RATS
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

M. Ounsworth
Cryptic Forest
J.-P. Fiset
Crypto4A
H. Tschofenig
H-BRS
H. Birkholz
Fraunhofer SIT
M. Wiseman

N. Smith
Intel Corporation
2 March 2026

Evidence Encoding for Hardware Security Modules
draft-ietf-rats-pkix-key-attestation-03

Abstract

This document specifies a vendor-agnostic format for Evidence produced and verified within a PKIX context. The Evidence produced this way includes claims collected about a cryptographic module, such as a Hardware Security Module (HSM), and elements found within it such as cryptographic keys.

One scenario envisaged is that the state information about the cryptographic module can be securely presented to a remote operator or auditor in a vendor-agnostic verifiable format. A more complex scenario would be to submit this Evidence to a Certification Authority to aid in determining whether the storage properties of this key meet the requirements of a given certificate profile.

This specification also offers a format for requesting a cryptographic module to produce Evidence tailored for expected use.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://ietf-rats-wg.github.io/key-attestation/draft-ietf-rats-pkix-key-attestation.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-rats-pkix-key-attestation/.

Discussion of this document takes place on the RATS Working Group mailing list (mailto:rats@ietf.org), which is archived at https://datatracker.ietf.org/wg/rats/about/. Subscribe at https://www.ietf.org/mailman/listinfo/rats/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-rats-wg/key-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 3 September 2026.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a format to transmit Evidence from an Attester to a Verifier within a PKIX environment. This environment refers to the components generally used to support PKI applications such as Certification Authorities and their clients, or more generally that rely upon X.509 certificates. As outlined in Section 3, this specification uses a necessary mixture of RATS and PKI terminology in order to map concepts between the two domains.

Within this specification, the concepts found in the Remote ATtestation procedureS Architecture ([RFC9334]) are mapped to the PKIX environment. There are many other specifications that are based on the RATS Architecture which offer formats to carry Evidence. This specification deals with peculiar aspects of the PKIX environment which make the existing Evidence formats inappropriate:

* ASN.1 is the preferred encoding format in this environment. X.509 certificates ([RFC5280]) are used widely within this environment and the majority of tools are designed to support ASN.1. There are many specialized devices (Hardware Security Modules) that are inflexible in adopting other formats because of internal constraints or validation difficulties. This specification defines the format in ASN.1 to ease the adoption within the community.

* The claims reported within the generated Evidence are generally a small subset of all possible claims about the Target Environment. The claims relate to elements such as "platform" and "keys" which are more numerous than what a Verifier requires for a specific function. This specification provides the means to moderate the information disseminated as part of the generated Evidence.

This specification also aims at providing an extensible framework to encode, as part of the generated Evidence, claims other than the one proposed in this document. This allows implementations to introduce new claims and their associated semantics to the Evidence produced.

2. Use Cases

This section covers use cases that motivated the development of this specification.

2.1. Remote audit of a Hardware Security Module (HSM)

There are situations where it is necessary to verify the current running state of an HSM as part of operational or auditing procedures. For example, there are devices that are certified to work in an environment only if certain versions of the firmware are loaded or only if user keys are protected with specific policies.

The Evidence format offered by this specification allows a platform to report its firmware level along with other collected claims necessary in critical deployments.

2.2. Key import and HSM clustering

Consider that an HSM is being added to a logical HSM cluster. Part of the onboarding process could involve the newly-added HSM providing Evidence of its running state, for example that it is a genuine device from the same manufacturer as the existing clustered HSMs, firmware patch level, FIPS mode, etc. It could also be required to provide information about any system-level keys required to establish secure cluster communication. In this scenario, the Verifier and Relying Party will typically be other HSMs in the cluster deciding whether or not to admit the new HSM.

A related scenario is when performing a key export-import across HSMs. If the key is being imported with certain properties, for example an environment running in FIPS mode at FIPS Level 3, and the key is set to certain protection properties such as Non-Exportable and Dual-Control, then the HSM might wish to verify that the key was previously stored under the same properties. This specification provides an Evidence format with sufficient details to support this type of implementation across HSM vendors.

These scenarios motivate the design requirements to have an ASN.1 based Evidence format and a data model that more closely matches typical HSM architecture since, as shown in both scenarios, an HSM is acting as Verifier and Relying Party.

2.3. Attesting subject of a certificate issuance

Prior to a Certification Authority (CA) issuing a certificate on behalf of a subject, a number of procedures are required to verify that the subject of the certificate is associated with the key that is certified. In some cases, such as issuing a code signing certificate [CNSA2.0] [CSBR], a CA must ensure that the subject key is located in a Hardware Security Module (HSM).

The Evidence format offered by this specification is designed to carry the information necessary for a CA to assess the location of the subject key along a number of commonly-required attributes. More specifically, a CA could determine which HSM was used to generate the subject key, whether this device adheres to certain jurisdiction policies (such as FIPS mode) and the constraints applied to the key (such as whether is it extractable).

For relatively simple HSM devices, storage properties such as "extractable" may always be false for all keys since the devices are not capable of key export and so the Evidence could be essentially a hard-coded template asserting these immutable attributes. However, more complex HSM devices require a more complex Evidence format that encompasses the mutability of these attributes.

Also, a client requesting a key attestation might wish to scope-down the content of the produced Evidence as the HSM contains much more information than that which is relevant to the transaction. Not reducing the scope of the generated Evidence could, in some scenarios, constitute a privacy violation.

3. Conventions and 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 specification uses a necessary mixture of PKI terminology and RATS Architecture definitions in order to map concepts between the two domains.

The reader is assumed to be familiar with the vocabulary and concepts defined in the RATS Architecture ([RFC9334]) such as Attester, Relying Party, Verifier.

The reader is assumed to be familiar with common vocabulary and concepts defined in [RFC5280] such as certificate, signature, attribute, verification and validation.

In order to avoid confusion, this document generally capitalizes RATS terms such as Attester, Relying Party, and Claim. Therefore, for example, a "Verifier" should be assumed to be an entity that checks the validity of Evidence as per [RFC9334], whereas a "verifier" could be a more general reference to a PKI entity that checks the validity of an X.509 certificate or other digital signature as per [RFC5280].

The following terms are used in this document:

Attestation Key (AK):
: Cryptographic key controlled solely by the Attester and used only for the purpose of producing Evidence. In other words, it is used to digitally sign the claims collected by the Attester.

Attester:
: The term Attester respects the definition offered in [RFC9334]. In this specification, it is also interchangeable with "platform" or "HSM".

Attesting Environment:
: As defined in [RFC9334], the Attesting Environment collects the information to be represented in Claims. In practical terms, an implementation may be designed with services to perform this function. To remain consistent with the RATS Architecture, the term "Attesting Environment" is used throughout this specification.

Evidence:
: The term Evidence respects the definition offered in [RFC9334]. In this specification, it refers to claims, encoded according to the format defined within this document, and signed using Attestation Keys.

Hardware Security Module (HSM):
: A physical computing device that safeguards and manages secrets, such as cryptographic keys, and performs cryptographic operations based on those secrets. This specification takes a broad definition of what counts as an HSM to include smartcards, USB tokens, TPMs, cryptographic co-processors (PCI cards) and "enterprise-grade" or "cloud-service grade" HSMs (possibly rack mounted). In this specification, it is interchangeable with "platform" or "Attester".

Key Attestation:
: Process of producing Evidence containing claims pertaining to user keys found within an HSM. In general, the claims include enough information about a user key and its hosting platform to allow a Relying Party to make judicious decisions about the key, such as whether to issue a certificate for the key.

RATS:

Remote ATtestation procedureS. Refers to a working group within IETF and all the documents developed under this umbrella of efforts. This specification is developed using concepts developed in RATS but more particularly refers to the RATS Architecture as introduced in [RFC9334].

PKIX:
Public Key Infrastructure based on X.509 certificates. Refers to the framework of technology, policies and procedures used to manage and distribute digital certificate based on [RFC5280] and related specifications.

Platform:
The module or device that embodies the Attester. In this specification, it is interchangeable with "Attester" or "HSM".

Platform Attestation:
Evidence containing claims pertaining to measured values associated with the platform itself. In general, the claims include enough information about the platform to allow a Relying Party to make judicious decisions about the platform, such as those carried out during audit reviews.

Presenter:
Role that facilitates communication between the Attester and the Verifier. The Presenter initiates the operation of generating Evidence at the Attester and passes the generated Evidence to the Verifier. In the case of HSMs, the Presenter is responsible of selecting the claims that are part of the generated Evidence.

Trust Anchor:
As defined in [RFC6024] and [RFC9019], a Trust Anchor "represents an authoritative entity via a public key and associated data. The public key is used to verify digital signatures, and the associated data is used to constrain the types of information for which the trust anchor is authoritative." The Trust Anchor may be a certificate, a raw public key, or other structure, as appropriate.

Trusted Platform Module (TPM):
A tamper-resistant processor generally located on a computer’s motherboard used to enhance attestation functions for the hosting platform. TPMs are very specialized Hardware Security Modules and generally use other protocols (than the one presented in this specification) to transmit Evidence.

User Key:

A user key consists of a key hosted by an HSM (the platform) and intended to be used by a client of the HSM. Other terms used for a user key are "application key", "client key" or "operational key". The access and operations on a user key is controlled by the HSM.

3.1. Claims and measurements in generated Evidence

The RATS Architecture [RFC9334] states that Evidence is made up of claims and that a claim is "a piece of asserted information, often in the form of a name/value pair". The RATS Architecture also mentions the concept of "measurements" that "can describe a variety of attributes of system components, such as hardware, firmware, BIOS, software, etc., and how they are hardened."

Some HSMs have a large amount of memory and can therefore contain a substantial amount of elements that can be observed independently by the Attesting Environment. Each of those elements, in turn, can contain a number of measurable attributes.

A certain level of complexity arises as multiple elements of the same class can be reported simultaneously in generated Evidence. In this case, multiple similar claims are reported simultaneously but associated with different elements.

For example, two independent user keys could be reported simultaneously in Evidence. Each key is associated with a SPKI (Subject Public Key Identifier). The measured values for the SPKI of the respective keys are different.

To that end, in this specification, the claims are organized as collections where each claim is the association of a claim type with the measured value. The collections, in turn, are organized by entities. An entity represents one of the elements that is observed in the Target Environment.

Thus, an entity is associated with a collection of claims. Each claim is the association of a claim type with a measured value.

The grouping of claims into entities facilitates the comprehension of a large addressable space into elements recognizable by the user. More importantly, it curtails the produced Evidence to portions of the Target Environment that relate to the needs of the Verifier. See Section 10.4.

