Workgroup: LAMPS
Internet-Draft:
draft-ounsworth-pkix-key-attestation-02
Published: 13 March 2023
Intended Status: Standards Track
Expires: 14 September 2023
Authors: M. Ounsworth, Entrust; R. Kettlewell, Entrust - nCipher; B. Couillard, Crypto4A; JP Fiset, Crypto4A

PKIX Key Attestation Format

Abstract

This document describes syntax for conveying key origin attestation information to a Certification Authority (CA) or other entity, so that they may decide how much trust to place in the management of the private key. For example, a reliant party may use this information to support a decision about whether to issue a certificate. In contrast to other key attestation formats, the one defined in this document requires only ASN.1 and the standard PKIX modules.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ounsworth-pkix-key-attestation/.

Discussion of this document takes place on the Limited Additional Mechanisms for PKIX and SMIME (lamps) Working Group mailing list (mailto:spasm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/spasm/. Subscribe at https://www.ietf.org/mailman/listinfo/spasm/.

Source for this draft and an issue tracker can be found at https://github.com/EntrustCorporation/draft-ounsworth-pq-composite-keys.

Status of This Memo

This Internet-Draft will expire on 14 September 2023.

Copyright Notice

Table of Contents

1. Introduction

Key attestation refers to the originator of a cryptographic key pair providing information about the provenance of that key pair, in a manner that can be cryptographically verified. The information provided may include, for example, the model and identity of the device that created the key pair and any policies that may be enforced upon the use of the private key, contained in a cryptographic envelope that can be chained to a manufacturing public key of the device vendor.

This information can be used by a Certification Authority (CA) to decide whether to issue a certificate, to apply a given policy or certificate template, or by other entities for their own purposes. The CA may choose to publish some or all of the key attestation data in the certificate for the use of parties that will rely on this certificate.

Many devices, including Hardware Security Modules, provide attestation information of some form in proprietary formats. A common syntax for key attestations is required to reduce the implementation burden on CA implementors and operators. Furthermore it is desirable that the syntax is sympathetic to existing CA implementations.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Cryptographic Keys

This section describes the cryptographic keys referenced in this document.

3.1. Trust Anchor Key

A trust anchor key is a signing key held by a vendor. For the purposes of this document, a trust anchor may be a proper Trust Anchor as defined in [RFC5914], or a root certification authority as defined in [RFC5280]. It is used either to directly sign device identity keys as defined in Section 3.3 or to sign intermediate CA keys. A trust anchor key MUST be associated with a vendor identity.

Constraints:

* A trust anchor key MUST only be used for purposes consistent with signing intermediate CA keys or devices (i.e. signing delegation certificates, CRLs, etc).

3.2. Intermediate CA Key

An intermediate CA key is a signing key held by a vendor and certified by that vendor's trust anchor.

It can be used for one of two purposes:

* To certify device identity keys (see Section 3.3) by signing device identity certificates (see Section 4.3)

* To certify further intermediate CA keys

The exact configuration and management of trust anchor keys and intermediate CA keys is beyond the scope of this document. An example configuration is that a vendor have an offline trust anchor, and an intermediate CA in each of its manufacturing sites, certified by the trust anchor key when a manufacturing site is created or during maintenance or recovery activities.

It may be impossible recertify a device after manufacture, and it may be impossible for a manufacturer to know when a device has been retired from use. Therefore:

* An intermediate CA need not track and public revocation information

* Intermediate CA keys MAY have an expiration date of 99991231235959Z ([RFC5280] section 4.1.2.5).

Constraints:

* An intermediate CA key MUST only be used for purposes consistent with certifying intermediate CA keys (i.e. signing delegation certificates, CRLs, etc) or devices.

3.3. Device Identity Key

A device identity key is a signing key held by a device. It is assumed that the key is unique to the device and cannot be extracted or used for any purpose other than the ones listed below. It is envisaged that this key will persist for the lifetime of the device.

It can be used for one of two purposes:

* To sign key attestations directly

* To sign device delegation certificates (see Section 4.4), which are used to certify device certification subkeys (see Section 3.4).

Constraints:

* A device identity key MUST NOT be used for any purpose other than signing key attestation certificates or device delegation certificates.

3.4. Device Certification Subkey

A device certification key is a signing key held by a device. It is assumed that the key is unique to the device and cannot be extracted or used for any purpose other than the ones listed below. Depending on the device architecture, it may also be limited to a particular context or partition of the device; in this case it is assumed to be unique to the context. A device certification key may have any lifetime, from single use to the lifetime of the device.

It can be used for one of two purposes:

* To sign key attestations directly

* To sign further device delegation certificates.

