to assert their current state, but which can not accept any inputs from the outside. The corollary system is a black-box (such as in an aircraft), which needs to log the state of a system, but which can never initiate a handshake.

Message Flow: background check

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: These requirements motivate the creation of the Time-Base Unidirectional Attestation (TUDA) [I-D.birkholz-rats-tuda], the output of TUDA is typically a secure audit log, where freshness is determined by synchronization to a trusted source of external time.

The freshness is preserved in the Evidence by the use of a Time Stamp Authority (TSA) which provides Time Stamp Tokens (TST).

3.4. Virtualized Multi-Tenant Hosts

Use case name: Multi-Tenant Hosts

Who will use it: Virtual machine systems

Message Flow: passport

Attester: virtual machine hypervisor

Relying Party: network operators

Description: The host system will do verification as per Section 3.1

The tenant virtual machines will do verification as per Section 3.1.

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.

3.5. Cryptographic Key Attestation

Cryptographic Attestation includes subcategories such as Device Type Attestation (the FIDO use case), and Key storage Attestation (the Android Keystore use case), and End-User Authorization.

Use case name: Key Attestation

Who will use it: network authentication systems

Message Flow: passport

Attester: device platform

Relying Party: internet peers

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.

The Device Type Attestation is provided by a Firmware TPM performing the Verifier function, creating Attestation Results that indicate a particular model/type of device. In TCG terms, this is called Implicit Attestation, in this case the Attested Environment is the (smartphone) Rich Execution Environment (REE) ([I-D.ietf-teep-architecture] section 2), and the Attesting Environment is within the TEE.

3.6. Geographic Evidence

Use case name: Location Evidence

Who will use it: geo-fenced systems

Message Flow: passport (probably)

Attester: secure GPS system(s)

Relying Party: internet peers

Description: The relying party wants to know the physical location (on the planet earth, using a geodetic system) of the device. This may be provided directly by a GPS/GLONASS/BeiDou/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.

3.7. 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

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.

4. Conceptual Overview

In network protocol exchanges, it is often the case that one entity (a Relying Party) requires an assessment of the trustworthiness of a remote entity (an Attester or specific system components [RFC4949]

thereof). Remote ATtestation procedureS (RATS) enable Relying Parties to establish a level of confidence in the trustworthiness of Attesters through the

- o Creation,
- o Conveyance, and
- o Appraisal

of attestation Evidence.

Qualities of Evidence: Evidence is composed of Claims about trustworthiness (the set of Claims is unbounded). The system characteristics of Attesters - the Environments they are composed of, and their continuous development - have an impact on the veracity of trustworthiness Claims included in valid Evidence.

Valid Evidence about the intactness of an Attester must be impossible to create by an untrustworthy or compromised Environment of an Attester.

Qualities of Environments: The resilience of Environments that are part of an Attester can vary widely - ranging from those highly resistant to attacks to those having little or no resistance to attacks. Configuration options, if set poorly, can result in a highly resistant environment being operationally less resistant. When a trustworthy Environment changes, it is possible that it transitions from being trustworthy to being untrustworthy.

An untrustworthy or compromised Environment must never be able to create valid Evidence expressing the intactness of an Attester.

The architecture provides a framework for anticipating when a relevant change with respect to a trustworthiness attribute occurs, what exactly changed and how relevant it is. The architecture also creates a context for enabling an appropriate response by applications, system software and protocol endpoints when changes to trustworthiness attributes do occur.

Detailed protocol specifications for message flows are defined in separate documents.

4.1. Two Types of Environments

An Attester produces Evidence about its own integrity, which means "it measures itself". To disambiguate this recursive or circular

looking relationships, two types of Environments inside an Attester are distinguished:

The attest-ED Environments and the attest-ING Environments.

Attested Environments are measured. They provide the raw values and the information to be represented in Claims and ultimately expressed as Evidence.

Attesting Environments conduct the measuring. They collect the Claims, format them appropriately, and typically use key material and cryptographic functions, such as signing or cipher algorithms, to create Evidence.

Attesting Environments use system components that have to be trusted. As a result, Evidence includes Claims about the Attested and the Attesting Environments. Claims about the Attested Environments are appraised using Reference Values and Claims about the Attesting Environments are appraised using Endorsements. It is not mandated that both Environments have to be separate, but it is highly encouraged. Examples of separated Environments that can be used as Attesting Environments include: Trusted Execution Environments (TEE), embedded Secure Elements (eSE), or Hardware Security Modules (HSM).

In summary, the majority of the creation of evidence can take place in an Attested Environments. Exemplary duties include the collection and formatting of Claim values, or the trigger for creating Evidence. A trusted sub-set of the creation of evidence can take place in an Attesting Environment, that provide special protection with respect to key material, identity documents, or primitive functions to create the Evidence itself.

4.2. Evidence Creation Prerequisites

One or more Environments that are part of an Attester must be able to conduct the following duties in order to create Evidence:

- o monitoring trustworthiness attributes of other Environments,
- o collecting trustworthiness attributes and create Claims about them,
- o serialize Claims using interoperable representations,
- o provide integrity protection for the sets of Claims, and
- o add appropriate attestation provenance attributes about the sets of Claims.

4.3. Trustworthiness

The trustworthiness of an Attester and therefore the believability of the Evidence it creates relies on the protection methods in place to shield and restrict the use of key material and the duties conducted by the Attesting Environment. In order to assess trustworthiness effectively, it is mandatory to understand the trustworthiness properties of the environments of an Attester. The corresponding appraisal of Evidence that leads to such an assessment of trustworthiness is the duty of a Verifier.

Trusting the assessment of a Verifier might come from trusting the Verifier's key material (direct trust), or trusting an Entity that the Verifier is associated with via a certification path (indirect trust).

The trustworthiness of corresponding Attestation Results also relies on trust towards manufacturers and those manufacturer's hardware in order to assess the integrity and resilience of that manufacturer's devices.

A stronger level of security comes when information can be vouched for by hardware or by (unchangeable) firmware, especially if such hardware is physically resistant to hardware tampering. The component that is implicitly trusted is often referred to as a Root of Trust.

4.4. Workflow

The basic function of RATS is creation, conveyance and appraisal of attestation Evidence. An Attester creates attestation Evidence that are conveyed to a Verifier for appraisal. The appraisals compare Evidence with expected Known-Good-Values obtained from Asserters (e.g. Principals that are Supply Chain Entities). There can be multiple forms of appraisal (e.g., software integrity verification, device composition and configuration verification, device identity and provenance verification). Attestation Results are the output of appraisals. Attestation Results are signed and conveyed to Relying Parties. Attestation Results provide the basis by which the Relying Party may determine a level of confidence to place in the application data or operations that follow.

The architecture defines attestation Roles: Attester, Verifier, Asserter and Relying Party. Roles exchange messages, but their structure is not defined in this document. The detailed definition of the messages is in an appropriate document, such as [I-D.ietf-rats-eat] or other protocols to be defined. Roles can be combined in various ways into Principals, depending upon the needs of

the use case. Information Model representations are realized as data structure and conveyance protocol specifications.

4.5. Interoperability between RATS

The RATS architecture anticipates use of information modeling techniques that describe computing structures - their components/ computational elements and corresponding capabilities - so that verification operations may rely on the information model as an interoperable way to navigate the structural complexity.

5. RATS Architecture

5.1. Goals

RATS architecture has the following goals:

- o Enable semantic interoperability of attestation semantics through information models about computing environments and trustworthiness.
- o Enable data structure interoperability related to claims, endpoint composition / structure, and end-to-end integrity and confidentiality protection mechanisms.
- o Enable programmatic assessment of trustworthiness. (Note: Mechanisms that manage risk, justify a level of confidence, or determine a consequence of an attestation result are out of scope).
- o Provide the building blocks, including Roles and Principals that enable the composition of service-chains/hierarchies and workflows that can create and appraise evidence about the trustworthiness of devices and services.
- o Use-case driven architecture and design (see [I-D.richardson-rats-usecases] and Section 3)
- o Terminology conventions that are consistently applied across RATS specifications.
- o Reinforce trusted computing principles that include attestation.

5.2. Attestation Principles

Specifications developed by the RATS working group apply the following principles:

- o Freshness - replay of previously asserted Claims about an Attested Environment can be detected.
- o Identity - the Attesting Environment is identifiable (non-anonymous).
- o Context - the Attested Environment is well-defined (unambiguous).
- o Provenance - the origin of Claims with respect to the Attested and Attesting Environments are known.
- o Validity - the expected lifetime of Claims about an Attested Environment is known.
- o Veracity - the believability (level of confidence) of Claims is based on verifiable proofs.

5.3. Attestation Workflow

Attestation workflow helps a Relying Party make better decisions by providing insight about the trustworthiness of endpoints participating in a distributed system. The workflow consists primarily of four roles; Relying Party, Verifier, Attester and Asserter. Attestation messages contain information useful for appraising the trustworthiness of an Attester endpoint and informing the Relying Party of the appraisal result.

This section details the primary roles of an attestation workflow and the messages they exchange.

5.3.1. Roles

An endpoint system (a.k.a., Entity) may implement one or more attestation Roles to accommodate a variety of possible message flows. Exemplary message flows are described in Section 1.6.1 and Section 1.6.2. Role messages are secured by the Entity that generated it. Entities possess credentials (e.g., cryptographic keys) that authenticate, integrity protect and optionally confidentiality protect attestation messages.

5.3.1.1. Attester

The Attester consists of both an Attesting Environment and an Attested Environment. In some implementations these environments may be combined. Other implementations may have multiples of Attesting and Attested environments. Although endpoint environments can be complex, and that complexity is security relevant, the basic function

of an Attester is to create Evidence that captures operational conditions affecting trustworthiness.

5.3.1.2. Asserter

The Asserter role is out of scope. The mechanism by which an Asserter communicates Known-Good-Values to a Verifier is also not subject to standardization. Users of the RATS architecture are assumed to have pre-existing mechanisms.

5.3.1.3. Verifier

The Verifier workflow function accepts Evidence from an Attester and accepts Reference Values from one or more Asserters. Reference values may be supplied a priori, cached or used to create policies. The Verifier performs an appraisal by matching Claims found in Evidence with Claims found in Reference Values and policies. If an attested Claim value differs from an expected Claim value, the Verifier flags this as a change possibly impacting trust level.

Endorsements may not have corresponding Claims in Evidence (because of their intrinsic nature). Consequently, the Verifier need only authenticate the endpoint and verify the Endorsements match the endpoint identity.

The result of appraisals and Endorsements, informed by owner policies, produces a new set of Claims that a Relying Party is suited to consume.

5.3.1.4. Relying Party

A Role in an attestation workflow that accepts Attestation Results from a Verifier that may be used by the Relying Party to inform application specific decision making. How Attestation Results are used to inform decision making is out-of-scope for this architecture.

5.3.2. Role Messages

5.3.2.1. Evidence

Claims that are formatted and protected by an Attester.

Evidence SHOULD satisfy Verifier expectations for freshness, identity, context, provenance, validity, and veracity.

5.3.2.2. Reference Values

Reference-values are Claims that a manufacturer, vendor or other supply chain entity makes that affects the trustworthiness of an Attester endpoint.

Claims may be persistent properties of the endpoint due to the physical nature of how it was manufactured or may reflect the processes that were followed as part of moving the endpoint through a supply-chain; e.g., validation or compliance testing. This class of Reference-values is known as Endorsements.

Another class of Reference-values identifies the firmware and software that could be installed in the endpoint after its manufacture. A digest of the the firmware or software can be an effective identifier for keeping track of the images produced by vendors and installed on an endpoint. This class of Reference-value is referred to as Known-Good-Value (KGV).

Known-Good-Values: Claims about the Attested Environment.

Typically, Known-Good-Value (KGV) Claims are message digests of firmware, software or configuration data supplied by various vendors. If an Attesting Environment implements cryptography, they include Claims about key material.

Like Claims, Known-Good-Values SHOULD satisfy a Verifier's expectations for freshness, identity, context, provenance, validity, relevance and veracity. Known-Good-Values are reference Claims that are - like Evidence - well formatted and protected (e.g. signed).

Endorsements: Claims about immutable and implicit characteristics of the Attesting Environment. Typically, endorsement Claims are created by manufacturing or supply chain entities.

Endorsements are intended to increase the level of confidence with respect to Evidence created by an Attester.

5.3.2.3. Attestation Results

Statements about the output of an appraisal of Evidence that are created, formatted and protected by a Verifier.

Attestation Results provide the basis for a Relying Party to establish a level of confidence in the trustworthiness of an Attester. Attestation Results SHOULD satisfy Relying Party expectations for freshness, identity, context, provenance, validity, relevance and veracity.

5.4. Principals (Entities?) - Containers for the Roles

[The authors are unhappy with the term Principal, and have been looking for something else. JOSE/JWT uses the term Principal]

Principals are Containers for the Roles.

Principals are users, organizations, devices and computing environments (e.g., devices, platforms, services, peripherals).

Principals may implement one or more Roles. Message flows occurring within the same Principal are out-of-scope.

The methods whereby Principals may be identified, discovered, authenticated, connected and trusted, though important, are out-of-scope.

Principal operations that apply resiliency, scaling, load balancing or replication are generally believed to be out-of-scope.

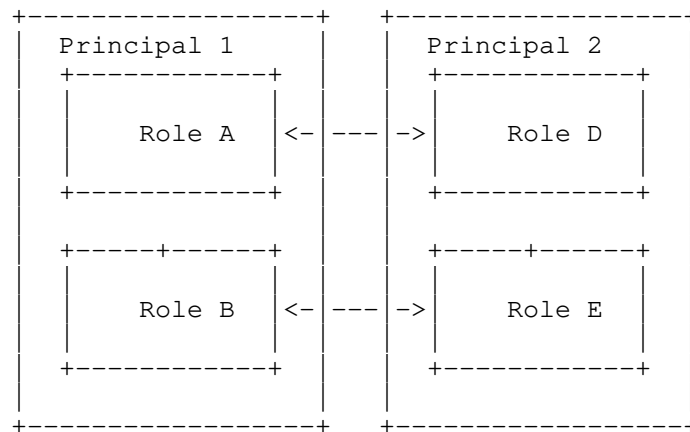


Figure 4: Principals-Role Composition

Principals have the following properties:

- o Multiplicity - Multiple instances of Principals that possess the same Roles can exist.
- o Composition - Principals possessing different Roles can be combined into a singleton Principal possessing the union of Roles. Message flows between combined Principals is uninteresting.

- o Decomposition - A singleton Principal possessing multiple Roles can be divided into multiple Principals.

6. Privacy Considerations

The conveyance of Evidence and the resulting Attestation Results reveal a great deal of information about the internal state of a device. In many cases the whole point of the Attestation process is to provide reliable evidence about the type of the device and the firmware that the device is running. This information is particularly interesting to many attackers: knowing that a device is running a weak version of a the firmware provides a way to aim attacks better.

Just knowing the existence of a device is itself a disclosure.

Conveyance protocols must detail what kinds of information is disclosed, and to whom it is exposed.

7. Security Considerations

Evidence, Verifiable Assertions and Attestation Results SHOULD use formats that support end-to-end integrity protection and MAY support end-to-end confidentiality protection.

Replay attacks are a concern that protocol implementations MUST deal with. This is typically done via a Nonce Claim, but the details belong to the protocol.

All other attacks involving RATS structures are not explicitly addressed by the architecture.

Additional security protections MAY be required of conveyance mechanisms. For example, additional means of authentication, confidentiality, integrity, replay, denial of service and privacy protection of RATS payloads and Principals may be needed.

8. Acknowledgements

Dave Thaler created the concepts of "Passport" and "Background Check".

9. References

9.1. Normative References

- [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>>.
- [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>>.

9.2. Informative References

- [ABLP] Abadi, M., Burrows, M., Lampson, B., and G. Plotkin, "A Calculus for Access Control in Distributed Systems", Springer Annual International Cryptology Conference, page 1-23, DOI 10.1.1.36.691, 1991.
- [DOLEV-YAO] Dolev, D. and A. Yao, "On the security of public key protocols", IEEE Transactions on Information Theory Vol. 29, pp. 198-208, DOI 10.1109/tit.1983.1056650, March 1983.
- [fido] FIDO Alliance, ., "FIDO Specification Overview", 2019, <<https://fidoalliance.org/specifications/>>.
- [I-D.birkholz-rats-tuda] Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", draft-birkholz-rats-tuda-01 (work in progress), September 2019.
- [I-D.fedorkow-rats-network-device-attestation] Fedorkow, G. and J. Fitzgerald-McKay, "Network Device Attestation Workflow", draft-fedorkow-rats-network-device-attestation-00 (work in progress), July 2019.
- [I-D.ietf-rats-eat] Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", draft-ietf-rats-eat-01 (work in progress), July 2019.
- [I-D.ietf-teep-architecture] Pei, M., Tschofenig, H., Wheeler, D., Atyeo, A., and D. Liu, "Trusted Execution Environment Provisioning (TEEP) Architecture", draft-ietf-teep-architecture-03 (work in progress), July 2019.

- [I-D.richardson-rats-usecases]
Richardson, M., Wallace, C., and W. Pan, "Use cases for Remote Attestation common encodings", draft-richardson-rats-usecases-05 (work in progress), October 2019.
- [iothreats]
GDN, ., "The Internet of Things or the Internet of threats?", 2016, <<https://gcn.com/articles/2016/05/03/internet-of-threats.aspx>>.
- [keystore]
Google, ., "Android Keystore System", 2019, <<https://developer.android.com/training/articles/keystore>>.
- [Lampson2007]
Lampson, B., "Practical Principles for Computer Security", IOSPress Proceedings of Software System Reliability and Security, page 151-195, DOI 10.1.1.63.5360, 2007.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<https://www.rfc-editor.org/info/rfc4949>>.
- [RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <<https://www.rfc-editor.org/info/rfc5209>>.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Monty Wiseman
GE Global Research
USA

Email: monty.wiseman@ge.com

Hannes Tschofenig
ARM Ltd.
110 Fulbourn Rd
Cambridge CB1 9NJ
UK

Email: hannes.tschofenig@gmx.net

Ned Smith
Intel Corporation
USA

Email: ned.smith@intel.com

Michael Richardson
Sandelman Software Works
Canada

Email: mcr+ietf@sandelman.ca

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

H. Birkholz
M. Eckel
Fraunhofer SIT
S. Bhandari
B. Sulzen
E. Voit
Cisco
L. Xia
Huawei
T. Laffey
HPE
G. Fedorkow
Juniper
July 08, 2019

YANG Module for Basic Challenge-Response-based Remote Attestation
Procedures
draft-birkholz-rats-basic-yang-module-01

Abstract

This document defines a YANG RPC and a minimal datastore tree required to retrieve attestation evidence about integrity measurements from a composite device with one or more roots of trust for reporting. Complementary measurement logs are also provided by the YANG RPC originating from one or more roots of trust of measurement. The module defined requires a TPM 2.0 and corresponding Trusted Software Stack included in the device components of the composite device the YANG server is running on.

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 January 9, 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	2
1.1. Requirements notation	3
2. The YANG Module for Basic Remote Attestation Procedures . . .	3
2.1. Tree format	3
2.2. Raw Format	7
3. IANA considerations	29
4. Security Considerations	29
5. Acknowledgements	29
6. Change Log	29
7. References	29
7.1. Normative References	29
7.2. Informative References	30
Authors' Addresses	30

1. Introduction

This document is based on the terminology defined in the [I-D.birkholz-attestation-terminology] and uses the interaction model and information elements defined in the [I-D.birkholz-rats-reference-interaction-model] document. The currently supported hardware security module (HWM) - sometimes also referred to as an embedded secure element (eSE) - is the Trusted Platform Module (TPM) 2.0 specified by the Trusted Computing Group (TCG). One or more TPM 2.0 embedded in the components of a composite device - sometimes also referred to as an aggregate device - are required in order to use the YANG module defined in this document. A TPM 2.0 is used as a root of trust for reporting (RTR) in order to retrieve attestation evidence from a composite device. Additionally, it is used as a root of trust for measurement (RTM) in order to provide event logs - sometimes also referred to as measurement logs.

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 RFC 2119, BCP 14 [RFC2119].

2. The YANG Module for Basic Remote Attestation Procedures

One or more TPM 2.0 MUST be embedded in the composite device that is providing attestation evidence via the YANG module defined in this document. The ietf-basic-remote-attestation YANG module enables a composite device to take on the role of Claimant and Attester in accordance with the Remote Attestation Procedures (RATS) architecture [I-D.birkholz-attestation-terminology] and the corresponding challenge-response interaction model defined in the [I-D.birkholz-rats-reference-interaction-model] document. A fresh nonce with an appropriate amount of entropy MUST be supplied by the YANG client in order to enable a proof-of-freshness with respect to the attestation evidence provided by the attester running the YANG datastore. The functions of this YANG module are restricted to 0-1 TPM 2.0 per hardware component.

