Use cases for Remote Attestation common encodings
draft-richardson-rats-usecases-04

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 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 ............................................. 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
3. Requirements Language ..................................... 4
4. Overview of Sources of Use Cases .......................... 4
5. Use case summaries ........................................ 5
   5.1. Device Capabilities/Firmware Attestation ............. 5
      5.1.1. Relying on an Attestation Server ............... 5
      5.1.2. Autonomous Relying Party ....................... 5
      5.1.3. Proxy Root of Trust ............................. 5
      5.1.4. network scaling – small ........................ 6
      5.1.5. network scaling – medium ....................... 6
      5.1.6. network scaling – large ........................ 6
   5.2. Hardware resiliency / watchdogs ....................... 6
   5.3. IETF TEEP WG use case ............................... 7
   5.4. Confidential Machine Learning (ML) model ............. 7
   5.5. Critical infrastructure ............................. 7
      5.5.1. Computation characteristics .................... 7
   5.6. Cryptographic Key Attestation ....................... 8
      5.6.1. Device Type Attestation ......................... 8
      5.6.2. Key storage attestation ........................ 8
      5.6.3. End user authorization ........................... 9
   5.7. Geographic attestation ............................... 9
      5.7.1. I am here ....................................... 9
      5.7.2. I am near ..................................... 9
      5.7.3. You are here ................................... 10
   5.8. Connectivity attestation ............................ 10
6. Technology users for RATS ................................. 10
   6.1. Trusted Computing Group (TCG) ....................... 10
   6.2. Android Keystore system ............................. 12
   6.3. Fast IDentity Online (FIDO) Alliance ................ 12
7. Examples of Existing Attestation Formats. ................ 14
   7.1. Android Keystore .................................... 14
      7.1.1. TEE ........................................... 15
      7.1.2. Secure Element ................................. 19
   7.2. Windows 10 TPM ..................................... 24
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

(TBD: Seeking something useful here.)

3. Requirements Language

This document is not a standards track document and does not make any normative protocol requirements using terminology described in [RFC2119].

4. Overview of Sources of Use Cases

The following specifications have been covered in this document:

- The Trusted Computing Group "Network Attestation System" (private document)
- Android Keystore
- Fast Identity Online (FIDO) Alliance attestation,

This document will be expanded to include summaries from:

- Trusted Computing Group (TCG) Trusted Platform Module (TPM)/Trusted Software Stack (TSS)
- ARM "Platform Security Architecture" [I-D.tschofenig-rats-psa-token]
- Intel SGX attestation [intelsgx]
5. Use case summaries

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

5.1. Device Capabilities/Firmware Attestation

A network operator wants to know the qualities of the hardware and software on the machines attached to their network. The process starts with some kind of Root of Trust, 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.

5.1.1. Relying on an Attestation Server

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

The signed measurements are sent to a relying party which must validate them directly. (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

A variety of devices provide measurements via their Root of Trust. A 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

An entire network of systems needs to be validated (such as all the desktops in an enterprise’s building). The infrastructure is in 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

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

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 maintain identities of connected devices more than to validate correct firmware, but both situations are reasonable.

5.2. Hardware resiliency / watchdogs

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

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

Microsoft talked about this category of use cases at the recent Microsoft //build conference.

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

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

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. Cryptographic Key Attestation

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.

5.6.1. Device Type Attestation

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 features such as a hardware TPM, software TPM via TEE, or software TPM without TEE. Other details such as the availability of fingerprint readers or HDMI outputs may also be inferred.

5.6.2. Key storage attestation

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.

5.6.3. End user authorization

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

5.7. Geographic attestation

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.

5.7.1. I am here

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

5.7.2. 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.

5.7.3. 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.

5.8. Connectivity attestation

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.

6. Technology users for RATS

6.1. Trusted Computing Group (TCG)

The TCG is trying to solve 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 a work-in-progress, and is available to TCG members only. The goal is to be multi-vendor, scalable and extensible. The proposal intentionally limits itself to:

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

Service providers and enterprises deploy hundreds of routers, many of them in remote locations where they’re difficult to access or secure. The point of remote attestation is to:

- identify a remote box in a way that’s hard to spoof
o report the inventory of software was launched on the box in a way that can not be spoofed

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, assemblies of hardware/software created by end-customers, and equipment that is sleepy. There is an intention to cover some of these are topics in future versions of the documents.