3.2. Attestation Key Certificate Chain

The data format in this specification represents Evidence and requires third-party endorsement in order to establish trust. Part of this endorsement is a trust anchor that chains to the HSM's attestation key (AK) which signs the Evidence. In practice the trust anchor usually is a manufacturing CA belonging to the device vendor which proves that the device is genuine and not counterfeit. The trust anchor can also belong to the device operator as would be the case when the AK certificate is replaced as part of onboarding the device into a new operational environment.

The AK certificate that signs the evidence MUST include the Extended Key Usage (EKU) certificate extension, and the EKU certificate extension MUST include the id-kp-attest, as defined in [I-D.jpfiset-lamps-attestationkey-eku].

Note that the data format specified in Section 5 allows for zero, one, or multiple 'SignatureBlock's, so a single Evidence statement could be un-protected, or could be endorsed by multiple AK chains leading to different trust anchors. See Section 6 for a discussion of handling multiple SignatureBlocks.

4. Information Model

The Evidence format is composed of two main sections:

* An Evidence section which describes the list of reported entities.

* A signature section where one or more digital signatures are offered to prove the origin of the Evidence and maintain its integrity.

The details of the signature section is left to the data model. The remainder of this section deals with the way the information is organized to form the claims.

The claims are associated with entities to help with the organization and comprehension of the information. Entities are elements observed in the Target Environment by the Attesting Environment. Each entity, in turn, is associated with a collection of claims that describes the attributes of the element.

Therefore, the Claim description section is a set of entities and each entity is composed of a claim set.

4.1. Entity

An entity is a logical construct that refers to a portion of the Target Environment's state. It is addressable via an identifier such as a UUID or a handle (as expressed in [PKCS11]). In general, an entity refers to a component recognized by users of the HSM, such as a key or the platform itself.

An entity is composed of a type, the entity type, and a collection of claims. The entity type describes the class of the entity while the collection of claims defines its state.

An entity MUST be reported at most once in a claim description. The claim description can have multiple entities of the same type (for example reporting multiple keys), but each entity MUST relate to different portions of the Target Environment.

It is possible for two entities to be quite similar such as in a situation where a key is imported twice in a HSM. In this case, the two related entities could be associated similar claims. However, they are treated as different entities as they are reporting different portions of the Target Environment.

The number of entities reported in a claim description, and their respective type, is left to the implementer. For a simple device where there is only one key, the list of reported entities could be fixed. For larger and more complex devices, the list of reported entities should be tailored to what is demanded by the Presenter.

In particular, note that the nonce claim contained with the Transaction entity is optional, and therefore it is possible that an extremely simple device that holds one static key could have its Evidence generated at manufacturing time and injected statically into the device instead of being generated on-demand. This model would essentially off-board the Attesting Environment to be part of the manufacturing infrastructure. In the RATS Architecture, this configuration would refer to the the information provided by the HSM as an Endorsement provided by the manufacturer as opposed to Evidence generated by the Attesting Environment.

4.2. Entity Type

An entity is defined by its type. This specification defines three entity types:

* Platform : This entity holds claims that describes the state of the platform (or device) itself. Entities of this type hold claims that are global in nature within the Target Environment.

* Key : The entities of this type represent a cryptographic key protected within the Target Environment and hold claims that describes that specific key.

* Transaction : This entity is logical in nature since it is associated with claims that does not describe anything found in the Target Environment. Instead, these claims relate to the current request for Evidence such as a nonce to support freshness.

Although this document defines a short list of entity types, this list is extensible to allow implementers to report on entities found in their implementation and not covered by this specification. By using an Object Identifiers (OID) for specifying entity types and claim types, this format is inherently extensible; implementers of this specification MAY define new custom or proprietary entity types and place them alongside the standardized entities, or define new claim types and place them inside standardized entities.

Verifiers SHOULD ignore and skip over unrecognized entity or claim types and continue processing normally. In other words, if a given Evidence would have been acceptable without the unrecognized entities or claims, then it SHOULD still be acceptable with them.

4.3. Claim Type

Each claim found in an entity is composed of the claim type and a value. Each claim describes a portion of the state of the associated entity. For example, a platform entity could have a claim which indicates the firmware version currently running. Another example is a key entity with a claim that reports whether the key is extractable or not.

A value provided by a claim is to be interpreted within the context of its entity and in relation to the claim type.

It is RECOMMENDED that a claim type be defined for a specific entity type, to reduce confusion when it comes to interpretation of the value. In other words, a claim type SHOULD NOT be used by multiple entity types. For example, if a concept of "revision" is applicable to a platform and a key, the claim for one entity type (platform revision) should have a different identifier than the one for the other entity type (key revision).

The nature of the value (boolean, integer, string, bytes) is dependent on the claim type.

This specification defines a limited set of claim types. However, the list is extensible through the IANA registration process or private OID allocation, enabling implementers to report additional claims not covered by this specification.

The number of claims reported within an entity, and their respective type, is left to the implementer. For a simple device, the reported list of claims for an entity might be fixed. However, for larger and more complex devices, the list of reported claims should be tailored to what is demanded by the Presenter.

Claims of a particular type MAY be repeated within an entity while others MUST NOT. For example, for a platform entity, there can only be one "firmware version" claim. Therefore, the associated claim MUST NOT be repeated as it may lead to confusion. However, a claim relating to a "ak-spki" MAY be repeated, each claim describing a different attesting key. Therefore, the definition of a claim specifies whether or not multiple copies of that claim are allowed within an entity claim set.

If a Verifier detects, within a single entity, multiple copies of a claim type that should not be repeated, it MUST reject the Evidence as malformed. Since a Verifier is encouraged to ignore unrecognized claim types, it is possible that a potential rejection is missed.

If a Verifier encounters, within the context of an entity, a repeated claim for a type where it is allowed, it MUST treat each one as an independent claim and MUST NOT consider later ones to overwrite the previous one.

5. Data Model

This section describes the data model associated with generated Evidence. For ease of deployment within the target ecosystem, ASN.1 definitions and DER encoding are used. A complete ASN.1 module is provided in Section 8.

The top-level structures, as ASN.1 fragments, are:

```
Evidence ::= SEQUENCE {
    tbs                          TbsEvidence,
    signatures                   SEQUENCE SIZE (0..MAX) OF SignatureBlock,
    intermediateCertificates  [0] SEQUENCE OF Certificate OPTIONAL
                                 -- As defined in RFC 5280
}

TbsEvidence ::= SEQUENCE {
    version INTEGER,
    reportedEntities SEQUENCE SIZE (1..MAX) OF ReportedEntity
}

SignatureBlock ::= SEQUENCE {
   sid                  SignerIdentifier,
   signatureAlgorithm   AlgorithmIdentifier,
   signatureValue       OCTET STRING
}

SignerIdentifier ::= SEQUENCE {
   keyId                [0] EXPLICIT OCTET STRING OPTIONAL,
   subjectKeyIdentifier [1] EXPLICIT SubjectPublicKeyInfo OPTIONAL,
                            -- As defined in RFC 5280
   certificate          [2] EXPLICIT Certificate OPTIONAL
                            -- As defined in RFC 5280
}
```

An Evidence message is composed of a protected section known as the To-Be-Signed (TBS) section where the Evidence reported by the Attesting Environment is assembled. The integrity of the TBS section is ensured with one or multiple cryptographic signatures over the content of this section. There is a provision to carry X.509 certificates supporting each signature. The SEQUENCE OF SignatureBlock allows for both multi-algorithm protection and for counter-signatures of the Evidence. In an effort to keep the Evidence format simple, distinguishing between these two cases is left up to Verifier policy, potentially by making use of the certificates that accompany each signature.

This design also does not prevent an attacker from removing, adding or re-ordering signatures without leaving trace. This is discussed as part of the security considerations in Section 10.3.

The TBS section is composed of a version number, to ensure future extensibility, and a sequence of reported entities. For compliance with this specification, TbsEvidence.version MUST be 1. This envelope format is not extensible; future specifications which make compatibility-breaking changes MUST increment the version number.

A SignatureBlock is included for each signature submitted against the TBS section. The SignatureBlock includes the signature algorithm (signatureAlgorithm) and the signature itself (signatureValue). It also includes information to identify the authority that provided the signature which is the structure SignerIdentifier (sid). The signer identifier includes a combination of X.509 certificate, SubjectPublicKeyInfo (SPKI) and/or key identifier (keyId). It is expected that a X.509 certificate will be generally used, as it provides the public key needed to verify the signature and clearly identifies the subject that provided the signature. The SPKI and keyId are allowed to support environments where X.509 certificates are not used.

The optional certificate list provided in Evidence.intermediateCertificates enables the insertion of X.509 certificates to support trusting the signatures found in signature blocks. This information is intended to provide the certificates required by the Verifier to validate the endorsement on the certificates included with the signatures. intermediateCertificates MAY include any or all intermediate CA certificates needed to build paths. It is not required to include trust anchors. Order is not significant.

As described in Section 4, the TbsEvidence is a collection of entities. Each entity is associated with a type that defines its class. The entity types are represented by object identifiers (OIDs). The following ASN.1 definition defines the structures associated with entities:

```
ReportedEntity ::= SEQUENCE {
    entityType  OBJECT IDENTIFIER,
    claims      SEQUENCE SIZE (1..MAX) OF ReportedClaim
}

id-evidence                    OBJECT IDENTIFIER ::= { 1 2 3 999 }
id-evidence-entity             OBJECT IDENTIFIER ::= { id-evidence 0 }
id-evidence-entity-transaction OBJECT IDENTIFIER ::= { id-evidence-entity 0 }
id-evidence-entity-platform    OBJECT IDENTIFIER ::= { id-evidence-entity 1 }
id-evidence-entity-key         OBJECT IDENTIFIER ::= { id-evidence-entity 2 }
```

In turn, entities are composed of a collection of claims. Each claim is composed of a type and a value. The claim types are represented by object identifiers (OIDs). The following ASN.1 definition defines the structures associated with claims:

```
ReportedClaim ::= SEQUENCE {
    claimType      OBJECT IDENTIFIER,
    value          ClaimValue OPTIONAL
}

ClaimValue ::= CHOICE {
   bytes       [0] IMPLICIT OCTET STRING,
   utf8String  [1] IMPLICIT UTF8String,
   bool        [2] IMPLICIT BOOLEAN,
   time        [3] IMPLICIT GeneralizedTime,
   int         [4] IMPLICIT INTEGER,
   oid         [5] IMPLICIT OBJECT IDENTIFIER
}
```

Each claim type SHOULD be associated with a single entity type. Therefore, it is encouraged to define claim types grouped with their respective entity type.