2.1. Tree format

<CODE BEGINS>

```
module: ietf-basic-remote-attestation
  +--ro rats-support-structures
    +--ro supported-algos*   uint16
    +--ro tpms* [tpm_name]
      +--ro tpm_name          string
      +--ro tpm-physical-index? int32 {ietfhw:entity-mib}?
      +--ro certificates* []
        +--ro certificate
          +--ro certificate-name?   string
          +--ro certificate-type?   enumeration
          +--ro certificate-value?  ietfct:end-entity-cert-cms
    +--ro compute-nodes* [node-name]
      +--ro node-name          string
      +--ro node-physical-index? int32 {ietfhw:entity-mib}?

  rpcs:
    +---x tpm12-challenge-response-attestation
      +---w input
        +---w tpm1-attestation-challenge
          +---w pcr-indices*          uint8
          +---w nonce-value           binary
          +---w TPM_SIG_SCHEME-value  uint8
```

```

+---w (key-identifier)?
+---:(public-key)
| +---w pub-key-id?          binary
+---:(TSS_UUID)
+---w TSS_UUID-value
+---w ulTimeLow?            uint32
+---w usTimeMid?            uint16
+---w usTimeHigh?           uint16
+---w bClockSeqHigh?        uint8
+---w bClockSeqLow?         uint8
+---w rgbNode*              uint8
+---w add-version?          boolean
+---w tpm_name?              string
+---w tpm-physical-index?    int32 {ietfhw:entity-mib}?
+---ro output
+---ro tpm12-attestation-response* [tpm_name]
+---ro tpm_name              string
+---ro tpm-physical-index?    int32 {ietfhw:entity-mib}?
+---ro up-time?              uint32
+---ro node-name?            string
+---ro node-physical-index?   int32 {ietfhw:entity-mib}?
+---ro fixed?                binary
+---ro external-data?        binary
+---ro signature-size?       uint32
+---ro signature?            binary
+---ro (tpm12-quote)
+---:(tpm12-quote1)
| +---ro version* []
| | +---ro major?          uint8
| | +---ro minor?          uint8
| | +---ro revMajor?       uint8
| | +---ro revMinor?       uint8
| +---ro digest-value?      binary
| +---ro TPM_PCR_COMPOSITE* []
| | +---ro pcr-indices*    uint8
| | +---ro value-size?     uint32
| | +---ro tpm12-pcr-value* binary
+---:(tpm12-quote2)
+---ro tag?                  uint8
+---ro pcr-indices*          uint8
+---ro locality-at-release?  uint8
+---ro digest-at-release?    binary
+---x tpm20-challenge-response-attestation
+---w input
+---w tpm20-attestation-challenge
+---w pcr-list* []
+---w pcr
+---w pcr-indices*          uint8

```

```

      +---w (algo-registry-type)
      +---: (tcg)
      |   +---w tcg-hash-algo-id?          uint16
      +---: (ietf)
      |   +---w ietf-ni-hash-algo-id?      uint8
      +---w nonce-value                    binary
      +---w (signature-identifier-type)
      |   +---: (TPM_ALG_ID)
      |   |   +---w TPM_ALG_ID-value?      uint16
      +---: (COSE_Algorithm)
      |   +---w COSE_Algorithm-value?      int32
      +---w (key-identifier)?
      |   +---: (public-key)
      |   |   +---w pub-key-id?            binary
      +---: (uuid)
      |   +---w uuid-value?                binary
      +---w tpms* [tpm_name]
      +---w tpm_name                        string
      +---w tpm-physical-index?            int32 {ietfhw:entity-mib}?
+--ro output
+--ro tpm20-attestation-response* [tpm_name]
+--ro tpm_name                            string
+--ro tpm-physical-index?                  int32 {ietfhw:entity-mib}?
+--ro up-time?                            uint32
+--ro node-name?                          string
+--ro node-physical-index?                  int32 {ietfhw:entity-mib}?
+--ro tpms-attest
|   +--ro pcrdigest?                      binary
|   +--ro tpms-attest-result?              binary
|   +--ro tpms-attest-result-length?      uint32
+--ro tpmt-signature?                      binary
+---x basic-trust-establishment
+---w input
+---w nonce-value                          binary
+---w (signature-identifier-type)
|   +---: (TPM_ALG_ID)
|   |   +---w TPM_ALG_ID-value?            uint16
|   +---: (COSE_Algorithm)
|   |   +---w COSE_Algorithm-value?        int32
+---w tpm_name?                            string
+---w tpm-physical-index?                    int32 {ietfhw:entity-mib}?
+---w certificate-name?                      string
+--ro output
+--ro attestation-certificates* [tpm_name]
+--ro tpm_name                            string
+--ro tpm-physical-index?                    int32 {ietfhw:entity-mib}?
+--ro up-time?                            uint32
+--ro node-name?                          string

```

```

|         +---ro node-physical-index?          int32 {ietfhw:entity-mib}?
|         +---ro certificate-name?             string
|         +---ro attestation-certificate?      ietfct:end-entity-cert-cms
|         +---ro (key-identifier)?
|             +---:(public-key)
|                 | +---ro pub-key-id?         binary
|             +---:(uuid)
|                 +---ro uuid-value?           binary
+---x log-retrieval
+---w input
|   +---w log-selector* [node-name]
|   |   +---w node-name                       string
|   |   +---w node-physical-index?           int32 {ietfhw:entity-mib}?
|   |   +---w (index-type)?
|   |       +---:(last-entry)
|   |           | +---w last-entry-value?     binary
|   |       +---:(index)
|   |           | +---w index-number?         uint64
|   |       +---:(timestamp)
|   |           +---w timestamp?              yang:date-and-time
|   +---w log-type                           identityref
|   +---w pcr-list* []
|   |   +---w pcr
|   |       +---w pcr-indices*                 uint8
|   |       +---w (algo-registry-type)
|   |           +---:(tcg)
|   |               | +---w tcg-hash-algo-id?  uint16
|   |           +---:(ietf)
|   |               +---w ietf-ni-hash-algo-id? uint8
|   +---w log-entry-quantity?  uint16
+---ro output
+---ro system-event-logs
|   +---ro node-data* [node-name tpm_name]
|   |   +---ro node-name                     string
|   |   +---ro node-physical-index?          int32 {ietfhw:entity-mib}?
|   |   +---ro up-time?                      uint32
|   |   +---ro tpm_name                      string
|   |   +---ro tpm-physical-index?           int32 {ietfhw:entity-mib}?
|   |   +---ro log-result
|   |       +---ro (log-type)
|   |           +---:(bios)
|   |               +---ro bios-event-logs
|   |                   +---ro bios-event-entry* [event-number]
|   |                       +---ro event-number    uint32
|   |                       +---ro event-type?     uint32
|   |                       +---ro pcr-index?      uint16
|   |                       +---ro digest-list* []
|   |                           | +---ro (algo-registry-type)

```



```
| | | +--:(tcg)
| | | | +--ro tcg-hash-algo-id?          uint16
| | | +--:(ietf)
| | | | +--ro ietf-ni-hash-algo-id?      uint8
| | +--ro digest*                        binary
+--ro event-size?                       uint32
+--ro event-data*                       uint8
+--:(ima)
    +--ro ima-event-logs
        +--ro ima-event-entry* [event-number]
            +--ro event-number           uint64
            +--ro ima-template?         string
            +--ro filename-hint?       string
            +--ro filedata-hash?       binary
            +--ro template-hash-algorithm? string
            +--ro template-hash?       binary
            +--ro pcr-index?           uint16
            +--ro signature?           binary
```

<CODE ENDS>

2.2. Raw Format

<CODE BEGINS>

```

module ietf-basic-remote-attestation {
  namespace "urn:ietf:params:xml:ns:yang:ietf-basic-remote-attestation";
  prefix "yang-brat";

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-hardware {
    prefix ietfhw;
  }
  import ietf-crypto-types {
    prefix ietfct;
  }

  organization
    "Fraunhofer SIT";
  contact
    "Henk Birkholz
    Fraunhofer Institute for Secure Information Technology
    Email: henk.birkholz@sit.fraunhofer.de";
  description
    "A YANG module to enable TPM 1.2 and TPM 2.0 based
    remote attestation procedures.
    Copyright (C) Fraunhofer SIT (2019).";
  revision "2019-07-08" {

```

```
description
  "Second version";
reference
  "draft-birkholz-rats-basic-yang-module";
}

grouping hash-algo {
  description
    "A selector for the hashing algorithm";
  choice algo-registry-type {
    mandatory true;
    description
      "Unfortunately, both IETF and TCG have registries here.
      Choose your weapon wisely.";
    case tcg {
      description
        "you chose the east door, the tcg space opens up to
        you.";
      leaf tcg-hash-algo-id {
        type uint16;
        description
          "This is an index referencing the TCG Algorithm
          Registry based on TPM_ALG_ID.";
      }
    }
    case ietf {
      description
        "you chose the west door, the ietf space opens up to
        you.";
      leaf ietf-ni-hash-algo-id {
        type uint8;
        description
          "This is an index referencing the Named Information
          Hash Algorithm Registry.";
      }
    }
  }
}

grouping hash {
  description
    "The hash value including hash-algo identifier";
  list hash-digests {
    description
      "The list of hashes.";
    container hash-digest {
      description
        "A hash value based on a hash algorithm registered by an
```

```
        SDO.";
        uses hash-algo;
        leaf hash-value {
            type binary;
            description
                "The binary representation of the hash value.";
        }
    }
}

grouping nonce {
    description
        "A nonce to show freshness and counter replays.";
    leaf nonce-value {
        type binary;
        mandatory true;
        description
            "This nonce SHOULD be generated via a registered
            cryptographic-strength algorithm. In consequence, the length
            of the nonce depends on the hash algorithm used. The algorithm
            used in this case is independent from the hash algorithm used to
            create the hash-value in the response of the attester.";
    }
}

grouping tpm12-pcr-selection {
    description
        "A Verifier can request one or more PCR values using its
        individually created Attestation Key Certificate (AC).
        The corresponding selection filter is represented in this grouping.
        Requesting a PCR value that is not in scope of the AC used, detailed
        exposure via error msg should be avoided.";
    leaf-list pcr-indices {
        type uint8;
        description
            "The numbers/indexes of the PCRs. At the moment this is limited
            to 32.";
    }
}

grouping tpm20-pcr-selection {
    description
        "A Verifier can request one or more PCR values uses its
        individually created AC. The corresponding selection filter is
        represented in this grouping. Requesting a PCR value that is not
        in scope of the AC used, detailed exposure via error msg should
        be avoided.";
```

```
list pcr-list {
  description
    "For each PCR in this list an individual list of banks
    (hash-algo) can be requested. It depends on the datastore, if
    every bank in this grouping is included per PCR (crude), or if
    each requested bank set is returned for each PCR individually
    (elegant).";
  container pcr {
    description
      "The composite of a PCR number and corresponding bank
      numbers.";
    leaf-list pcr-indices {
      type uint8;
      description
        "The number of the PCR. At the moment this is limited
        32";
    }
    uses hash-algo;
  }
}

grouping pcr-selector {
  description
    "A Verifier can request the generation of an attestation
    certificate (a signed public attestation key
    (non-migratable, tpm-resident) wrt one or more PCR values.
    The corresponding creation input is represented in this grouping.
    Requesting a PCR value that is not supported results in an error,
    detailed exposure via error msg should be avoided.";
  list pcr-list {
    description
      "For each PCR in this list an individual hash-algo can be
      requested.";
    container pcr {
      description
        "The composite of a PCR number and corresponding bank
        numbers.";
      leaf-list pcr-index {
        type uint8;
        description
          "The numbers of the PCRs that are associated with
          the created key. At the moment the highest number is 32";
      }
      uses hash-algo;
    }
  }
}
```

```
grouping tpm12-signature-scheme {
  description
    "The signature scheme used to sign the evidence via a TPM 1.2.";
  leaf TPM_SIG_SCHEME-value {
    type uint8;
    mandatory true;
    description
      "Selects the signature scheme that is used to sign the TPM quote
      information response. Allowed values can be found in the table at
      the bottom of page 32 in the TPM 1.2 Structures specification
      (Level 2 Revision 116, 1 March 2011).";
  }
}

grouping tpm20-signature-scheme {
  description
    "The signature scheme used to sign the evidence.";
  choice signature-identifier-type {
    mandatory true;
    description
      "There are multiple ways to reference a signature type.
      This used to select the signature algo to sign the quote
      information response.";
    case TPM_ALG_ID {
      description
        "This references the indices of table 9 in the TPM 2.0
        structure specification.";
      leaf TPM_ALG_ID-value {
        type uint16;
        description
          "The TPM Algo ID.";
      }
    }
    case COSE_Algorithm {
      description
        "This references the IANA COSE Algorithms Registry indices.
        Every index of this registry to be used must be mapable to a
        TPM_ALG_ID value.";
      leaf COSE_Algorithm-value {
        type int32;
        description
          "The TPM Algo ID.";
      }
    }
  }
}

grouping tpm12-attestation-key-identifier {
```

```
description
  "A selector for a suitable key identifier for a TPM 1.2.";
choice key-identifier {
  description
    "Identifier for the attestation key to use for signing
    attestation evidence.";
  case public-key {
    leaf pub-key-id {
      type binary;
      description
        "The value of the identifier for the public key.";
    }
  }
  case TSS_UUID {
    description
      "Use a YANG agent generated (and maintained) attestation
      key UUID that complies with the TSS_UUID datatype of the TCG
      Software Stack (TSS) Specification, Version 1.10 Golden,
      August 20, 2003.";
    container TSS_UUID-value {
      description
        "A detailed structure that is used to create the
        TPM 1.2 native TSS_UUID as defined in the TCG Software
        Stack (TSS) Specification, Version 1.10 Golden,
        August 20, 2003.";
      leaf ulTimeLow {
        type uint32;
        description
          "The low field of the timestamp.";
      }
      leaf usTimeMid {
        type uint16;
        description
          "The middle field of the timestamp.";
      }
      leaf usTimeHigh {
        type uint16;
        description
          "The high field of the timestamp multiplexed with the
          version number.";
      }
      leaf bClockSeqHigh {
        type uint8;
        description
          "The high field of the clock sequence multiplexed with
          the variant.";
      }
      leaf bClockSeqLow {
```

```
        type uint8;
        description
            "The low field of the clock sequence.";
    }
    leaf-list rgbNode {
        type uint8;
        description
            "The spatially unique node identifier.";
    }
}
}
}
}

grouping tpm20-attestation-key-identifier {
    description
        "A selector for a suitable key identifier.";
    choice key-identifier {
        description
            "Identifier for the attestation key to use for signing
            attestation evidence.";
        case public-key {
            leaf pub-key-id {
                type binary;
                description
                    "The value of the identifier for the public key.";
            }
        }
        case uuid {
            description
                "Use a YANG agent generated (and maintained) attestation
                key UUID.";
            leaf uuid-value {
                type binary;
                description
                    "The UUID identifying the corresponding public key.";
            }
        }
    }
}

grouping tpm-name {
    description
        "In a system with multiple-TPMs get the data from a specific TPM
        identified by the name and physical-index.";
    leaf tpm_name {
        type string;
        description
```

```
        "Name of the TPM or All";
    }
    leaf tpm-physical-index {
        if-feature ietfhw:entity-mib;
        type int32 {
            range "1..2147483647";
        }
        config false;
        description
            "The entPhysicalIndex for the TPM.";
        reference
            "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
    }
}

grouping compute-node {
    description
        "In a distributed system with multiple compute nodes
        this is the node identified by name and physical-index.";
    leaf node-name {
        type string;
        description
            "Name of the compute node or All";
    }
    leaf node-physical-index {
        if-feature ietfhw:entity-mib;
        type int32 {
            range "1..2147483647";
        }
        config false;
        description
            "The entPhysicalIndex for the compute node.";
        reference
            "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
    }
}

grouping tpml2-pcr-info-short {
    description
        "This structure is for defining a digest at release when the only
        information that is necessary is the release configuration.";
    uses tpml2-pcr-selection;
    leaf locality-at-release {
        type uint8;
        description
            ".This SHALL be the locality modifier required to release the
            information (TPM 1.2 type TPM_LOCALITY_SELECTION)";
    }
    leaf digest-at-release {
```



```
    type binary;
    description
        "This SHALL be the digest of the PCR indices and PCR values
        to verify when revealing auth data (TPM 1.2 type
        TPM_COMPOSITE_HASH).";
}
}

grouping tpml2-version {
    description
        "This structure provides information relative the version of
        the TPM.";
    list version {
        description
            "This indicates the version of the structure
            (TPM 1.2 type TPM_STRUCT_VER). This MUST be 1.1.0.0.";
        leaf major {
            type uint8;
            description
                "Indicates the major version of the structure.
                MUST be 0x01.";
        }
        leaf minor {
            type uint8;
            description
                "Indicates the minor version of the structure.
                MUST be 0x01.";
        }
        leaf revMajor {
            type uint8;
            description
                "Indicates the rev major version of the structure.
                MUST be 0x00.";
        }
        leaf revMinor {
            type uint8;
            description
                "Indicates the rev minor version of the structure.
                MUST be 0x00.";
        }
    }
}

grouping tpml2-quote-info-common {
    description
        "These statements are used in bot quote variants of the TPM 1.2";
    leaf fixed {
        type binary;
    }
}
```

```
        description
            "This SHALL always be the string 'QUOT' or 'QUO2'
            (length is 4 bytes).";
    }
    leaf external-data {
        type binary;
        description
            "160 bits of externally supplied data, typically a nonce.";
    }
    leaf signature-size {
        type uint32;
        description
            "The size of TPM 1.2 'signature' value.";
    }
    leaf signature {
        type binary;
        description
            "Signature over SHA-1 hash of tpml2-quote-info2'.";
    }
}

grouping tpml2-quote-info {
    description
        "This structure provides the mechanism for the TPM to quote the
        current values of a list of PCRs (as used by the TPM_Quote2
        command).";
    uses tpml2-version;
    leaf digest-value {
        type binary;
        description
            "This SHALL be the result of the composite hash algorithm using
            the current values of the requested PCR indices
            (TPM 1.2 type TPM_COMPOSITE_HASH.)";
    }
}

grouping tpml2-quote-info2 {
    description
        "This structure provides the mechanism for the TPM to quote the
        current values of a list of PCRs
        (as used by the TPM_Quote2 command).";
    leaf tag {
        type uint8;
        description
            "This SHALL be TPM_TAG_QUOTE_INFO2.";
    }
    uses tpml2-pcr-info-short;
}
```

```
grouping tpm12-cap-version-info {
  description
    "TPM returns the current version and revision of the TPM 1.2 .";
  list TPM_PCR_COMPOSITE {
    description
      "The TPM 1.2 TPM_PCRVALUES for the pcr-indices.";
    uses tpm12-pcr-selection;
    leaf value-size {
      type uint32;
      description
        "This SHALL be the size of the 'tpm12-pcr-value' field
        (not the number of PCRs).";
    }
    leaf-list tpm12-pcr-value {
      type binary;
      description
        "The list of TPM_PCRVALUES from each PCR selected in sequence
        of tpm12-pcr-selection.";
    }
  }
  list version-info {
    description
      "An optional output parameter from a TPM 1.2 TPM_Quote2.";
    leaf tag {
      type uint16;
      description
        "The TPM 1.2 version and revision
        (TPM 1.2 type TPM_STRUCTURE_TAG).
        This MUST be TPM_CAP_VERSION_INFO (0x0030)";
    }
    uses tpm12-version;
    leaf spec-level {
      type uint16;
      description
        "A number indicating the level of ordinals supported.";
    }
    leaf errata-rev {
      type uint8;
      description
        "A number indicating the errata version of the
        specification.";
    }
    leaf tpm-vendor-id {
      type binary;
      description
        "The vendor ID unique to each TPM manufacturer.";
    }
    leaf vendor-specific-size {
      type uint16;
    }
  }
}
```

```
        description
            "The size of the vendor-specific area.";
    }
    leaf vendor-specific {
        type binary;
        description
            "Vendor specific information.";
    }
}

}

}

grouping tpm12-pcr-composite {
    description
        "The actual values of the selected PCRs (a list of TPM_PCRVALUES
        (binary) and associated metadata for TPM 1.2.";
    list TPM_PCR_COMPOSITE {
        description
            "The TPM 1.2 TPM_PCRVALUES for the pcr-indices.";
        uses tpm12-pcr-selection;
        leaf value-size {
            type uint32;
            description
                "This SHALL be the size of the 'tpm12-pcr-value' field
                (not the number of PCRs).";
        }
        leaf-list tpm12-pcr-value {
            type binary;
            description
                "The list of TPM_PCRVALUES from each PCR selected in sequence
                of tpm12-pcr-selection.";
        }
    }
}

grouping node-uptime {
    description
        "Uptime in seconds of the node.";
    leaf up-time {
        type uint32;
        description
            "Uptime in seconds of this node reporting its data";
    }
}

identity log-type {
    description
        "The type of logs available.";
```

```
}

identity bios {
  base log-type;
  description
    "Measurement log created by the BIOS/UEFI.";
}

identity ima {
  base log-type;
  description
    "Measurement log created by IMA.";
}

grouping log-identifier {
  description
    "Identifier for type of log to be retrieved.";
  leaf log-type {
    type identityref {
      base log-type;
    }
    mandatory true;
    description
      "The corresponding measurement log type identity.";
  }
}

grouping boot-event-log {
  description
    "Defines an event log corresponding to the event that extended the
    PCR";
  leaf event-number {
    type uint32;
    description
      "Unique event number of this event";
  }
  leaf event-type {
    type uint32;
    description
      "log event type";
  }
  leaf pcr-index {
    type uint16;
    description
      "Defines the PCR index that this event extended";
  }
  list digest-list {
    description "Hash of event data";
  }
}
```

```
    uses hash-algo;
    leaf-list digest {
        type binary;
        description
            "The hash of the event data";
    }
}
leaf event-size {
    type uint32;
    description
        "Size of the event data";
}
leaf-list event-data {
    type uint8;
    description
        "the event data size determined by event-size";
}
}

grouping ima-event {
    description
        "Defines an hash log extend event for IMA measurements";
    leaf event-number {
        type uint64;
        description
            "Unique number for this event for sequencing";
    }
    leaf ima-template {
        type string;
        description
            "Name of the template used for event logs
            for e.g. ima, ima-ng";
    }
    leaf filename-hint {
        type string;
        description
            "File that was measured";
    }
    leaf filedata-hash {
        type binary;
        description
            "Hash of filedata";
    }
    leaf template-hash-algorithm {
        type string;
        description
            "Algorithm used for template-hash";
    }
}
```

```
leaf template-hash {
    type binary;
    description
        "hash(filedata-hash, filename-hint)";
}
leaf pcr-index {
    type uint16;
    description
        "Defines the PCR index that this event extended";
}
leaf signature {
    type binary;
    description
        "The file signature";
}
}

grouping bios-event-log {
    description
        "Measurement log created by the BIOS/UEFI.";
    list bios-event-entry {
        key event-number;
        description
            "Ordered list of TCG described event log
             that extended the PCRs in the order they
             were logged";
        uses boot-event-log;
    }
}

grouping ima-event-log {
    list ima-event-entry {
        key event-number;
        description
            "Ordered list of ima event logs by event-number";
        uses ima-event;
    }
    description
        "Measurement log created by IMA.";
}

grouping event-logs {
    description
        "A selector for the log and its type.";
    choice log-type {
        mandatory true;
        description
            "Event log type determines the event logs content.";
    }
}
```

```
    case bios {
      description
        "BIOS/UEFI event logs";
      container bios-event-logs {
        description
          "This is an index referencing the TCG Algorithm
          Registry based on TPM_ALG_ID.";
        uses bios-event-log;
      }
    }
  case ima {
    description
      "IMA event logs";
    container ima-event-logs {
      description
        "This is an index referencing the TCG Algorithm
        Registry based on TPM_ALG_ID.";
      uses ima-event-log;
    }
  }
}

rpc tpm12-challenge-response-attestation {
  description
    "This RPC accepts the input for TSS TPM 1.2 commands of the
    managed device. ComponentIndex from the hardware manager YANG
    module to refer to dedicated TPM in composite devices,
    e.g. smart NICs, is still a TODO.";
  input {
    container tpm1-attestation-challenge {
      description
        "This container includes every information element defined
        in the reference challenge-response interaction model for
        remote attestation. Corresponding values are based on
        TPM 1.2 structure definitions";
      uses tpm12-pcr-selection;
      uses nonce;
      uses tpm12-signature-scheme;
      uses tpm12-attestation-key-identifier;
      leaf add-version {
        type boolean;
        description
          "Whether or not to include TPM_CAP_VERSION_INFO; if true,
          then TPM_Quote2 must be used to create the response.";
      }
      uses tpm-name;
    }
  }
}
```



```
    }
    output {
      list tpm12-attestation-response {
        key tpm_name;
        description
          "The binary output of TPM 1.2 TPM_Quote/TPM_Quote2, including
            the PCR selection and other associated attestation evidence
            metadata";
        uses tpm-name;
        uses node-uptime;
        uses compute-node;
        uses tpm12-quote-info-common;
        choice tpm12-quote {
          mandatory true;
          description
            "Either a tpm12-quote-info or tpm12-quote-info2, depending
              on whether TPM_Quote or TPM_Quote2 was used
              (cf. input field add-version).";
          case tpm12-quotel {
            description
              "BIOS/UEFI event logs";
            uses tpm12-quote-info;
            uses tpm12-pcr-composite;
          }
          case tpm12-quote2 {
            description
              "BIOS/UEFI event logs";
            uses tpm12-quote-info2;
          }
        }
      }
    }
  }
}

rpc tpm20-challenge-response-attestation {
  description
    "This RPC accepts the input for TSS TPM 2.0 commands of the
      managed device. ComponentIndex from the hardware manager YANG
      module to refer to dedicated TPM in composite devices,
      e.g. smart NICs, is still a TODO.";
  input {
    container tpm20-attestation-challenge {
      description
        "This container includes every information element defined
          in the reference challenge-response interaction model for
          remote attestation. Corresponding values are based on
          TPM 2.0 structure definitions";
      uses tpm20-pcr-selection;
    }
  }
}
```

```
    uses nonce;
    uses tpm20-signature-scheme;
    uses tpm20-attestation-key-identifier;
  }
  list tpms {
    key tpm_name;
    description
      "TPMs to fetch the attestation information.";
    uses tpm-name;
  }
}
output {
  list tpm20-attestation-response {
    key tpm_name;
    description
      "The binary output of TPM2b_Quote. An TPMS_ATTEST structure
      including a length, encapsulated in a signature";
    uses tpm-name;
    uses node-uptime;
    uses compute-node;
    container tpms-attest {
      leaf pcrdigest {
        type binary;
        description
          "split out value of TPMS_QUOTE_INFO for convenience";
      }
      leaf tpms-attest-result {
        type binary;
        description
          "The complete TPM generate structure including
          signature.";
      }
      leaf tpms-attest-result-length {
        type uint32;
        description
          "Length of attest result provided by the TPM structure.";
      }
    }
    description
      "A composite of value and length and list of selected
      pcrcs (original name: [type]attested)";
  }
  leaf tpmt-signature {
    type binary;
    description
      "Split out value of the signature for convenience.
      TODO: check for length values that complement binary value
      data node leafs.";
  }
}
```

```
    }
  }
}

rpc basic-trust-establishment {
  description
    "This RPC creates a tpm-resident, non-migratable key to be used
    in TPM_Quote commands, an attestation certificate.";
  input {
    uses nonce;
    uses tpm20-signature-scheme;
    uses tpm-name;
    leaf certificate-name {
      type string;
      description
        "An arbitrary name for the identity certificate chain
        requested.";
    }
  }
  output {
    list attestation-certificates {
      key tpm_name;
      description
        "Attestation Certificate data from a TPM identified by the TPM
        name";
      uses tpm-name;
      uses node-uptime;
      uses compute-node;
      leaf certificate-name {
        type string;
        description
          "An arbitrary name for this identity certificate or
          certificate chain.";
      }
      leaf attestation-certificate {
        type ietfct:end-entity-cert-cms;
        description
          "The binary signed certificate chain data for this identity
          certificate.";
      }
      uses tpm20-attestation-key-identifier;
    }
  }
}

rpc log-retrieval {
  description
    "Logs Entries are either identified via indices or via providing
```

```
the last line received. The number of lines returned can be
limited. The type of log is a choice that can be augmented.";
input {
  list log-selector {
    key node-name;
    description
      "Selection of log entries to be reported.";
    uses compute-node;
    choice index-type {
      description
        "Last log entry received, log index number, or timestamp.";
      case last-entry {
        description
          "The last entry of the log already retrieved.";
        leaf last-entry-value {
          type binary;
          description
            "Content of an log event which matches 1:1 with a
            unique event record contained within the log. Log
            entries subsequent to this will be passed to the
            requester. Note: if log entry values are not unique,
            this MUST return an error.";
        }
      }
      case index {
        description
          "Numeric index of the last log entry retrieved, or zero.";
        leaf index-number {
          type uint64;
          description
            "The numeric index number of a log entry. Zero means
            to start at the beginning of the log. Entries
            subsequent to this will be passed to the
            requester.";
        }
      }
    }
  }
  case timestamp {
    leaf timestamp {
      type yang:date-and-time;
      description
        "Timestamp from which to start the extraction. The next
        log entry subsequent to this timestamp is to be sent.";
    }
    description
      "Timestamp from which to start the extraction.";
  }
}
```

```

    uses log-identifier;
    uses tpm20-pcr-selection;
    leaf log-entry-quantity {
        type uint16;
        description
            "The number of log entries to be returned. If omitted, it
            means all of them.";
    }
}
output {
    container system-event-logs {
        description
            "The requested data of the measurement event logs";
        list node-data {
            key "node-name tpm_name";
            description
                "Event logs of a node in a distributed system
                identified by the node name";
            uses compute-node;
            uses node-uptime;
            uses tpm-name;
            container log-result {
                description
                    "The requested entries of the corresponding log.";
                uses event-logs;
            }
        }
    }
}

container rats-support-structures {
    config false;
    description
        "The datastore definition enabling verifiers or relying
        parties to discover the information necessary to use the
        remote attestation RPCs appropriately.";
    leaf-list supported-algos {
        type uint16;
        description
            "Supported TPM_ALG_ID values for the TPM in question.
            Will include ComponentIndex soon.";
    }
    list tpms {
        key tpm_name;
        uses tpm-name;
        description
            "A list of TPMs in this composite