Constraints:

* A device certification subkey MUST NOT be used for any purpose other than signing key attestation certificates (see Section 4.5) or device delegation certificates (see Section 4.4).

3.5. Application Key

An application key is a key created and managed by a device (excluding the device identity key and device certification subkey described above). Its purpose and lifetime are arbitrary - in other words, it can be used for any purpose a user of the device wishes.

(MikeO: maybe I'm a noob here, but the distinction between this an a Device Certification Subkey could be stated more clearly. Maybe the distinction "This is envisioned for cases where a device needs an attested key which may be used for arbitrary purposes".)

(RJK: it's not really about what the device desires - these are the keys that we are trying to attest the origin of. The user has some higher-level purpose, e.g. code signing, which requires them to define a code signing key and attest to its origins in an HSM; from the point of view of this spec, their code-signing key is an application key. Keeping this comment open in the hope we can find a clear way of articulating this.)

4. Key Attestations

A verifier is an entity which wishes to verify the origin of a key, based on its trust in a trust anchor.

For example, it could be a certificate authority with an operational constraint that it only certifies hardware-protected keys.

4.1. Key Attestation Bundle

A key attestation consists of a nonempty sequence of [RFC5280] certificates, containing key attestation extensions as described below.

Specifically, a key attestation consists of:

* Zero or more intermediate certificates (see Section 4.2)

* Exactly one device identity certificate (see Section 4.3)

* Zero or more device delegation certificates (see Section 4.4)

* Exactly one key attestation certificate (see Section 4.5)

The first certificate (whether it is an intermediate certificate or the device identity certificate) is signed by a trust anchor key. A verifier must decide through its own policies and processes which trust anchors keys to trust and what policies to accept in key attestations certified by them. A trust anchor key MUST be associated with a vendor identity.

Constraints:

* A verifier MUST verify that each certificate is well-formed (except that expiry and revocation information need not be present)

* A verifier MUST verify that the first certificate is signed by a trust anchor key

* A verifier MUST verify that each certificate, apart from the first, is certified by the previous certificate in the key attestation.

* A verifier MUST verify that the ordering of certificates is as described above.

4.2. Intermediate CA Certificate

An intermediate CA delegation certificate certifies an intermediate CA. Apart from the absence of any constraints on expiry time and revocation, it is little different from any other intermediate CA's certificate.

It MUST have the [RFC5280] basic constraints extension with the cA boolean set to true.

It MAY have the [RFC5280] pathLenConstraint, and there is no change to the [RFC5280] interpretation this field. Therefore, if it is present, it must permit sufficiently many following certificates to account for certificates signed by the device i.e. device identity certificates (see Section 4.3) and device delegation certificates (see Section 4.4).

It MUST NOT have any of the extensions defined in the following sections (Section 4.3, Section 4.4 and Section 4.5). A verifier may detect an intermediate CA delegation by the presence of a true cA boolean and the absence of these extensions.

Constraints:

* A verifier MUST honor pathLenConstraint if present.

* There may be any number of intermediate CA certificates, including 0.

4.3. Device Identity Certificate

A device identity certificate certifies a specific device by binding its public device identity key (defined in Section 3.3) to a vendor-specific representation of device identity such as vendor name, model, and serial number. For a hardware device, it is envisaged that a manufacturing facility will use its trust anchor or intermediate CA to sign a device identity certificate for each device as it is manufactured.

A device identity certificate MUST contain a DeviceInformation extension, identified by id-device-information. This extension contains the vendor identity, device model and device serial. Together these are called the device identity and MUST uniquely define a particular device.

```
id-device-information OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1567 }

DeviceInformation ::= SEQUENCE {
  vendor UTF8STring   -- manufacturer of device
  model UTF8STring    -- device model information
  serial UTF8STring   -- device instance information
}
```

EDNOTE: this is a temporary OID for the purposes of prototyping. We are requesting IANA to assign a permanent OID, see Section 8.

A device identity certificate MUST have the [RFC5280] basic constraints extension with the cA boolean set to true (since the device is acting as a CA).

No significance is attached to the subject field of a device identity certificate.

Constraints:

* A verifier MUST reject any key attestation that does not contain exactly one device identity certificate.

* A verifier MUST reject any device identity certificate whose vendor identity as indicated in the vendor field does not match the one associated with the trust anchor used to verify the key attestation.

* Two distinct devices from the same vendor MUST NOT have the same device identity, i.e. they must have different values for at least one field of DeviceIdentity.

* Two distinct devices MUST NOT have the same device identity key.

