RATS
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

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

N. Smith
Intel Corporation

7 July 2025

PKIX Evidence for Remote Attestation of Hardware Security Modules
draft-ietf-rats-pkix-key-attestation-01

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 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

Copyright Notice

Table of Contents

1. Introduction 4
2. Use Cases 5
  2.1. Remote audit of a Hardware Security Module (HSM) 5
  2.2. Key import and HSM clustering 5
  2.3. Attesting subject of a certificate issuance 6
3. Terminology 6
  3.1. Attestation Key Certificate Chain 9

4. Information Model 9
4.1. Entity 10
4.2. Entity Type 10
4.3. Attribute and Attribute Type 11
5. Data Model 12
5.1. Platform Entity 15
5.1.1. vendor 18
5.1.2. oemid, hwmodel, swversion, dbgstat, uptime, bootcount 18
5.1.3. hwserial 19
5.1.4. usermods 19
5.1.5. fipsboot, fipsver and fipslevel 19
5.1.6. envid 20
5.1.7. envdesc 20
5.2. Key Entity 20
5.2.1. identifier 22
5.2.2. spki 22
5.2.3. purpose, extractable, sensitive, never-extractable, local 23
5.2.4. expiry 23
5.2.5. protection 23
5.3. Transaction Entity 24
5.3.1. nonce 25
5.3.2. timestamp 25
5.4. Additional Entity and Attribute Types 25
5.5. Encoding 26
6. Signing Procedure 26
7. Verification Procedure 26
8. Appraisal Policies and Profiles 27
8.1. Key Import into an HSM 27
8.2. CA/Browser Forum Code-Signing 27
9. Attestation Requests 28
9.1. Request Attributes with Specified Values 30
9.1.1. Key Identifiers 31
9.1.2. Nonce 31
9.1.3. Custom Key Selection 31
9.1.4. Custom Transaction Entity Attributes 32
9.2. Processing an Attestation Request 32
9.3. Verification by Presenter 33
10. ASN.1 Module 33
11. IANA Considerations 36
12. Security Considerations 36
12.1. Policies relating to Verifier and Relying Party 36
12.2. Simple to Implement 37
12.3. Detached Signatures 37
12.4. Privacy 38
12.5. Authenticating and Authorizing the Presenter 38
12.6. Proof-of-Possession of User Keys 39

13. References 40
13.1. Normative References 40
13.2. Informative References 41
Appendix A. Samples 42
Appendix B. Acknowledgements 50
Authors' Addresses 50

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 (RATS [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 within the Evidence are about internal entities such as "platforms" and "keys" which are not necessarily distinct from the Attesting Environment. Therefore, although the concept of "measurement" is present within the PKIX environment, it is not always clear that one attesting environment is measuring another distinct target environment the way it is envisioned in the RATS Architecture. Therefore, the emphasis and structure of this specifications is adjusted accordingly. Specifically, this specification assumes that the Attesting Environment and the Target Environment, as outlined in [RFC9334], are the same. This might not be the case for all devices encountered, but is sufficient for the proposed specification.

This specification also aims at providing an extensible framework to encode within 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 application keys are protected in with a certain set of protection 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 proof 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 attestation of any system-level keys required for secure establishment of cluster communication. In this scenario, the Verifier and Relying Party will be the 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 a way to do this 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 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], [codesigningbrsv3.8], 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 such as TPM-like devices, storage properties such as "extractable" may always be true for all keys since the devices are not capable of key export and so the attestation could be essentially a hard-coded template asserting these immutable attributes. However, more complex HSM devices require a more complex key attestation format that encompasses the mutability of these attributes. Also, the client requesting the key attestation might wish to scope-down the content of the key attestation as the HSM contains many keys and only a certain subset are relevant for attesting a given transaction, or only certain claims are relevant. Lack of ability to scope-down the key attestation contents could, in some scenarios, constitute a privacy violation. This motivates the design choice for a key attestation request mechanism. The same objective could have been accomplished via a selective disclosure mechanism. However, since a request is necessary to transmit the attestation nonce to the HSM, a standardized request format fits the use case better and is generally simpler.

3. Terminology

This specification uses a necessary mixture of RATS and PKI terminology 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, verifier.

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.

Attestation Service (AttS):
: A logical module within the HSM that is responsible for generating Evidence compatible with the format outlined in this specification. It collects claims from the platform and uses the Attestation Key to digitally sign the collection.

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

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 the Attestation Key.

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 application keys found within an HSM. In general, the claims includes enough information about an application 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.

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 attributes 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 HSM, in this case the Attester, and the Verifier. The Presenter initiates the operation of generating evidence at the HSM and passes the generated evidence to the Verifier. This role is supported by a combination of one or multiple human operators or automated processes.

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. It can be a non-root certificate when it is a certificate.