```

```
device that rats can be conducted with.";
list certificates {
  description
    "The TPM's endorsement-certificate.";
  container certificate {
    leaf certificate-name {
      type string;
      description
        "An arbitrary name for this identity certificate or
        certificate chain.";
    }
    leaf certificate-type {
      type enumeration {
        enum endorsement-cert {
          value 0;
          description
            "EK Cert type.";
        }
        enum attestation-cert {
          value 1;
          description
            "AK Cert type.";
        }
      }
      description "Type of this certificate";
    }
    leaf certificate-value {
      type ietfct:end-entity-cert-cms;
      description
        "The binary signed public endorsement key (EK),
        attestation key (AK) and corresponding assertions
        (EK, AK Certificate). In a TPM 2.0 the EK, AK Certificate
        resides in a well-defined NVRAM location by the TPM
        vendor.";
    }
    description
      "Two kinds of certificates can be accessed via this
      statement. An Attestation Key Certificate and a
      Endorsement Key Certificate.";
  }
}
list compute-nodes {
  key node-name;
  uses compute-node;
  description
    "A list names of hardware components in this composite
    device that rats can be conducted with.";
```

```
    }  
  }  
}  
<CODE ENDS>
```

3. IANA considerations

This document will include requests to IANA:

To be defined yet.

4. Security Considerations

There are always some.

5. Acknowledgements

Not yet.

6. Change Log

Changes from version 00 to version 01:

- o Addressed author's comments
- o Extended complementary details about attestation-certificates
- o Relabeled chunk-size to log-entry-quantity
- o Relabeled location with compute-node or tpm-name where appropriate
- o Added a valid entity-mib physical-index to compute-node and tpm-name to map it back to hardware inventory
- o Relabeled name to tpm_name
- o Removed event-string in last-entry

7. References

7.1. Normative References

[I-D.birkholz-rats-reference-interaction-model]
Birkholz, H. and M. Eckel, "Reference Interaction Model for Challenge-Response-based Remote Attestation", draft-birkholz-rats-reference-interaction-model-00 (work in progress), March 2019.

- [I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for
Cryptography", draft-ietf-netconf-crypto-types-10 (work in
progress), July 2019.
- [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>>.

7.2. Informative References

- [I-D.birkholz-attestation-terminology]
Birkholz, H., Wiseman, M., and H. Tschofenig, "Reference
Terminology for Remote Attestation Procedures", draft-
birkholz-attestation-terminology-02 (work in progress),
July 2018.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Michael Eckel
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: michael.eckel@sit.fraunhofer.de

Shwetha Bhandari
Cisco Systems

Email: shwethab@cisco.com

Bill Sulzen
Cisco Systems

Email: bsulzen@cisco.com

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Liang Xia (Frank)
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.Xialiang@huawei.com

Tom Laffey
Hewlett Packard Enterprise

Email: tom.laffey@hpe.com

Guy C. Fedorkow
Juniper Networks
10 Technology Park Drive
Westford, Massachusetts 01886

Email: gfedorkow@juniper.net

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 12, 2020

H. Birkholz
M. Eckel
Fraunhofer SIT
January 09, 2020

An Information Model for Claims used in RATS
draft-birkholz-rats-information-model-01

Abstract

This document defines a standardized information model (IM) for Claims that can be used in remote attestation procedures (RATS). The information elements defined include attestation Claims which provide information about system components characteristics, as well as commonly used attributes and attribute structures that are required by protocols facilitating remote attestation.

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 July 12, 2020.

Copyright Notice

Copyright (c) 2020 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	2
1.1. Document Structure	3
2. RATS Information Elements	4
3. Security Considerations	8
4. Acknowledgments	8
5. Change Log	8
6. Contributors	8
7. References	8
7.1. Normative References	8
7.2. Informative References	9
Authors' Addresses	9

1. Introduction

Remote attestation procedures (RATS) are used to increase the trust in the trustworthiness of an Attester. This is typically accomplished by conveying Attestation Evidence from an Attester to a Verifier that is able to appraise the Evidence. The exact definitions of RATS roles, such as an Attester or a Verifier, are specified in the RATS architecture [I-D.ietf-rats-architecture]. This document defines the common information elements (IE) that are able to express the characteristics of an Attester. Ultimately, these IE can be used to compose Attestation Evidence (attestation Claims that are accompanied by a proof of their validity).

In general, RATS convey information elements that:

- o enable the functionality of remote attestation protocols,
- o are able to express Claims about an Attester's composition, configuration, or operational state,
- o represent the provenance of Claims, including entities that provide assertions on behalf of the Attester,
- o compose a type of proof of validity with respect to other Claims, and that
- o are either verifiable (via comparison with trusted reference values) or non-verifiable.

1.1. Document Structure

Every information element listed is annotated with one or more of these attributes:

Protocol (P): This IE is used on a remote attestation protocol layer, typically on the control plane or as protocol-specific data plane content.

Hardware (H): This IE expresses characteristics about an Attester's hardware components or the composition of its hardware components.

Software (S): This IE expresses characteristics about an Attester's software components or their semantic relationship. The term software component - in the scope of this document - subsumes firmware, bootloader, BIOS/(U)EFI, and microcode.

Operational State (O): This IE is used to convey information about the combination of applied configuration and system state as defined in [RFC8342].

Verifiable (V): This IE requires reference integrity measurements (RIM), compliance-policy, certification-path, or another type of trust-chain in order to be appraised appropriately by a Verifier.

Additionally, every IE definition includes a reference to the source of its definition, if it is not specified in this document for the first time (which is the most likely case). If a source of a definition is not a specification or (proposed) standard, but a draft, a web resource, or source that cannot be attribute with a DOI or ISSN, the following attribute is associated.

Unstable (U): The source of the definition of this IE may change in the future and is not considered to be stable at the time of publication of this document.

Information elements might reference other information elements or have to be associated in a set (with or without a specific order) in order to convey the intended meaning to a Verifier. Reference to other IE inside this documents simply use their name as reference. In consequence, an IE can be a superstructure composed of other IE with its own name (and potentially additional definition text that defines its purpose and or usage).

The RATS Information Model allows for expressing a hierarchical taxonomy. If an IE is a specialisation of another IE, the last sentence in the definition includes a "This IE is a specialization of _IE NAME_".

The ordering of IE is in descending alphabetical order; independent of source or semantic relationship to other IE, or other types of hierarchy.

2. RATS Information Elements

Age: The latency between the creation of a Claim value (e.g. by asserters such as hardware sensors or the Linux Integrity Measurement Architecture) including its composition into Attestation Evidence and its following conveyance to another RATS Actor/Role in RATS. The Age IE does not require a threshold at which point another information element is considered "old" and an age information element has to be included.

Reference: [I-D.ietf-rats-eat]

Claim Selection: [P]

A filter expression that enables the conveyance of a subset of all attestation Claims available to the Attester, if requested by a Verifier.

Attestation Evidence: [H, S, O, V]

A composite IE that must include at least an Authentication-Secret Identifier, an Attester Identity, and at least one Attestation Claim. Attestation Evidence is always signed via the Authentication Secret and thereby binds the listed information elements cryptographically. Attestation Evidence can only be trusted by a Verifier if it is associated with a trust anchor the Verifier also trusts.

Attester Identifier: [P, O, V]

A value associated or bound to a distinguishable Attester that is intended to uniquely identify it, but is not directly associated with a trust anchor. Additional Endorsement Documents can increase the level of confidence in an Attester Identifier.

Attester Identity: [P, S, V]

A document about a distinguishable Attester issued and signed by a third party. If not cryptographically associated with a trust anchor directly or indirectly, this IE is a specialization of Attester Identifier.

Attestation Result: [P]

A set of one or more values that are created by an appraisal action of a Verifier. Attestation Result is the most generic definition of the output of RATS and are typically consumed by relying parties.

Authentication-Secret Identifier: [O, V]

An identifier that is associated with an authentication secret used to sign Attestation Evidence.

Authorization Challenge: [P]

The input to an challenge-response protocol hand-shake. This IE can be Nonce, but also the output of a local attestation procedure.

Reference [I-D.tschofenig-rats-psa-token]

Endorsement Document: [P, H, S, V]

A document about the capabilities and functionality of one or more sub-components of a distinguishable Attester issued and signed by a third party. Endorsement Documents are intended to render Attestation Evidence trustworthy. If not cryptographically associated with a trust anchor directly or indirectly, this IE is a specialization of System Component Identifier.

Location: A global standardized set of coordinates and related attributes representing the geographic position of a device based on a geodetic system, such as Navstar GPS. The coordinate values can have different meaning with respect to the geographic position of a device depending on the geodetic system used. The default is WGS-84.

The basic location attributes include: latitude, longitude, altitude, accuracy, altitude accuracy, heading, and velocity.

Reference [I-D.ietf-rats-eat]

Measured Boot Characteristics: [H, S, V]

If every piece of software is measured by a root-of-trust for measuring during boot time and across staged execution environments (e.g. UEFI, Bootloader, Kernel, Rich OS), associated information about how and in which operational states these measurements are conducted is vital to RATS. This IE represents several states of a (composite) device with respect to measured boot (previously often called secure boot) including: "Secure Boot

Enabled", "Debug Disabled", "Debug Disabled Since Boot", "Debug Permanent Disable", "Debug Full Permanent Disable".

Nonce: [P]

An information element with two major uses: the prevention of replay-attacks and as an IE that can be used in a challenge-response interaction model. It is created by the requester to provide Evidence about the freshness of the corresponding response. It is important to highlight that a nonce by itself does not protect from relay-attacks.

OEM Identifier: [H, S, V]

A organizationally unique identifier (OUI) assigned by the IEEE Registration Authority (IEEE RA). This IE is associated with a device or a distinguishable sub-component of a composite device with its own execution environment. It intended to identify a device(component) during its life-cycle. This is a specialization of System Component Identifier.

Reference [I-D.ietf-rats-eat]

Origination: [P, S, V]

An IE representing attestation provenance. Attestation Claims or Attestation Evidence are produced by a specific source of information that is intended to be uniquely identifiable. The source of information is a distinguishable type of execution environment (see [I-D.ietf-rats-architecture]) of a device or the sub-components of a composite device.

Reference [I-D.ietf-rats-eat]

Universal Entity ID: [P, H, V]

A unique identifier permanently associated with an individual manufactured entity / device, such as a mobile phone, a water meter, a Bluetooth speaker or a networked security camera. This IE is intended to either identify an device or a submodule or subsystem of a device. It does not identify types, models or classes of devices. It is akin to a serial number, though it does not have to be sequential. This IE is a specialization of System Component Identifier.

Reference [I-D.ietf-rats-eat]

Uptime: [H, S]

An IE representing the number of seconds since the first execution environment of a (composite) device is able to measure it.

Reference [I-D.ietf-rats-eat]

Security Level: [H, S, V]

A level of confidence with respect to the resilience against attacks intended to compromise Attestation Evidence. A Security Level can be associated with an Origination. This IE is context specific and requires a scope-specific definition of values as part of a security framework. The [I-D.ietf-rats-eat] document, for example, provides an enumeration of security levels that is similar to the Metadata Service defined by the Fast Identity Online (FIDO) Alliance.

Reference [I-D.ietf-rats-eat]

Software Component Identifier: [S, V]

An IE representing one or more distinguishable Software Components [I-D.ietf-sacm-terminology] that were loaded and measured by an appropriate root-of-trust. The use of this IE typically requires the use of Measured Boot.

Reference [I-D.tschofenig-rats-psa-token]

System Component Identifier: [H, S, V]

An identifier intended to uniquely identify a distinguishable system component. System components can be hardware components or software components (e.g. a virtual machine). The system component can be an "atomic" device (i.e. a composite device with only one hardware component) or a part of a composite device.

Timestamp: [P, S]

A generic information element that represents a certain point of time in the past. The level of confidence in the value of a timestamp is based on the trustworthiness of the source of time, which can be local or remote, a composite of multiple time sources to represent the state synchronization, as well as the precision and the accuracy of the source of time itself.

Timestamps can be time-zone specific and therefore change their meaning if the definition of time zones changes.

Verification Service Indicator: [P, S, V]

This IE provides a hint (typically consumed by a Relying Party) that enables the discovery of an appropriate Verification Service or Remote Attestation Service, e.g. a URL.

Reference [I-D.tschofenig-rats-psa-token]

3. Security Considerations

Probably none

4. Acknowledgments

TBD

5. Change Log

Initial version -00

Refresh to version -01 for visibility

6. Contributors

TBD

7. References

7.1. Normative References

[I-D.birkholz-rats-tuda]

Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", draft-birkholz-rats-tuda-01 (work in progress), September 2019.

[I-D.ietf-rats-architecture]

Birkholz, H., Thaler, D., Richardson, M., and N. Smith, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-00 (work in progress), December 2019.

[I-D.ietf-rats-eat]

Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", draft-ietf-rats-eat-01 (work in progress), July 2019.

- [I-D.ietf-rats-yang-tpm-charra]
Birkholz, H., Eckel, M., Bhandari, S., Sulzen, B., Voit, E., Xia, L., Laffey, T., and G. Fedorkow, "A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs", draft-ietf-rats-yang-tpm-charra-00 (work in progress), January 2020.
- [I-D.tschofenig-rats-psa-token]
Tschofenig, H., Frost, S., Brossard, M., Shaw, A., and T. Fossati, "Arm's Platform Security Architecture (PSA) Attestation Token", draft-tschofenig-rats-psa-token-04 (work in progress), November 2019.
- [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>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.

7.2. Informative References

- [I-D.birkholz-rats-reference-interaction-model]
Birkholz, H. and M. Eckel, "Reference Interaction Models for Remote Attestation Procedures", draft-birkholz-rats-reference-interaction-model-02 (work in progress), January 2020.
- [I-D.ietf-sacm-terminology]
Birkholz, H., Lu, J., Strassner, J., Cam-Winget, N., and A. Montville, "Security Automation and Continuous Monitoring (SACM) Terminology", draft-ietf-sacm-terminology-16 (work in progress), December 2018.
- [I-D.richardson-rats-usecases]
Richardson, M., Wallace, C., and W. Pan, "Use cases for Remote Attestation common encodings", draft-richardson-rats-usecases-06 (work in progress), November 2019.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Michael Eckel
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: michael.eckel@sit.fraunhofer.de

RATS Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2021

H. Birkholz
M. Eckel
Fraunhofer SIT
C. Newton
L. Chen
University of Surrey
July 08, 2020

Reference Interaction Models for Remote Attestation Procedures
draft-birkholz-rats-reference-interaction-model-03

Abstract

This document describes interaction models for remote attestation procedures (RATS). Three conveying mechanisms - Challenge/Response, Uni-Directional, and Streaming Remote Attestation - are illustrated and defined. Analogously, a general overview about the information elements typically used by corresponding conveyance protocols are highlighted. Privacy preserving conveyance of Evidence via Direct Anonymous Attestation is elaborated on for each interaction model, individually.

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 January 9, 2021.

Copyright Notice

Copyright (c) 2020 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
3. Disambiguation	4
4. Scope and Intent	4
5. Direct Anonymous Attestation	5
5.1. Endorsers	5
5.2. Endorsers for Direct Anonymous Attestation	6
6. Normative Prerequisites	6
7. Generic Information Elements	7
8. Interaction Models	9
8.1. Challenge/Response Remote Attestation	10
8.2. Uni-Directional Remote Attestation	11
8.3. Streaming Remote Attestation	13
9. Additional Application-Specific Requirements	15
9.1. Confidentiality	15
9.2. Mutual Authentication	15
9.3. Hardware-Enforcement/Support	15
10. Implementation Status	15
10.1. Implementer	16
10.2. Implementation Name	16
10.3. Implementation URL	16
10.4. Maturity	16
10.5. Coverage and Version Compatibility	16
10.6. License	16
10.7. Implementation Dependencies	16
10.8. Contact	17
11. Security and Privacy Considerations	17
12. Acknowledgments	17
13. Change Log	17
14. References	19
14.1. Normative References	19
14.2. Informative References	20
Appendix A. CDDL Specification for a simple CoAP Challenge/Response Interaction	20
Authors' Addresses	21

1. Introduction

Remote Attestation procedures (RATS, [I-D.ietf-rats-architecture]) are workflows composed of roles and interactions, in which Verifiers create Attestation Results about the trustworthiness of an Attester's system component characteristics. The Verifier's assessment in the form of Attestation Results is created based on Attestation Policies and Evidence - trustable and tamper-evident Claims Sets about an Attester's system component characteristics - created by an Attester. The roles `_Attester_` and `_Verifier_`, as well as the Conceptual Messages `_Evidence_` and `_Attestation Results_` are terms defined by the RATS Architecture [I-D.ietf-rats-architecture]. This documents captures interaction models that can be used in specific RATS-related solution documents. The primary focus of this document is the conveyance of attestation Evidence. Specific goals of this document are to:

- o prevent inconsistencies in descriptions of these interaction models in other documents (due to text cloning over time),
- o enable to highlight an exact delta/divergence between the core set of characteristics captured here in this document and variants of these interaction models used in other specifications or solutions, and to
- o illustrate the application of Direct Anonymous Attestation (DAA) for each of the interaction models described.

In summary, this document enables the specification and design of trustworthy and privacy preserving conveyance methods for attestation Evidence from an Attester to a Verifier. While the conveyance of other Conceptual Messages is out-of-scope the methods described can also be applied to the conveyance of Endorsements or Attestation Results.

2. Terminology

This document uses the terms, roles, and concepts defined in [I-D.ietf-rats-architecture]:

Attester, Verifier, Relying Party, Conceptual Message, Evidence, Endorsement, Attestation Result, Appraisal Policy, Attesting Environment, Target Environment

A PKIX Certificate is an X.509v3 format certificate as specified by [RFC5280].

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.

3. Disambiguation

The term "Remote Attestation" is a common expression and often associated or connoted with certain properties. The term "Remote" in this context does not necessarily refer to a remote entity in the scope of network topologies or the Internet. It rather refers to a decoupled system or entities that exchange the payload of the Conceptual Message type called Evidence [I-D.ietf-rats-architecture]. This conveyance can also be "local", if the Verifier is part of the same entity as the Attester, e.g., separate system components of a Composite Device (a single RATS Entity). Examples of these types of co-located environments include: a Trusted Execution Environment (TEE), Baseboard Management Controllers (BMCs), as well as other physical or logical protected/isolated/shielded Computing Environments (e.g. embedded Secure Elements (eSE) or Trusted Platform Modules (TPM)).