The type of a claim value is dictated by the claim type. When a claim type is defined, the definition must include the type of the value, its semantic and interpretation.

The remainder of this section describes the entity types and their associated claims.

5.1. Platform Entity

A platform entity reports information about the device where the Evidence is generated and is composed of a collection of claims that are global to the Target Environment. It is associated with the type identifier id-evidence-entity-platform.

A platform entity, if provided, MUST be included only once within the reported entities. If a Verifier encounters multiple entities of type id-evidence-entity-platform, it MUST reject the Evidence as malformed.

The following table lists the claims for a platform entity (platform claims) defined within this specification. In cases where the claim is borrowed from another specification, the "Reference" column refers to the specification where the semantics for the claim value can be found. Claims defined in this specification have further details below.

Claim Type	Claim Value	Reference	Multiple?	OID
vendor	utf8String	RFCthis	No	id-evidence-claim-platform-vendor
oemid	bytes	[RFC9711]	No	id-evidence-claim-platform-oemid
hwmodel	bytes	[RFC9711]	No	id-evidence-claim-platform-hwmodel
hwversion	utf8String	[RFC9711]	No	id-evidence-claim-platform-hwversion
hwserial	utf8String	RFCthis	No	id-evidence-claim-platform-hwserial
swname	utf8String	[RFC9711]	No	id-evidence-claim-platform-swname
swversion	utf8String	[RFC9711]	No	id-evidence-claim-platform-swversion
dbgstat	int	[RFC9711]	No	id-evidence-claim-platform-debugstat
uptime	int	[RFC9711]	No	id-evidence-claim-platform-uptime

bootcount	int	[RFC9711]	No	id-evidence-claim-platform-bootcount
fipsboot	bool	[FIPS140-3]	No	id-evidence-claim-platform-fipsboot
fipsver	utf8String	[FIPS140-3]	No	id-evidence-claim-platform-fipsver
fipslevel	int	[FIPS140-3]	No	id-evidence-claim-platform-fipslevel
fipsmodule	utf8String	[FIPS140-3]	No	id-evidence-claim-platform-fipsmodule

Table 1

Each claim defined in the table above is described in the following sub-sections.

5.1.1. vendor

A human-readable string that reports the name of the device’s manufacturer. This field is for informational purposes only and should not be used in any automated mechanism to compare the Evidence. For the purposes of comparison, the claims oemid and hwmodel should be used.

If the device is submitted to FIPS validation, this string should correspond to the vendor field of the submission.

5.1.2. oemid, hwmodel, hwversion, swname, swversion, dbgstat, uptime, bootcount

These claims are defined in [RFC9711] and are reused in this specification for interoperability. Small descriptions are offered for each to ease the reading of this specification. In case of confusion between the description offered here and the one in [RFC9711], the definition offered in the latter shall prevail.

The claim "oemid" uniquely identifies the Original Equipment Manufacturer (OEM) of the HSM. This is a sequence of bytes and is not meant to be a human readable string.

The claim "hwmodel" differentiates models, products, and variants manufactured by a particular OEM. A model must be unique within a given "oemid". This is a sequence of bytes and is not meant to be a human readable string.

The claim "hwversion" is a text string reporting the version of the hardware. This claim must be interpreted along with the claim "hwmodel".

The claim "swname" is a text string reporting the name of the firmware running on the platform.

The claim "swversion" differentiates between the various revisions of a firmware offered for the platform. This is a string that is expected to be human readable.

The claim "dbgstat" refers to the state of the debug facilities offered by the HSM. This is an integer value describing the current state as described in [RFC9711].

The claim "uptime" reports the number of seconds that have elapsed since the HSM was last booted.

The claim "bootcount" reports the number of times the HSM was booted.

5.1.3. hwserial

A human-readable string that reports the serial number of the hardware module. This serial number often matches the number engraved on the case or on an applied sticker.

5.1.4. fipsboot, fipsver, fipslevel and fipsmodule

FIPS 140-3 CMVP validation places stringent requirements on the mode of operation of the device and the cryptography offered by the module, including only enabling FIPS-approved algorithms, certain requirements on entropy sources, and extensive start-up self-tests. FIPS 140-3 offers compliance levels 1 through 4 with increasingly strict requirements. Many HSMs include a configuration setting that allows the device to be taken out of FIPS mode and thus enable additional functionality or performance, and some offer configuration settings to change between compliance levels.

The boolean claim fipsboot indicates whether the device is currently operating in FIPS mode. When the claim value is "true", the HSM is running in compliance with the FIPS 140 restrictions. Among other restrictions, it means that only FIPS-approved algorithms are available. If the value of this claim is "false", then the HSM is not restricted to the behavior limited by compliance.

The textual claim fipsver indicates the version of the FIPS CMVP specification with which the device’s operational mode is compliant. At the time of writing, the strings "FIPS 140-2" or "FIPS 140-3" SHOULD be used.

The integer claim fipslevel indicates the compliance level to which the device is currently operating and MUST only be 1, 2, 3, or 4. The fipslevel claim has no meaning if fipsboot is absent or false.

The claim fipsmodule is a textual field used to represent the name of the module that was submitted to CMVP for validation. The information derived by combining this claim with the vendor name shall be sufficient to find the associated records in the CMVP database.

The FIPS status information found in Evidence indicates only the mode of operation of the device and is not authoritative of its validation status. This information is available on the NIST CMVP website or by contacting the device vendor. As an example, some devices may have the option to enable FIPS mode in configuration even if the vendor has not submitted this model for validation. As another example, a device may be running in a mode consistent with FIPS Level 3 but the device was only validated and certified to Level 2. A Relying Party wishing to know the validation status of the device MUST couple the device state information contained in the Evidence with a valid FIPS CMVP certificate for the device.

5.2. Key Entity

A key entity is associated with the type id-evidence-entity-key. Each instance of a key entity represents a different addressable key found in the Target Environment. There can be multiple key entities found in Evidence, but each reported key entity MUST describe a different key from the Target Environment. Two key entities may represent the same underlying cryptographic key (keys with the exact same value) but they must be different portions of the Target Environment.

A key entity is composed of a collection of claims relating to the cryptographic key. At minimum, a key entity MUST report the claim "identifier" to uniquely identify this cryptographic key from any others found in the same Target Environment.

A Verifier that detects Evidence with multiple key entities referring to the same addressable key MUST reject the Evidence.

The following table lists the claims for a key entity defined within this specification. The "Reference" column refers to the specification where the semantics for the claim can be found.

Claim Type	Claim Value	Reference	Multiple?	OID
identifier	utf8String	RFCthis	Yes	id-evidence-claim-key-identifier
spki	bytes	RFCthis	No	id-evidence-claim-key-spki
extractable	bool	[PKCS11]	No	id-evidence-claim-key-extractable
sensitive	bool	[PKCS11]	No	id-evidence-claim-key-sensitive
never-extractable	bool	[PKCS11]	No	id-evidence-claim-key-never-extractable
local	bool	[PKCS11]	No	id-evidence-claim-key-local
expiry	time	RFCthis	No	id-evidence-claim-key-expiry
purpose	bytes	RFCthis	No	id-evidence-claim-key-purpose

Table 2

An attestation key might be visible to a client of the device and be reported along with other cryptographic keys. Therefore, it is acceptable to include a key entity providing claims about an attestation key like any other cryptographic key. An implementation MAY reject the generation of Evidence if it relates to an attestation key.

5.2.1. identifier

A human-readable string that uniquely identifies the cryptographic key. This value often contains a UUID but could also have a numeric value expressed as text or any other textual description.

This claim MAY be repeated as some environments have more than one way to refer to a cryptographic key.

5.2.2. spki

The value of this claim contains the DER-encoded field SubjectPublicKeyInfo (see [RFC5280]) associated with the cryptographic key.

5.2.3. extractable, sensitive, never-extractable, local

These claims are defined as key attributes in [PKCS11] and reused in this specification for interoperability. Small descriptions are offered for each to ease the reading of this specification. In case of confusion between the description offered here and the one in [PKCS11], the definition offered in the latter shall prevail.

The claim "extractable" indicates that the key can be exported from the HSM. Corresponds directly to the attribute CKA_EXTRACTABLE found in PKCS#11.

The claim "sensitive" indicates that the value of key cannot leave the HSM in plaintext. Corresponds directly to the attribute CKA_SENSITIVE found in PKCS#11.

The claim "never-extractable" indicates if the key was never extractable from the HSM throughout the life of the key. Corresponds directly to the attribute CKA_NEVER_EXTRACTABLE found in PKCS#11.

The claim "local" indicates whether the key was generated locally or imported. Corresponds directly to the attribute CKA_LOCAL found in PKCS#11.

5.2.4. expiry

Reports a time after which the key is not to be used. The device MAY enforce this policy based on its internal clock.

Note that security considerations should be taken relating to HSMs and their internal clocks. See Section 10.7.

5.2.5. purpose

Reports the key capabilities associated with the subject key. Since multiple capabilities can be associated with a single key, the value of this claim is a list of capabilities, each reported as an object identifier (OID).

The value of this claim is the DER encoding of the following structure:

<CODE STARTS>

```
EvidenceKeyCapabilities ::= SEQUENCE OF OBJECT IDENTIFIER
```

<CODE ENDS>

The following table describes the key capabilities defined in this specification. The key capabilities offered are based on key attributes provided by PKCS#11. Each capability is assigned an object identifier (OID).

Capability	PKCS#11	OID
encrypt	CKA_ENCRYPT	id-evidence-key-capability-encrypt
decrypt	CKA_DECRYPT	id-evidence-key-capability-decrypt
wrap	CKA_WRAP	id-evidence-key-capability-wrap
unwrap	CKA_UNWRAP	id-evidence-key-capability-unwrap
sign	CKA_SIGN	id-evidence-key-capability-sign
sign-recover	CKA_SIGN_RECOVER	id-evidence-key-capability-sign-recover
verify	CKA_VERIFY	id-evidence-key-capability-verify
verify-recover	CKA_VERIFY_RECOVER	id-evidence-key-capability-verify-recover
derive	CKA_DERIVE	id-evidence-key-capability-derive

Table 3

The use of an object identifier to report a capability allows third parties to extend this list to support implementations that have other key capabilities.

5.3. Transaction Entity

A transaction entity is associated with the type id-evidence-entity-transaction. This is a logical entity and does not relate to any state found in the Target Environment. Instead, it groups together claims that relate to the request of generating the Evidence.