Usage Protocol
: A (security) protocol that requires demonstrating possession of the private component of the application key.

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 is "application key", "client key" or "operational key". The access and operations on a user key is controlled by the HSM.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3.1. Attestation Key Certificate Chain

The data format in this specification represents PKIX 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 will usually be 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 network.

The AK certificate that signs the evidence MUST have the Extended Key Usage id-kp-attest defined in [TODO-submit-2-pager-to-lamps].

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 7 for a discussion of handling multiple SignatureBlocks.

4. Information Model

The PKIX Evidence format is composed of two main sections:

* A claim description section which describes the information transmitted as Evidence.

* A signature section where one or more digital signatures are offered to prove the origin of the claims and maintain their 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 organized into a set of entities to help with the organization and comprehension of the information. Entities are elements observed in the Target Environment by the Attester. Each entity, in turn, is associated with a set of attributes.

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

4.1. Entity

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

An entity SHOULD be reported only 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 be relating to different elements. For example, if a given application public key appears in two different entities, these MUST be interpreted as two distinct and independent entities that happen to have the same public key, and MUST NOT be interpreted as adding additional attributes to the already-described entity. This restriction is to ease the implementation of Verifiers for the provided Evidence.

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 the demands of the Presenter.

In particular, note that the nonce attribute 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 key attestation object generated at manufacture time and injected statically into the device and act as a kind of certificate, instead of being generated on-demand. This model would essentially off-board the Target Environment to be part of the manufacturing infrastructure.

4.2. Entity Type

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

* Platform : This entity holds attributes relating to the state of the platform, or device, where the Attester is located. Entities of this type hold attributes 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 attributes relating to that key.

* Transaction : This entity is logical in nature since it is associated with attributes that are not found in the Target Environment. The attributes found in this entity 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 Identifier (OID) for identifying both entity types and the attribute types that they contain, this format is inherently extensible; implementers of Attesters MAY define new custom or proprietary entity types and place them alongside the standardized entities, or define new attribute types and place them inside standardized entities.

Verifiers SHOULD ignore and skip over unrecognized entity or attribute types and continue processing normally. In other words, if a given Evidence would have been acceptable without the unrecognized entity or attribute, then it SHOULD still be acceptable. In PKI terminology, all custom entities and attributes not defined in this document SHOULD be considered non-critical unless a further specification indicates differently.

4.3. Attribute and Attribute Type

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

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

It is RECOMMENDED that an attribute type be defined for a specific entity type, to reduce confusion when it comes to interpretation of the value. In other words, an attribute 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 attribute 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 attribute type.

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

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

Some attributes MAY be repeated within an entity while others MUST NOT. For example, for a platform entity, there can only be one "firmware version" attribute. Therefore, the associated attribute MUST NOT be repeated as it may lead to confusion. However, an attribute relating to a "loaded module" MAY be repeated, each attribute describing a different loaded module. Therefore, the definition of an attribute specifies whether or not multiple copies of that attribute are allowed.

If a Verifier encounters, within a single entity, multiple copies of an attribute specified as "Multiple Allowed: No", it MUST reject the evidence as malformed.

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

5. Data Model

This section describes the data model associated with PKIX 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 10.

The top-level structures are:

```
PkixEvidence ::= SEQUENCE {
    tbs                          TbsPkixEvidence,
    signatures                   SEQUENCE SIZE (0..MAX) of SignatureBlock,
    intermediateCertificates  [0] IMPLICIT SEQUENCE of Certificate OPTIONAL
                                 -- As defined in RFC 5280
}

TbsPkixEvidence ::= 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
}
```

A PkixEvidence message is composed of a protected section known as the To-Be-Signed (TBS) section where the evidence reported by the HSM 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 against stripping attacks where an attacker removes a signature without leaving evidence in the message that an additional signature had been there or signature re-ordering attacks. Again, this is left up to Verifier and its policy to enforce the expected number of algorithms or signatures. Consequently, Verifiers MUST NOT make any inferences about the lack of a signature. For example, enumerating counter-signatures on an Evidence MUST NOT be considered to be a complete list of HSMs in a given cluster. Similarly, the presence and order of counter-signatures MUST NOT be taken as proof of the path that the evidence traversed over the network.

The TBS section is composed of a version number, to ensure future extensibility, and a sequence of reported entities. For compliance with this specification, TbsPkixEvidence.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, Subject Public Key Identifier (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.

SignatureBlock.certChain MUST contain at least one X.509 certificate as per [RFC5280]. While there might exist Target Environments which use out-of-band or non-X.509 mechanisms for communicating the AK public key to the Verifier, these mechanisms are insufficient to comply with this specification.

The optional certificates provided in PkixEvidence.intermediateCertificates enables the insertion of X.509 certificates to support trusting the signatures. This information is intended to provide the certificates required by the Verifier to verified the endorsement on the certificates included with the signatures.