As a matter of interpretation, it is envisaged that the uniqueness requirement on device identity keys (and all other keys in this specification) is achieved by generating keys of adequate size and using cryptographically secure pseudorandom number generators, rather than by maintaining an industry-wide database of all device identity keys.

4.4. Device Delegation Certificate

A device delegation certificate certifies that a specific certification subkey (defined in Section 3.4) belongs to a specific device by binding it to a vendor-specific representation of the device and the subkey's purpose. It is envisaged that a single hardware device may have multiple certification subkeys each being restricted to, for example, a single partition or application context. The device may create new certification subkeys and therefore new device delegation certificates over time, for instance when the device is re-initialized, or if the device supports dynamic creation of users or application contexts and needs to create distinct certification subkeys for each.

A device delegation certificate MUST contain a DeviceSubkeyInformation extension, identified by id-device-subkey-information. This contains the vendor identity, device model, device serial and key purpose. Note that this does not uniquely define the certification subkey.

```
id-device-subkey-information OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1568 }

DeviceSubkeyInformation ::= SEQUENCE {
  vendor UTF8STring   -- manufacturer of device
  model UTF8STring    -- device model information
  serial UTF8STring   -- device instance information
  purpose UTF8String  -- description of subkey purpose
}
```

EDNOTE: this is a temporary OID for the purposes of prototyping. We are requesting IANA to assign a permanent OID, see Section 8.

The meaning of the purpose field is entirely dependent on the device.

It MUST have the [RFC5280] basic constraints extension with the cA boolean set to true (since the device is acting as a CA).

No significance is attached to the subject field of a device delegation certificate.

Constraints:

* A verifier MUST reject any device delegation certificate whose device identity as indicated in the vendor, model and serial fields does not match the values from the device identity certificate.

* The purpose field may have any value.

* Two device delegation certificates signed by the same key MAY have the same purpose field.

4.5. Key Attestation Certificate

A key attestation certificate certifies that an application key was created in a particular device and is managed according to a particular policy.

A key attestation certificate MUST contain a ApplicationKeyInformation extension identified by id-application-key-information. This contains the vendor identity, device model, device serial and vendor-specific information.

A key attestation certificate MUST contain an [RFC5280] Extended Key Usage extension documenting how the device will permit the key to be used. See Section 5 for more details.

```
ApplicationKeyInformation ::= SEQUENCE {
  vendor UTF8STring         -- manufacturer of device
  model UTF8STring          -- device model information
  vendorinfo OCTET STRING   -- vendor-specific information
}
```

EDNOTE: this is a temporary OID for the purposes of prototyping. We are requesting IANA to assign a permanent OID, see Section 8.

If the key attestation certificate contains the [RFC5280] Basic Constraints extension then it MUST have the cA boolean set to false.

No significance is attached to the subject field of a key attestation certificate.

Constraints:

* A verifier MUST reject any key attestation certificate whose device identity as indicated in the vendor, model and serial fields does not match the values from the device identity certificate.

* A verifier MUST reject any key attestation certificate which does not contain exactly one [RFC5280] Extended Key Usage extension

* A verifier MUST reject any key attestation certificate which permits operations inconsistent with its acceptable policies.

4.5.1. Vendor-Specific Information

The ApplicationKeyInformation.vendorInfo field of the key attestation certificate MAY contain any octet string (including the empty string). The interpretation is up to the vendor. For example, it may be used to convey information about how the key was generated or a vendor-specific description of the policies that govern its use.

5. Key Usage

Key attestation certificates contain an [RFC5280] s4.2.1.12 Extended Key Usage extension describing how the device will permit the key to be used.

The standard ExtendedKeyUsage purposes defined in [RFC5280] are not necessarily suitable in this context. For example the standard ExtendedKeyUsage OIDs are also not necessarily suitable. For example the device may have no information about whether a signing key is intended to be used for server authentication, client authentication, or any other application of digital signatures. For this reason an additional set of key usage purposes are defined here.

```
id-Signature       OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1613 }
-- the device will generate signatures with the key

id-Decryption      OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1614 }
-- the device will decrypt messages with the key and return the plaintex

id-KeyAgreement    OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1615 }
-- the device will use the key for key agreement

id-KeyTransport    OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1616 }
-- the device will use the key for key transport

id-Recoverable     OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1612 }
-- the key is can be recovered under administrative authorization
```

EDNOTE: these are a temporary OIDs for the purposes of prototyping. We are requesting IANA to assign a permanent OID, see Section 8.

EDNOTE: We should consult particularly with CAs to see if there are other properties that would be beneficial to include in this list.

Constraints:

* If the device does not include id-Signature in the list of key use purposes then it MUST NOT generate signatures with the key.