For example, it is possible to include a nonce as part of the request to produce Evidence. This nonce is repeated as part of the Evidence to support the freshness of the Evidence. The nonce is not related to any element in the Target Environment and the transaction entity is used to gather those values into claims.

A transaction entity, if provided, MUST be included only once within the reported entities. If a Verifier detects multiple entities of type id-evidence-entity-transaction, it MUST reject the Evidence.

The following table lists the claims for a transaction entity defined within this specification. The "Reference" column refers to the specification where the semantics for the claim value can be found.

Claim Type	Claim Value	Reference	Multiple?	OID
nonce	bytes	[RFC9711]	No	id-evidence-claim-transaction-nonce
timestamp	time	[RFC9711]	No	id-evidence-claim-transaction-timestamp
ak-spki	bytes	RFCthis	Yes	id-evidence-claim-transaction-ak-spki

Table 4

5.3.1. nonce

The claim nonce is used to provide "freshness" quality as to the generated Evidence. A Presenter requesting Evidence MAY provide a nonce value as part of the request. This nonce value, if specified, SHOULD be repeated in the generated Evidence as a claim within the transaction entity.

This claim is similar to the "eat_nonce" as defined in [RFC9711]. According to that specification, this claim may be specified multiple times with different values. However, within the scope of this specification, the "nonce" value can be specified only once within a transaction.

5.3.2. timestamp

The time at which the Evidence was generated, according to the internal system clock of the Attesting Environment. This is similar to the "iat" claim in [RFC9711].

Note that security considerations should be taken relating to the evaluation of timestamps generated by HSMs. See Section 10.7.

5.3.3. ak-spki

This claim contains the encoded Subject Public Key Information (SPKI) for the attestation key used to sign the Evidence. The definition and encoding for SPKIs are defined in X.509 certificates ([RFC5280]).

This transaction claim is used to bind the content of the Evidence with the key(s) used to sign that Evidence. The importance of this binding is discussed in Section 10.3.

This claim can be reported multiple times. Each included claim MUST refer to a different attestation key. In other words, this claim should be repeated only if multiple attestation keys are used to sign the Evidence.

5.4. Additional Entity and Claim Types

It is expected that HSM vendors will register additional Entity and Claim types by assigning OIDs from their own proprietary OID arcs to hold data describing additional proprietary key properties.

When new entity and claim types are used, documentation similar to the one produced in this specification SHOULD be distributed to explain the semantics of the claims and the frequency that values can be provided.

See Section 7.2, Section 7.3 and Section 10.1 for handling of unrecognized custom types.

5.5. Encoding

The structure Evidence is to be DER encoded [X.690].

If a textual representation is required, then the DER encoding MAY be subsequently encoded into Standard Base64 as defined in [RFC4648].

PEM-like representations are also allowed where a MIME-compliant Base64 transformation of the DER encoding is used, provided that the header label is "EVIDENCE". For example:

```
-----BEGIN EVIDENCE-----
(...)
-----END EVIDENCE-----
```

6. Signing and Verification Procedures

The SignatureBlock.signatureValue signs over the DER-encoded to-be-signed Evidence data Evidence.tbs and MUST be validated with the subject public key of the end entity X.509 certificate contained in the SignerIdentifier.certificate. Verifiers MAY also use Evidence.intermediateCertificates to build a certification path to a trust anchor.

Note that the structure Evidence MAY contain zero or more SignatureBlocks. A structure Evidence with zero SignatureBlocks is unsigned and unprotected; Verifiers MUST treat it as untrusted and MUST NOT rely on its claims.

More than one SignatureBlock MAY be used to convey a number of different semantics. For example, the HSM's Attesting Environment might hold multiple Attestation Keys using different cryptographic algorithms in order to provide resilience against cryptographic degradation. In this case a Verifier would be expected to validate all SignatureBlocks. Alternatively, the HSM's Attesting Service may hold multiple Attestation Keys (or multiple X.509 certificates for the same key) from multiple operational environments to which it belongs. In this case a Verifier would be expected to only validate the SignatureBlock corresponding to its own environment. Alternatively, multiple SignatureBlocks could be used to convey counter-signatures from external parties, in which case the Verifier will need to be equipped with environment-specific verification logic. Multiple of these cases, and potentially others, could be supported by a single Evidence object.

Note that each SignatureBlock is a fully detached signature over the tbs content with no binding between the signed content and the SignatureBlocks meaning that a third-party can add a counter-signature of the Evidence after the fact, or an attacker can remove a SignatureBlock without leaving any artifact. See Section 10.3 for further discussion.

If any transaction.ak-spki claims are present, the Verifier SHOULD verify that each SignerIdentifier' s SubjectPublicKeyInfo (or the SPKI of its certificate) matches at least one ak-spki value.

7. Attestation Requests

This section is informative in nature and implementers of this specification do not need to adhere to it. The aim of this section is to provide a standard interface between a Presenter and an HSM producing Evidence. The authors hope that this standard interface will yield interoperable tools between offerings from different vendors.

The interface presented in this section might be too complex for manufacturers of HSMs with limited capabilities such as smartcards or personal ID tokens. For devices with limited capabilities, a fixed Evidence endorsed by the vendor might be installed during manufacturing. Other approaches for constrained HSMs might be to report entities and claims that are fixed or offer limited variations.

On the other hand, an enterprise-grade HSM with the capability to hold a large number of private keys is expected to be capable of generating Evidence catered to the specific constraints imposed by a Verifier and without exposing extraneous information. The aim of the request interface is to provide the means to select and report specific information in the generated Evidence.

This section introduces the role of "Presenter" as shown in Figure 1. The Presenter is the role that initiates the generation of Evidence. Since HSMs are generally servers (client/server relationship) or peripherals (controller/peripheral relationship), a Presenter is required to launch the process of creating the Evidence and capturing it to forward it to the Verifier.

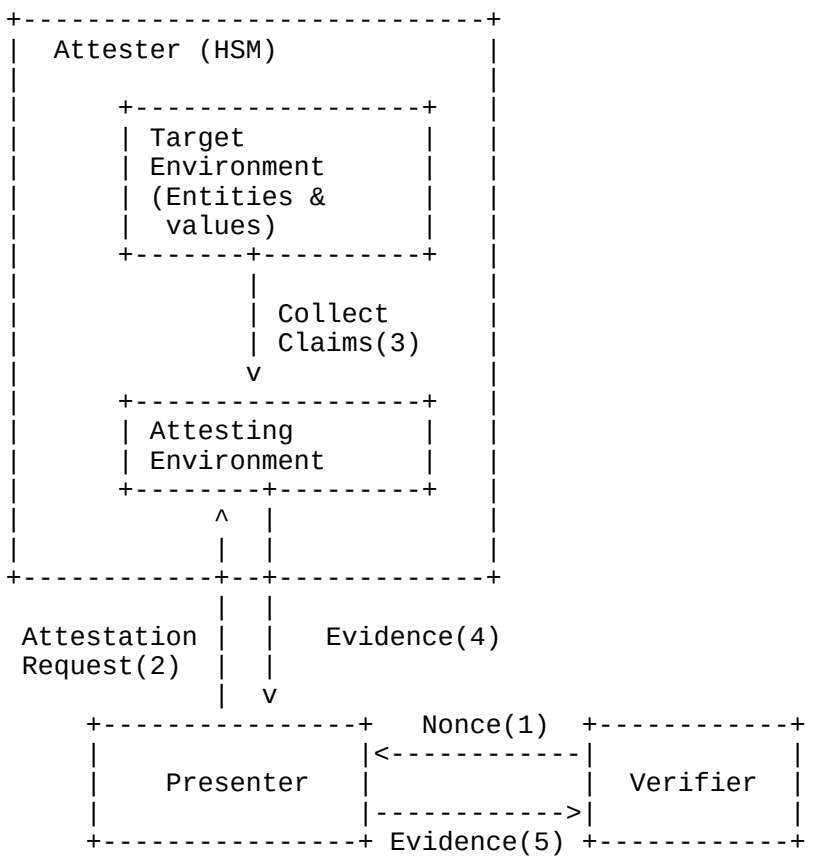

Figure 1: Architecture

The process of generating Evidence generally starts at the Verifier with the generation of a nonce. The nonce is used to ensure freshness of the Evidence and this quality is guaranteed by the Verifier. Therefore, if a nonce is used, it must be provided to the Presenter by the Verifier (1).

An Attestation Request (request) is assembled by the Presenter and submitted to the HSM (2). The Attesting Environment parses the request and collects the appropriate measurements from the Target Environment (3).

In the previous figure, the HSM is represented as being composed of an Attesting Environment and a Target Environment. This representation is offered as a simplified view and implementations are not required to adhere to this separation of concerns.

The Attesting Environment produces Evidence based on the collected information and returns it to the Presenter for distribution (4). Finally, the Presenter forwards the Evidence to the Verifier (5).

The aim of the figure is to depict the position of the Presenter as an intermediate role between the Attester and the Verifier. The role of "Presenter" is privileged as it controls the claims included in the Evidence being generated by the Attester. However, the role is not "trusted" as the Verifier does not have to take into account the participation of the Presenter as part of the function of appraising the Evidence.

The attestation request, shown in the figure, consists of a structure TbsEvidence containing one ReportedEntity for each entity expected to be included in the Evidence produced by the HSM.

Each instance of ReportedEntity included in the request is referred to as a requested entity. A requested entity contains a number of instances of ReportedClaim known as requested claims. The collection of requested entities and requested claims represent the information desired by the Presenter.

In most cases the value of a requested claim should be left unspecified by the Presenter. In the process of generating the Evidence, the values of the desired claims are measured by the Attesting Environment within the HSM and reported accordingly. For the purpose of creating a request, the Presenter does not specify the value of the requested claims and leaves them empty. This is possible because the definition of the structure ReportedClaim specifies the element value as optional.

On the other hand, there are circumstances where the value of a requested claim should be provided by the Presenter. For example, when a particular cryptographic key is to be included in the Evidence, the request must include a key entity with one of the "identifier" claim set to the value corresponding to the desired key.

Some instances of ReportedEntity, such as those representing the platform or the transaction, do not need identifiers as the associated elements are implicit in nature. Custom entity types might need selection during an attestation request and related documentation should specify how this is achieved.

The instance of TbsEvidence is unsigned and does not provide any means to maintain integrity when communicated from the Presenter to the HSM. These details are left to the implementer. However, it is worth pointing out that the structure offered by Evidence could be reused by an implementer to provide those capabilities, as described in Section 10.5.