As described in the Section 4 section, the TbsPkixEvidence is a set 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,
    reportedAttributes SEQUENCE SIZE (1..MAX) OF ReportedAttribute
}

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

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

```
ReportedAttribute ::= SEQUENCE {
    attributeType      OBJECT IDENTIFIER,
    value              OPTIONAL AttributeValue
}

AttributeValue :== 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
}
```

The attributes associated with an entity are dependent on the type of entity. Therefore, it is encouraged to define attribute types grouped with their respective entity type.

The type of an attribute value is dictated by the attribute type. When an attribute 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 attributes.

5.1. Platform Entity

A platform entity is associated with the type identifier id-pkix-attest-entity-platform. It is composed of a set of attributes that are global to the Target Environment.

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

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

Attribute	AttributeValue	Reference	Multiple?	OID
vendor	utf8String	RFCthis	No	id-pkix-evidence-attribute-platform-vendor
oemid	bytes	[RFC9711]	No	id-pkix-evidence-attribute-platform-oemid
hwmodel	utf8String	[RFC9711]	No	id-pkix-evidence-attribute-platform-model
hwserial	utf8String	RFCthis	No	id-pkix-evidence-attribute-platform-hwserial
swversion	utf8String	[RFC9711]	No	id-pkix-evidence-attribute-platform-swversion
dbgstat	int	[RFC9711]	No	id-pkix-evidence-attribute-platform-debugstat
uptime	int	[RFC9711]	No	id-pkix-evidence-attribute-platform-uptime
bootcount	int	[RFC9711]	No	id-pkix-evidence-attribute-

				platform-bootcount
usermods	utf8String	RFCthis	Yes	id-pkix-evidence-attribute-platform-usermods
fipsboot	bool	[FIPS.140-3]	No	id-pkix-evidence-attribute-platform-fipsboot
fipsver	utf8String	[FIPS.140-3]	No	id-pkix-evidence-attribute-platform-fipsver
fipslevel	int	[FIPS.140-3]	No	id-pkix-evidence-attribute-platform-fipslevel
envid	utf8String	RFCthis	Yes	id-pkix-evidence-attribute-platform-envid
envdesc	utf8String	RFCthis	Yes	id-pkix-evidence-attribute-platform-envdesc

Table 1

TODO: find the actual reference for "FIPS Mode" -- FIPS 140-3 does not define it (at least not the 11 page useless version of 140-3 that I found).

Each attribute 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.

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

These attributes are defined in [RFC9711] 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 [RFC9711], the definition offered in the latter shall prevail.

The attribute "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 attribute "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.

EDNOTE: JPF: "hwmodel" in EAT is not human readable. We have "vendor" that duplicates in human readable for "oemid". Should we duplicate "hwmodel" in a human readable form? Should we define it here for ourselves?

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

EDNOTE: JPF: In EAT, "swversion" requires "swname". Should we add "swname" or disassociate from the EAT definition?

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

The attribute "bootcount" reported 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. usermods

Most HSMs have some concept of trusted execution environment where user software modules can be loaded inside the HSM to run with some level of privileged access to the application keys. This attribute lists user modules currently loaded onto the HSM in a human readable format, preferably JSON.

EDNOTE: JPF if JSON, why have multiple attributes.

5.1.5. fipsboot, fipsver and fipslevel

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 attribute fipsboot indicates whether the device is currently operating in FIPS mode. When the attribute 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 attribute is "false", then the HSM is not restricted to the behavior limited by compliance.

The UTF8String attribute 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 attribute fipslevel indicates the compliance level to which the device is currently operating and MUST only be 1, 2, 3, or 4. The fipslevel attribute has no meaning if fipsboot is absent or false.

The FIPS status information in PKIX 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.

5.1.6. envid

An identifier for an environment to which the attested keys belong. These will be in a vendor-chosen format, but are constrained to ASCII as URIs, UUID, and similar types of identifiers are envisioned.

There MAY be multiple envid attributes if the attested keys simultaneously belong to multiple environments.

Note that by including envid as a platform attribute, this implies that it applies to all attested key entities. If the HSM needs to attest multiple keys across multiple disjoint environments, then multiple PkixEvidences are required. This naturally enforces privacy constraints of only attesting a single environment at a time.

EDNOTE: JPF I do not understand this sub-section

If an envid request attribute contains a value, this means that the Presenter is requesting that only keys belonging to the given environment be included in the returned attestation.

5.1.7. envdesc

Further description of the environment beyond hwvendor, hwmodel, hwserial, swversion; for example if there is a need to describe multiple logical partitions within the same device. Contents could be a human-readable description or other identifiers.