* If the device does not include id-Decryption in the list of key use purposes then it MUST NOT decrypt ciphertexts with the key.

* If the device does not include id-KeyAgreement in the list of key use purposes then it MUST NOT use the key for key agreement.

* If the device does not include id-KeyTransport in the list of key use purposes then it MUST NOT use the key for key transport.

* If the device does not include id-Recoverable in the list of key use purposes then it MUST NOT permit recovery operations on the key.

5.1. Distinctions between Key Use Policies

* Decryption means using the key to decrypt a ciphertext and returning the plaintext to the caller, outside the device

* Key Transport means using the key to decrypt a ciphertext and using the plaintext as key material, managed by the device

* Key Agreement means using the key to agree a secret shared with another party, as prelude to further secure communication

5.2. Recoverable Keys

The id-Recoverable key use purpose indicates that the policies controlling use of the key may be modified by a suitably authorized administrator. This may be necessary, for example, to ensure that the key remains available for use even when an authentication token is lost or destroyed.

The scope of possible modifications, and the kind of authorization required, are intentionally vague.

See Section 9.3 for further discussion.

5.3. Key Protection

These key use purposes are not intended to describe how applications keys is protected by the device. For example one device may protect keys by maintaining them inside a hardened boundary at all times; another may allow keys to be used across multiple devices by encrypting them under a shared master key, or by sharing them with other authorized devices via a secure channel.

Provided the device is able to guarantee that the key use policy it signs will be honored, the mechanism is uses to protect application keys is not relevant.

5.4. Vendor-Defined Key Use Policies

A vendor may define key use policies outside the list above, for example reflecting policies not envisaged by this document or to cover device-specific functionality. For example they may describe a policy in terms of their device's proprietary policy or access control syntax and publish an OID reflecting that policy.

A verifier MUST NOT accept such a vendor-defined policy unless they fully understand the intended meaning.

6. Embedding Key Attestations in Certification Requests

A convenient way to convey a key attestation is to embed it into a [RFC2986] certification request. This may be done via the AttestationBundle extension, identified by the OID id-attestation-bundle.

Constraints:

- A certification request SHOULD only have one embedded key attestation.
- A CA MUST follow meet all the constraints on verifiers described above.
- A CA MUST verify that the subject public key in the certification request is the same as the subject public key in the key attestation certificate.

*RJK TODO tidy up all this section

(MikeO: We'll need to be explicit about how to bundle this into a [RFC2986] Attribute. Do we need an OID for the type? I assume the values is straight-forward: it'll be a single item, which is the OCTET STRING of the AttestationBundle?

Attribute { ATTRIBUTE:IOSet } ::= SEQUENCE { type ATTRIBUTE.&id({IOSet}), values SET SIZE(1..MAX) OF ATTRIBUTE.&Type({IOSet}{@type}) }

)

```
id-attestation-bundle OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1571 }

AttestationBundle ::= SEQUENCE OF Certificate
```

EDNOTE: this is a temporary OID for the purposes of prototyping. We are requesting IANA to assign a permanent OID, see Section 8.

7. Implementation Considerations

... TODO document any (non-security) GOTCHAs ...

8. IANA Considerations

The following Object Identifiers are to be assigned by IANA:

```
id-device-information OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1567 }

id-device-subkey-information OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1568 }

id-application-key-information OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1569 }

id-attestation-bundle OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1571 }

id-Signature       OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1613 }

id-Decryption      OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1614 }

id-KeyAgreement    OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1615 }

id-KeyTransport    OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1616 }

id-Recoverable     OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1612 }
```

TODO: suggest to IANA which public arc we want these in (these are just placeholders).

TODO update for our new EKU OIDs

*RJK: the OIDs are assigned by a free OID assignment service. If I can have something under Entrust then I'll replace them with that.

9. Security Considerations

9.1. Key Use Constraints

The key use constraints describe above are essential. For example if a device identity key could be used by a user to sign arbitrary messages, that user could forge key attestations.

9.2. Verification Model

An API that verifies a key attestation may be designed in a number of different ways.

1. It may accept just a key attestation. It will verify it, and return either an error indicator or the public trust anchor key, vendor identity, public application key, and the policy governing is use. The caller must check at least that the trust anchor key is acceptable; the vendor identity from the key attestation matches the one associated with the trust anchor; and that the policy is acceptable, before using the application key. If the caller is running in a context where there are multiple copies of the application key (for example, the certification request verification described in Section 6 it must also check that all copies of the application key match.