4. Scope and Intent

This document focuses on generic interaction models between Attesters and Verifiers in order to convey Evidence. Complementary procedures, functions, or services that are required for a complete semantic binding of the concepts defined in [I-D.ietf-rats-architecture] are out-of-scope of this document. Examples include: identity establishment, key distribution and enrollment, time synchronization, as well as certificate revocation.

Furthermore, any processes and duties that go beyond carrying out remote attestation procedures are out-of-scope. For instance, using the results of a remote attestation that are created by the Verifier, e.g., how to triggering remediation actions or recovery processes, as well as such remediation actions and recovery processes themselves, are also out-of-scope.

The interaction models illustrated in this document are intended to provide a stable basis and reference for other solutions documents inside or outside the IETF. Solution documents of any kind can reference the interaction models in order to avoid text clones and to avoid the danger of subtle discrepancies. Analogously, deviations from the generic model descriptions in this document can be illustrated in solutions documents to highlight distinct contributions.

5. Direct Anonymous Attestation

DAA [DAA] is a signature scheme used in RATS that allows preservation of the privacy of users that are associated with an Attester (e.g. its owner). Essentially, DAA can be seen as a group signature scheme with the feature that given a DAA signature no-one can find out who the signer is, i.e., the anonymity is not revocable. To be able to sign anonymously an Attester has to obtain a credential from a DAA Issuer. The DAA Issuer uses a private/public key pair to generate a credential for an Attester and makes the public key (in the form of a public key certificate) available to the verifier to enable them to validate the DAA signature obtained as part of the Evidence.

In order to support these DAA signatures, the DAA Issuer MUST associate a single key pair with each group of Attesters and use the same key pair when creating the credentials for all of the Attesters in this group. The DAA Issuer's public key certificate for the group replaces the Attester Identity documents in the verification of the Evidence (instead of unique Attester Identity documents). This is in contrast to intuition that there has to be a unique Attester Identity per device.

This document extends the duties of the Endorser role as defined by the RATS architecture with respect to the provision of these Attester Identity documents to Attesters. The existing duties of the Endorser role and the duties of a DAA Issuer are quite similar as illustrated in the following subsections.

5.1. Endorsers

Via its Attesting Environments, an Attester can only create Evidence about its Target Environments. After being appraised to be trustworthy, a Target Environment may become a new Attesting Environment in charge of creating Evidence for further Target Environments. [I-D.ietf-rats-architecture] explains this as Layered Attestation. Layered Attestation has to start with an initial Attesting Environment (i.e., there cannot be turtles all the way down [turtles]). At this rock bottom of Layered Attestation, the Attesting Environments are called Roots of Trust (RoT). An Attester cannot create Evidence about its own RoTs by design. As a consequence, a Verifier requires trustable statements about this subset of Attesting Environments from a different source than the Attester itself. The corresponding trustable statements are called Endorsements and originate from external, trustable entities that take on the role of an Endorser (e.g., supply chain entities).

5.2. Endorsers for Direct Anonymous Attestation

In order to enable DAA to be used, an Endorser role takes on the duties of a DAA Issuer in addition to its already defined duties. DAA Issuers offer zero-knowledge proofs based on public key certificates used for a group of Attesters [DAA]. Effectively, these certificates share the semantics of Endorsements. The differences are:

- o The associated private keys are used by the DAA Issuer to provide an Attester with a credential that it can use to convince the Verifier that its Evidence is valid. To keep their anonymity the Attester randomises this credential each time that it is used.
- o The Verifier can use the DAA Issuer's public key certificate, together with the randomised credential from the Attester, to confirm that the Evidence comes from a valid Attester.
- o A credential is conveyed from an Endorser to an Attester together with the transfer of the public key certificates from Endorser to Verifier.

The zero-knowledge proofs required cannot be created by an Attester alone - like the Endorsements of RoTs - and have to be created by a trustable third entity - like an Endorser. Due to that vast semantic overlap (XXX-mcr:explain), an Endorser in this document can convey trustable third party statements both to a Verifier and an Attester.

6. Normative Prerequisites

In order to ensure an appropriate conveyance of Evidence, the following set of prerequisites MUST be in place to support the implementation of interaction models:

Attester Identity: The provenance of Evidence with respect to a distinguishable Attesting Environment MUST be correct and unambiguous.

An Attester Identity MAY be a unique identity, it MAY be included in a zero-knowledge proof (ZKP), or it MAY be part of a group signature, or it MAY be a randomised DAA credential.

Attestation Evidence Authenticity: Attestation Evidence MUST be correct and authentic.

In order to provide proofs of authenticity, Attestation Evidence SHOULD be cryptographically associated with an identity document (e.g. an PKIX certificate or trusted key material, or a randomised

DAA credential), or SHOULD include a correct and unambiguous and stable reference to an accessible identity document.

Authentication Secret: An Authentication Secret MUST be available exclusively to an Attester's Attesting Environment.

The Attester MUST protect Claims with that Authentication Secret, thereby proving the authenticity of the Claims included in Evidence. The Authentication Secret MUST be established before RATS can take place.

Evidence Freshness: Evidence MUST include an indicator about its Freshness that can be understood by a Verifier. Analogously, interaction models MUST support the conveyance of proofs of freshness in a way that is useful to Verifiers and their appraisal procedures.

Evidence Protection: Evidence MUST be a set of well-formatted and well-protected Claims that an Attester can create and convey to a Verifier in a tamper-evident manner.

7. Generic Information Elements

This section defines the information elements that are vital to all kinds interaction models. Varying from solution to solution, generic information elements can be either included in the scope of protocol messages or can be included in their payload. Ultimately, the following information elements are required by any kind of scalable remote attestation procedure using one or more of the interaction models provided.

Attester Identity ('attesterIdentity'): _mandatory_

A statement about a distinguishable Attester made by an Endorser without accompanying evidence about its validity - used as proof of identity.

In DAA the Attester's identity is not revealed to the verifier. The Attester is issued with a credential by the Endorser that is randomised and then used to anonymously confirm the validity of their evidence. The evidence is verified using the Endorser's public key.

Authentication Secret IDs ('authSecID'): _mandatory_

A statement representing an identifier list that MUST be associated with corresponding Authentication Secrets used to protect Evidence. In DAA, Authentication Secret IDs are

represented by the Endorser (DAA issuer)'s public key that MUST be used to create DAA credentials for the corresponding Authentication Secrets used to protect Evidence.

Each Authentication Secret is uniquely associated with a distinguishable Attesting Environment. Consequently, an Authentication Secret ID also identifies an Attesting Environment. In DAA an Authentication Secret ID does not identify a unique Attesting Environment but associated with a group of Attesting Environments. This is because an Attesting Environment should not be distinguishable and the DAA credential which represents the Attesting Environment is randomised each time it used.

Handle ('handle'): _mandatory_

A statement that is intended to uniquely distinguish received Evidence and/or determine the Freshness of Evidence.

A Verifier can also use a Handle as an indicator for authenticity or attestation provenance, as only Attesters and Verifiers that are intended to exchange Evidence should have knowledge of the corresponding Handles. Examples include Nonces or signed timestamps.

Claims ('claims'): _mandatory_

Claims are assertions that represent characteristics of an Attester's Target Environment.

Claims are part Conceptual Message and are, for example, used to appraise the integrity of Attesters via a Verifiers. The other information elements in this section can be expressed as Claims in any type of Conceptual Messages.

Reference Claims ('refClaims') _mandatory_

Reference Claims are a specific subset of Appraisal Policies as defined in [I-D.ietf-rats-architecture].

Reference Claims are used to appraise the Claims received from an Attester via appraisal by direct comparison. For example, Reference Claims MAY be Reference Integrity Measurements (RIM) or assertions that are implicitly trusted because they are signed by a trusted authority (see Endorsements in [I-D.ietf-rats-architecture]). Reference Claims typically represent (trusted) Claim sets about an Attester's intended platform operational state.

Claim Selection ('claimSelection'): _optional_

A statement that represents a (sub-)set of Claims that can be created by an Attester.

Claim Selections can act as filters that can specify the exact set of Claims to be included in Evidence. An Attester MAY decide whether or not to provide all Claims as requested via a Claim Selection.

Evidence ('signedAttestationEvidence'): _mandatory_

A set of Claims that consists of a list of Authentication Secret IDs that each identifies an Authentication Secret in a single Attesting Environment, the Attester Identity, Claims, and a Handle. Attestation Evidence MUST cryptographically bind all of these information elements. The Evidence MUST be protected via the Authentication Secret. The Authentication Secret MUST be trusted by the Verifier as authoritative.

Attestation Result ('attestationResult'): _mandatory_

An Attestation Result is produced by the Verifier as the output of the appraisal of Evidence. Attestation Results include condensed assertions about integrity or other characteristics of the corresponding Attester.

8. Interaction Models

The following subsections introduce and illustrate the interaction models:

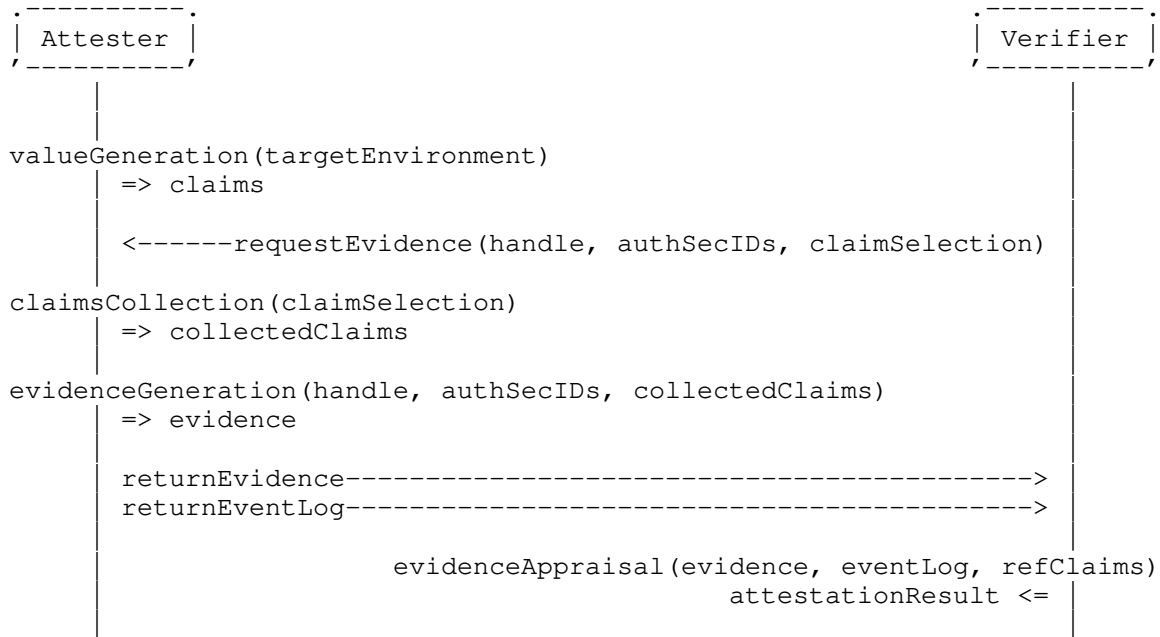
1. Challenge/Response Remote Attestation
2. Uni-Directional Remote Attestation
3. Streaming Remote Attestation

Each section starts with a sequence diagram illustrating the interactions between Attester and Verifier. The other roles RATS roles - mainly Relying Parties and Endorsers - are not relevant for this interaction model. While the interaction models presented focus on the conveyance of Evidence, future work could apply this to the conveyance of other Conceptual Messages, namely Attestation Results, Endorsements, or Appraisal Policies.

All interaction model have a strong focus on the use of a handle to incorporate a proof of freshness. The ways these handles are

processed is the most prominent difference between the three interaction models.

8.1. Challenge/Response Remote Attestation



This Challenge/Response Remote Attestation procedure is initiated by the Verifier, by sending a remote attestation request to the Attester. A request includes a Handle, a list of Authentication Secret IDs, and a Claim Selection.

In the Challenge/Response model, the handle is composed of qualifying data in the form of a cryptographically strongly randomly generated, and therefore unpredictable, nonce. The Verifier-generated nonce is intended to guarantee Evidence freshness.

The list of Authentication Secret IDs selects the attestation keys with which the Attester is requested to sign the Attestation Evidence. Each selected key is uniquely associated with an Attesting Environment of the Attester. As a result, a single Authentication Secret ID identifies a single Attesting Environment.

Analogously, a particular set of Evidence originating from a particular Attesting Environments in a composite device can be requested via multiple Authentication Secret IDs. Methods to acquire

Authentication Secret IDs or mappings between Attesting Environments to Authentication Secret IDs are out-of-scope of this document.

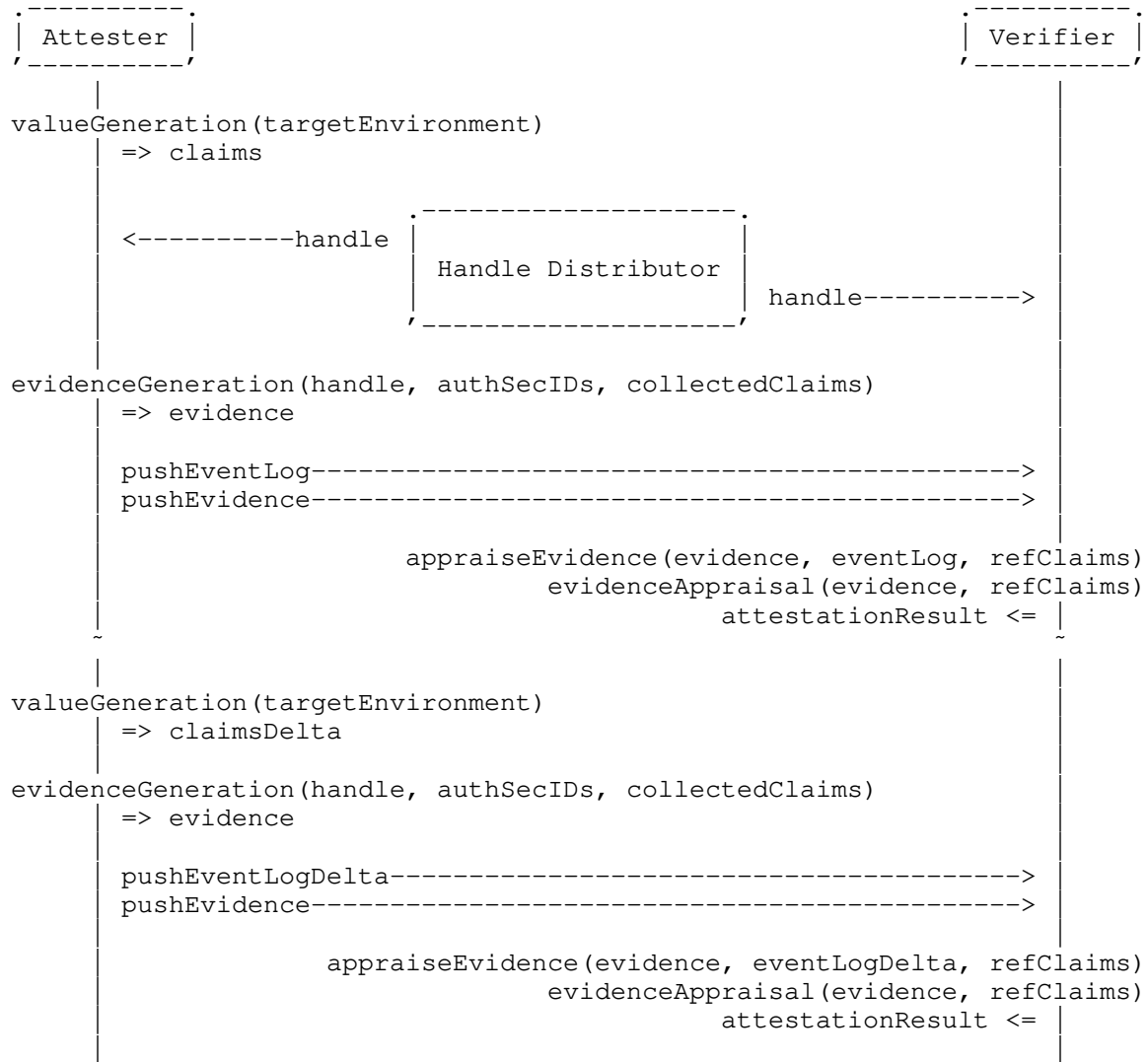
The Claim Selection narrows down the set of Claims collected and used to create Evidence to those that the Verifier requires. If the Claim Selection is omitted, then by default all Claims that are known and available on the Attester MUST be used to create corresponding Evidence. For example when performing a boot integrity evaluation, a Verifier may only be requesting a particular subset of claims about the Attester, such as Evidence about BIOS and firmware the Attester booted up, and not include information about all currently running software.

While it is crucial that Claims, the Handle, as well as the Attester Identity information MUST be cryptographically bound to the signature of Evidence, they may be presented in an encrypted form.

Cryptographic blinding MAY be used at this point. For further reference see section Section 11.

As soon as the Verifier receives signed Evidence, it validates the signature, the Attester Identity, as well as the Nonce, and appraises the Claims. Appraisal procedures are application-specific and can be conducted via comparison of the Claims with corresponding Reference Claims, such as Reference Integrity Measurements. The final output of the Verifier are Attestation Results. Attestation Results constitute new Claims Sets about an Attester's properties and characteristics that enables Relying Parties, for example, to assess an Attester's trustworthiness.

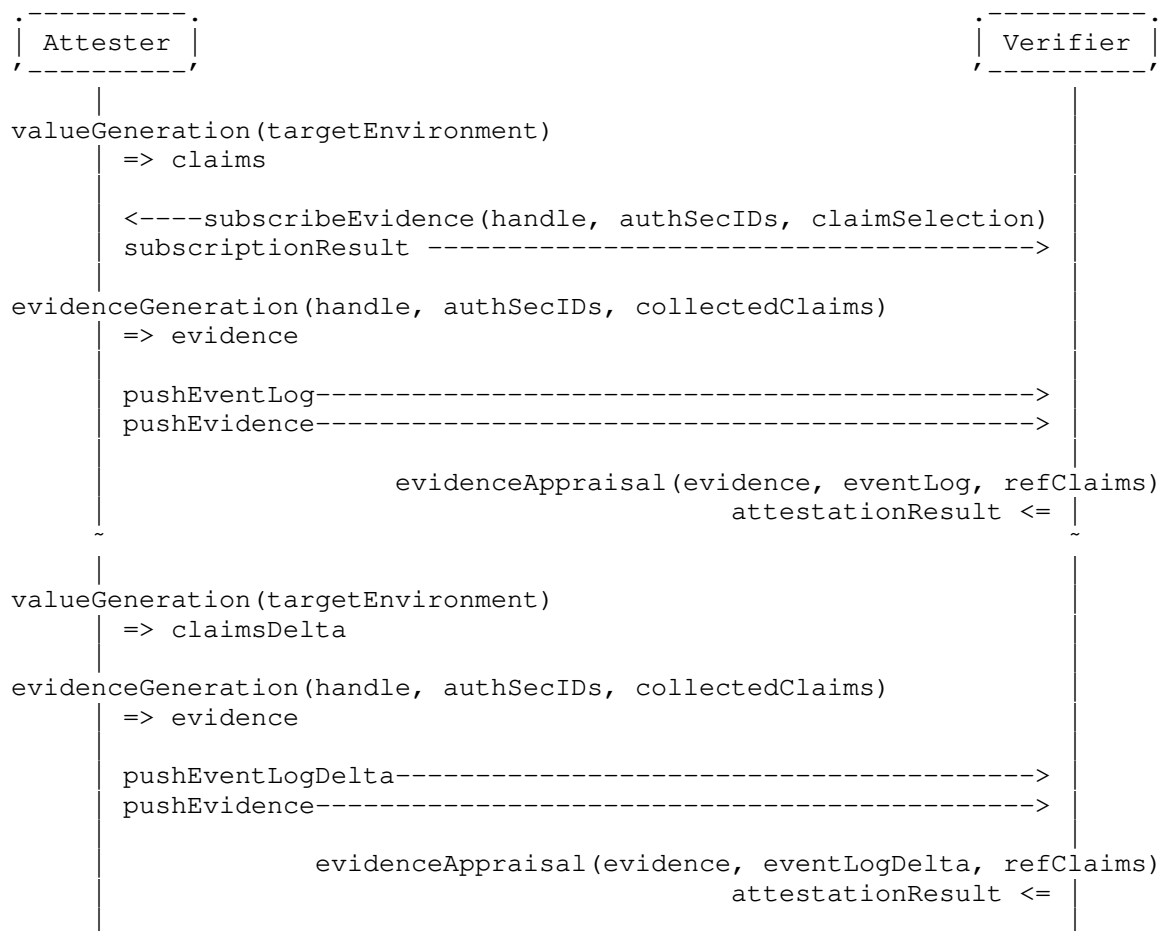
8.2. Uni-Directional Remote Attestation



Uni-Directional Remote Attestation procedures can be initiated both by the Attester and by the Verifier. Initiation by the Attester can result in unsolicited pushes of Evidence to the Verifier. Initiation by the Verifier always results in solicited pushes to the Verifier. The Uni-Directional model uses the same information elements as the Challenge/Response model. In the sequence diagram above, the Attester initiates the conveyance of Evidence (comparable with a RESTful POST operation or the emission of a beacon). While a request of evidence from the Verifier would result in a sequence diagram more similar to the Challenge/Response model (comparable with a RESTful

GET operation), the specific manner how handles are created and used always remains as the distinguishing quality of this model. In the Uni-Directional model, handles are composed of trustable signed timestamps as shown in [I-D.birkholz-rats-tuda], potentially including other qualifying data. The handles are created by an external 3rd entity - the Handle Distributor - that includes a trustworthy source of time and takes on the role of a Time Stamping Authority (TSA, as initially defined in [RFC3161]). Timestamps created from local clocks (absolute clocks using a global timescale, as well as relative clocks, such as tick-counters) of Attesters and Verifiers MUST be cryptographically bound to fresh Handles received from the Handle Distributor. This binding provides a proof of synchronization that MUST be included in every evidence created. Correspondingly, evidence created for conveyance via this model provides a proof that it was fresh at a certain point in time. Effectively, this allows for series of evidence to be pushed to multiple Verifiers, simultaneously. Methods to detect excessive time drift that would mandate a fresh Handle to be received by the Handle Distributor, as well as timing of handle distribution are out-of-scope of this document.

8.3. Streaming Remote Attestation



Streaming Remote Attestation procedures require the setup of subscription state. Setting up subscription state between a Verifier and an Attester is conducted via a subscribe operation. This subscribe operation is used to convey the handles required for Evidence generation. Effectively, this allows for series of evidence to be pushed to a Verifier similar to the Uni-Directional model. While a Handle Distributor is not required in this model, it is also limited to bi-lateral subscription relationships, in which each Verifier has to create and provide its individual handle. Handles provided by a specific subscribing Verifier MUST be used in Evidence generation for that specific Verifier. The Streaming model uses the same information elements as the Challenge/Response and the Uni-Directional model. Methods to detect excessive time drift that would mandate a refreshed Handle to be conveyed via another subscribe operation are out-of-scope of this document.