5.2. Key Entity

A key entity is associated with the type id-pkix-attest-entity-key. Each instance of a key entity represents a different cryptographic key found in the Target Environment. There can be multiple key entities found in a claim description, but each reported key entity MUST describe a different cryptographic key.

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

A Verifier that encounters a claim description with multiple key entities referring to the same cryptographic key MUST reject the Evidence.

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

Attribute	AttributeValue	Reference	Multiple?	OID
identifier	utf8String	RFCthis	Yes	id-pkix-evidence-attribute-key-identifier
spki	bytes	RFCthis	No	id-pkix-evidence-attribute-key-spki
purpose	bytes	[PKCS11]	No	id-pkix-evidence-attribute-key-purpose
extractable	bool	[PKCS11]	No	id-pkix-evidence-attribute-key-extractable
sensitive	bool	[PKCS11]	No	id-pkix-evidence-attribute-key-sensitive
never-extractable	bool	[PKCS11]	No	id-pkix-evidence-attribute-

				key-never-extractable
local	bool	[PKCS11]	No	id-pkix-evidence-attribute-key-local
expiry	time	RFCthis	No	id-pkix-evidence-attribute-key-expiry
protection	bytes	RFCthis	No	id-pkix-evidence-attribute-key-protection

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 PKIX 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 attribute 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 attribute contains the DER-encoded field SubjectPublicKeyInfo (see [RFC5280]) associated with the cryptographic key.

5.2.3. purpose, extractable, sensitive, never-extractable, local

These attributes are defined 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 attribute "purpose" defines the intended usage for the key.

EDNOTE: JPF: I do not see "purpose" as part of PKCS#11

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

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

The attribute "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 attribute "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.

5.2.5. protection

Indicates any additional key protection properties around use or modification of this key. These are generalized properties and will not apply the same way to all HSM vendors. Consult vendor documentation for the in-context meaning of these flags.

TODO: define a bit-indexed byte array

BIT MASK / Boolean Array {DualControl (0), CardControl (1), PasswordControl (2), ...}

We may need to say that the first X are reserved for use by future RFCs that update this specification, and beyond that is private use.

5.3. Transaction Entity

A transaction entity is associated with the type id-pkix-attest-entity-transaction. This is a logical entity and does not relate to an element found in the Target Environment. Instead, it groups together attributes 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, within the portion protected for integrity, to prove the freshness of the claims. This "nonce" is not related to any element in the Target Environment and the transaction entity is used to gather those values into attributes.

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

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

A default and vendor-agnostic set of transaction entity attributes is defined in this section.

These attribute types MAY be contained within a transaction entity; i.e. an entity identified by id-pkix-attest-entity-transaction.

Attribute	AttributeValue	Reference	Multiple?	OID
nonce	bytes	[RFC9711]	Yes	id-pkix-evidence-attribute-transaction-nonce
timestamp	time	[RFC9711]	No	id-pkix-evidence-attribute-transaction-timestamp

Table 3

5.3.1. nonce

The attribute "nonce" is used to provide "freshness" quality as to the claims provided in the PkixEvidence message. A Presenter requesting a PkixEvidence message MAY provide a nonce value as part of the request. This nonce value, if provided, SHOULD be repeated as an attribute within the transaction entity.

This is similar to the attribute "eat_nonce" as defined in [RFC9711]. According to this specification, this attribute may be specified multiple times with different values. In that case, all different values shall be repeated in the PKIXEvidence.

5.3.2. timestamp

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

EDNOTE: JPF: Does the following paragraph belong to Security Considerations?

Note that it is common for HSMs to not have an accurate system clock; consider an HSM for a root CA kept offline and booted up infrequently in an local network segregated from all other network, or a smart card which boots up only when held against an NFC reader. Implementers of emitters SHOULD include this attribute only if the device reliably knows its own time (for example has had recent contact with an NTP server). Implementers of parsers SHOULD be wary of trusting the contents of this attribute. A challenge-response protocol that makes use of the nonce attribute is a far more reliable way of establishing freshness.

5.4. Additional Entity and Attribute Types

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

An Attester (HSM) which is requested to provide information about unrecognized Entity or Attribute types MUST fail the operation.

A Verifier which encounters an unrecognized Entity or Attribute type MAY ignore it.

5.5. Encoding

A PkixEvidence is to be DER encoded [X.690].

If a textual representation is required, then the DER encoding MAY be subsequently encoded into Base64.

EDNOTE: I think we have to be precise about which flavour of Base64 we are referring to.

6. Signing Procedure

The SignatureBlock.signatureValue signs over the DER-encoded to-be-signed evidence data PkixEvidence.tbs and MUST be validated with the subject public key of the leaf X.509 certificate contained in the SignatureBlock.certChain.