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

The TCG uses the following terminology:

- Device Manufacturer
- Attester ("device under attestation")
- Verifier (Network Management Station)
- "Explicit Attestation" is the TCG term for a static (platform) attestation
- "Implicit Attestation" is the TCG term for a session attestation
- Reference Integrity Measurements (RIM), which are signed my device manufacturer and integrated into firmware.
- Quotes: measured values (having been signed), and RIMs
- Reference Integrity Values (RIV)
- devices have a Initial Attestation Key (IAK), which is provisioned at the same time as the IDevID [IEEE802-1AR]
- PCR - Platform Configuration Registry (deals with hash chains)
The TCG document builds upon a number of IETF technologies: SNMP (Attestation MIB), YANG, XML, JSON, CBOR, NETCONF, RESTCONF, CoAP, TLS and SSH. The TCG document leverages the 802.1AR IDevID and LDevID processes.

6.2. Android Keystore system

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

The uses described in section Section 5.6 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 provided protected confirmation messages. This uses hardware access from the TPM/TEE to display a message directly to the user, and receives confirmation directly from the user. A hash of the contents of the message can provided in an attestation that the device provides.

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

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

6.3. Fast IDentity Online (FIDO) Alliance

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

The use cases of Section 5.6 are primary.

FIDO specifications extend to various hardware second factor authentication devices.

Terminology includes:

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

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

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

FIDO does 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.

```java
'''
// KeyGenParameterSpec.Builder builder = null; if(hasStrongBox) {
  builder = new KeyGenParameterSpec.Builder( m_alias,
    KeyProperties.PURPOSE_SIGN | KeyProperties.PURPOSE_VERIFY |
    KeyProperties.PURPOSE_ENCRYPT | KeyProperties.PURPOSE_DECRYPT)
    .setKeySize(2048) .setDigests(KeyProperties.DIGEST_NONE,
    KeyProperties.DIGEST_SHA256)
    .setBlockModes(KeyProperties.BLOCK_MODE_CBC,
    KeyProperties.BLOCK_MODE_GCM)
    .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_RSA_PKCS1,
    KeyProperties.ENCRYPTION_PADDING_RSA_OAEP)
    .setSignaturePaddings(KeyProperties.SIGNATURE_PADDING_RSA_PSS,
    KeyProperties.SIGNATURE_PADDING_RSA_PKCS1)
    .setUserAuthenticationRequired(false) .setIsStrongBoxBacked(true)
    .setUnlockedDeviceRequired(true); } else { builder = new
    KeyGenParameterSpec.Builder( m_alias, KeyProperties.PURPOSE_SIGN |
    KeyProperties.PURPOSE_VERIFY | KeyProperties.PURPOSE_ENCRYPT |
    KeyProperties.PURPOSE_DECRYPT) .setKeySize(2048)
    .setDigests(KeyProperties.DIGEST_NONE, KeyProperties.DIGEST_SHA256,
    KeyProperties.DIGEST_SHA384, KeyProperties.DIGEST_SHA512)
    .setBlockModes(KeyProperties.BLOCK_MODE_CBC,
    KeyProperties.BLOCK_MODE_CTR,KeyProperties.BLOCK_MODE_GCM)
    .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_RSA_PKCS1,
    KeyProperties.ENCRYPTION_PADDING_RSA_OAEP)
    .setSignaturePaddings(KeyProperties.SIGNATURE_PADDING_RSA_PSS,
    KeyProperties.SIGNATURE_PADDING_RSA_PKCS1)
    .setUserAuthenticationRequired(false) .setIsStrongBoxBacked(false)
    .setUnlockedDeviceRequired(true); }

builder.setAttestationChallenge(challenge_bytes);

KeyGenParameterSpec keySpec = builder.build(); AttestedKeyPair akp =
  dpm.generateKeyPair(componentName, algorithm, keySpec,
  idAttestationFlags);```
7.1.1.  TEE

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

679 166: [3] { 682 163: SEQUENCE { 685 29: SEQUENCE { 687 3: OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14) 692 22: OCTET STRING, encapsulates { 694 20: OCTET STRING : 36 61 E0 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] : 36 61 E1 00 7C 88 05 09 51 8B 44 6C 47 FF 1A 4C : C9 EA 4F 12 : } : } : } 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 B8 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 7B 5F 7E 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 EA 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 1 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 '90e8da3cdaf7c7820' : } : } 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 01/01/1970 00:00:00 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
```
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:

```csharp
request.KeyAlgorithmName = KeyAlgorithmNames.Rsa;
request.KeyStorageProviderName = "Microsoft Smart Card Key Storage Provider";
request.UseExistingKey = true; request.Exportable = ExportOption.NotExportable; request.ContainerName = prj.GetContainerName();
request.Subject = subject_name; request.KeyUsages = keyUsages;
request.SmartcardReaderName = smartCardReaderName;

string privacyCa = "MIIDezCCAmOgAwIBAgIBATANBgkqhkiG9w0BAQsFADBUMBQswCQYDVQQGEwJVUzEYMBYGA1UEChMPVS5TLiBHb3Zlcm5tZW50MQ0wCwYDVQQDDA1fmFyZmFjdGVkMB4XDTE4MDQwMjE5MDAwMFoXDTI4MDAwMDAwMFowVDELMAkGA1UEBhMCVVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubVREMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMROV8sQ707OSvRxoXS5L6MaB0r4+5fj9J0sA3sA22pctONaXu6Ze1v55Ft+u5E2onSTG9CJl9yD0D+nWcu/tbpihJZU47p6a+qa75e6dGkRn2h0c9fIS3h0uyYc+9mK+i4b4WQ+yw+DzD2UeQW0JhEVLJ0tvlO+LQ2M80v9g991ZDiiqbR2TpyhcmP8nLsL+2hEYsYw==";
byte[] privacyCaBytes = Convert.FromBase64String(privacyCa); IBuffer buffer = privacyCaBytes.AsBuffer();
request.AttestationCredentialCertificate = new Certificate(buffer); 
```