7.1. Requested Claims with Specified Values

This section deals with the requested claims specified in this document where a value should be provided by a Presenter. In other words, this section defines all requested claims that should set a value in the structure ReportedClaim. Requested claims not covered in this sub-section should not have a specified value (left empty).

Since this section is non-normative, implementers may deviate from those recommendations.

7.1.1. Key Identifiers

A Presenter may choose to select which cryptographic keys are reported as part of the generated Evidence. For each selected cryptographic key, the Presenter includes a requested entity of type id-evidence-entity-key. Among the requested claims for this entity, the Presenter includes one claim with the type id-evidence-claim-key-identifier. The value of this claim should be set to the utf8String that represents the identifier for the specific key.

An HSM receiving an attestation request which selects a key via this approach SHOULD fail the transaction if it cannot find the cryptographic key associated with the specified identifier.

7.1.2. Nonce

A Presenter may choose to include a nonce as part of the attestation request. When producing the Evidence, the HSM repeats the nonce that was provided as part of the request.

When providing a nonce, a Presenter includes, in the attestation request, an entity of type id-evidence-entity-transaction with a claim of type id-evidence-claim-transaction-nonce. This claim is set with the value of the nonce as "bytes".

It is important to note that the Presenter, as an untrusted participant, should not be generating the value for the nonce. In fact, the nonce should be generated by the Verifier so that the freshness of the Evidence can be trusted by the Verifier.

7.1.3. Custom Key Selection

An implementer might desire to select multiple cryptographic keys based on a shared attribute. A possible approach is to include a single request entity of type id-evidence-entity-key including a claim with a set value. This claim would not be related to the key identifier as this is unique to each key. A HSM supporting this scheme could select all the cryptographic keys matching the specified claim and report them in the generated Evidence.

This is a departure from the base request interface, as multiple key entities are reported from a single requested entity.

More elaborate selection schemes can be envisaged where multiple requested claims specifying values would be tested against cryptographic keys. Whether these claims are combined in a logical "and" or in a logical "or" would need to be specified by the implementer.

7.1.4. Custom Transaction Entity Claims

The extensibility offered by the proposed request interface allows an implementer to add custom claims to the transaction entity in order to influence the way that the Evidence generation is performed.

In such an approach, a new custom claim for requested entities of type "transaction" is defined. Then, a claim of that type is included in the attestation request (as part of the transaction entity) while specifying a value. This value is considered by the HSM while generating the Evidence.

7.1.5. Reporting of Attestation Keys

There is a provision for the Attesting Environment to report the attestation key(s) used during the generation of the Evidence. To this end, the transaction claim "ak-spki" is used.

A Presenter invokes this provision by submitting an attestation request with a transaction claim of type "ak-spki" with a non-specified value (left empty).

In this case, the Attesting Environment adds a transaction claim of type "ak-spki" for each Attestation Key used to sign the Evidence. The value of this claim is an octet string (bytes) which is the encoding of the Subject Public Key Information (SPKI) associated with the Attestation Key. Details on SPKIs and their encoding can be found in X.509 certificates ([RFC5280]).

This reporting effectively binds the signature blocks to the content (see Section 10.3).

7.2. Processing an Attestation Request

This sub-section deals with the rules that should be considered when an Attesting Environment processes a request to generate Evidence. This section is non-normative and implementers MAY choose to not follow these recommendations.

These recommendations apply to any attestation request schemes and are not restricted solely to the request interface proposed here.

An Attesting Environment SHOULD fail an attestation request if it contains an unrecognized entity type. This is to ensure that all the semantics expected by the Presenter are fully understood by the Attesting Environment.

An Attesting Environment MUST fail an attestation request if it contains a requested claim with an unrecognized type with a specified a value (not empty). This represents a situation where the Presenter is selecting specific information that is not understood by the Attesting Environment.

An Attesting Environment SHOULD ignore unrecognized claim types in an attestation request. In this situation, the Attesting Environment SHOULD NOT include the claim as part of the response. This guidance is to increase the likelihood of interoperability between tools of various vendors.

An Attesting Environment MUST NOT include entities and claims in the generated Evidence if these entities and claims were not specified as part of the request. This is to give control to the Presenter as to what information is disclosed by the Attesting Environment.

An Attesting Environment MUST fail an attestation request if the Presenter does not have the appropriate access rights to the entities or claims included in the request.

7.3. Verification by Presenter

This sub-section deals with the rules that should be considered when a Presenter receives Evidence from the Attester (the HSM) prior to distribution. This section is non-normative and implementers MAY choose to not follow these recommendations.

These recommendations apply to any Evidence and are not restricted solely to Evidence generated from the proposed request interface.

A Presenter MUST review the Evidence produced by an Attester for fitness prior to distribution.

A Presenter MUST NOT disclose Evidence if it contains information it cannot parse. This restriction applies to entity types and claim types. This is to ensure that the information provided by the Attester can be evaluated by the Presenter.

A Presenter MUST NOT disclose Evidence if it contains entities others than the ones that were requested of the Attester. This is to ensure that only the selected entities are exposed to the Verifier.

A Presenter MUST NOT disclose Evidence if it contains an entity with a claim that was not requested of the Attester. This is to ensure that only the selected information is disclosed to the Verifier.

Further privacy concerns are discussed in Section 10.4.

8. ASN.1 Module

<CODE STARTS>

```
================ NOTE: '\' line wrapping per RFC 8792 ================

PKIX-Evidence-2025
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-pkix-evidence-2025(TBDMOD) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

Evidence ::= SEQUENCE {
    tbs                           TbsEvidence,
    signatures                    SEQUENCE SIZE (0..MAX) OF \
                                                    SignatureBlock,
    intermediateCertificates  [0] SEQUENCE OF Certificate OPTIONAL
                                  -- As defined in RFC 5280
}

TbsEvidence ::= SEQUENCE {
    version INTEGER,
    reportedEntities SEQUENCE SIZE (1..MAX) OF ReportedEntity
}

ReportedEntity ::= SEQUENCE {
    entityType         OBJECT IDENTIFIER,
    claims             SEQUENCE SIZE (1..MAX) OF ReportedClaim
```

```
   }

   ReportedClaim ::= SEQUENCE {
       claimType          OBJECT IDENTIFIER,
       value              ClaimValue OPTIONAL
   }

   ClaimValue ::= CHOICE {
      bytes       [0] OCTET STRING,
      utf8String  [1] UTF8String,
      bool        [2] BOOLEAN,
      time        [3] GeneralizedTime,
      int         [4] INTEGER,
      oid         [5] OBJECT IDENTIFIER,
      null        [6] NULL
   }

   SignatureBlock ::= SEQUENCE {
      sid                  SignerIdentifier,
      signatureAlgorithm   AlgorithmIdentifier,
      signatureValue       OCTET STRING
   }

   SignerIdentifier ::= SEQUENCE {
      keyId                [0] EXPLICIT OCTET STRING OPTIONAL,
      subjectKeyIdentifier [1] EXPLICIT SubjectPublicKeyInfo OPTIONAL,
                               -- As defined in RFC 5280
      certificate          [2] EXPLICIT Certificate OPTIONAL
                               -- As defined in RFC 5280
   }

   EvidenceKeyCapabilities ::= SEQUENCE OF OBJECT IDENTIFIER

   id-evidence OBJECT IDENTIFIER ::= { 1 2 3 999 }

   id-evidence-entity             OBJECT IDENTIFIER ::= { id-evidence \
                                                                  0 }
   id-evidence-entity-transaction OBJECT IDENTIFIER ::= { id-evidence-\
                                                          entity 0 }
   id-evidence-entity-platform    OBJECT IDENTIFIER ::= { id-evidence-\
                                                          entity 1 }
   id-evidence-entity-key         OBJECT IDENTIFIER ::= { id-evidence-\
                                                          entity 2 }

   id-evidence-claim    OBJECT IDENTIFIER ::= { id-evidence 1 }

   id-evidence-claim-transaction           OBJECT IDENTIFIER ::= { id-\
                                                      evidence-claim 0 }
```

```
   id-evidence-claim-transaction-nonce     OBJECT IDENTIFIER ::= { id-\
                                          evidence-claim-transaction 0 }
   id-evidence-claim-transaction-timestamp OBJECT IDENTIFIER ::= { id-\
                                          evidence-claim-transaction 1 }
   id-evidence-claim-transaction-ak-spki   OBJECT IDENTIFIER ::= { id-\
                                          evidence-claim-transaction 2 }

   id-evidence-claim-platform            OBJECT IDENTIFIER ::= { id-\
                                                    evidence-claim 1 }
   id-evidence-claim-platform-vendor     OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 0 }
   id-evidence-claim-platform-oemid      OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 1 }
   id-evidence-claim-platform-hwmodel    OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 2 }
   id-evidence-claim-platform-hwversion  OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 3 }
   id-evidence-claim-platform-hwserial   OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 4 }
   id-evidence-claim-platform-swname     OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 5 }
   id-evidence-claim-platform-swversion  OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 6 }
   id-evidence-claim-platform-debugstat  OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 7 }
   id-evidence-claim-platform-uptime     OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 8 }
   id-evidence-claim-platform-bootcount  OBJECT IDENTIFIER ::= { id-\
                                        evidence-claim-platform 9 }
   id-evidence-claim-platform-usermods   OBJECT IDENTIFIER ::= { id-\
                                       evidence-claim-platform 10 }
   id-evidence-claim-platform-fipsboot   OBJECT IDENTIFIER ::= { id-\
                                       evidence-claim-platform 11 }
   id-evidence-claim-platform-fipsver    OBJECT IDENTIFIER ::= { id-\
                                       evidence-claim-platform 12 }
   id-evidence-claim-platform-fipslevel  OBJECT IDENTIFIER ::= { id-\
                                       evidence-claim-platform 13 }
   id-evidence-claim-platform-fipsmodule OBJECT IDENTIFIER ::= { id-\
                                       evidence-claim-platform 14 }


   id-evidence-claim-key                   OBJECT IDENTIFIER ::= { id-\
                                                    evidence-claim 2 }
   id-evidence-claim-key-identifier        OBJECT IDENTIFIER ::= { id-\
                                              evidence-claim-key 0 }
   id-evidence-claim-key-spki              OBJECT IDENTIFIER ::= { id-\
                                              evidence-claim-key 1 }
   id-evidence-claim-key-extractable       OBJECT IDENTIFIER ::= { id-\
```

```
                                                evidence-claim-key 2 }
   id-evidence-claim-key-sensitive         OBJECT IDENTIFIER ::= { id-\
                                                evidence-claim-key 3 }
   id-evidence-claim-key-never-extractable OBJECT IDENTIFIER ::= { id-\
                                                evidence-claim-key 4 }
   id-evidence-claim-key-local             OBJECT IDENTIFIER ::= { id-\
                                                evidence-claim-key 5 }
   id-evidence-claim-key-expiry            OBJECT IDENTIFIER ::= { id-\
                                                evidence-claim-key 6 }
   id-evidence-claim-key-purpose           OBJECT IDENTIFIER ::= { id-\
                                                evidence-claim-key 7 }


   id-evidence-key-capability                  OBJECT IDENTIFIER ::= { \
                                                        id-evidence 2 }
   id-evidence-key-capability-encrypt          OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 0 }
   id-evidence-key-capability-decrypt          OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 1 }
   id-evidence-key-capability-wrap             OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 2 }
   id-evidence-key-capability-unwrap           OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 3 }
   id-evidence-key-capability-sign             OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 4 }
   id-evidence-key-capability-sign-recover     OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 5 }
   id-evidence-key-capability-verify           OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 6 }
   id-evidence-key-capability-verify-recover   OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 7 }
   id-evidence-key-capability-derive           OBJECT IDENTIFIER ::= { \
                                          id-evidence-key-capability 8 }

   END
```

   <CODE ENDS>

9. IANA Considerations

   Please replace "RFCthis" with the RFC number assigned to this document.