7. Verification Procedure

The SignatureBlock.signatureValue signs over the DER-encoded to-be-signed evidence data PkixEvidence.tbs and MUST be validated with the subject public key of the leaf X.509 certificate contained in the SignatureBlock.certChain.

Note that a PkixEvidence MAY contain zero or more SignatureBlocks. A PkixEvidence with zero SignatureBlocks is unsigned, MUST be treated as un-protected and un-trusted, and any signature validation procedure MUST fail.

More than one SignatureBlocks MAY be used to convey a number of different semantics. For example, the HSM’s Attesting Service might hold multiple Attestation Keys on different cryptographic algorithms in order to provide algorithm redundancy in the case that one algorithm becomes cryptographically broken. 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 PkixEvidence object.

Note that each SignatureBlock is a fully detached signature over the tbs content with no binding between the signed content and the SignatureBlocks, or between 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 {#sec-detached-sigs} for further discussion.

8. Appraisal Policies and Profiles

This section provides some sample profiles of appraisal policies that verifiers MAY apply when evaluating evidence. These appraisal policy profiles represent environment-specific requirements on the contents of the evidence and / or endorsement certificate chain.

8.1. Key Import into an HSM

An HSM which is compliant with this draft SHOULD validate any PKIX evidence that is provided along with the key being imported.

The SignatureBlocks MUST be validated and MUST chain to a trust anchor known to the HSM. In most cases this will be the same trust anchor that endorsed the HSM’s own AK, but the HSM MAY be configured with set of third-party trust anchors from which it will accept key attestations.

If the HSM is operating in FIPS Mode, then it MUST only import keys from HSMs also operating in FIPS Mode.

The claims key-purpose, key-extractable, key-never-extractable, and key-local MUST be checked and honoured during key import, which typically means that after import, the key MUST NOT claim a stronger protection property than it had within the previous HSM. In other words, Key Attestation allows and requires that key protection properties be preserved over export / import operations between different HSMs, and this format provides a vendor-agnostic way to achieve this.

How to handle errors is outside the scope of this specification and is left to implementors; for example the key import MAY be aborted, or a prompt MAY be given to the user administrator, or any similar reasonable error handling logic may be used.

8.2. CA/Browser Forum Code-Signing

TODO: ... intro text

The subscriber MUST:

* Provide the CA with a CSR containing the subscriber key.

* Provide PKIX evidence, as per this specification, describing the private key protection properties of the subscriber's private key. This evidence MAY be transported inside the CSR as per draft-ietf-lamps-csr-attest, or it MAY be transported adjacent to the CSR over any other certificate enrollment mechanism.

The CA / RA / RP / Verifier MUST:

* Ensure that the subscriber key which is the subject of the CSR is also described by a KAT by matching either the key fingerprint or full SubjectPublicKeyInfo.

* The hardware root-of-trust described by a PAT has a valid and active FIPS certificate according to the NIST CMVP database.

* The attestation key (AK) which has signed the PKIX evidence chains to a root certificate that A) belongs to the hardware vendor described in the PAT token, and B) is trusted by the CA / RA / RP / Verifier to endorse hardware from this vendor, for example through a CA's partner program or through a network operator's device on-boarding process.

* The key is protected by a module running in FIPS mode. The parsing logic is to start at the leaf KAT token that matches the key in the CSR and parsing towards the root PAT ensuring that there is at least one fipsboot=true and no fipsboot=false on that path.

9. 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 PKIX 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 attestation message endorsed by the vendor might be installed during manufacturing. Other approaches for constrained HSMs might be to report entities and attributes 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 attestation messages 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 an attestation message.

This section introduces the role of "Presenter" as shown in Figure 1. The Presenter is the role that initiates the generation of an attestation message. Since HSMs are generally servers (client/server relationship) or slaves (master/slave relationship), a Presenter is required to launch the process of creating the attestation message and capturing its results. The results are then forwarded to the Verifier.

