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 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 finger-
print 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
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
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
```
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]
```

```
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.

```plaintext
```

Internet-Draft                 useful RATS                     July 2019

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' : }
1259 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
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 BF : 56 AF 77 D3 CE 3C DC 48 52 DD B2
86 0D 76 17 7C : FD EE B4 E6 E6 0A 08 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:
29: SEQUENCE { 1635 3: OBJECT IDENTIFIER subjectKeyIdentifier (2 5 4
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 DC : } : ) } 
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 : } : ) ) 1697 15: 
SEQUENCE ( 1699 3: OBJECT IDENTIFIER basicConstraints (2 5 29 19
1704 1: BOOLEAN TRUE 1707 5: OCTET STRING, encapsulates { 1709 3:
SEQUENCE { 1711 1: BOOLEAN TRUE : : ) : : ) 1714 14: SEQUENCE { 1716
3: OBJECT IDENTIFIER keyUsage (2 5 29 15) 1721 1: BOOLEAN TRUE 1724
4: OCTET STRING, encapsulates { 1726 2: BIT STRING 2 unused bits :
‘100000’B (bit 5) : ) : ) ) 1730 84: SEQUENCE { 1732 3: OBJECT
IDENTIFIER cRLDistributionPoints (2 5 29 31) 1737 77: OCTET STRING, 
encapsulates { 1739 75: SEQUENCE { 1741 73: SEQUENCE { 1743 71: [0] { 1745 69: [0] : 1747 03: [6] :
'https://android.googleapis.com/attestation/crl/1' :
SEQUENCE ( 1818 9: OBJECT IDENTIFIER : sha256WithRSAEncryption (1 2
840 113549 1 1 11) 1829 0: NULL : ) 1831 513: 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 CE 68 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

null

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 = "MIIDezCCAmOgAwIBAgIBATANBgkqhkiG9w0BAQsFADBUMQswCQYDVQQGEwJVUzEyMA4GA1UEA
MBWgZ2V0MRMwEQYDVQQDa Wh0MDQ4NjE4MjUyNTQxOjQwMQswCQYDVQQGEwJXYWdhb3UETABCBg
MB0GA1UdDgQcBhAcBAgN1kYIgIRGwoCAwEAgYIKwYBBQUHAwIwAAYHKoUBiNRAgIBAgIazuI
yTzJpYzYyMB0GCSqGSIb3DQEBCwUAA4G/KBQb/nVvJ2aIwD2/gEo124b634vB/a3P9uq8J
Fon6v17U1/hG6ncqR312H7H16B5xYK7yj/4IcUaQhHk06D1CTdM6jKbVp5

byte[] privacyCaBytes = Convert.FromBase64String(privacyCa); IBuffer buffer = privacyCaBytes.AsBuffer();
request.AttestationCredentialCertificate = new Certificate(buffer);`
```


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.

```
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 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 BD E8 9A 85 04 D7 F3 7B B7 4C FF : 32 CD
B4 F1 80 CA A6 35 B9 2C 39 87 B7 96 03 C3 : A3 00 22 00 DB 6C 88 60
```
```
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:
```
```
```
```
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 3B 4A 5D 6C 27 F5 69 7D 17 AD F6 C0 03 27 09 3D 93 F5 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 B8 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 0B 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 11) 82 0: NULL : } 84 43: SEQUENCE { 86 41: SET { 88 39: SEQUENCE { 90 3: OBJECT IDENTIFIER commonName (2 5 4 3) 95 32:
STRING 01 01 -- PIN and 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 DC F4 3A 68 FD 2A 34 OD 80 3C F7
8F : B8 79 B0 76 E5 4D 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 { 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) 93 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 data (1 2 840
113549 1 7 1) : } 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 BE 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 : DD 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 128 bytes
17: SEQUENCE { 498 10: OBJECT IDENTIFIER '1 3 6 1 4 1 41482 3 3):
510 3: OCTET STRING 05 01 02 FF 02 01 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 01 00 : ) : ) : ) : ) 535 13: SEQUENCE
{ 537 9: OBJECT IDENTIFIER : sha256WithRSAEncryption (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 BF 88 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 { 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 0D 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 B6 4A 5E C7 :
38 F0 AF C9 28 5C D8 A2 80 C9 43 93 2D FA 39 7F : E9 39 2D 18 B7 A7
A2 76 8F D4 6C D0 75 96 99 OD : 06 37 9D 90 D5 71 00 6E F8 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:
OCfT
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 44 7A 3C 0C F0 7C 67 11 ED C5 84 07 C8 19 1B
71 34 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 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]
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
fido-technotes-the-truth-about-attestation/>.

[I-D.gutmann-scep]

[I-D.tschofenig-rats-psa-token]
Tschofenig, H., Frost, S., Brossard, M., and A. Shaw,
"Arm’s Platform Security Architecture (PSA) Attestation
Token", draft-tschofenig-rats-psa-token-01 (work in
progress), April 2019.

[ieee802-1AR]
IEEE Standard, ., "IEEE 802.1AR Secure Device Identifier",

[intelsgx]
Intel, ., "Intel(R) Software Guard Extensions: Attestation
& Provisioning Services", n.d.,
attestation-services>.

[keystore]
<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,
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