9. Additional Application-Specific Requirements

Depending on the use cases covered, there can be additional requirements. An exemplary subset is illustrated in this section.

9.1. Confidentiality

Confidentiality of exchanged attestation information may be desirable. This requirement usually is present when communication takes place over insecure channels, such as the public Internet. In such cases, TLS may be used as a suitable communication protocol that preserves confidentiality. In private networks, such as carrier management networks, it must be evaluated whether or not the transport medium is considered confidential.

9.2. Mutual Authentication

In particular use cases mutual authentication may be desirable in such a way that a Verifier also needs to prove its identity to the Attester, instead of only the Attester proving its identity to the Verifier.

9.3. Hardware-Enforcement/Support

Depending on given usage scenarios, hardware support for secure storage of cryptographic keys, crypto accelerators, as well as protected or isolated execution environments can be mandatory requirements. Well-known technologies in support of these requirements are roots of trusts, such as Hardware Security Modules (HSM), Physically Unclonable Functions (PUFs), Shielded Secrets, or Trusted Executions Environments (TEEs).

10. Implementation Status

Note to RFC Editor: Please remove this section as well as references to [BCP205] before AUTH48.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [BCP205]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their

features. Readers are advised to note that other implementations may exist.

According to [BCP205], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

10.1. Implementer

The open-source implementation was initiated and is maintained by the Fraunhofer Institute for Secure Information Technology - SIT.

10.2. Implementation Name

The open-source implementation is named "CHallenge-Response based Remote Attestation" or in short: CHARRA.

10.3. Implementation URL

The open-source implementation project resource can be located via:
<https://github.com/Fraunhofer-SIT/charra>

10.4. Maturity

The code's level of maturity is considered to be "prototype".

10.5. Coverage and Version Compatibility

The current version (commit '847bcde') is aligned with the exemplary specification of the CoAP FETCH bodies defined in section Appendix A of this document.

10.6. License

The CHARRA project and all corresponding code and data maintained on github are provided under the BSD 3-Clause "New" or "Revised" license.

10.7. Implementation Dependencies

The implementation requires the use of the official Trusted Computing Group (TCG) open-source Trusted Software Stack (TSS) for the Trusted Platform Module (TPM) 2.0. The corresponding code and data is also maintained on github and the project resources can be located via:
<https://github.com/tpm2-software/tpm2-tss/>

The implementation uses the Constrained Application Protocol [RFC7252] (<http://coap.technology/>) and the Concise Binary Object Representation [RFC7049] (<https://cbor.io/>).

10.8. Contact

Michael Eckel (michael.eckel@sit.fraunhofer.de)

11. Security and Privacy Considerations

In a remote attestation procedure the Verifier or the Attester MAY want to cryptographically blind several attributes. For instance, information can be part of the signature after applying a one-way function (e. g. a hash function).

There is also a possibility to scramble the Nonce or Attester Identity with other information that is known to both the Verifier and Attester. A prominent example is the IP address of the Attester that usually is known by the Attester itself as well as the Verifier. This extra information can be used to scramble the Nonce in order to counter certain types of relay attacks.

12. Acknowledgments

Olaf Bergmann, Michael Richardson, and Ned Smith

13. Change Log

- o Initial draft -00
- o Changes from version 00 to version 01:
 - * Added details to the flow diagram
 - * Integrated comments from Ned Smith (Intel)
 - * Reorganized sections and
 - * Updated interaction model
 - * Replaced "claims" with "assertions"
 - * Added proof-of-concept CDDL for CBOR via CoAP based on a TPM 2.0 quote operation
- o Changes from version 01 to version 02:

- * Revised the relabeling of "claims" with "assertion" in alignment with the RATS Architecture I-D.
- * Added Implementation Status section
- * Updated interaction model
- * Text revisions based on changes in [I-D.ietf-rats-architecture] and comments provided on rats@ietf.org.
- o Changes from version 02 to version 00 RATS related document
 - * update of the challenge/response diagram
 - * minor rephrasing of Prerequisites section
 - * rephrasing to information elements and interaction model section
- o Changes from version 00 to version 01
 - * added Attestation Authenticity, updated Identity and Secret
 - * relabeled Secret ID to Authentication Secret ID + rephrasing
 - * relabeled Claim Selection to Assertion Selection + rephrasing
 - * relabeled Evidence to (Signed) Attestation Evidence
 - * Added Attestation Result and Reference Assertions
 - * update of the challenge/response diagram and expositional text
 - * added CDDL spec for CoAP FETCH operation proof-of-concept
- o Changes from version 01 to version 02
 - * prepared the inclusion of additional reference models
 - * update to Introduction and Scope section
 - * major update to (Normative) Prerequisites
 - * relabeled Attestation Authenticity to Att. Evidence Authenticity
 - * relabeled Assertion term back to Claim terms

- * added BCP205 Implementation Status section related to Appendix CDDL
- o Changes from version 02 to version 03
 - * major refactoring to now accommodate three interaction models
 - * updated existing and added two new diagrams for models
 - * major refactoring of existing and adding of new diagram description
 - * incorporated content about Direct Anonymous Attestation
 - * integrated comments from Michael Richardson
 - * updated roster

14. References

14.1. Normative References

- [BCP205] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [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>>.
- [RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <<https://www.rfc-editor.org/info/rfc3161>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.

- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.
- [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>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.

14.2. Informative References

- [DAA] Brickell, E., Camenisch, J., and L. Chen, "Direct Anonymous Attestation", ACM Proceedings of the 11rd ACM conference on Computer and Communications Security , page 132-145, 2004.
- [I-D.birkholz-rats-tuda] Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", draft-birkholz-rats-tuda-02 (work in progress), March 2020.
- [I-D.ietf-rats-architecture] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-04 (work in progress), May 2020.
- [turtles] Wikipedia, "Turtles all the way down", July 2020, <https://en.wikipedia.org/wiki/Turtles_all_the_way_down>.

Appendix A. CDDL Specification for a simple CoAP Challenge/Response Interaction

The following CDDL specification is an exemplary proof-of-concept to illustrate a potential implementation of the Challenge/Response Interaction Model. The transfer protocol used is CoAP using the FETCH operation. The actual resource operated on can be empty. Both the Challenge Message and the Response Message are exchanged via the FETCH operation and corresponding FETCH Request and FETCH Response body.

In this example, evidence is created via the root-of-trust for reporting primitive operation "quote" that is provided by a TPM 2.0.

RAIM-Bodies = CoAP-FETCH-Body / CoAP-FETCH-Response-Body

```
CoAP-FETCH-Body = [ hello: bool, ; if true, the AK-Cert is conveyed
                    nonce: bytes,
                    pcr-selection: [ + [ tcg-hash-alg-id: uint .size 2, ; TPM2_AL
G_ID
                                [ + pcr: uint .size 1 ],
                                ],
                    ],
```

```
CoAP-FETCH-Response-Body = [ attestation-evidence: TPMS_ATTEST-quote,
                             tpm-native-signature: bytes,
                             ? ak-cert: bytes, ; attestation key certificate
                             ]
```

```
TPMS_ATTEST-quote = [ qualifiediSigner: uint .size 2, ;TPM2B_NAME
                     TPMS_CLOCK_INFO,
                     firmwareVersion: uint .size 8
                     quote-responses: [ * [ pcr: uint .size 1,
                                             + [ pcr-value: bytes,
                                                ? hash-alg-id: uint .size 2,
                                                ],
                                             ],
                                             ? pcr-digest: bytes,
                                             ],
                     ]
```

```
TPMS_CLOCK_INFO = [ clock: uint .size 8,
                    resetCounter: uint .size 4,
                    restartCounter: uint .size 4,
                    save: bool,
                    ]
```

Authors' Addresses

Henk Birkholz
 Fraunhofer SIT
 Rheinstrasse 75
 Darmstadt 64295
 Germany

Email: henk.birkholz@sit.fraunhofer.de

Michael Eckel
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: michael.eckel@sit.fraunhofer.de

Christopher Newton
University of Surrey

Email: cn0016@surrey.ac.uk

Liqun Chen
University of Surrey

Email: liqun.chen@surrey.ac.uk

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2020

G. Mandyam
Qualcomm Technologies Inc.
L. Lundblade
Security Theory LLC
M. Ballesteros
J. O'Donoghue
Qualcomm Technologies Inc.
July 04, 2019

The Entity Attestation Token (EAT)
draft-ietf-rats-eat-01

Abstract

An Entity Attestation Token (EAT) provides a signed (attested) set of claims that describe state and characteristics of an entity, typically a device like a phone or an IoT device. These claims are used by a relying party to determine how much it wishes to trust the entity.

An EAT is either a CWT or JWT with some attestation-oriented claims. To a large degree, all this document does is extend CWT and JWT.

Contributing

TBD

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 January 5, 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
1.1. CDDL, CWT and JWT	4
1.2. Entity Overview	4
1.3. EAT Operating Models	5
1.4. What is Not Standardized	6
1.4.1. Transmission Protocol	6
1.4.2. Signing Scheme	6
2. Terminology	7
3. The Claims Information Model	8
3.1. Nonce Claim (cti and jti)	8
3.2. Timestamp claim (iat)	8
3.3. Universal Entity ID Claim (ueid)	8
3.4. Origination Claim (origination)	11
3.4.1. CDDL	12
3.5. OEM identification by IEEE OUI (oemid)	12
3.5.1. CDDL	12
3.6. The Security Level Claim (security_level)	12
3.6.1. CDDL	13
3.7. Secure Boot and Debug Enable State Claims (boot_state)	13
3.7.1. Secure Boot Enabled	13
3.7.2. Debug Disabled	13
3.7.3. Debug Disabled Since Boot	14
3.7.4. Debug Permanent Disable	14
3.7.5. Debug Full Permanent Disable	14
3.7.6. CDDL	14
3.8. The Location Claim (location)	14
3.8.1. CDDL	15
3.9. The Age Claim (age)	15
3.10. The Uptime Claim (uptime)	15
3.10.1. CDDL	15
3.11. Nested EATs, the EAT Claim (nested_eat)	15

3.11.1. CDDL	16
3.12. The Submods Claim (submods)	16
3.12.1. The submod_name Claim	16
3.12.2. CDDL	16
4. Data Model	17
4.1. Common CDDL Types	17
4.2. CDDL for CWT-defined Claims	17
4.3. JSON	18
4.3.1. JSON Labels	18
4.3.2. JSON Interoperability	19
4.4. CBOR	19
4.4.1. Labels	19
4.4.2. CBOR Interoperability	20
4.5. Collected CDDL	21
5. IANA Considerations	21
5.1. Reuse of CBOR Web Token (CWT) Claims Registry	21
5.1.1. Claims Registered by This Document	22
6. Privacy Considerations	22
6.1. UEID Privacy Considerations	22
7. Security Considerations	23
8. References	23
8.1. Normative References	23
8.2. Informative References	24
Appendix A. Examples	26
A.1. Very Simple EAT	26
A.2. Example with Submodules, Nesting and Security Levels	26
Appendix B. Changes from Previous Drafts	27
B.1. From draft-mandyam-rats-eat-00	27
Authors' Addresses	27

1. Introduction

Remote device attestation is a fundamental service that allows a remote device such as a mobile phone, an Internet-of-Things (IoT) device, or other endpoint to prove itself to a relying party, a server or a service. This allows the relying party to know some characteristics about the device and decide whether it trusts the device.

Remote attestation is a fundamental service that can underlie other protocols and services that need to know about the trustworthiness of the device before proceeding. One good example is biometric authentication where the biometric matching is done on the device. The relying party needs to know that the device is one that is known to do biometric matching correctly. Another example is content protection where the relying party wants to know the device will protect the data. This generalizes on to corporate enterprises that

might want to know that a device is trustworthy before allowing corporate data to be accessed by it.

The notion of attestation here is large and may include, but is not limited to the following:

- o Proof of the make and model of the device hardware (HW)
- o Proof of the make and model of the device processor, particularly for security-oriented chips
- o Measurement of the software (SW) running on the device
- o Configuration and state of the device
- o Environmental characteristics of the device such as its GPS location

1.1. CDDL, CWT and JWT

An EAT token is either a CWT as defined in [RFC8392] or a JWT as defined in [RFC7519]. This specification defines additional claims for entity attestation.

This specification uses CDDL, [RFC8610], as the primary formalism to define each claim. The implementor then interprets the CDDL to come to either the CBOR [RFC7049] or JSON [ECMAScript] representation. In the case of JSON, Appendix E of [RFC8610] is followed. Additional rules are given in Section 4.3.2 of this document where Appendix E is insufficient. (Note that this is not to define a general means to translate between CBOR and JSON, but only to define enough such that the claims defined in this document can be rendered unambiguously in JSON).

1.2. Entity Overview

An "entity" can be any device or device subassembly ("submodule") that can generate its own attestation in the form of an EAT. The attestation should be cryptographically verifiable by the EAT consumer. An EAT at the device-level can be composed of several submodule EAT's. It is assumed that any entity that can create an EAT does so by means of a dedicated root-of-trust (RoT).

Modern devices such as a mobile phone have many different execution environments operating with different security levels. For example, it is common for a mobile phone to have an "apps" environment that runs an operating system (OS) that hosts a plethora of downloadable apps. It may also have a TEE (Trusted Execution Environment) that is

distinct, isolated, and hosts security-oriented functionality like biometric authentication. Additionally, it may have an eSE (embedded Secure Element) – a high security chip with defenses against HW attacks that can serve as a RoT. This device attestation format allows the attested data to be tagged at a security level from which it originates. In general, any discrete execution environment that has an identifiable security level can be considered an entity.

1.3. EAT Operating Models

At least the following three participants exist in all EAT operating models. Some operating models have additional participants.

The Entity. This is the phone, the IoT device, the sensor, the sub-assembly or such that the attestation provides information about.

The Manufacturer. The company that made the entity. This may be a chip vendor, a circuit board module vendor or a vendor of finished consumer products.

The Relying Party. The server, service or company that makes use of the information in the EAT about the entity.

In all operating models, the manufacturer provisions some secret attestation key material (AKM) into the entity during manufacturing. This might be during the manufacturer of a chip at a fabrication facility (fab) or during final assembly of a consumer product or any time in between. This attestation key material is used for signing EATs.

In all operating models, hardware and/or software on the entity create an EAT of the format described in this document. The EAT is always signed by the attestation key material provisioned by the manufacturer.

In all operating models, the relying party must end up knowing that the signature on the EAT is valid and consistent with data from claims in the EAT. This can happen in many different ways. Here are some examples.

- o The EAT is transmitted to the relying party. The relying party gets corresponding key material (e.g. a root certificate) from the manufacturer. The relying party performs the verification.
- o The EAT is transmitted to the relying party. The relying party transmits the EAT to a verification service offered by the manufacturer. The server returns the validated claims.

- o The EAT is transmitted directly to a verification service, perhaps operated by the manufacturer or perhaps by another party. It verifies the EAT and makes the validated claims available to the relying party. It may even modify the claims in some way and re-sign the EAT (with a different signing key).

All these operating models are supported and there is no preference of one over the other. It is important to support this variety of operating models to generally facilitate deployment and to allow for some special scenarios. One special scenario has a validation service that is monetized, most likely by the manufacturer. In another, a privacy proxy service processes the EAT before it is transmitted to the relying party. In yet another, symmetric key material is used for signing. In this case the manufacturer should perform the verification, because any release of the key material would enable a participant other than the entity to create valid signed EATs.

1.4. What is Not Standardized

The following is not standardized for EAT, just the same they are not standardized for CWT or JWT.

1.4.1. Transmission Protocol

EATs may be transmitted by any protocol the same as CWTs and JWTs. For example, they might be added in extension fields of other protocols, bundled into an HTTP header, or just transmitted as files. This flexibility is intentional to allow broader adoption. This flexibility is possible because EAT's are self-secured with signing (and possibly additionally with encryption and anti-replay). The transmission protocol is not required to fulfill any additional security requirements.

For certain devices, a direct connection may not exist between the EAT-producing device and the Relying Party. In such cases, the EAT should be protected against malicious access. The use of COSE and JOSE allows for signing and encryption of the EAT. Therefore, even if the EAT is conveyed through intermediaries between the device and Relying Party, such intermediaries cannot easily modify the EAT payload or alter the signature.

1.4.2. Signing Scheme

The term "signing scheme" is used to refer to the system that includes end-end process of establishing signing attestation key material in the entity, signing the EAT, and verifying it. This might involve key IDs and X.509 certificate chains or something

similar but different. The term "signing algorithm" refers just to the algorithm ID in the COSE signing structure. No particular signing algorithm or signing scheme is required by this standard.

There are three main implementation issues driving this. First, secure non-volatile storage space in the entity for the attestation key material may be highly limited, perhaps to only a few hundred bits, on some small IoT chips. Second, the factory cost of provisioning key material in each chip or device may be high, with even millisecond delays adding to the cost of a chip. Third, privacy-preserving signing schemes like ECDAA (Elliptic Curve Direct Anonymous Attestation) are complex and not suitable for all use cases.

Over time to facilitate interoperability, some signing schemes may be defined in EAT profiles or other documents either in the IETF or outside.

2. Terminology

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.

This document reuses terminology from JWT [RFC7519], COSE [RFC8152], and CWT [RFC8392].

Claim Name. The human-readable name used to identify a claim.

Claim Key. The CBOR map key or JSON name used to identify a claim.

Claim Value. The CBOR map or JSON object value representing the value of the claim.

CWT Claims Set. The CBOR map or JSON object that contains the claims conveyed by the CWT or JWT.

Attestation Key Material (AKM). The key material used to sign the EAT token. If it is done symmetrically with HMAC, then this is a simple symmetric key. If it is done with ECC, such as an IEEE DevID [IDevID], then this is the private part of the EC key pair. If ECDAA is used, (e.g., as used by Enhanced Privacy ID, i.e. EPID) then it is the key material needed for ECDAA.

3. The Claims Information Model

This section describes new claims defined for attestation. It also mentions several claims defined by CWT and JWT are particularly important for EAT.

Note also: * Any claim defined for CWT or JWT may be used in an EAT including those in the CWT [IANA.CWT.Claims] and JWT IANA [IANA.JWT.Claims] claims registries. * All claims are optional * No claims are mandatory * All claims that are not understood by implementations MUST be ignored

CDDL along with text descriptions is used to define the information model. Each claim is defined as a CDDL group (the group is a general aggregation and type definition feature of CDDL). In the data model, described in the Section 4, the CDDL groups turn into CBOR map entries and JSON name/value pairs.

3.1. Nonce Claim (cti and jti)

All EATs should have a nonce to prevent replay attacks. The nonce is generated by the relying party, sent to the entity by any protocol, and included in the token. Note that intrinsically by the nature of a nonce no security is needed for its transport.

CWT defines the "cti" claim. JWT defines the "jti" claim. These carry the nonce in an EAT.

TODO: what about the JWT claim "nonce"?

3.2. Timestamp claim (iat)

The "iat" claim defined in CWT and JWT is used to indicate the date-of-creation of the token.

3.3. Universal Entity ID Claim (ueid)

UEID's identify individual manufactured entities / devices such as a mobile phone, a water meter, a Bluetooth speaker or a networked security camera. It may identify the entire device or a submodule or subsystem. It does not identify types, models or classes of devices. It is akin to a serial number, though it does not have to be sequential.

UEID's must be universally and globally unique across manufacturers and countries. UEIDs must also be unique across protocols and systems, as tokens are intended to be embedded in many different protocols and systems. No two products anywhere, even in completely

different industries made by two different manufacturers in two different countries should have the same UEID (if they are not global and universal in this way, then relying parties receiving them will have to track other characteristics of the device to keep devices distinct between manufacturers).

There are privacy considerations for UEID's. See Section 6.1.

The UEID should be permanent. It should never change for a given device / entity. In addition, it should not be reprogrammable. UEID's are variable length. The recommended maximum is 33 bytes (1 type byte and 256 bits). The recommended minimum is 17 bytes (1 type and 128 bits) because fewer bytes endanger the universal uniqueness.

When the entity constructs the UEID, the first byte is a type and the following bytes the ID for that type. Several types are allowed to accommodate different industries and different manufacturing processes and to give options to avoid paying fees for certain types of manufacturer registrations.

Creation of new types requires a Standards Action [RFC8126].

Type Byte	Type Name	Specification
0x01	RAND	This is a 128- to 256-bit random number generated once and stored in the device. This may be constructed by concatenating enough identifiers to be universally unique and then feeding the concatenation through a cryptographic hash function. It may also be a cryptographic quality random number generate once at the beginning of the life of the device and stored.
0x02	IEEE EUI	This makes use of the IEEE company identification registry. An EUI is made up of an OUI and OUI-36 or a CID, different registered company identifiers, and some unique per-device identifier. EUIs are often the same as or similar to MAC addresses. (Note that while devices with multiple network interfaces may have multiple MAC addresses, there is only one UEID for a device)
0x03	IMEI	TODO: normative references to IEEE. This is a 14-digit identifier consisting of an 8-digit Type Allocation Code and a 6-digit serial number allocated by the manufacturer, which SHALL be encoded as a binary integer over 48 bits. The IMEI value encoded SHALL NOT include Luhn checksum or SVN information.
0x04	EUI-48	This is a 48-bit identifier formed by concatenating the 24-bit OUI with a 24-bit identifier assigned by the organisation that purchased the OUI.
0x05	EUI-60	This is a 60-bit identifier formed by concatenating the 24-bit OUI with a 36-bit identifier assigned by the organisation that purchased the OUI.
0x06	EUI-64	This is a 64-bit identifier formed by concatenating the 24-bit OUI with a 40-bit identifier assigned by the organisation that purchased the OUI.

Table 1: UEID Composition Types

UEID's are not designed for direct use by humans (e.g., printing on the case of a device), so no textual representation is defined.

The consumer (the relying party) of a UEID MUST treat a UEID as a completely opaque string of bytes and not make any use of its

internal structure. For example, they should not use the OUI part of a type 0x02 UEID to identify the manufacturer of the device. Instead they should use the OUI claim that is defined elsewhere. The reasons for this are:

- o UEIDs types may vary freely from one manufacturer to the next.
- o New types of UEIDs may be created. For example, a type 0x07 UEID may be created based on some other manufacturer registration scheme.
- o Device manufacturers are allowed to change from one type of UEID to another anytime they want. For example, they may find they can optimize their manufacturing by switching from type 0x01 to type 0x02 or vice versa. The main requirement on the manufacturer is that UEIDs be universally unique.

CDDL

```
ueid_claim = (
  ueid: bstr )
```

3.4. Origination Claim (origination)

This claim describes the parts of the device or entity that are creating the EAT. Often it will be tied back to the device or chip manufacturer. The following table gives some examples:

Name	Description
Acme-TEE	The EATs are generated in the TEE authored and configured by "Acme"
Acme-TPM	The EATs are generated in a TPM manufactured by "Acme"
Acme-Linux-Kernel	The EATs are generated in a Linux kernel configured and shipped by "Acme"
Acme-TA	The EATs are generated in a Trusted Application (TA) authored by "Acme"

TODO: consider a more structure approach where the name and the URI and other are in separate fields.

TODO: This needs refinement. It is somewhat parallel to issuer claim in CWT in that it describes the authority that created the token.