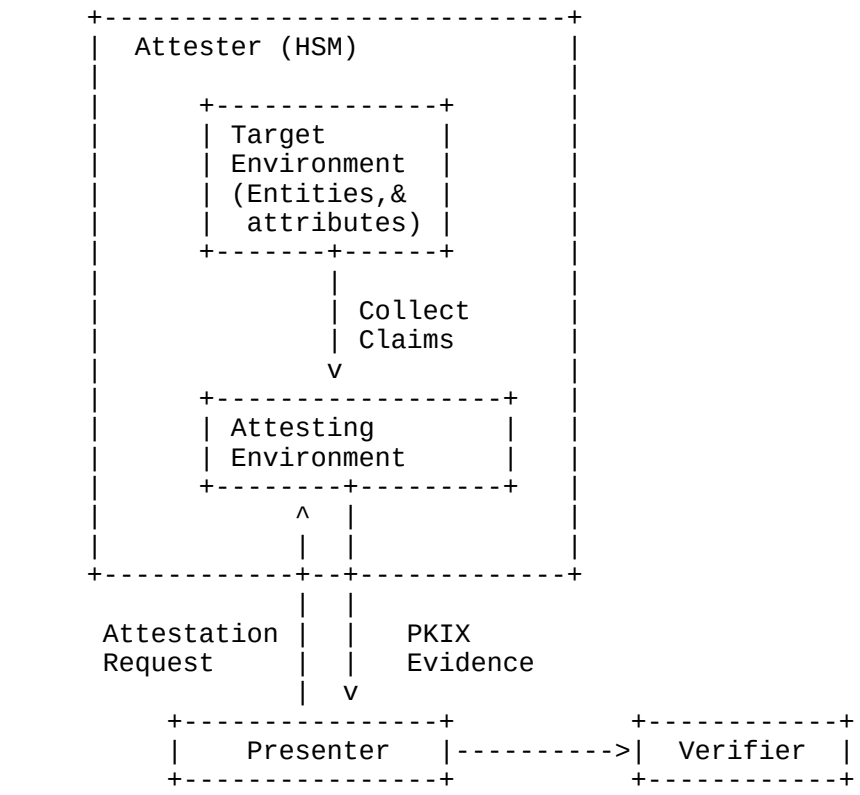

Figure 1: Architecture

An Attestation Request (request) is assembled by the Presenter and submitted to the HSM. The HSM parses the request and produces PKIX evidence which is returned to the Presenter for distribution.

The request consists of a structure TbsPkixEvidence 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 request entity. A request entity contains a number of instances of ReportedAttribute known as request attributes. The collection of request entities and request attributes represent the information desired by the Presenter.

In most cases the value of a request attribute should be left unspecified by the Presenter. In the process of generating the evidence, the values of the desired attributes are observed by the Attestation Service within the HSM and reported accordingly. For the purpose of creating a request, the Presenter sets the values of the attributes to null. This is a departure from the values specified for each attribute but serves well the purposes of the request.

On the other hand, there are circumstances where the value of a request attribute 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 request entity with one of its attributes set with a type id-pkix-evidence-attribute-key-identifier. The value of this attribute is set to the key identifier associated with the cryptographic key to be reported.

Some instances of ReportedEntity, such as those representing the platform or the transaction, do not need identifiers as the associated entities 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 TbsPkixEvidence is unsigned and does not provided 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 PkixEvidence could be reused by an implementer to provide those capabilities, as described in Section 12.5.

9.1. Request Attributes with Specified Values

This section deals with the request attributes specified in this document where a value should be provided by a Presenter. In other words, this sub-section defines all request attributes that should not be null. Request attributes not covered in this sub-section should have a value of null.

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

9.1.1. Key Identifiers

A Presenter may choose to select which cryptographic keys are reported as part of the PKIX evidence. For each selected cryptographic key, the Presenter includes a request entity of type id-pkix-evidence-entity-key. Among the request attributes for this entity, the Presenter includes one attribute with the type id-pkix-evidence-attribute-key-identifier. The value of this attribute 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 MUST fail the transaction if it cannot find the cryptographic key associated with the specified identifier.

9.1.2. Nonce

A Presenter may choose to include a nonce as part of the attestation request. When producing the PKIX 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-pkix-evidence-entity-transaction with an attribute of type id-pkix-evidence-attribute-transaction-nonce. This attribute is set with the value of the nonce as "bytes".

9.1.3. 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-pkix-evidence-entity-key including an attribute with a set value. This attribute 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 attribute and report them in the PKIX evidence.

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

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

9.1.4. Custom Transaction Entity Attributes

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

In such an approach, a new custom attribute for request entities of type id-pkix-evidence-entity-transaction is defined. Then, an attribute 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 PKIX evidence.

9.2. Processing an Attestation Request

This sub-section deals with the rules that should be considered when an Attester (the HSM) processes a request to generate an attestation request. 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 Attester MUST 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 Attester.

An Attester MUST fail an attestation request if it contains a request attribute of an unrecognized type while specifying a value (other than null). This represents a situation where the Presenter is selecting specific information that is not understood by the Attester.

An Attester SHOULD fail an attestation request if it contains a request attribute with an unrecognized type. An environment with an Attester that ignores unrecognized attributes forces the Presenter to review the generated evidence for necessary information.

An Attester MUST NOT include entities and attributes in the generated attestation message if these entities and attributes were not specified as part of the request. This is to give the Presenter the control on what information is disclosed by the Attester.

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

9.3. Verification by Presenter

This sub-section deals with the rules that should be considered when a Presenter receives an PKIX 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 PKIX evidence and are not restricted solely 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 an attestation message if it contains information it cannot parse. This restriction applies to entity types and attributes type. This is to ensure that the information provided by the Attester can be evaluated by the Presenter.