   The following OIDs are defined in this document and will require IANA registration under the assigned arc:

   * id-evidence

* id-evidence-entity
* id-evidence-entity-transaction
* id-evidence-entity-platform
* id-evidence-entity-key
* Claim OIDs referenced in the Platform, Key, and Transaction tables (e.g., id-evidence-claim-platform-*, id-evidence-claim-key-*, id-evidence-claim-transaction-*).

10. Security Considerations

10.1. Policies relating to Verifier and Relying Party

The generation of Evidence by an HSM is to provide sufficient information to a Verifier and, ultimately, a Relying Party to appraise the Target Environment (the HSM) and make decisions based on this appraisal.

The Appraisal Policy associated with the Verifier influences the generation of the Attestation Results. Those results, in turn, are consumed by the Relying Party to make decisions about the HSM, which might be based on a set of rules and policies. Therefore, the interpretation of the provided Evidence may greatly influence the outcome of some decisions.

A Verifier MAY reject Evidence if it lacks the claims required per the Verifier’s appraisal policy. For example, if a Relying Party mandates a FIPS-certified device, it SHOULD reject Evidence lacking sufficient information to verify the device’s FIPS certification status.

If a Verifier encounters a claim with an unrecognized claim type, it MAY ignore it and treat it as extraneous information. By ignoring a claim, the Verifier may accept Evidence that would be deemed malformed to a Verifier with different policies. However, this approach fosters a higher likelihood of achieving interoperability.

10.2. Simple to Implement

The nature of attestation requires the Attesting Environment to be implemented in an extremely privileged position within the HSM so that it can collect the required measurements such as hardware registers and the user keys. For many HSM architectures, this will place the Attesting Environment inside the "security kernel" and potentially subject to FIPS 140-3 or Common Criteria validation and

change control. For both security and compliance reasons, there is an incentive for the generation and parsing logic to be simple and easy to implement correctly. Additionally, when the data formats contained in this specification are parsed within an HSM boundary -- that would be parsing Evidence produced by a different HSM -- implementers SHOULD opt for simple logic that rejects any data that does not match the expected format, instead of attempting to be flexible.

In particular, the Attesting Environment SHOULD generate Evidence from scratch and avoid copying any content from the request. The Attesting Environment MUST generate Evidence only from information and measurements that are directly observable by it.

10.3. Detached Signatures

The construction of the Evidence structure includes a collection of signature blocks that are not explicitly bound to the content. This approach was influenced by the following motivations:

* Multiple simultaneous signature blocks are desired to support hybrid environments where multiple keys using different cryptographic algorithms are required to support appraisal policies.

* Provide the ability to add counter-signatures without having to define an envelop scheme.

The concept of counter-signatures is important for environments where a number of heterogeneous devices are deployed. In those environments, it is possible for a trusted actor, intermediary between the Attester and the Verifier, to validate the original signature(s) and apply its own afterwards.

The ability to add signature blocks to the Evidence after the original generation by the Attester leads to the unfortunate situation where signature blocks can also be removed without leaving any trace. Therefore, the signature blocks can be deemed as "detachable" or "stapled".

Manipulation of the Evidence after it was generated can lead to undesired outcomes at the Verifier.

Therefore, Verifiers MUST be designed to accept Evidence based on their appraisal policies, regardless of the presence or absence of certain signature(s). Consequently, Verifiers MUST NOT make any inferences based on a missing signature, as the signature could have been removed in transit.

This specification provides the transaction claim "ak-spki" to effectively bind the content with the signature blocks that were inserted by the Attesting Environment. When this claim is provided, it reports the SPKI of one of the attestation keys used by the Attesting Environment to produce the Evidence. This claim is repeated for each of the attestation keys used by the Attesting Environment.

10.4. Privacy

Some HSMs have the capacity of supporting cryptographic keys controlled by separate entities referred to as "tenants", and when the HSM is used in that mode it is referred to as a multi-tenant configuration.

For example, an enterprise-grade HSM in a large multi-tenant cloud service could host TLS keys fronting multiple un-related web domains. Providing Evidence for claims of any one of the keys would involve a Presenter that could potentially access any of the hosted keys. In such a case, privacy violations could occur if the Presenter was to disclose information that does not relate to the subject key.

Implementers SHOULD be careful to avoid over-disclosure of information, for example by authenticating the Presenter as described in Section 10.5 and only returning results for keys and portions of the Target Environment for which it is authorized. In absence of an existing mechanism for authenticating and authorizing administrative connections to the HSM, the attestation request MAY be authenticated by embedding the TbsEvidence of the request inside a Evidence signed with a certificate belonging to the Presenter.

Furthermore, enterprise and cloud-services grade HSMs SHOULD support the full set of attestation request functionality described in Section 7 so that Presenters can fine-tune the content of the generated Evidence such that it is appropriate for the intended Verifier.

10.5. Authenticating and Authorizing the Presenter

The Presenter represents a privileged role within the architecture of this specification as it gets to learn about the existence of user keys and their protection properties, as well as details of the platform. The Presenter is in the position of deciding how much information to disclose to the Verifier, and to request a suitably redacted Evidence from the HSM.

For personal cryptographic tokens it might be appropriate for the attestation request interface to be un-authenticated. However, for enterprise and cloud-services grade HSMs the Presenter SHOULD be authenticated using the HSM's native authentication mechanism. The details are HSM-specific and are thus left up to the implementer. However, it is RECOMMENDED to implement an authorization framework similar to the following.

A Presenter SHOULD be allowed to request Evidence for any user keys which it is allowed to use. For example, a TLS application that is correctly authenticated to the HSM in order to use its TLS keys SHOULD be able to request Evidence related to those same keys without needing to perform any additional authentication or requiring any additional roles or permissions. HSMs that wish to allow a Presenter to request Evidence of keys which is not allowed to use, for example for the purposes of displaying HSM status information on an administrative console or UI, SHOULD have a "Attestation Requester" role or permission and SHOULD enforce the HSM's native access controls such that the Presenter can only retrieve Evidence for keys for which it has visibility.

In the absence of an existing mechanism for authenticating and authorizing administrative connections to the HSM, the attestation request MAY be authenticated by embedding the TbsEvidence of the request inside a strcture Evidence signed with a certificate belonging to the Presenter.

10.6. Proof-of-Possession of User Keys

With asymmetric keys within a Public Key Infrastructure (PKI) it is common to require a key holder to prove that they are in control of the private key by using it. This is called "proof-of-possession (PoP)". This specification intentionally does not provide a mechanism for PoP of user keys and relies on the Presenter, Verifier, and Relying Party trusting the Attester to correctly report the cryptographic keys that it is holding.

It would be trivial to add a PoP Key claim that uses the attested user key to sign over, for example, the Transaction Entity. However, this approach leads to undesired consequences, as explained below.

First, a user key intended for TLS, as an example, SHOULD only be used with the TLS protocol. Introducing a signature oracle whereby the TLS application key is used to sign Evidence could lead to cross-protocol attacks. In this example, an attacker could submit a "nonce" value which is in fact not random but is crafted in such a way as to appear as a valid message in some other protocol context or exploit some other weakness in the signature algorithm.

Second, the Presenter who has connected to the HSM to request Evidence may have permissions to list the requested application keys but not permission to use them, as in the case where the Presenter is an administrative UI displaying HSM status information to a system's administrator or auditor.

Requiring the Attesting Environment to use the reported application keys to generate Evidence could, in some architectures, require the Attesting Environment to resolve complex access control logic and handle complex error conditions, which violates the "simple to implement" design principle outlined in Section 10.2. More discussions on authenticating the Presenter can be found in Section 10.5.

10.7. Timestamps and HSMs

It is common for HSMs to have an inaccurate system clock. Most clocks have a natural drift and must be corrected periodically. HSMs, like any other devices, are subject to these issues.

There are many situations where HSMs can not naturally correct their internal system clocks. For example, consider a HSM hosting a trust anchor and usually kept offline and booted up infrequently in a network without a reliable time management service. Another example is a smart card which boots up only when held against an NFC reader.

When a timestamp generated from a HSM is evaluated, the expected behavior of the system clock SHOULD be considered.

More specifically, the timestamp SHOULD NOT be relied on for establishing the freshness of the Evidence generated by a HSM. Instead, Verifiers SHOULD rely on other provisions such as the "nonce" claim of the "transaction" entity, introduced in this specification.

Furthermore, the internal system clock of HSMs SHOULD NOT be relied on to enforce expiration policies.

11. References

11.1. Normative References

[FIPS140-3] NIST, Information Technology Laboratory, "Security Requirements for Cryptographic Modules", FIPS 140-3, n.d., <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf>.

[I-D.jpfiset-lamps-attestationkey-eku]
Fiset, J., Ounsworth, M., Tschofenig, H., and M. Wiseman, "X.509 Certificate Extended Key Usage (EKU) for Attestation Keys", Work in Progress, Internet-Draft, draft-jpfiset-lamps-attestationkey-eku-02, 1 March 2026, <https://datatracker.ietf.org/doc/html/draft-jpfiset-lamps-attestationkey-eku-02>.

[PKCS11] Bong, D., Cox, T., and OASIS PKCS 11 TC, "PKCS #11 Specification Version 3.1", 11 August 2022, <https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.1/cs01/pkcs11-spec-v3.1-cs01.html>.