3.4.1. CDDL

```
origination_claim = (  
  origination: string_or_uri )
```

3.5. OEM identification by IEEE OUI (oemid)

This claim identifies a device OEM by the IEEE OUI. Reference TBD. It is a byte string representing the OUI in binary form in network byte order (TODO: confirm details).

Companies that have more than one IEEE OUI registered with IEEE should pick one and prefer that for all their devices.

Note that the OUI is in common use as a part of MAC Address. This claim is only the first bits of the MAC address that identify the manufacturer. The IEEE maintains a registry for these in which many companies participate.

3.5.1. CDDL

```
oemid_claim = (  
  oemid: bstr )
```

3.6. The Security Level Claim (security_level)

EATs have a claim that roughly characterizes the device / entities ability to defend against attacks aimed at capturing the signing key, forging claims and at forging EATs. This is done by roughly defining four security levels as described below. This is similar to the security levels defined in the Metadata Service defined by the Fast Identity Online (FIDO) Alliance (TODO: reference).

These claims describe security environment and countermeasures available on the end-entity / client device where the attestation key reside and the claims originate.

- 1 - Unrestricted There is some expectation that implementor will protect the attestation signing keys at this level. Otherwise the EAT provides no meaningful security assurances.
- 2- Restricted Entities at this level should not be general-purpose operating environments that host features such as app download systems, web browsers and complex productivity applications. It is akin to the Secure Restricted level (see below) without the security orientation. Examples include a Wi-Fi subsystem, an IoT camera, or sensor device.

- 3 - Secure Restricted Entities at this level must meet the criteria defined by FIDO Allowed Restricted Operating Environments (TODO: reference). Examples include TEE's and schemes using virtualization-based security. Like the FIDO security goal, security at this level is aimed at defending well against large-scale network / remote attacks against the device.
- 4 - Hardware Entities at this level must include substantial defense against physical or electrical attacks against the device itself. It is assumed any potential attacker has captured the device and can disassemble it. Example include TPMs and Secure Elements.

This claim is not intended as a replacement for a proper end-device security certification schemes such as those based on FIPS (TODO: reference) or those based on Common Criteria (TODO: reference). The claim made here is solely a self-claim made by the Entity Originator.

3.6.1. CDDL

```
security_level_type = (  
  unrestricted: 1,  
  restricted: 2,  
  secure_restricted: 3,  
  hardware: 4  
)  
  
security_level_claim = (  
  security_level: security_level_type )
```

3.7. Secure Boot and Debug Enable State Claims (boot_state)

This claim is an array of five Boolean values indicating the boot and debug state of the entity.

3.7.1. Secure Boot Enabled

This indicates whether secure boot is enabled either for an entire device or an individual submodule. If it appears at the device level, then this means that secure boot is enabled for all submodules. Secure boot enablement allows a secure boot loader to authenticate software running either in a device or a submodule prior allowing execution.

3.7.2. Debug Disabled

This indicates whether debug capabilities are disabled for an entity (i.e. value of 'true'). Debug disablement is considered a prerequisite before an entity is considered operational.

3.7.3. Debug Disabled Since Boot

This claim indicates whether debug capabilities for the entity were not disabled in any way since boot (i.e. value of 'true').

3.7.4. Debug Permanent Disable

This claim indicates whether debug capabilities for the entity are permanently disabled (i.e. value of 'true'). This value can be set to 'true' also if only the manufacturer is allowed to enable debug, but the end user is not.

3.7.5. Debug Full Permanent Disable

This claim indicates whether debug capabilities for the entity are permanently disabled (i.e. value of 'true'). This value can only be set to 'true' if no party can enable debug capabilities for the entity. Often this is implemented by blowing a fuse on a chip as fuses cannot be restored once blown.

3.7.6. CDDL

```
boot_state_type = [  
    secure_boot_enabled=> bool,  
    debug_disabled=> bool,  
    debug_disabled_since_boot=> bool,  
    debug_permanent_disable=> bool,  
    debug_full_permanent_disable=> bool  
]  
  
boot_state_claim = (  
    boot_state: boot_state_type  
)
```

3.8. The Location Claim (location)

The location claim is a CBOR-formatted object that describes the location of the device entity from which the attestation originates. It is comprised of a map of additional sub claims that represent the actual location coordinates (latitude, longitude and altitude). The location coordinate claims are consistent with the WGS84 coordinate system [WGS84]. In addition, a sub claim providing the estimated accuracy of the location measurement is defined.

3.8.1. CDDL

```
location_type = {  
    latitude => number,  
    longitude => number,  
    altitude => number,  
    accuracy => number,  
    altitude_accuracy => number,  
    heading_claim => number,  
    speed_claim => number  
}  
  
location_claim = (  
    location: location_type )
```

3.9. The Age Claim (age)

The "age" claim contains a value that represents the number of seconds that have elapsed since the token was created, measurement was made, or location was obtained. Typical attestable values are sent as soon as they are obtained. However, in the case that such a value is buffered and sent at a later time and a sufficiently accurate time reference is unavailable for creation of a timestamp, then the age claim is provided.

```
age_claim = (  
    age: uint)
```

3.10. The Uptime Claim (uptime)

The "uptime" claim contains a value that represents the number of seconds that have elapsed since the entity or submod was last booted.

3.10.1. CDDL

```
uptime_claim = (  
    uptime: uint )
```

3.11. Nested EATs, the EAT Claim (nested_eat)

It is allowed for one EAT to be embedded in another. This is for complex devices that have more than one subsystem capable of generating an EAT. Typically, one will be the device-wide EAT that is low to medium security and another from a Secure Element or similar that is high security.

The contents of the "eat" claim must be a fully signed, optionally encrypted, EAT token.

3.11.1. CDDL

```
nested_eat_claim = (  
  nested_eat: nested_eat_type)
```

A `nested_eat_type` is defined in words rather than CDDL. It is either a full CWT or JWT including the COSE or JOSE signing.

3.12. The Submods Claim (submods)

Some devices are complex, having many subsystems or submodules. A mobile phone is a good example. It may have several connectivity submodules for communications (e.g., Wi-Fi and cellular). It may have subsystems for low-power audio and video playback. It may have one or more security-oriented subsystems like a TEE or a Secure Element.

The claims for each these can be grouped together in a submodule.

Specifically, the "submods" claim is an array. Each item in the array is a CBOR map containing all the claims for a particular submodule.

The security level of the submod is assumed to be at the same level as the main entity unless there is a security level claim in that submodule indicating otherwise. The security level of a submodule can never be higher (more secure) than the security level of the EAT it is a part of.

3.12.1. The submod_name Claim

Each submodule should have a `submod_name` claim that is descriptive name. This name should be the CBOR txt type.

3.12.2. CDDL

In the following a `generic_claim_type` is any CBOR map entry or JSON name/value pair.

```
submod_name_type = (  
  submod_name: tstr )  
  
submods_type = [ * submod_claims ]  
  
submod_claims = {  
  submod_name_type,  
  * generic_claim_type  
}  
  
submods_claim = (  
  submods: submod_type )
```

4. Data Model

This makes use of the types defined in CDDL Appendix D, Standard Prelude.

4.1. Common CDDL Types

string_or_uri = #6.32(tstr) / tstr; See JSON section below for JSON encoding of string_or_uri

4.2. CDDL for CWT-defined Claims

This section provides CDDL for the claims defined in CWT. It is non-normative as [RFC8392] is the authoritative definition of these claims.

```
cwt_claim = (  
    issuer_claim //  
    subject_claim //  
    audience_claim //  
    expiration_claim //  
    not_before_claim //  
    issued_at_claim //  
    cwt_id_claim  
)  
  
issuer_claim = (  
    issuer: string_or_uri )  
  
subject_claim = (  
    subject: string_or_uri )  
  
audience_claim = (  
    audience: string_or_uri )  
  
expiration_claim = (  
    expiration: time )  
  
not_before_claim = (  
    not_before: time )  
  
issued_at_claim = (  
    issued_at: time )  
  
cwt_id_claim = (  
    cwt_id: bstr )  
  
issuer = 1  
subject = 2  
audience = 3  
expiration = 4  
not_before = 5  
issued_at = 6  
cwt_id = 7
```

4.3. JSON

4.3.1. JSON Labels

```
uuid = "uuid"
origination = "origination"
oemid = "oemid"
security_level = "security_level"
boot_state = "boot_state"
location = "location"
age = "age"
uptime = "uptime"
nested_eat = "nested_eat"
submods = "submods"
```

4.3.2. JSON Interoperability

JSON should be encoded per RFC 8610 Appendix E. In addition, the following CDDL types are encoded in JSON as follows:

- o bstr - must be base64url encoded
- o time - must be encoded as NumericDate as described section 2 of [RFC7519].
- o string_or_uri - must be encoded as StringOrURI as described section 2 of [RFC7519].

4.4. CBOR

4.4.1. Labels

```
uuid = 8
origination = 9
oemid = 10
security_level = 11
boot_state = 12
location = 13
age = 14
uptime = 15
nested_eat = 16
submods = 17
submod_name = 18

latitude 1
longitude 2
altitude 3
accuracy 4
altitude_accuracy 5
heading_claim 6
speed_claim 7
```

4.4.2. CBOR Interoperability

Variations in the CBOR serializations supported in CBOR encoding and decoding are allowed and suggests that CBOR-based protocols specify how this variation is handled. This section specifies what formats MUST be supported in order to achieve interoperability.

The assumption is that the entity is likely to be a constrained device and relying party is likely to be a very capable server. The approach taken is that the entity generating the token can use whatever encoding it wants, specifically encodings that are easier to implement such as indefinite lengths. The relying party receiving the token must support decoding all encodings.

These rules cover all types used in the claims in this document. They also are recommendations for additional claims.

Canonical CBOR encoding, Preferred Serialization and Deterministically Encoded CBOR are explicitly NOT required as they would place an unnecessary burden on the entity implementation, particularly if the entity implementation is implemented in hardware.

- o Integer Encoding (major type 0, 1) - The entity may use any integer encoding allowed by CBOR. The server MUST accept all integer encodings allowed by CBOR.
- o String Encoding (major type 2 and 3) - The entity can use any string encoding allowed by CBOR including indefinite lengths. It may also encode the lengths of strings in any way allowed by CBOR. The server must accept all string encodings.
- o Major type 2, bstr, SHOULD be have tag 21 to indicate conversion to base64url in case that conversion is performed.
- o Map and Array Encoding (major type 4 and 5) - The entity can use any array or map encoding allowed by CBOR including indefinite lengths. Sorting of map keys is not required. Duplicate map keys are not allowed. The server must accept all array and map encodings. The server may reject maps with duplicate map keys.
- o Date and Time - The entity should send dates as tag 1 encoded as 64-bit or 32-bit integers. The entity may not send floating-point dates. The server must support tag 1 epoch-based dates encoded as 64-bit or 32-bit integers. The entity may send tag 0 dates, however tag 1 is preferred. The server must support tag 0 UTC dates.

- o URIs - URIs should be encoded as text strings and marked with tag 32.
- o Floating Point - The entity may use any floating-point encoding. The relying party must support decoding of all types of floating-point.
- o Other types - Use of Other types like bignums, regular expressions and such, SHOULD NOT be used. The server MAY support them but is not required to so interoperability is not guaranteed.

4.5. Collected CDDL

A `generic_claim` is any CBOR map entry or JSON name/value pair.

```
eat_claims = { ; the top-level payload that is signed using COSE or JOSE
               * claim
}
```

```
claim = (
  uuid_claim //
  origination_claim //
  oemid_claim //
  security_level_claim //
  boot_state_claim //
  location_claim //
  age_claim //
  uptime_claim //
  nested_eat_claim //
  cwt_claim //
  generic_claim_type //
)
```

TODO: copy the rest of the CDDL here (wait until the CDDL is more settled so as to avoid copying multiple times)

5. IANA Considerations

5.1. Reuse of CBOR Web Token (CWT) Claims Registry

Claims defined for EAT are compatible with those of CWT so the CWT Claims Registry is re used. No new IANA registry is created. All EAT claims should be registered in the CWT and JWT Claims Registries.

5.1.1.1. Claims Registered by This Document

- o Claim Name: UEID
- o Claim Description: The Universal Entity ID
- o JWT Claim Name: N/A
- o Claim Key: 8
- o Claim Value Type(s): byte string
- o Change Controller: IESG
- o Specification Document(s): *this document*

TODO: add the rest of the claims in here

6. Privacy Considerations

Certain EAT claims can be used to track the owner of an entity and therefore, implementations should consider providing privacy-preserving options dependent on the intended usage of the EAT. Examples would include suppression of location claims for EAT's provided to unauthenticated consumers.

6.1. UEID Privacy Considerations

A UEID is usually not privacy-preserving. Any set of relying parties that receives tokens that happen to be from a single device will be able to know the tokens are all from the same device and be able to track the device. Thus, in many usage situations ueid violates governmental privacy regulation. In other usage situations UEID will not be allowed for certain products like browsers that give privacy for the end user. it will often be the case that tokens will not have a UEID for these reasons.

There are several strategies that can be used to still be able to put UEID's in tokens:

- o The device obtains explicit permission from the user of the device to use the UEID. This may be through a prompt. It may also be through a license agreement. For example, agreements for some online banking and brokerage services might already cover use of a UEID.
- o The UEID is used only in a particular context or particular use case. It is used only by one relying party.

- o The device authenticates the relying party and generates a derived UEID just for that particular relying party. For example, the relying party could prove their identity cryptographically to the device, then the device generates a UEID just for that relying party by hashing a proofed relying party ID with the main device UEID.

Note that some of these privacy preservation strategies result in multiple UEIDs per device. Each UEID is used in a different context, use case or system on the device. However, from the view of the relying party, there is just one UEID and it is still globally universal across manufacturers.

7. Security Considerations

TODO: Perhaps this can be the same as CWT / COSE, but not sure yet because it involves so much entity / device security that those do not.

8. References

8.1. Normative References

[IANA.CWT.Claims]

IANA, "CBOR Web Token (CWT) Claims", n.d.,
<<http://www.iana.org/assignments/cwt>>.

[IANA.JWT.Claims]

IANA, "JSON Web Token (JWT) Claims", n.d.,
<<https://www.iana.org/assignments/jwt>>.

[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>>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.
- [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>>.
- [RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <<https://www.rfc-editor.org/info/rfc8392>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [TIME_T] The Open Group Base Specifications, "Vol. 1: Base Definitions, Issue 7", Section 4.15 'Seconds Since the Epoch', IEEE Std 1003.1, 2013 Edition, 2013, <http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_15>.
- [WGS84] National Imagery and Mapping Agency, "National Imagery and Mapping Agency Technical Report 8350.2, Third Edition", 2000, <<http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf>>.

8.2. Informative References

- [ASN.1] International Telecommunication Union, "Information Technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1994.
- [ECMAScript] "Ecma International, "ECMAScript Language Specification, 5.1 Edition", ECMA Standard 262", June 2011, <<http://www.ecma-international.org/ecma-262/5.1/ECMA-262.pdf>>.
- [IDevID] "IEEE Standard, "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.

[Webauthn]

Worldwide Web Consortium, "Web Authentication: A Web API for accessing scoped credentials", 2016.

Appendix A. Examples

A.1. Very Simple EAT

This is shown in CBOR diagnostic form. Only the payload signed by COSE is shown.

```
{
  / nonce (cti) /           7:h'948f8860d13a463e8e',
  / UEID /                 8:h'0198f50a4ff6c05861c8860d13a638ea4fe2f',
  / boot_state /          12:{true, true, true, true, false}
  / time stamp (iat) /     6:1526542894,
}
```

A.2. Example with Submodules, Nesting and Security Levels

```
{
  / nonce /               7:h'948f8860d13a463e8e',
  / UEID /               8:h'0198f50a4ff6c05861c8860d13a638ea4fe2f',
  / boot_state /         12:{true, true, true, true, false}
  / time stamp (iat) /   6:1526542894,
  / seclevel /           11:3, / secure restricted OS /

  / submods / 17:
  [
    / 1st submod, an Android Application / {
      / submod_name /    18:'Android App "Foo"',
      / seclevel /       11:1, / unrestricted /
      / app data /       -70000:'text string'
    },
    / 2nd submod, A nested EAT from a secure element / {
      / submod_name /    18:'Secure Element EAT',
      / eat /            16:61( 18(
        / an embedded EAT / [ /...COSE_Sign1 bytes with payload.../ ]
      ) )
    }
    / 3rd submod, information about Linux Android / {
      / submod_name/ 18:'Linux Android',
      / seclevel /    11:1, / unrestricted /
      / custom - release / -80000:'8.0.0',
      / custom - version / -80001:'4.9.51+'
    }
  ]
}
```

Appendix B. Changes from Previous Drafts

The following is a list of known changes from the previous drafts. This list is non-authoritative. It is meant to help reviewers see the significant differences.

B.1. From draft-mandyam-rats-eat-00

This is a fairly large change in the orientation of the document, but not new claims have been added.

- o Separate information and data model using CDDL.
- o Say an EAT is a CWT or JWT
- o Use a map to structure the boot_state and location claims

Authors' Addresses

Giridhar Mandyam
Qualcomm Technologies Inc.
5775 Morehouse Drive
San Diego, California
USA

Phone: +1 858 651 7200
EMail: mandyam@qti.qualcomm.com

Laurence Lundblade
Security Theory LLC

EMail: lgl@island-resort.com

Miguel Ballesteros
Qualcomm Technologies Inc.
5775 Morehouse Drive
San Diego, California
USA

Phone: +1 858 651 4299
EMail: mballest@qti.qualcomm.com

Jeremy O'Donoghue
Qualcomm Technologies Inc.
279 Farnborough Road
Farnborough GU14 7LS
United Kingdom

Phone: +44 1252 363189
EMail: jodonogh@qti.qualcomm.com

RATS Working Group
Internet-Draft
Intended status: Informational
Expires: May 6, 2021

M. Richardson
Sandelman Software Works
C. Wallace
Red Hound Software
W. Pan
Huawei Technologies
November 02, 2020

Use cases for Remote Attestation common encodings
draft-richardson-rats-usecases-08

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 6, 2021.

Copyright Notice

Copyright (c) 2020 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. Safety Critical Systems	19
5.14. Trusted Path Routing	19
6. Technology users for RATS.	20
6.1. Trusted Computing Group Remove Integrity Verification	

(TCG-RIV)	20
6.2. Android Keystore system	22
6.3. Fast IDentity Online (FIDO) Alliance	23
7. Examples of Existing Attestation Formats.	24
7.1. Android Keystore	24
7.1.1. TEE	25
7.1.2. Secure Element	37
7.2. Windows 10 TPM	50
7.2.1. Attestation statement	52
7.3. Yubikey	56
7.3.1. Yubikey 4	56
7.3.2. Yubikey 5	60
8. Privacy Considerations.	64
9. Security Considerations	64
10. IANA Considerations	64
11. Acknowledgements	64
12. References	64
12.1. Normative References	64
12.2. Informative References	64
Appendix A. Changes	67
Authors' Addresses	68

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:

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

This document will be expanded to include summaries from:

- o Trusted Computing Group (TCG) Trusted Platform Module (TPM)/Trusted Software Stack (TSS)
- o ARM "Platform Security Architecture"
[I-D.tschofenig-rats-psa-token]
- o Intel SGX attestation [intelsgx]
- o Windows Defender System Guard attestation [windowsdefender]
- o Windows Device Health Attestation [windowshealth]
- o Azure Sphere Attestation [azureattestation]:
<https://azure.microsoft.com/enus/resources/azure-sphere-device-authentication-andattestation-service/en-us/>
- o 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/device-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. Safety Critical Systems

Use case name: Safety Critical Systems

Who will use it: Power plants and other systems that need to assert their current state, but which can not accept any inputs from the outside. The corollary system is a black-box (such as in an aircraft), which needs to log the state of a system, but which can never initiate a handshake.

Message Flow: background check

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: These requirements motivate the creation of the Time Base Unidirectional Attestation (TUDA) [I-D.birkholz-rats-tuda], 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 by the use of a Time Stamp Authority (TSA) which provides Time Stamp Tokens (TST).

5.14. Trusted Path Routing

Use case name: Trusted Path Routing

Who will use it: Service Providers want to offer a trustworthy transport service to Government, Military, Financial, and Medical end-users.

Message Flow: background check model for a centralized controller based alternative, and passport model for a router/switch distributed alternative.

Attester: Routers/switches

Relying Party: Network Controllers and Peer Routers/Switches

Claims used as evidence: TPM Quotes, log entries passed into TPM PCRs, trustworthiness levels appraised by Verifiers, and included in passports.

Description: There are end-users who believe encryption technologies like IPsec alone are insufficient to protect the confidentiality of their highly sensitive traffic flows. These end-users want their sensitive flows to be forwarded across just those network devices currently appraised as trustworthy by the TCG-RIV use case.

[I-D.voit-rats-trusted-path-routing] discusses two alternatives for exchanging traffic with end-user customer identified "sensitive subnets". Traffic going to and from these subnets will transit a path where the IP layer and above are only interpretable by those network devices recently evaluated as trustworthy.

These two alternatives are:

Centralized Trusted Path Routing: For sensitive subnets, trusted end-to-end paths are pre-assigned through a network provider domain. Along these paths, attestation evidence of potentially transited components has been assessed. Each path is guaranteed to only include devices meeting the needs of a formally defined trustworthiness level.

Distributed Trusted Path Routing: Through the exchange of attestation evidence between peering network devices, a trusted topology is established and maintained. Only devices meeting the needs of a formally defined trustworthiness level are included as members of this topology. Traffic exchanged with sensitive subnets is forwarded into this topology.