A Presenter MUST NOT disclose an attestation message 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 evidencee if it contains an entity with an attribute 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 12.4.

10. 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) }


PkixEvidence ::= SEQUENCE {
    tbs                          TbsPkixEvidence,
    signatures                   SEQUENCE SIZE (0..MAX) of \
                                                 SignatureBlock,
    intermediateCertificates  [0] IMPLICIT SEQUENCE of Certificate \
```

```
                                                              OPTIONAL
                                    -- As defined in RFC 5280
   }

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

   ReportedEntity ::= SEQUENCE {
       entityType         OBJECT IDENTIFIER,
       reportedAttributes SEQUENCE SIZE (1..MAX) OF ReportedAttribute
   }

   ReportedAttribute ::= SEQUENCE {
       attributeType      OBJECT IDENTIFIER,
       value              AttributeValue
   }

   AttributeValue :== 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,
      null        [6] IMPLICIT 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
   }

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

   id-pkix-evidence-entity-type        OBJECT IDENTIFIER ::= { id-pkix-\
                                                               evidence- 0 }
   id-pkix-evidence-entity-transaction OBJECT IDENTIFIER ::= { id-pkix-\
```

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

   id-pkix-evidence-attribute-type OBJECT IDENTIFIER ::= { id-pkix-\
                                                           evidence- 1 }

   id-pkix-evidence-attribute-transaction           OBJECT IDENTIFIER :\
                              := { id-pkix-evidence-attribute-type 0 }
   id-pkix-evidence-attribute-transaction-nonce     OBJECT IDENTIFIER :\
                         := { id-pkix-evidence-attribute-transaction 0 }
   id-pkix-evidence-attribute-transaction-timestamp OBJECT IDENTIFIER :\
                         := { id-pkix-evidence-attribute-transaction 1 }

   id-pkix-evidence-attribute-platform            OBJECT IDENTIFIER ::\
                               = { id-pkix-evidence-attribute-type 1 }
   id-pkix-evidence-attribute-platform-vendor     OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 0 }
   id-pkix-evidence-attribute-platform-hwserial   OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 1 }
   id-pkix-evidence-attribute-platform-fipsboot   OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 2 }
   id-pkix-evidence-attribute-platform-model      OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 3 }
   id-pkix-evidence-attribute-platform-swversion  OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 4 }
   id-pkix-evidence-attribute-platform-oemid      OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 5 }
   id-pkix-evidence-attribute-platform-debugstat  OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 6 }
   id-pkix-evidence-attribute-platform-uptime     OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 7 }
   id-pkix-evidence-attribute-platform-bootcount  OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 8 }
   id-pkix-evidence-attribute-platform-usermods   OBJECT IDENTIFIER ::\
                            = { id-pkix-evidence-attribute-platform 9 }
   id-pkix-evidence-attribute-platform-envid      OBJECT IDENTIFIER ::\
                           = { id-pkix-evidence-attribute-platform 10 }
   id-pkix-evidence-attribute-platform-envdesc    OBJECT IDENTIFIER ::\
                           = { id-pkix-evidence-attribute-platform 11 }
   id-pkix-evidence-attribute-platform-fipsver    OBJECT IDENTIFIER ::\
                           = { id-pkix-evidence-attribute-platform 12 }
   id-pkix-evidence-attribute-platform-fipslevel  OBJECT IDENTIFIER ::\
                           = { id-pkix-evidence-attribute-platform 13 }
```

```
   id-pkix-evidence-attribute-key                    OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-type 2 }
   id-pkix-evidence-attribute-key-identifier         OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 0 }
   id-pkix-evidence-attribute-key-spki               OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 1 }
   id-pkix-evidence-attribute-key-purpose            OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 2 }
   id-pkix-evidence-attribute-key-extractable        OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 3 }
   id-pkix-evidence-attribute-key-never-extractable OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 4 }
   id-pkix-evidence-attribute-key-local              OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 5 }
   id-pkix-evidence-attribute-key-expiry             OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 6 }
   id-pkix-evidence-attribute-key-protection         OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 7 }
   id-pkix-evidence-attribute-key-sensitive          OBJECT IDENTIFIER :\
                                 := { id-pkix-evidence-attribute-key 8 }
```

<CODE ENDS>

11. IANA Considerations

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

TODO: list out all the OIDs that need IANA registration.

12. Security Considerations

12.1. Policies relating to Verifier and Relying Party

The generation of PKIX evidence by an HSM is to provide sufficient information to a Verifier and 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 PKIX evidence may greatly influence the outcome of some decisions.