[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/rfc/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[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/rfc/rfc5280>.

[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/rfc/rfc8174>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <https://www.rfc-editor.org/rfc/rfc9711>.

[X.690] ITU-T, "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, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690>.

[X680] ITU-T, "Information technology — ASN.1: Specification of basic notation", n.d., <https://www.itu.int/rec/T-REC-X.680>.

[X690] ITU-T, "Information technology — ASN.1 encoding rules: BER, CER, DER", n.d., <https://www.itu.int/rec/T-REC-X.690>.

11.2. Informative References

[CNSA2.0] National Security Agency, "Commercial National Security Algorithm Suite 2.0", n.d., <https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF>.

[CSBR] CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Code Signing Certificates Version 3.8.0", n.d., <https://cabforum.org/working-groups/code-signing/documents/>.

[I-D.fossati-tls-attestation] Tschofenig, H., Sheffer, Y., Howard, P., Mihalcea, I., Deshpande, Y., Niemi, A., and T. Fossati, "Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-fossati-tls-attestation-09, 30 April 2025, <https://datatracker.ietf.org/doc/html/draft-fossati-tls-attestation-09>.

[I-D.ietf-lamps-csr-attestation] Ounsworth, M., Tschofenig, H., Birkholz, H., Wiseman, M., and N. Smith, "Use of Remote Attestation with Certification Signing Requests", Work in Progress, Internet-Draft, draft-ietf-lamps-csr-attestation-22, 11 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-csr-attestation-22>.

[I-D.ietf-rats-msg-wrap] Birkholz, H., Smith, N., Fossati, T., Tschofenig, H., and D. Glaze, "RATS Conceptual Messages Wrapper (CMW)", Work in Progress, Internet-Draft, draft-ietf-rats-msg-wrap-23, 11 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-msg-wrap-23>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/rfc/rfc2986>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/rfc/rfc4211>.

[RFC6024] Reddy, R. and C. Wallace, "Trust Anchor Management Requirements", RFC 6024, DOI 10.17487/RFC6024, October 2010, <https://www.rfc-editor.org/rfc/rfc6024>.

[RFC9019] Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", RFC 9019, DOI 10.17487/RFC9019, April 2021, <https://www.rfc-editor.org/rfc/rfc9019>.

Appendix A. Samples

A reference implementation of this specification can be found at https://github.com/ietf-rats-wg/key-attestation

It produces the following sample Evidence:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

PkixAttestation:
 tbs=TbsPkixAttestation:
  version=2
  reportedEntities=SequenceOf:
   ReportedEntity:
    entityType=1.2.3.999.0.0
    reportedAttributes=SequenceOf:
     ReportedAttribute:
      attributeType=1.2.3.999.1.0.0
      value=AttributeValue:
       bytes=0102030405


   ReportedEntity:
    entityType=1.2.3.999.0.1
    reportedAttributes=SequenceOf:
     ReportedAttribute:
      attributeType=1.2.3.999.1.1.1
      value=AttributeValue:
       utf8String=HSM-123

     ReportedAttribute:
      attributeType=1.2.3.999.1.1.2
      value=AttributeValue:
       bool=True
```

```
        ReportedAttribute:
         attributeType=1.2.3.999.1.1.3
         value=AttributeValue:
          utf8String=Model ABC

        ReportedAttribute:
         attributeType=1.2.3.999.1.1.4
         value=AttributeValue:
          utf8String=3.1.9


      ReportedEntity:
       entityType=1.2.3.999.0.2
       reportedAttributes=SequenceOf:
        ReportedAttribute:
         attributeType=1.2.3.999.1.2.0
         value=AttributeValue:
          utf8String=26d765d8-1afd-4dfb-a290-cf867ddecfa1

        ReportedAttribute:
         attributeType=1.2.3.999.1.2.3
         value=AttributeValue:
          bool=False

        ReportedAttribute:
         attributeType=1.2.3.999.1.2.1
         value=AttributeValue:
          bytes=\
   0x3059301306072a8648ce3d020106082a8648ce3d03010703420004422548f88fb7\
   82ffb5eca3744452c72a1e558fbd6f73be5e48e93232cc45c5b16c4cd10c4cb8d5b8\
                       a17139e94882c8992572993425f41419ab7e90a42a494272


      ReportedEntity:
       entityType=1.2.3.999.0.2
       reportedAttributes=SequenceOf:
        ReportedAttribute:
         attributeType=1.2.3.999.1.2.0
         value=AttributeValue:
          utf8String=49a96ace-e39a-4fd2-bec1-13165a99621c

        ReportedAttribute:
         attributeType=1.2.3.999.1.2.3
         value=AttributeValue:
          bool=True

        ReportedAttribute:
         attributeType=1.2.3.999.1.2.1
```

```
         value=AttributeValue:
          bytes=\
   0x3059301306072a8648ce3d020106082a8648ce3d03010703420004422548f88fb7\
   82ffb5eca3744452c72a1e558fbd6f73be5e48e93232cc45c5b16c4cd10c4cb8d5b8\
                       a17139e94882c8992572993425f41419ab7e90a42a494272


      ReportedEntity:
       entityType=1.2.3.888.0
       reportedAttributes=SequenceOf:
        ReportedAttribute:
         attributeType=1.2.3.888.1
         value=AttributeValue:
          utf8String=partition 1




    signatures=SequenceOf:
     SignatureBlock:
      certChain=SequenceOf:
       Certificate:
        tbsCertificate=TBSCertificate:
         version=v3
         serialNumber=510501933685942792810365453374472870755160518925
         signature=AlgorithmIdentifier:
          algorithm=1.2.840.113549.1.1.11
          parameters=0x0500

         issuer=Name:
          rdnSequence=RDNSequence:
           RelativeDistinguishedName:
            AttributeTypeAndValue:
             type=2.5.4.10
             value=0x0c0449455446
           RelativeDistinguishedName:
            AttributeTypeAndValue:
             type=2.5.4.11
             value=0x0c0452415453
           RelativeDistinguishedName:
            AttributeTypeAndValue:
             type=2.5.4.3
             value=0x0c06414b20525341


         validity=Validity:
          notBefore=Time:
           utcTime=250117171303Z
```

```
          notAfter=Time:
           generalTime=20520604171303Z


         subject=Name:
          rdnSequence=RDNSequence:
           RelativeDistinguishedName:
            AttributeTypeAndValue:
             type=2.5.4.10
             value=0x0c0449455446
           RelativeDistinguishedName:
            AttributeTypeAndValue:
             type=2.5.4.11
             value=0x0c0452415453
           RelativeDistinguishedName:
            AttributeTypeAndValue:
             type=2.5.4.3
             value=0x0c06414b20525341


         subjectPublicKeyInfo=SubjectPublicKeyInfo:
          algorithm=AlgorithmIdentifier:
           algorithm=1.2.840.113549.1.1.1
           parameters=0x0500

          subjectPublicKey=\
   3179526881036662712546805998442714593178454291971073358719080815289 3\
   6065422142080963288830772256071363933627956099976019683120390012513 3\
   9428349101203532726047646450301142882318337709398316574407647199690 0\
   0068924511373955261527953452814577609081331482231201260756773607305 7\
   9368207137330909288490926721109373003007555617978080004381348394580 4\
   3673852453722969649609202093945235393494912138691342219564365300965 3\
   8774370157050711206440175821831476015308127198134081235036566346651 3\
   6208533265342524247069928410336528174613546323161293125978255428205 6\
   9667842318342646457447037125609399476844336456206583416539426479221 1\
                                    6497136978846472730791582076791848960 1

         extensions=Extensions:
          Extension:
           extnID=2.5.29.14
           critical=False
           extnValue=0x04148919595e0ef169f5cbbd47e134fce298cc693091
          Extension:
           extnID=2.5.29.35
           critical=False
           extnValue=0x301680148919595e0ef169f5cbbd47e134fce298cc693091
          Extension:
           extnID=2.5.29.19
```

```
          critical=True
          extnValue=0x30030101ff


       signatureAlgorithm=AlgorithmIdentifier:
        algorithm=1.2.840.113549.1.1.11
        parameters=0x0500

       signature=\
   129777754246317682895425391026533829824317955511461452817501895537 57\
   940982572813264428982985997740595878077027853994515775116752030963 85\
   846965154876580877526985727116774851279501791628486705130288446531 57\
   510109136580166401706084139357801193498669861701480333019557531169 84\
   041271273907756544780231564646860424999020990745523383622980115200 44\
   626010317310350064783875819761023855234905306452542024082619355339 53\
   788737252565842696669185047936744977484555748222380220850547521856 87\
   440807655337724821853332688158460379554906105417720665175648371832 82\
   593957703987473044279033772600410587816837599812311033199334883362 93\
                                                                     25492

      signatureAlgorithm=AlgorithmIdentifier:
       algorithm=1.2.840.113549.1.1.10
       parameters=RSASSA_PSS_params:
        hashAlgorithm=AlgorithmIdentifier:
         algorithm=2.16.840.1.101.3.4.2.1

        maskGenAlgorithm=AlgorithmIdentifier:
         algorithm=1.2.840.113549.1.1.8

        saltLength=20
        trailerField=1


      signatureValue=\
   0xab7fd2b0f854daa4e867fd16955cd3b9910e93b70c7403cfa8077f04193909d14e\
   c6bed859b67476c84cc2c28842b9a087d5c39e11ca95f6961d272d97297cb6ed3c06\
   2717696b032f4bf1f0f41ac20ae9706a8a4c17845ae2512950774173737010d6692c\
   b726d1ab3a022092efcf27f0dd875b62e4df546814186f9e744cc34cf0778c877c57\
   1d006be094aa683a5f66d6816d22dba104334163020c62d81903c41d353eaba94212\
   47fc354fd3288a01921d93014100960324c3122feebfffc1007c83e98136e1b1fca1\
   15835b9e67fa9056f290208fb99e1c8144839a5e13ccb1217dceeecc253fc7785bc8\
                                  308382e052ffb867b40a0cd593176ed6ddc7b0
     SignatureBlock:
      certChain=SequenceOf:
       Certificate:
        tbsCertificate=TBSCertificate:
         version=v3
         serialNumber=43752118382009037811618748949928339462896457144
```