csrToDiscard = await 
CertificateEnrollmentManager.UserCertificateEnrollmentManager.
CreateRequestAsync(request); ```

The structure is essentially a Full PKI Request as described in RFC 5272.

- 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.
```
```

The format is structured as follows:

```c
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:

```c
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 ED 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_CLOCKINFO.clock AD 8C 8C 
60 - TPMS_CLOCKINFO.resetCount E6 C2 70 80 - 
TPMS_CLOCKINFO.restartCount 00 - - TPMS_CLOCKINFO.safe 
D4 2C 65 4C 6B 95 ED 95 - firmwareVersion
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 

Richardson, et al. Expires January 9, 2020
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 B1 7F 76 48
E4 : A9 4C BC D5 6B E1 1F 0A 60 93 C6 FE AA DD 82 8D B6 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: 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 6B 11 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 1B 64 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: OCTET STRING { 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 B2 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 2D 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 OD 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 -- firmware 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 and touch policy : : : } : } : } 1302 13:
SEQUENCE { 1304 9: OBJECT IDENTIFIER : sha256WithRSAEncryption (1 2 840 113549 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 DC F4 3A 68 FD 2A 34 OD 80 3C F7 8F : B8 79 B0 76 E5 4D 96 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 { 49 762: SEQUENCE { 53 3: [0] { 55 1: INTEGER 00 86 77 17 E0 1D 19 2B 26 69 13:
SEQUENCE { 71 9: OBJECT IDENTIFIER : sha256WithRSAEncryption (1 2 840 113549 1 11) 82 0: NULL : ) 84 43: SEQUENCE { 86 41: SET { 88 39: SEQUENCE { 90 3: OBJECT IDENTIFIER commonName (2 5 4 3) 93 95: SEQUENCE {} 129 32: UTF8String 'Yubico PIV Root CA Serial 263751' : ) : ) 131 13: UTCString 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 publicKeyAlgorithm (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) 215 0: NULL : ) 217 271: BIT STRING : 30 B2 01 0A 02 82 01 01 00 C5 5B 8D 6B 97 99 09 3B : 69 8B 88 FE DA 70 FC 5C 89 78 41 25 A2 1D 7B 84 : 8E 93 36 AD 67 2B 4C AB 45 BE B2 E0 D5 9C 1B A1 : 6B D5 6B F8 63 5C 83 CB 83 38 62 B7 64 AE 83 37 : 37 BE C8 60 8D E6 01 F8 75 AA AE F6 6E A7 D5 76 : C5 C1 25 AD AA 9E 9D DC B5 7E 9E 8E 2A B4 3F 99 : 0D F7 9F 20 A0 28 A0 9F 03 B1 22 5F AF 38 FB 73 : 46 F4 C7 93 30 DD FA D0 86 E0 C9 C6 72 9F 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 FF 02 D1 00 : ) 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 D1 00 : ) : ) : ) 535 13: SEQUENCE { 537 9: OBJECT IDENTIFIER : sha256WithRSAEncryption (1 2 840 113549 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 BF 58 5B 9A 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 : )

Privacy Considerations.
TBD.

Security Considerations.
TBD.
10. IANA Considerations

TBD.

11. Acknowledgements

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

12. References

12.1. Normative References


12.2. Informative References


[fidosignature]

[fidotechnote]

[I-D.gutmann-scep]

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

[ieee802-1AR]

[intelsgx]

[keystore]

[keystore_attestation]

[RFC4210]
Appendix A. Changes

- created new section for target use cases
- 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