6. Technology users for RATS.

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:

- o "non-privacy-preserving applications (i.e., networking, Industrial IoT)",
- o the firmware is provided by the device manufacturer
- o there is a manufacturer installed hardware root of trust (such as a TPM and boot ROM)

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:

- o identify a remote box in a way that's hard to spoof
- o 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 are topics 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:

- o Device Manufacturer

- o Attester ("device under attestation")
- o Verifier (Network Management Station)
- o "Explicit Attestation" is the TCG term for a static (platform) attestation
- o "Implicit Attestation" is the TCG term for a session attestation
- o Reference Integrity Measurements (RIM), which are signed by device manufacturer and integrated into firmware.
- o Quotes: measured values (having been signed), and RIMs
- o Reference Integrity Values (RIV)
- o devices have a Initial Attestation Key (IAK), which is provisioned at the same time as the IDevID [IEEE 802-1AR]
- o 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 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 be 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:

- o "relying party" validates a claim
- o "relying party application" makes FIDO Authn calls
- o "browser" provides the Web Authentication JS API
- o "platform" is the base system
- o "internal authenticator" is some credential built-in to the device
- o "external authenticator" may be connected by USB, bluetooth, wifi, and may be a 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 provides 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 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.DIGE
ST_SHA256)
        .setBlockModes(KeyProperties.BLOCK_MODE_CBC, KeyPropertie
s.BLOCK_MODE_GCM)
        .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_R
SA_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.DIGE
ST_SHA256, KeyProperties.DIGEST_SHA384, KeyProperties.DIGEST_SHA512)
        .setBlockModes(KeyProperties.BLOCK_MODE_CBC, KeyPropertie
s.BLOCK_MODE_CTR, KeyProperties.BLOCK_MODE_GCM)
        .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_R
SA_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, idAt
testationFlags);

```

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)

```

29    0:      NULL
      :      }
31    27:    SEQUENCE {
33    25:      SET {
35    23:        SEQUENCE {
37    3:          OBJECT IDENTIFIER serialNumber (2 5 4 5)
42    16:          PrintableString 'c6047571d8f0d17c'
      :      }
      :      }
      :      }
60    32:    SEQUENCE {
62    13:      UTCTime 01/01/1970 00:00:00 GMT
77    15:      GeneralizedTime 07/02/2106 06:28:15 GMT
      :      }
94    31:    SEQUENCE {
96    29:      SET {
98    27:        SEQUENCE {
100   3:          OBJECT IDENTIFIER commonName (2 5 4 3)
105   20:          UTF8String 'Android Keystore Key'
      :      }
      :      }
      :      }
127  290:    SEQUENCE {
131   13:      SEQUENCE {
133   9:        OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
144   0:        NULL
      :      }
146  271:    BIT STRING, encapsulates {
151  266:      SEQUENCE {
155  257:        INTEGER
      :          00 B5 3A 83 61 A2 85 CC D2 D6 25 7F 07 0B B4 A0
      :          F6 FE 05 01 C9 55 CB 0D 18 D2 C6 79 BA 82 12 67
      :          75 8D 5B F3 24 D3 F8 EA 99 82 7D 1F 5E CD 77 D6
      :          99 11 13 FF 18 C9 3D 4D 01 C5 8E E9 04 E7 17 E2
      :          88 12 2B B9 A1 77 2F C2 4F 57 78 98 4E E3 DE 7A
      :          1B 18 BE D3 ED C9 59 A0 24 50 E1 FA AC 81 B6 DA
      :          80 B0 BD 48 AD 26 9C 4A 4E CE 54 17 58 C1 F4 F8
      :          7F 3C 5D 8F C8 2C 2A 7B 18 95 B3 D4 E0 3A C8 9D
      :          [ Another 129 bytes skipped ]
416   3:        INTEGER 65537
      :      }
      :      }
      :      }
421  347:    [3] {
425  343:      SEQUENCE {
429   14:        SEQUENCE {
431   3:          OBJECT IDENTIFIER keyUsage (2 5 29 15)
436   1:          BOOLEAN TRUE

```



```

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 { -- KeyDescription
469  1:      INTEGER 2 -- attestationVersion (KM3)
472  1:      ENUMERATED 1 -- attestationSecurityLevel (TrustedE
nv.)
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 { -- softwareEnforced
496  8:      [701] { -- creationDateTime
500  6:      INTEGER 01 64 47 2A 4B 64
      :      }
508 28:      [709] { -- attestationApplicationId
512 26:      OCTET STRING, encapsulates {
514 24:      SEQUENCE { -- AttestationApplicationId
516 20:      SET { -- package_infos
518 18:      SEQUENCE { -- AttestationPackageInfo
520 13:      OCTET STRING 'AndroidSystem' -- package_nam
e
535  1:      INTEGER 1 -- version
      :      }
      :      }
538  0:      SET {} -- signature_digests
      :      }
      :      }
      :      }
      :      }
540 229:    SEQUENCE { -- hardwareEnforced
543 14:      [1] { -- purpose
545 12:      SET {
547  1:      INTEGER 0 -- KeyPurpose.ENCRYPT
550  1:      INTEGER 1 -- KeyPurpose.DECRYPT
553  1:      INTEGER 2 -- KeyPurpose.SIGN
556  1:      INTEGER 3 -- KeyPurpose.VERIFY
      :      }
      :      }
559  3:      [2] { -- algorithm
561  1:      INTEGER 1 -- Algorithm.RSA
      :      }
564  4:      [3] { -- keySize
566  2:      INTEGER 2048

```

```

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

```

```

:      }
716  9:      [712] {      -- attestationIdProduct
720  7:      OCTET STRING 'walleye'
:      }
729  14:     [713] {      -- attestationIdSerial
733  12:     OCTET STRING 'HT83K1A03849'
:     }
747  8:      [716] {      -- attestationIdManufacturer
751  6:      OCTET STRING 'Google'
:      }
759  9:      [717] {      -- attestationIdModel
763  7:      OCTET STRING 'Pixel 2'
:      }
:    }
:  }
: }
: }
: }
: }
: }
772  13: SEQUENCE {
774  9:   OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
785  0:   NULL
:   }
787  385: BIT STRING
:      05 41 B9 13 11 53 93 A2 02 62 1F 15 35 8E D9 7C
:      A1 D5 2E ED 13 AC 24 26 B2 A1 2F EE B4 0C 4D 71
:      DC 9F 55 EC A1 F6 64 62 F2 73 A8 7E FC 48 63 29
:      1E F5 0D 48 F3 73 43 0C 00 E0 D4 07 86 A6 A4 38
:      0E A8 47 0F 27 01 01 31 52 F6 62 8A 4B 80 BE 72
:      FB 02 E7 56 84 CA CA 4D C3 6C 7C B2 BA C7 D7 9B
:      C5 9D 90 65 4E F5 54 8F 25 CC 11 7F 8E 77 10 6A
:      6E 9F 80 89 48 8B 1D 51 AA 3B B7 C5 24 3C 28 B1
:      [ Another 256 bytes skipped ]
:    }
0 1304: SEQUENCE {
4 768:   SEQUENCE {
8 3:     [0] {
10 1:      INTEGER 2
:      }
13 10:     INTEGER 10 34 53 32 94 08 68 79 38 72
25 13:     SEQUENCE {
27 9:      OBJECT IDENTIFIER
:      sha256WithRSAEncryption (1 2 840 113549 1 1 11)
38 0:      NULL
:      }
40 27:     SEQUENCE {
42 25:      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 86 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 20:       [0]
      :         0E 55 6F 46 F5 3B 77 67 E1 B9 73 DC 55 E6 AE EA
      :         B4 FD 27 DD
      :       }
      :     }
      :   }
622 12: SEQUENCE {
624 3:   OBJECT IDENTIFIER basicConstraints (2 5 29 19)
629 1:   BOOLEAN TRUE
632 2:   OCTET STRING, encapsulates {
634 0:     SEQUENCE {}
      :   }
      : }
636 14: SEQUENCE {
638 3:   OBJECT IDENTIFIER keyUsage (2 5 29 15)
643 1:   BOOLEAN TRUE
646 4:   OCTET STRING, encapsulates {
648 2:     BIT STRING 7 unused bits
      :     '1'B (bit 0)
      :   }
      : }
652 36: SEQUENCE {
654 3:   OBJECT IDENTIFIER nameConstraints (2 5 29 30)
659 29:   OCTET STRING, encapsulates {
661 27:     SEQUENCE {
663 25:       [0] {
665 23:         SEQUENCE {
667 21:           [2] 'invalid;email:invalid'
      :         }
      :       }
      :     }
      :   }
      : }
690 84: SEQUENCE {
692 3:   OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
697 77:   OCTET STRING, encapsulates {
699 75:     SEQUENCE {
701 73:       SEQUENCE {
703 71:         [0] {
705 69:           [0] {
707 67:             [6]
      :             'https://android.googleapis.com/attestation/crl/1'
      :             '0345332940868793872'

```

```

:      }
:    }
:  }
: }
: }
: }
: }
: }
: }
: }
776 13: SEQUENCE {
778 9:   OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
789 0:   NULL
: }
791 513: 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 ]
: }
0 1393: SEQUENCE {
4 857: SEQUENCE {
8 3:   [0] {
10 1:   INTEGER 2
: }
13 10:   INTEGER 03 88 26 67 60 65 89 96 85 74
25 13:   SEQUENCE {
27 9:   OBJECT IDENTIFIER
:       sha256WithRSAEncryption (1 2 840 113549 1 1 11)
38 0:   NULL
: }
40 27:   SEQUENCE {
42 25:   SET {
44 23:     SEQUENCE {
46 3:     OBJECT IDENTIFIER serialNumber (2 5 4 5)
51 16:     PrintableString 'f92009e853b6b045'
: }
: }
: }
69 30:   SEQUENCE {
71 13:     UTCTime 26/05/2016 17:01:32 GMT
86 13:     UTCTime 24/05/2026 17:01:32 GMT
: }
101 27: SEQUENCE {

```

```

103 25:      SET {
105 23:      SEQUENCE {
107 3:        OBJECT IDENTIFIER serialNumber (2 5 4 5)
112 16:        PrintableString '87f4514475ba0a2b'
      :      }
      :    }
      :  }
130 546:    SEQUENCE {
134 13:      SEQUENCE {
136 9:        OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
147 0:        NULL
      :      }
149 527:    BIT STRING, encapsulates {
154 522:      SEQUENCE {
158 513:        INTEGER
      :          00 D2 60 D6 45 85 E3 E2 23 79 5A DA 45 57 A7 D8
      :          5B AF BD 9A 37 CB FA 97 C0 65 44 9D 3A C6 47 F6
      :          0D 0B A2 74 12 CA F7 4B B9 5F FB B4 EC 5A 2B D0
      :          16 01 DE BE E2 FE D2 76 0D 75 C4 B1 6A CB 3A 67
      :          07 21 E0 D5 19 68 C8 1B 01 A2 24 02 FE AD 40 D6
      :          A7 98 16 0F A2 98 2E A7 AD 75 34 84 6F F8 CF 8A
      :          A1 0E 90 33 40 9E D0 86 26 57 71 CE FF CF 52 E1
      :          F0 F9 2B 7E 68 62 03 D8 FD FD 02 53 03 19 AC 28
      :          [ Another 385 bytes skipped ]
675 3:        INTEGER 65537
      :      }
      :    }
      :  }
680 182:    [3] {
683 179:      SEQUENCE {
686 29:        SEQUENCE {
688 3:          OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
693 22:          OCTET STRING, encapsulates {
695 20:            OCTET STRING
      :              0E 55 6F 46 F5 3B 77 67 E1 B9 73 DC 55 E6 AE EA
      :              B4 FD 27 DD
      :            }
      :          }
717 31:        SEQUENCE {
719 3:          OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
724 24:          OCTET STRING, encapsulates {
726 22:            SEQUENCE {
728 20:              [0]
      :                36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C
      :                C9 EA 4F 12
      :              }
      :            }
      :          }
      :    }

```

```

750      15:      SEQUENCE {
752      3:      OBJECT IDENTIFIER basicConstraints (2 5 29 19)
757      1:      BOOLEAN TRUE
760      5:      OCTET STRING, encapsulates {
762      3:      SEQUENCE {
764      1:      BOOLEAN TRUE
      :      }
      :      }
      :      }
767      14:     SEQUENCE {
769      3:      OBJECT IDENTIFIER keyUsage (2 5 29 15)
774      1:      BOOLEAN TRUE
777      4:      OCTET STRING, encapsulates {
779      2:      BIT STRING 1 unused bit
      :      '1100001'B
      :      }
      :      }
783      80:     SEQUENCE {
785      3:      OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
790      73:      OCTET STRING, encapsulates {
792      71:      SEQUENCE {
794      69:      SEQUENCE {
796      67:      [0] {
798      65:      [0] {
800      63:      [6]
      :      'https://android.googleapis.com/attestation/crl/E'
      :      '8FA196314D2FA18'
      :      }
      :      }
      :      }
      :      }
      :      }
      :      }
      :      }
      :      }
      :      }
865      13:     SEQUENCE {
867      9:      OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
878      0:      NULL
      :      }
880      513:    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 13:     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 75 31 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
      :      }
      :    }
      :  [3] {
679  166:      SEQUENCE {
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 {

```

```

793 53: SEQUENCE {
795 51:   [0] {
797 49:     [0] {
799 47:       [6]
      :       'https://android.googleapis.com/attestation/crl/'
      :     }
      :   }
      : }
      : }
      : }
      : }
      : }
      : }
      : }
848 13: SEQUENCE {
850 9:   OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
861 0:   NULL
      : }
863 513: BIT STRING
      :   20 C8 C3 8D 4B DC A9 57 1B 46 8C 89 2F FF 72 AA
      :   C6 F8 44 A1 1D 41 A8 F0 73 6C C3 7D 16 D6 42 6D
      :   8E 7E 94 07 04 4C EA 39 E6 8B 07 C1 3D BF 15 03
      :   DD 5C 85 BD AF B2 C0 2D 5F 6C DB 4E FA 81 27 DF
      :   8B 04 F1 82 77 0F C4 E7 74 5B 7F CE AA 87 12 9A
      :   88 01 CE 8E 9B C0 CB 96 37 9B 4D 26 A8 2D 30 FD
      :   9C 2F 8E ED 6D C1 BE 2F 84 B6 89 E4 D9 14 25 8B
      :   14 4B BA E6 24 A1 C7 06 71 13 2E 2F 06 16 A8 84
      :   [ Another 384 bytes skipped ]
      : }

```

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 {

```

```

53      3:      [0] {
55      1:      INTEGER 2
          :      }
58      1:      INTEGER 1
61      13:     SEQUENCE {
63      9:      OBJECT IDENTIFIER
          :      sha256WithRSAEncryption (1 2 840 113549 1 1 11)
74      0:      NULL
          :      }
76      47:     SEQUENCE {
78      25:     SET {
80      23:     SEQUENCE {
82      3:      OBJECT IDENTIFIER serialNumber (2 5 4 5)
87      16:     PrintableString '90e8da3cadfc7820'
          :      }
          :      }
105     18:     SET {
107     16:     SEQUENCE {
109     3:      OBJECT IDENTIFIER title (2 5 4 12)
114     9:      UTF8String 'StrongBox'
          :      }
          :      }
          :      }
125     30:     SEQUENCE {
127     13:     UTCTime 01/01/1970 00:00:00 GMT
142     13:     UTCTime 23/05/2028 23:59:59 GMT
          :      }
157     31:     SEQUENCE {
159     29:     SET {
161     27:     SEQUENCE {
163     3:      OBJECT IDENTIFIER commonName (2 5 4 3)
168     20:     UTF8String 'Android Keystore Key'
          :      }
          :      }
          :      }
190     290:    SEQUENCE {
194     13:     SEQUENCE {
196     9:      OBJECT IDENTIFIER
          :      rsaEncryption (1 2 840 113549 1 1 1)
207     0:      NULL
          :      }
209     271:    BIT STRING, encapsulates {
214     266:    SEQUENCE {
218     257:    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

```

```

:      3B 2B 83 0F B4 9C 3D C7 13 0B 9C 59 2F 1A 38 CE
:      A5 1D 95 A7 3C EE 70 6A CF 41 FF 55 3F E0 9C 69
:      E5 A0 C1 19 EF 40 E9 40 FC 74 D3 3B 96 D9 0E C1
:      C3 9D 14 10 0C A6 95 19 49 88 F4 AB 74 FC 86 A6
:      [ Another 129 bytes skipped ]
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
:              '1'B (bit 0)
:            }
:          }
508   375:      SEQUENCE {
512   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 (Stro
ngBox)
538   1:            INTEGER 4
541   1:            ENUMERATED 2    -- attestationSecurityLevel (Stro
ngBox)
544   9:          OCTET STRING 'challenge'
555   0:          OCTET STRING
:              Error: Object has zero length.
557   53:        SEQUENCE {
559   2:          [509] {
563   0:            NULL
:          }
565   11:        [701] {
569   9:          INTEGER 00 FF FF FF FF FF E5 99 78
:          }
580   28:        [709] {
584   26:          OCTET STRING, encapsulates {
586   24:            SEQUENCE {
588   20:              SET {
590   18:                SEQUENCE {
592   13:                  OCTET STRING 'AndroidSystem'
607   1:                  INTEGER 1
:                }
:              }
610   0:            SET {}
:          }

```

```

      :
      :
      :
612 271: SEQUENCE {
616 14: [1] {
618 12:   SET {
620 1:     INTEGER 0
623 1:     INTEGER 1
626 1:     INTEGER 2
629 1:     INTEGER 3
      :   }
      : }
632 3: [2] {
634 1:   INTEGER 1
      : }
637 4: [3] {
639 2:   INTEGER 2048
      : }
643 8: [4] {
645 6:   SET {
647 1:     INTEGER 2
650 1:     INTEGER 32
      :   }
      : }
653 8: [5] {
655 6:   SET {
657 1:     INTEGER 0
660 1:     INTEGER 4
      :   }
      : }
663 14: [6] {
665 12:   SET {
667 1:     INTEGER 2
670 1:     INTEGER 3
673 1:     INTEGER 4
676 1:     INTEGER 5
      :   }
      : }
679 2: [503] {
683 0:   NULL
      : }
685 3: [702] {
689 1:   INTEGER 0
      : }
692 76: [704] {
696 74:   SEQUENCE {
698 32:     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
732    1:      BOOLEAN TRUE
735    1:      ENUMERATED 0
738    32:     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
      :      }
      :
772    3:      [705] {
776    1:      INTEGER 0
      :      }
779    5:      [706] {
783    3:      INTEGER 201903
      :      }
788    8:      [710] {
792    6:      OCTET STRING 'google'
      :      }
800    10:     [711] {
804    8:      OCTET STRING 'blueline'
      :      }
814    10:     [712] {
818    8:      OCTET STRING 'blueline'
      :      }
828    11:     [713] {
832    9:      OCTET STRING '8A2X0KLUU'
      :      }
843    8:      [716] {
847    6:      OCTET STRING 'Google'
      :      }
855    9:      [717] {
859    7:      OCTET STRING 'Pixel 3'
      :      }
868    6:      [718] {
872    4:      INTEGER 20180905
      :      }
878    5:      [719] {
882    3:      INTEGER 201903
      :      }
      :      }
      :      }
      :      }
      :      }
      :      }
      :      }
887    13:     SEQUENCE {
889    9:      OBJECT IDENTIFIER
      :      sha256WithRSAEncryption (1 2 840 113549 1 1 11)

```

```

900    0:          NULL
      :          }
902  257:        BIT STRING
      :          83 EA 59 8D BE 37 4A D5 C0 FC F8 FB AC 8B 72 1E
      :          A5 C2 3B 0C C0 04 1B C0 5A 18 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 1A 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'
      :              }
      :            }

```



```

:
1313 18:      SET {
1315 16:      SEQUENCE {
1317 3:        OBJECT IDENTIFIER title (2 5 4 12)
1322 9:        UTF8String 'StrongBox'
:        }
:      }
:    }
1333 290:    SEQUENCE {
1337 13:    SEQUENCE {
1339 9:      OBJECT IDENTIFIER
:      rsaEncryption (1 2 840 113549 1 1 1)
1350 0:      NULL
:    }
1352 271:    BIT STRING, encapsulates {
1357 266:    SEQUENCE {
1361 257:    INTEGER
:      00 A5 09 D4 09 D2 30 19 36 34 71 FD 7D 41 89 E6
:      2C A5 9D 10 1B 4F 40 6A B0 5F 56 34 16 E6 EB D7
:      F3 E9 C5 DC 20 F3 86 D1 77 19 D7 15 1F E7 EC 62
:      DC 0A BC 64 E9 18 52 B0 AA B8 FF 58 6A E0 0F B8
:      56 AF 77 D3 CE 3C DC 48 52 DD B2 86 0D 76 17 7C
:      FD EE B4 E6 6E 0A 08 9E 06 CA 0F EC 4B B0 7C AF
:      EA 82 27 A8 C9 A7 63 DA 89 F6 30 BA 3C 3A E5 C6
:      EF 11 06 42 8A 2E FE 19 BE F2 C7 3B 34 16 B2 E2
:      [ Another 129 bytes skipped ]
1622 3:    INTEGER 65537
:    }
:  }
: }
1627 186: [3] {
1630 183: SEQUENCE {
1633 29: SEQUENCE {
1635 3:   OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
1640 22:   OCTET STRING, encapsulates {
1642 20:   OCTET STRING
:     77 A4 AD DF 1D 29 89 CA 92 E3 BA DE 27 3C 70 DF
:     36 03 7C 0C
:   }
: }
1664 31: SEQUENCE {
1666 3:   OBJECT IDENTIFIER
:   authorityKeyIdentifier (2 5 29 35)
1671 24:   OCTET STRING, encapsulates {
1673 22:   SEQUENCE {
1675 20:   [0]
:     1B 17 70 C6 97 DC 84 54 75 7C 3C 98 5C E6 1D 1D
:     08 59 5D 53

```

```

    :
    :
    :
    }
}
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 {
SEQUENCE {
[0] {
[0] {
[6]
'https://android.googleapis.com/attestation/crl/1'
'7102468407102977850'
}
}
}
}
}
}
}
}
}
}
SEQUENCE {
OBJECT IDENTIFIER
sha256WithRSAEncryption (1 2 840 113549 1 1 11)
NULL
}
BIT STRING
13 22 DA F2 92 93 CE C0 9F 70 40 C9 DA 85 6B 61
6F 8F BE E0 A4 04 55 C1 63 84 61 37 F5 4B 71 6D
62 AA 6F BF 6C E8 48 03 AD 28 85 21 9E 3C 1C 91