A Verifier MAY reject a PKIX evidence if it lacks required attributes 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 an attribute with an unrecognized attribute type, it MAY ignore it and treat it as extraneous information. By ignoring an attribute, the Verifier may accept PKIX evidence that would be deemed malformed to a Verifier with different policies. However, this approach fosters a higher likelihood of achieving interoperability.

12.2. Simple to Implement

The nature of attestation requires the Attestation Service to be implemented in an extremely privileged position within the HSM so that it can collect measurements of both the hardware environment and the user keys being attested. For many HSM and TPM architectures, this will place the Attestation Service inside the "HSM kernel" and potentially subject to FIPS 140-3 or Common Criteria validation and change control. For both security and compliance reasons there is 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 a request entity, or parsing an attestation 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 Attestation Service SHOULD generate the PKIX evidence from scratch and avoid copying any content from the request. The Attestation Service MUST generate PKIX evidence only from attributes and values that are observed by the service.

12.3. Detached Signatures

TODO: Editorial work needed.

No indication within the tbs content about what or how many signatures to expect.

A SignatureBlock can be trivially stripped off without leaving any evidence.

When multiple SignatureBlocks are used for providing third-party counter-signatures, note that the counter signature only covers the tbs content and not existing SignatureBlocks.

12.4. Privacy

Often, a TPM will host cryptographic keys for both the kernel and userspace of a local operating system but a Presenter may only represents a single user or application. Similarly, a single enterprise-grade Hardware Security Module will often host cryptographic keys for an entire multi-tenant cloud service and the Presenter or Receiver or Recipient belongs only to a single tenant. For example the HSM backing a TLS-terminating loadbalancer fronting thousands of un-related web domains. In these cases, disclosing that two different keys reside on the same hardware, or in some cases even disclosing the existance of a given key, let alone its attributes, to an unauthorized party would constitute an egregious privacy violation.

Implementions SHOULD be careful to avoid over-disclosure of information, for example by authenticating the Presenter as described in Section 12.5 and only returning results for keys and envirnments 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 TbsPkixEvidence of the request inside a PkixEvidence 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 9 so that Presenters can fine-tune the content of a PKIX evidence such that it is appropriate for the intended Verifier.

12.5. 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 attestation 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 of 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 read access.

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 ClaimDescriptionTbs of the request inside a PkixEvidence signed with a certificate belonging to the Presenter.

12.6. 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 easy to add a PoP Key Attribute that uses the attested user key to sign over, for example, the Transaction Entity. However, this is a bad idea and MUST NOT be added as a custom attribute for several reasons.

First, an application key intended, for example, for TLS SHOULD only be used with the TLS protocol and introducing a signature oracle whereby the TLS application key is used to sign PKIX evidence could lead to cross-protocol attacks whereby the attacker submits 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 PKIX evidence may have permissions to view 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 an systems administrator or auditor.

Requiring the Attestation Service to use the attested application keys could, in some architectures, require the Attestation Service to resolve complex access control logic and handle complex error conditions for each requested key, which violates the "simple to implement" design principle outlined in Section 12.2. More discussion of authenticating the Presenter can be found in Section 12.5.

13. References

13.1. Normative References

[FIPS.140-3] 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>.

[PKCS11] Cox, D. B. T., "PKCS #11 Specification Version 3.1", n.d., <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>.

[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.680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", n.d., <https://www.itu.int/rec/T-REC-X.680>.

[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)", ISO/IEC 8825-1:2015, November 2015.

13.2. Informative References

[CNSA2.0] "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>.

[codesigningbrsv3.8] "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-19, 25 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-csr-attestation-19>.

[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-16, 3 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-msg-wrap-16>.

[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\
                                    649713697884647273079158207679184896 01

         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=\
    1297777542463176828954253910265338298243179555114614528175018955375 7\
    9409825728132644289829859977405958780770278539945157751167520309638 5\
    8469651548765808775269857271167748512795017916284867051302884465315 7\
    5101091365801664017060841393578011934986698617014803330195575311698 4\
    0412712739077565447802315646468604249990209907455233836229801152004 4\
    6260103173103500647838758197610238552349053064525420240826193553395 3\
    7887372525658426966691850479367449774845557482223802208505475218568 7\
    4408076553377248218533326881584603795549061054177206651756483718328 2\
    5939577039874730442790337726004105878168375998123110331993348833629 3\
                                                                    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=437521183820090378116187489499283394628964571 44
         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=\
   57095560233504924588952816185508037812996307929249104847846164660564\
   88839712339087758567046283628572504126189755002031148112756265577433\
                                                        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=\
   18216751979714603574557504315480141511553297913673112867639918069266\
   48218048839904015520407896430131032024244860880583649829667093244967\
                                      82518079519267269438816178719668437

      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.

Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road - Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: mike.ounsworth@entrust.com

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: montywiseman32@gmail.com

Ned Smith
Intel Corporation
United States of America

Email: ned.smith@intel.com