```
          signature=AlgorithmIdentifier:
           algorithm=1.2.840.10045.4.3.2

          issuer=Name:
           rdnSequence=RDNSequence:
            RelativeDistinguishedName:
             AttributeTypeAndValue:
              type=2.5.4.10
              value=0x0c0449455446
            RelativeDistinguishedName:
             AttributeTypeAndValue:
              type=2.5.4.11
              value=0x0c0452415453
            RelativeDistinguishedName:
             AttributeTypeAndValue:
              type=2.5.4.3
              value=0x0c07414b2050323536


          validity=Validity:
           notBefore=Time:
            utcTime=250117171428Z

           notAfter=Time:
            generalTime=20520604171428Z


          subject=Name:
           rdnSequence=RDNSequence:
            RelativeDistinguishedName:
             AttributeTypeAndValue:
              type=2.5.4.10
              value=0x0c0449455446
            RelativeDistinguishedName:
             AttributeTypeAndValue:
              type=2.5.4.11
              value=0x0c0452415453
            RelativeDistinguishedName:
             AttributeTypeAndValue:
              type=2.5.4.3
              value=0x0c07414b2050323536


          subjectPublicKeyInfo=SubjectPublicKeyInfo:
           algorithm=AlgorithmIdentifier:
            algorithm=1.2.840.10045.2.1
            parameters=0x06082a8648ce3d030107
```

```
        subjectPublicKey=\
   570955602335049245889528161855080378129963079292491048478461646606 64\
   888397123390877585670462836285725041261897550020311481127562655774 33\
                                                        3675293173915140722

       extensions=Extensions:
        Extension:
         extnID=2.5.29.14
         critical=False
         extnValue=0x04145b70a79817f79ff637d2f7e3dc446c2109d7bbd4
        Extension:
         extnID=2.5.29.35
         critical=False
         extnValue=0x301680145b70a79817f79ff637d2f7e3dc446c2109d7bbd4
        Extension:
         extnID=2.5.29.19
         critical=True
         extnValue=0x30030101ff


      signatureAlgorithm=AlgorithmIdentifier:
       algorithm=1.2.840.10045.4.3.2

      signature=\
   182167519797146035745575043154801415115532979136731128676399180692 66\
   482180488399040155204078964301310320242448608805836498296670932449 67\
                                    8251807951926726943881617871966843 7

   signatureAlgorithm=AlgorithmIdentifier:
    algorithm=1.2.840.10045.2.1
    parameters=0x06082a8648ce3d030107

   signatureValue=\
   0x3046022100e416af2483667e73345ee297e563cf1639e41ab9bdcd01f98872fddb\
   101e779d022100d06c6e1054292640eea1873230a399af0936760cbfc8023a8a2874\
                                                           f9c5fc5ba8



   DER Base64:
   MIIIszCCAgsCAQIwggIEMCEGBioDh2cAADAXMBUGByoDh2cBAAAECjAxMDIwMzA0MDUw\
   VAYGKgOHZwABMEowEgYHKgOHZwEBAQwHSFNNLTEyMzAMBgcqA4dnAQECAQH/\
   MBQGByoDh2cBAQMMCU1vZGVsIEFCQzAQBgcqA4dnAQEEDAUzLjEuOTCBsgYGKgOHZwAC\
   MIGnMC8GByoDh2cBAgAMJDI2ZDc2NWQ4LTFhZmQtNGRmYi1hMjkwLWNmODY3ZGRlY2Zh\
   MTAMBgcqA4dnAQIDAQEAMGYGByoDh2cBAgEEWzBZMBMGByqGSM49AgEGCCqGSM49AwEH\
   A0IABEIlSPiPt4L/teyjdERSxyoeVY+9b3O+\
   XkjpMjLMRcWxbEzRDEy41bihcTnpSILImSVymTQl9BQZq36QpCpJQnIwgbIGBioDh2cA\
   AjCBpzAvBgcqA4dnAQIADCQ0OWE5NmFjZS1lMzlhLTRmZDItYmVjMS0xMzE2NWE5OTYy\
```

```
   MWMwDAYHKgOHZwECAwEB/\
   zBmBgcqA4dnAQIBBFswWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARCJUj4j7eC/\
   7Xso3REUscqHlWPvW9zvl5I6TIyzEXFsWxM0QxMuNW4oXE56UiCyJklcpk0JfQUGat+\
   kKQqSUJyMB8GBSoDhngAMBYwFAYFKgOGeAEMC3BhcnRpdGlvbiAxMIIGoDCCBHowggNF\
   MIIDQTCCAimgAwIBAgIUWWuyy9RGarWD+\
   k6k4ZswYmQ7cQ0wDQYJKoZIhvcNAQELBQAwLzENMAsGA1UECgwESUVURjENMAsGA1UEC\
   wwEUkFUUzEPMA0GA1UEAwwGQUsgUlNBMCAXDTI1MDExNzE3MTMwM1oYDzIwNTIwNjA0M\
   TcxMzAzWjAvMQ0wCwYDVQQKDARJRVRGMQ0wCwYDVQQLDARSQVRTMQ8wDQYDVQQDDAZBS\
   yBSU0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCw+\
   egZQ6eumJKq3hfKfED4dE/tL4FI5sjqont9ABVI+\
   1GSqyi1bFBgsRjM0THllIdMbKmJtWwnKW8J+5OgNN8y6Xxv8JmM/\
   Y5vQt2lis0fqXmG8UTz0VTWdlAXXmhUs6lSADvAaIe4RVrCsZ97L3ZQTryY7JRVcbB4k\
   hUN3Gp0yg+801SXzoFTTa+UGIRLE66jH51aa5VXu99hnv1OiH8tQrjdi8mH6uG/\
   icq4XuIeNWMF32wHqIOOPvQcWV3M5D2vxJEj702Ku6k9OQXkAo17qRSEonWW4HtLbtmS\
   8He1JNPc/n3dVUm+\
   fM6NoDXPoLP7j55G9zKyqGtGAWXAj1MTAgMBAAGjUzBRMB0GA1UdDgQWBBSJGVleDvFp\
   9cu9R+E0/OKYzGkwkTAfBgNVHSMEGDAWgBSJGVleDvFp9cu9R+E0/\
   OKYzGkwkTAPBgNVHRMBAf8EBTADAQH/\
   MA0GCSqGSIb3DQEBCwUAA4IBAQBmzcTIPYhVNtMdrOb9ee9qYADlTuQl1y1mdrDPcC+\
   zmwZuwKLJu89hvxmFdDrVNc6QsNKnH0fWtMZxU5UQTrqW2Wf0jLY3bjfJkCmTQahOK8X\
   D3oQqfXVKCe+MGFUSh71BUXc4FIQzMJ6phG+5qiCqsD9BL/gFXf4ao+BI4SQhVWi6FR+\
   JOBMxd91DYDyYr6NfddAbzaW7iDoVEWR1pvQAZbycWfv1KIY6ne2yQ0dSedOqIE9Odjq\
   i2QkW4kD7qXRLYKcMPqe1SPao2xoS2Kz8SIdoLInLu7Cb3QC7n/\
   oEbiK4JIVD29giMpudJ8gbBLLjwDrCls0yA+ng8n/\
   wkki0MCsGCSqGSIb3DQEBCjAeoA0wCwYJYIZIAWUDBAIBoQ0wCwYJKoZIhvcNAQEIBII\
   BAKt/0rD4VNqk6Gf9FpVc07mRDpO3DHQDz6gHfwQZOQnRTsa+\
   2Fm2dHbITMLCiEK5oIfVw54RypX2lh0nLZcpfLbtPAYnF2lrAy9L8fD0GsIK6XBqikwX\
   hFriUSlQd0Fzc3AQ1mkstybRqzoCIJLvzyfw3YdbYuTfVGgUGG+\
   edEzDTPB3jId8Vx0Aa+CUqmg6X2bWgW0i26EEM0FjAgxi2BkDxB01PqupQhJH/\
   DVP0yiKAZIdkwFBAJYDJMMSL+6//\
   8EAfIPpgTbhsfyhFYNbnmf6kFbykCCPuZ4cgUSDml4TzLEhfc7uzCU/x3hbyDCDguBS/\
   7hntAoM1ZMXbtbdx7AwggIeMIIBuzCCAbcwggFdoAMCAQICFAep6a/8hKR/\
   Xf8D7fMOi6OQH5W4MAoGCCqGSM49BAMCMDAxDTALBgNVBAoMBElFVEYxDTALBgNVBAsM\
   BFJBVFMxEDAOBgNVBAMMB0FLIFAyNTYwIBcNMjUwMTE3MTcxNDI4WhgPMjA1MjA2MDQx\
   NzE0MjhaMDAxDTALBgNVBAoMBElFVEYxDTALBgNVBAsMBFJBVFMxEDAOBgNVBAMMB0FL\
   IFAyNTYwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARCJUj4j7eC/\
   7Xso3REUscqHlWPvW9zvl5I6TIyzEXFsWxM0QxMuNW4oXE56UiCyJklcpk0JfQUGat+\
   kKQqSUJyo1MwUTAdBgNVHQ4EFgQUW3CnmBf3n/\
   Y30vfj3ERsIQnXu9QwHwYDVR0jBBgwFoAUW3CnmBf3n/\
   Y30vfj3ERsIQnXu9QwDwYDVR0TAQH/BAUwAwEB/\
   zAKBggqhkjOPQQDAgNIADBFAiEAkH8Erj/\
   TLNoEfJIvokEEDVmhH5f7UQHdrrCyQWEhJegCICRsy/1Vqjo3qg/WrHospwcB2PaHYy+\
   FnH79mznqO7jVMBMGByqGSM49AgEGCCqGSM49AwEHBEgwRgIhAOQWrySDZn5zNF7il+\
   VjzxY55Bq5vc0B+Yhy/dsQHnedAiEA0GxuEFQpJkDuoYcyMKOZrwk2dgy/yAI6iih0+\
                                                                  cX8W6g=
```

Appendix B. Acknowledgements

This specification is the work of a design team created by the chairs of the RATS working group. This specification has been developed based on discussions in that design team and also with great amounts of input taken from discussions on the RATS mailing list.

We would like to thank Jeff Andersen for the review comments.

We would like to thank Dave Thaler for his guidance.

Authors' Addresses

Mike Ounsworth
Cryptic Forest Software
Sioux Lookout
Canada
Email: mike@ounsworth.ca

Jean-Pierre Fiset
Crypto4A Inc.
1550A Laperriere Ave
Ottawa, Ontario K1Z 7T2
Canada
Email: jp@crypto4a.com

Hannes Tschofenig
University of Applied Sciences Bonn-Rhein-Sieg
Germany
Email: Hannes.Tschofenig@gmx.net

Henk Birkholz
Fraunhofer SIT
Email: henk.birkholz@ietf.contact

Monty Wiseman
United States of America
Email: mwiseman@computer.org

Ned Smith
Intel Corporation
United States of America
Email: ned.smith@intel.com