2. It may accept a key attestation, trust anchor, vendor identity and at least one acceptable policy. It will verify the key attestation using the trust anchor, and check that the vendor identities in the key attestation match the trust anchor, and check that the policy is acceptable. It will return either an error indicator or the application key. If the caller is running in a context where there are multiple copies of the application key then it must also check that all copies of the application key match. Apart from that it can use the application key without further checks.

3. It may accept a key attestation, trust anchor, vendor identity, application key and at least one acceptable policy. It will verify the key attestation using the trust anchor, and check that the vendor identities in the key attestation match the trust anchor, check that the policy is acceptable, and that the application key is the expected value. It will return either an error or a success indicator. The caller can use the application key without further checks.

In all of these models the same set of checks must be done, but in the first two some of the checks are delegated to the caller. The advantage of the later models is that they are more robust against the caller ommitting some of the necessary checks. For a publicly available API this robustness is particularly appropriate.

9.3. Recoverable Keys

The definition of recoverability is intentionally vague. Depending on the device it may mean that, for example, a signature-only RSA key could additionally be given decrypt permission, or it could mean that private key material could be extracted in plaintext. The range of possibilities is too broad to tie down in a device-independent specification.

It should be noted that placing trust in a key does mean generally placing trust in the operators and administrators of the device that contains it, even without any possibility of administrator override of the policy governing its use. For example, even if a key is not recoverable, there is nothing to prevent a key owner exposing a signature oracle for their key, allowing anyone to sign with it. As such, if the key owner and the device administrator belong to the same organization, and have aligned priorities, there is not much practical difference between recoverable and non-recoverable keys.

However, in the example where a device is owned and managed by a service provider but leased to an end user, the key owner and the device administrators belong to separate organizations and have different priorities. In that case a verifier may prefer to reject recoverable keys.

9.4. Uniqueness of Keys

It's generally assumed that all keys are unique. This is the expected outcome for properly generated cryptographic keys, and while a collision is in principle possible by chance, it's much more likely that a collision indicates a failure in the key generation process (for example, [DSA1571]).

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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/info/rfc2986>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation

List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", RFC 5914, DOI 10.17487/RFC5914, June 2010, <https://www.rfc-editor.org/info/rfc5914>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[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.

10.2. Informative References

[DSA1571] Debian Project, "DSA-1571-1 openssl - predictable random number generator", May 2008, <https://www.debian.org/security/2008/dsa-1571>.

Appendix A. Samples

... either place samples here inline, or reference on Github. I've got a script I've used in other I-Ds to inline include files, if that's useful here.

Appendix B. ASN.1 Module

... any ASN.1 that we are defining goes here ...

```
-- TODO probably need some ASN.1 furniture around this
-- TODO need to import Certificate from RFC5280

id-device-information OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1567 }

DeviceInformation ::= SEQUENCE {
  vendor UTF8STring   -- manufacturer of device
  model UTF8STring    -- device model information
  serial UTF8STring   -- device instance information
}

id-device-subkey-information OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1568 }

DeviceSubkeyInformation ::= SEQUENCE {
  vendor UTF8STring   -- manufacturer of device
  model UTF8STring    -- device model information
  serial UTF8STring   -- device instance information
  purpose UTF8String     -- description of subkey purpose
}

id-application-key-information OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1569 }

ApplicationKeyInformation ::= SEQUENCE {
  vendor UTF8STring         -- manufacturer of device
  model UTF8STring          -- device model information
  policy OBJECT IDENTIFIER  -- policy governing key use
  vendorinfo OCTET STRING   -- vendor-specific information
}

id-attestation-bundle OBJECT IDENTIFIER ::=
        { 1 3 6 1 4 1 54392 5 1571 }

AttestationBundle ::= SEQUENCE OF Certificate
```

Appendix C. Intellectual Property Considerations

... mention any IP considerations here ...

Appendix D. Contributors and Acknowledgements

This document incorporates contributions and comments from a large group of experts. The Editors would especially like to acknowledge the expertise and tireless dedication of the following people, who attended many long meetings and generated millions of bytes of

electronic mail and VOIP traffic over the past year in pursuit of this document:

Chris Trufan (Entrust).

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada

Email: mike.ounsworth@entrust.com

Richard Kettlewell
Entrust - nCipher Security Limited
One Station Square
Cambridge
CB1 2GA
United Kingdom

Email: richard.kettlewell@entrust.com

Bruno Couillard
Crypto4A
1550 Laperriere Ave
Ottawa, On K1Z 7T2
Canada

Email: bruno@crypto4a.com

Jean-Pierre Fiset
Crypto4A
1550 Laperriere Ave
Ottawa, On K1Z 7T2
Canada

Email: jp@crypto4a.com