```

```

      :      48 EE 65 28 65 70 D0 BD 5B CC DB CE B1 F5 B5 C3
      :      CA 7A A9 C8 8A 68 12 8A CA 6A 85 A6 BC DA 36 E9
      :      B9 94 35 82 5B CA BC B6 9F 83 03 7F 21 6C EE 82
      :      C1 3F BD C1 41 4B DD 1A 6F 6C AF 4A 52 FC 19 19
      :      17 AC 29 0C 5E D7 57 90 D5 B1 2B 36 29 1F 45 33
      :      [ Another 384 bytes skipped ]
      :    }
2348 1376: SEQUENCE {
2352 840:   SEQUENCE {
2356   3:     [0] {
2358   1:       INTEGER 2
      :     }
2361   9:     INTEGER 00 E8 FA 19 63 14 D2 FA 18
2372  13:   SEQUENCE {
2374   9:     OBJECT IDENTIFIER
      :       sha256WithRSAEncryption (1 2 840 113549 1 1 11)
2385   0:     NULL
      :   }
2387  27: SEQUENCE {
2389  25:   SET {
2391  23:     SEQUENCE {
2393   3:       OBJECT IDENTIFIER serialNumber (2 5 4 5)
2398  16:       PrintableString 'f92009e853b6b045'
      :     }
      :   }
      : }
2416  30: SEQUENCE {
2418  13:   UTCTime 26/05/2016 16:28:52 GMT
2433  13:   UTCTime 24/05/2026 16:28:52 GMT
      : }
2448  27: SEQUENCE {
2450  25:   SET {
2452  23:     SEQUENCE {
2454   3:       OBJECT IDENTIFIER serialNumber (2 5 4 5)
2459  16:       PrintableString 'f92009e853b6b045'
      :     }
      :   }
      : }
2477 546: SEQUENCE {
2481  13:   SEQUENCE {
2483   9:     OBJECT IDENTIFIER
      :       rsaEncryption (1 2 840 113549 1 1 1)
2494   0:     NULL
      :   }
2496 527: BIT STRING, encapsulates {
2501 522:   SEQUENCE {
2505 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 75 31 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 ]
3022  3:      INTEGER 65537
      :      }
      :    }
      :  [3] {
3027 166:      SEQUENCE {
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: SEQUENCE {  

3143           51: [0] {  

3145             49: [0] {  

3147               47: [6]  

                 : 'https://android.googleapis.com/attestation/crl/'  

                 : }  

                 : }  

                 : }  

                 : }  

                 : }  

                 : }  

                 : }  

                 : }  

                 : }  

                 : }  

3196   13: SEQUENCE {  

3198     9: OBJECT IDENTIFIER  

        : sha256WithRSAEncryption (1 2 840 113549 1 1 11)  

3209     0: NULL  

        : }  

3211   513: BIT STRING  

        : 20 C8 C3 8D 4B DC A9 57 1B 46 8C 89 2F FF 72 AA  

        : C6 F8 44 A1 1D 41 A8 F0 73 6C C3 7D 16 D6 42 6D  

        : 8E 7E 94 07 04 4C EA 39 E6 8B 07 C1 3D BF 15 03  

        : DD 5C 85 BD AF B2 C0 2D 5F 6C DB 4E FA 81 27 DF  

        : 8B 04 F1 82 77 0F C4 E7 74 5B 7F CE AA 87 12 9A  

        : 88 01 CE 8E 9B C0 CB 96 37 9B 4D 26 A8 2D 30 FD  

        : 9C 2F 8E ED 6D C1 BE 2F 84 B6 89 E4 D9 14 25 8B  

        : 14 4B BA E6 24 A1 C7 06 71 13 2E 2F 06 16 A8 84  

        :      [ Another 384 bytes skipped ]  

        : }  

3728 1413: SEQUENCE {  

3732   877: SEQUENCE {  

3736     3: [0] {  

3738     1: INTEGER 2  

        : }  

3741   10: INTEGER 03 88 26 67 60 65 89 96 85 99  

3753   13: SEQUENCE {  

3755     9: OBJECT IDENTIFIER  

        : sha256WithRSAEncryption (1 2 840 113549 1 1 11)  

3766     0: NULL  

        : }  

3768   27: SEQUENCE {
```

```

3770 25:          SET {
3772 23:            SEQUENCE {
3774 3:              OBJECT IDENTIFIER serialNumber (2 5 4 5)
3779 16:              PrintableString 'f92009e853b6b045'
          :          }
          :        }
          :      }
3797 30:    SEQUENCE {
3799 13:      UTCTime 20/06/2018 22:47:35 GMT
3814 13:      UTCTime 17/06/2028 22:47:35 GMT
          :    }
3829 47:    SEQUENCE {
3831 25:      SET {
3833 23:        SEQUENCE {
3835 3:          OBJECT IDENTIFIER serialNumber (2 5 4 5)
3840 16:          PrintableString 'ccd18b9b608d658e'
          :        }
          :      }
3858 18:    SET {
3860 16:      SEQUENCE {
3862 3:        OBJECT IDENTIFIER title (2 5 4 12)
3867 9:        UTF8String 'StrongBox'
          :      }
          :    }
          :  }
3878 546: SEQUENCE {
3882 13:   SEQUENCE {
3884 9:    OBJECT IDENTIFIER
          :      rsaEncryption (1 2 840 113549 1 1 1)
3895 0:    NULL
          :  }
3897 527: BIT STRING, encapsulates {
3902 522:   SEQUENCE {
3906 513:    INTEGER
          :      00 E8 22 0B F1 72 A6 01 63 D3 3C 44 9D DB 7A 87
          :      D6 3D 6F 6D 92 B7 C9 4A 70 96 5D 29 7A 8E 96 3E
          :      FE F3 10 53 B2 19 A5 BF 6E 54 AD D0 0A A2 8E 54
          :      E0 D4 B4 2E A6 E0 D4 30 F8 5A 47 CC 09 00 56 45
          :      BE DA 5A 84 59 90 18 CE 29 6C 8E 9E E6 90 98 BD
          :      D4 D8 F8 38 82 90 C9 79 DB 31 D3 7A A1 CA BA 6A
          :      8B 9D 15 91 E2 6C 41 A3 2B 25 DA 4F E4 B3 14 E5
          :      4B EC B7 89 06 44 18 67 C1 4C 03 35 18 D8 FD 7D
          :      [ Another 385 bytes skipped ]
4423 3:    INTEGER 65537
          :  }
          : }
          : }
4428 182: [3] {

```

```

4431 179:      SEQUENCE {
4434 29:          SEQUENCE {
4436 3:              OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
4441 22:              OCTET STRING, encapsulates {
4443 20:                  OCTET STRING
:                  1B 17 70 C6 97 DC 84 54 75 7C 3C 98 5C E6 1D 1D
:                  08 59 5D 53
:                  }
:              }
4465 31:      SEQUENCE {
4467 3:          OBJECT IDENTIFIER
:          authorityKeyIdentifier (2 5 29 35)
4472 24:          OCTET STRING, encapsulates {
4474 22:              SEQUENCE {
4476 20:                  [0]
:                  36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C
:                  C9 EA 4F 12
:                  }
:              }
:          }
4498 15:      SEQUENCE {
4500 3:          OBJECT IDENTIFIER basicConstraints (2 5 29 19)
4505 1:          BOOLEAN TRUE
4508 5:          OCTET STRING, encapsulates {
4510 3:              SEQUENCE {
4512 1:                  BOOLEAN TRUE
:                  }
:              }
:          }
4515 14:      SEQUENCE {
4517 3:          OBJECT IDENTIFIER keyUsage (2 5 29 15)
4522 1:          BOOLEAN TRUE
4525 4:          OCTET STRING, encapsulates {
4527 2:              BIT STRING 2 unused bits
:              '100000'B (bit 5)
:              }
:          }
4531 80:      SEQUENCE {
4533 3:          OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
4538 73:          OCTET STRING, encapsulates {
4540 71:              SEQUENCE {
4542 69:                  SEQUENCE {
4544 67:                      [0] {
4546 65:                          [0] {
4548 63:                              [6]
:                              'https://android.googleapis.com/attestation/crl/8'
:                              'F6734C9FA504789'
:                              }

```

[illegible]

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:

- * ContentInfo
 - * SignedData
 - * PKIData
 - * Empty controlSequence
 - * One TaggestRequest
 - * PKCS 10
 - * Basic request details along with encrypted attestation extension
 - * Empty cmsSequence
 - * Empty otherMsgSequence
 - * Certificates bag with two certs (one of which is revoked)

7.2.1. Attestation statement

This section provides an annotation attestation statement as extracted from an encrypted attestation extension. The structure of the attestation statement is defined here:
<https://msdn.microsoft.com/en-us/library/dn408990.aspx>.

```

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 00 01 00 00
:          00 03 00 00 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 14 DD AF 30 72 A6 E3 89 4D BF 7A 54 26
:          36 2F 10 D6 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 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

```

```

: A8 CE C0 64 71 F6 E3 81 F7 9D 79 E5 73 7B F3 A4
: 6E 65 8D 72 B4 0A 3E 5E 70 5F AB 2B 89 B9 5E 65
: 44 BF 44 7B FB 2E 29 39 64 36 85 63 46 62 AF 25
: A5 8B 19 30 AF 50 43 50 4D 38 00 00 00 02 00 00
: 00 03 00 00 00 38 01 00 00 E0 00 00 00 00 00 00
: 00 00 00 00 00 B0 00 00 00 00 00 00 00 00 00
: 00 00 00 00 00 00 00 00 00 00 00 00 00 01 36 00
: 01 00 0B 00 06 00 72 00 20 9D FF CB F3 6C 38 3A
: E6 99 FB 98 68 DC 6D CB 89 D7 15 38 84 BE 28 03
: 92 2C 12 41 58 BF AD 22 AE 00 10 00 10 08 00 00
: 00 00 00 01 00 9B B1 27 B7 E3 5D 0C 10 74 52 1B
: 60 59 96 5E B6 08 D4 76 26 17 B5 92 49 39 34 CD
: A4 2D 4D C9 3E 50 05 2E D8 9E 22 37 E2 05 D2 7F
: 3B 3E 4D 9F E0 E0 31 52 74 A0 D5 18 BE F1 9F 79
: 48 D6 24 69 35 3C D4 1F 55 73 75 ED 83 D6 3A E3
: 63 77 A6 5B 92 97 86 13 7C 69 3B DE AA E5 0E 9A
: 39 CF 53 DF 4C 7A E0 3C A3 EC 29 DA 18 5F 86 E6
: 22 D9 2C A3 8E D8 E2 3E 80 9C 69 52 FA 1E 90 3F
: BA 09 04 D0 91 6A 27 2B 44 8C FF E8 DE FF BD B9
: CE DD 95 67 70 FD 94 E5 3A E6 E4 EA 01 A5 AC 4A
: 79 5C 88 4D 07 43 C7 C0 B8 95 3E 7C 72 90 CD 35
: 99 B3 32 8A C7 8C 90 63 E3 46 88 62 35 A4 5B 54
: F1 E8 61 0E CF 85 B4 41 6F 06 94 B6 BA 6F 4B CE
: F7 8A 18 6C 5E 9A 6B 65 C3 F5 58 ED 7D 6A 3A E6
: 24 B6 21 6F 8C EE 1C 21 60 9E 2F 86 22 D2 2B 8F
: E0 3B 12 AC 6B F5 FF 54 C6 E8 D4 3C 2E D3 B6 8E
: 7A 30 36 29 3D 00 DE 00 20 13 F5 31 2B 87 50 19
: D3 95 1F F2 B6 00 95 5B 0A E2 54 7A A0 CF 6A 2C
: F5 4F AD 77 C6 D5 4F 52 CB 00 10 3B 41 34 BF D4
: FC 8B BE 87 14 47 81 4E 5C 5C 23 73 44 AF D6 56
: 6F A6 6E BE E7 63 9C 43 53 C4 3C 26 33 B6 AD 75
: 36 AC 91 98 C1 FF E3 B2 AF E6 3F 14 C0 2E 65 D7
: C1 AD F6 22 D9 59 96 B6 70 8C 30 2F DE 76 1B EB
: 9D 56 C1 77 F8 1D 38 5C 7D 13 9C FD 1E 3E 00 1B
: 5A 74 C4 8E 49 2B 0B B5 C5 0E E3 A7 2C 92 E2 96
: 1E 9D C8 43 02 2F 8F F8 6E 66 4A FA D8 56 57 59
: 48 A4 D5 B7 7F 49 52 CA FA 11 E4 AF 27 E7 64 21
: 76 79 9B 8A A3 1A A6 FA A1 03 3E CC CD 41 26 3C
: 0D 3C DC 81 21 21 DE 92 4D 2A EF 66 DE D6 77 FE
: 41 0C 5D 44 1A D0 C4 D7 8B EA 6D DE 01 EE 97 DB
: 61 0F FD 62 59 00 00 00 06 00 20 8F CD 21 69 AB
: 92 69 4E 0C 63 3F 1A B7 72 84 2B 82 41 BB C2 02
: 88 98 1F C7 AC 1E DD C1 FD DB 0E 00 20 E5 29 F5
: D6 11 28 72 95 4E 8E D6 60 51 17 B7 57 E2 37 C6
: E1 95 13 A9 49 FE E1 F2 04 C4 58 02 3A 00 20 AF
: 2C A5 69 69 9C 43 6A 21 00 6F 1C B8 A2 75 6C 98
: BC 1C 76 5A 35 59 C5 FE 1C 3F 5E 72 28 A7 E7 00
: 20 C4 13 A8 47 B1 11 12 B1 CB DD D4 EC A4 DA AA

```

```

:          15 A1 85 2C 1C 3B BA 57 46 1D 25 76 05 F3 D5 AF
:          53 00 00 00 20 04 8E 9A 3A CE 08 58 3F 79 F3 44
:          FF 78 5B BE A9 F0 7A C7 FA 33 25 B3 D4 9A 21 DD
:          51 94 C6 58 50
:          }

```

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 14 DD AF 30 72 A6 E3 89 4D BF 7A
54 26 36 2F 10 D6 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 - TPM2B_NAME.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

```

```

00 22 - TPMS_CERTIFY_INFO.qualifiedName.size 00 0B 6C 88 60 B2 80 E3
BE 7D - TPM2B_NAME.name 34 F2 85 DC 26 9D 1B 72 A8 0A 17 CF 31 08 F1
55 F2 9B 4E 82 C8 5B 49 7B ~~~~~

```

```

Signature (256 bytes) - generated using the AIK private key
~~~~~
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
A8 CE C0 64 71 F6 E3 81 F7 9D 79 E5 73 7B F3 A4 6E 65 8D 72 B4 0A 3E
5E 70 5F AB 2B 89 B9 5E 65 44 BF 44 7B FB 2E 29 39 64 36 85 63 46 62
AF 25 A5 8B 19 30 AF ~~~~~

```

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 {
49  462:        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 {

```

```

496 17:          SEQUENCE {
498 10:          OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 3'
510 3:          OCTET STRING 04 03 03
      :          }
      :        }
      :      }
      :    }
515 13:  SEQUENCE {
517 9:    OBJECT IDENTIFIER
      :    sha256WithRSAEncryption (1 2 840 113549 1 1 11)
528 0:    NULL
      :    }
530 257:  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 ]
      :  }
791 783: SEQUENCE {
795 503: SEQUENCE {
799 3:   [0] {
801 1:   INTEGER 2
      :   }
804 17:  INTEGER
      :    00 FE B9 AF 03 3B 0B A7 79 04 02 F5 67 AE DF 72
      :    ED
823 13:  SEQUENCE {
825 9:    OBJECT IDENTIFIER
      :    sha256WithRSAEncryption (1 2 840 113549 1 1 11)
836 0:    NULL
      :    }
838 33:  SEQUENCE {
840 31:    SET {
842 29:      SEQUENCE {
844 3:        OBJECT IDENTIFIER commonName (2 5 4 3)
849 22:        UTF8String 'Yubico PIV Attestation'
      :        }
      :      }
      :    }
873 32:  SEQUENCE {
875 13:    UTCTime 14/03/2016 00:00:00 GMT
890 15:    GeneralizedTime 17/04/2052 00:00:00 GMT
      :    }

```



```

907 37:      SEQUENCE {
909 35:      SET {
911 33:      SEQUENCE {
913 3:        OBJECT IDENTIFIER commonName (2 5 4 3)
918 26:        UTF8String 'YubiKey PIV Attestation 9e'
          :        }
          :      }
          :    }
946 290:    SEQUENCE {
950 13:      SEQUENCE {
952 9:        OBJECT IDENTIFIER
          :        rsaEncryption (1 2 840 113549 1 1 1)
963 0:        NULL
          :      }
965 271:    BIT STRING
          :    30 82 01 0A 02 82 01 01 00 93 C4 C0 35 95 7E 26
          :    2A 7E A5 D0 29 C4 D7 E9 39 67 22 B1 09 45 46 4D
          :    DB A4 77 CB 0B A3 F1 D0 69 3C 24 8D A2 72 72 27
          :    E1 7F DE CB 67 A4 1D D2 E5 43 44 6F 21 39 F8 57
          :    34 01 0E 7E C3 81 63 63 6A 6D D7 40 20 7B AF 35
          :    61 9C 8D C1 D1 2B 25 48 EE 52 FC F3 72 6A 74 96
          :    01 CB 1C 1A B2 AD F9 18 96 EB 59 EF E3 3A CA BC
          :    AA 9B 42 FE FF 60 6E 28 89 49 0D C1 B1 B0 25 AE
          :    [ Another 142 bytes skipped ]
          :    }
1240 60:    [3] {
1242 58:      SEQUENCE {
1244 17:        SEQUENCE {
1246 10:          OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 3'
1258 3:          OCTET STRING 04 03 03                                -- firmwa
re version
          :        }
1263 19:      SEQUENCE {
1265 10:        OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 7'
1277 5:        OCTET STRING 02 03 4F 9B B5                            -- serial number
          :      }
1284 16:      SEQUENCE {
1286 10:        OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 8'
1298 2:        OCTET STRING 01 01                                    -- PIN an
d touch policy
          :      }
          :    }
          :  }
          :    }
1302 13:    SEQUENCE {
1304 9:      OBJECT IDENTIFIER
          :      sha256WithRSAEncryption (1 2 840 113549 1 1 11)
1315 0:      NULL
          :    }
1317 257:    BIT STRING

```

```

      :      1F 2B B8 1C 95 A1 01 74 3F 87 27 F6 B3 A6 A9 9D
      :      11 B9 ED 68 92 B9 05 2D 22 36 51 28 23 3D B0 2F
      :      7A 17 D5 8C 0C F4 3A 68 FD 2A 34 0D 80 3C F7 8F
      :      B8 79 B0 76 E5 4D 61 94 C5 72 D6 9F 6E 26 76 5F
      :      03 94 55 40 93 5C 04 EF CC 58 41 EB 7C 86 64 23
      :      5F 23 5E 94 78 73 2E 77 8C 58 C5 45 87 22 CF BA
      :      69 06 B8 C7 06 37 10 21 8C 74 AD 08 B9 85 F2 7B
      :      99 02 4A 3E E8 96 09 D3 F4 C6 AB FA 49 68 E2 E0
      :      [ Another 128 bytes skipped ]
      :      }
      :      }
1578 0:      SET {}
      :      }
      :      }
      :      }

```

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)
      :       }
41 1570:   [0] {
45  762:     SEQUENCE {
49  482:       SEQUENCE {
53   3:         [0] {
55   1:           INTEGER 2
      :           }
58   9:           INTEGER 00 86 77 17 E0 1D 19 2B 26
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 C5 5B 8D E9 B9 3C 53
      :      69 82 88 FE DA 70 FC 5C 88 78 41 25 A2 1D 7B 84
      :      8E 93 36 AD 67 2B 4C AB 45 BE B2 E0 D5 9C 1B A1
      :      68 D5 6B F8 63 5C 83 CB 83 38 62 B7 64 AE 83 37
      :      37 8E C8 60 80 E6 01 F8 75 AA AE F6 6E A7 D5 76
      :      C5 C1 25 AD AA 9E 9D DC B5 7E E9 8E 2A B4 3F 99
      :      0D F7 9F 20 A0 28 A0 9F B3 B1 22 5F AF 38 FB 73
      :      46 F4 C7 93 30 DD FA D0 86 E0 C9 C6 72 99 AF FB
      :      [ 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
      :     sha256WithRSAAEncryption (1 2 840 113549 1 1 11)
548 0:     NULL
      :     }
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 8B 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 20 D1 9D F9 CE B3 B3 4D 1B
      :          61 B0 B4 E0 3F 44 19 72 88 8B 8D 9F 86 4A 5E C7
      :          38 F0 AF C9 28 5C D8 A2 80 C9 43 93 2D FA 39 7F
      :          E9 39 2D 18 1B 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 C2 4B 9D C4 D5 64
      :          32 9A 54 D6 23 B1 65 92 A3 D7 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 2B D6 CD A8 5F DC AE 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
      :          D8 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 ]
      :          }

```

```
1615      :      }
      0:      SET {}
      :      }
      :      }
      :      }
```

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 scalling 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

[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>>.

12.2. Informative References

[android_security]
Kraleovich, R., "The Android Platform Security Model", n.d., <<https://arxiv.org/pdf/1904.05572.pdf>>.

[azureattestation]
Microsoft, ., "Azure Sphere Attestation", n.d., <<https://azure.microsoft.com/enus/resources/azure-sphere-device-authentication-andattestation-service/en-us/>>.

- [fido] FIDO Alliance, ., "FIDO Specification Overview", n.d., <<https://fidoalliance.org/specifications/>>.
- [fido_w3c] W3C, ., "Web Authentication: An API for accessing Public Key Credentials Level 1", n.d., <<https://www.w3.org/TR/webauthn-1/>>.
- [fidoattestation] FIDO Alliance, ., "FIDO 2.0: Key Attestation", n.d., <<https://fidoalliance.org/specs/fido-v2.0-ps-20150904/fido-key-attestation-v2.0-ps-20150904.html>>.
- [fidosignature] FIDO Alliance, ., "FIDO 2.0: Signature Format", n.d., <<https://fidoalliance.org/specs/fido-v2.0-ps-20150904/fido-signature-format-v2.0-ps-20150904.html>>.
- [fidotechnote] FIDO Alliance, ., "FIDO TechNotes: The Truth about Attestation", n.d., <<https://fidoalliance.org/fido-technotes-the-truth-about-attestation/>>.
- [I-D.birkholz-rats-tuda] Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", draft-birkholz-rats-tuda-03 (work in progress), July 2020.
- [I-D.fedorkow-rats-network-device-attestation] Fedorkow, G., Voit, E., and J. Fitzgerald-McKay, "TPM-based Network Device Remote Integrity Verification", draft-fedorkow-rats-network-device-attestation-05 (work in progress), April 2020.
- [I-D.gutmann-scep] Gutmann, P., "Simple Certificate Enrolment Protocol", draft-gutmann-scep-16 (work in progress), March 2020.
- [I-D.tschofenig-rats-psa-token] Tschofenig, H., Frost, S., Brossard, M., Shaw, A., and T. Fossati, "Arm's Platform Security Architecture (PSA) Attestation Token", draft-tschofenig-rats-psa-token-05 (work in progress), March 2020.
- [I-D.voit-rats-trusted-path-routing] Voit, E., "Trusted Path Routing", draft-voit-rats-trusted-path-routing-02 (work in progress), June 2020.

- [ieee802-1AR] IEEE Standard, ., "IEEE 802.1AR Secure Device Identifier", 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [intelsgx] Intel, ., "Intel(R) Software Guard Extensions: Attestation & Provisioning Services", n.d., <<https://software.intel.com/en-us/sgx/attestation-services>>.
- [keystore] Google, ., "Android Keystore System", n.d., <<https://developer.android.com/training/articles/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, <<https://www.rfc-editor.org/info/rfc4210>>.
- [RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <<https://www.rfc-editor.org/info/rfc5209>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <<https://www.rfc-editor.org/info/rfc8555>>.

[SP800-147B]

NIST, ., "BIOS Protection Guidelines for Servers", n.d.,
<<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-147B.pdf>>.

[SP800-155]

NIST, ., "BIOS Integrity Measurement Guidelines (Draft)",
n.d., <https://csrc.nist.gov/CSRC/media/Publications/sp/800-155/draft/documents/draft-SP800-155_Dec2011.pdf>.

[tapinfomodel]

Group, T., "TCG Trusted Attestation Protocol (TAP)
Information Model for TPM Families 1.2 and 2.0 and DICE
Family 1.0", n.d., <https://trustedcomputinggroup.org/wp-content/uploads/TNC_TAP_Information_Model_v1.00_r0.29A_publicreview.pdf>.

[tcgglossary]

Group, T., "TCG Glossary, Version 1.1", n.d.,
<<https://trustedcomputinggroup.org/wp-content/uploads/TCG-Glossary-V1.1-Rev-1.0.pdf>>.

[tpmarchspec]

Group, T., "TPM 2.0 Mobile Reference Architecture", n.d.,
<<https://trustedcomputinggroup.org/resource/tpm-2-0-mobile-reference-architecture-specification/>>.

[windowsdefender]

Microsoft, ., "Windows Defender System Guard attestation",
n.d., <<https://www.microsoft.com/security/blog/2018/04/19/introducing-windows-defender-system-guard-runtime-attestation/>>.

[windowshealth]

Microsoft, ., "Windows Device Health Attestation", n.d.,
<<https://docs.microsoft.com/en-us/windowsserver/security/device-health-attestation>>.

[yubikey_attestation]

Yubico, ., "PIV Attestation", n.d.,
<https://developers.yubico.com/PIV/Introduction/PIV_attestation.html>.

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