Workgroup: LAMPS
Internet-Draft: draft-stjohns-csr-attest-02
Published: 7 June 2023
Intended Status: Standards Track
Expires: 9 December 2023
Authors: M. StJohns
NthPermutation Security LLC

Attestation Attributes for Use with Certification Signing Requests

Abstract

This document describes two ASN.1 Attribute definitions, and an ASN.1 CLASS definition for an attestation statement structure that may be used to encode key attestation data for inclusion in PKCS10 certificate requests and in other circumstances.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document creates two ATTRIBUTE/Attribute definitions. The first Attribute may be used to carry a set of certificates or public keys that may be necessary to validate an attestation. The second Attribute carries a structure that may be used to carry key attestation statements, signatures and related data. Both of these Attribute definitions are intended to be used to carry the attestation data a Certification Authority (CA) may need to decide to issue a certificate containg the attested key.

The AttestStatement structure provides an encoding that may be used regardless of the actual format and mechanisms used by an given type of attestation. In its simplest expansion it encodes a SEQUENCE of an OBJECT IDENTIFIER and a related ASN.1 type.

For the purposes of this document, a "certificate" is a signed binding of a public key and some identifying or use information. An X.509 Certificate is one example, but the structures described below allow for the carriage of any identifiable type of certificate. Examples include Card Verifiable Certificates [TR-03110-3] and [EQMV] implicit certificates.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Definitions

The following definitions should be used in the context of this document primarily to understand the relationship of an attestation to the ASN.1 structures used to carry an attestation in PKIX structures. As of the date of this document, the IETF lacked a common nomenclature for attestation-related terms.

Attestation Engine
The secure hardware and firmware used to compose and sign an attestation.

Attestation Key
Either the private key used to sign an attestation statement, or the related public key used to verify the attestation statement, depending on context.

Attester
The entity that directs the creation of an attestation statement and who owns, controls, or is permitted to use the private attestation key.

Ancillary Attestation Data
Any data provided from a source external to the attestation engine as part of the creation of an attestation statement and/or any additional data needed for the verification of an attestation statement. The externally provided data could be a relying party originated nonce, a time stamp, session information or other data meant to be bound in time to the attestation statement. Other additional data may be an internally formatted key, or other data needed to bridge between the attestation statement and PKIX or other relying or interpretating regimes. The format of externally provided data is not under the control of the attestation engine, but may need to be transformed, such as by hashing, before it may be incorporated within the attestation statement processing. For example, a relying party may need both the "externalData" argument for the TPM 2.0 TPM2_Certify command, and the TPMT_PUBLIC structure containing the key being certified to verify an attestation.

Attestation Statement
The object, any optional ancillary data incorporated during the creation of that object, and the signature over that object, created by an attestation engine at the request of an Attester to provide evidence of a fact or set of facts within the cognizance of the attestation engine at a particular point in time." Sometimes referred to simply as an "attestation".

Attestation
The implicit or explicit collection of an attestation statement, any ancillary data, an attestation key, and a chain of trust for that key. By convention, this contains at least the minimum data needed to cryptographically validate an attestation statement and extract any policy meaning.

Key Attestation
An attestation created with respect to a particular key or key pair.

3. ASN.1 Elements

3.1. Object Identifiers

Placeholder for now, waiting on guidance.

```
-- Root of IETF's PKIX OID tree
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
     dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

-- S/Mime attributes - can be used here.
id-aa OBJECT IDENTIFIER ::= {iso(1) member-body(2) usa(840)
     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) attributes(2)}


-- Branch for attestation statement types
id-ata OBJECT IDENTIFIER ::= { id-pkix (TBD1) }
```

3.2. CertificateChoice

This is an ASN.1 CHOICE construct used to represent an encoding of a broad variety of certificate types.

```
CertificateChoice ::=
   CHOICE {
      cert Certificate, -- typical X.509 cert
      opaqueCert    [0] IMPLICIT OCTET STRING, -- Format implicitly agre
                                                            -- by sender and
      typedCert     [1] IMPLICIT TypedCert,
      typedFlatCert [2] IMPLICIT TypedFlatCert
   }
```

"Certificate" is a standard X.509 certificate that **MUST** be compliant with RFC5280. Enforcement of this constraint is left to the relying parties.

"opaqueCert" should be used sparingly as it requires the receiving party to implictly know its format. It is encoded as an OCTET STRING.

"TypedCert" is an ASN.1 construct that has the charateristics of a certificate, but is not encoded as an X.509 certificate. The certTypeField indicates how to interpret the certBody field. While it is possible to carry any type of data in this structure, it's intended the content field include data for at least one public key formatted as a SubjectPublicKeyInfo (see [RFC5912]).

```
TYPED-CERT ::= TYPE-IDENTIFIER -- basically an object id and a matching
                               -- structure encoded as a sequence
CertType ::= TYPED-CERT.&id

TypedCert ::= SEQUENCE {
              certType     TYPED-CERT.&id({TypedCertSet}),
              content     TYPED-CERT.&Type ({TypedCertSet}{@certType})
          }

TypedCertSet TYPED-CERT ::= {
             ... -- Empty for now,
             }
```

"TypedFlatCert" is a certificate that does not have a valid ASN.1 encoding. Think compact or implicit certificates as might be used with smart cards. certType indicates the format of the data in the certBody field, and ideally refers to a published specification.

```
TypedFlatCert ::= SEQUENCE {
    certType OBJECT IDENTIFIER,
    certBody OCTET STRING
}
```

3.3. AttestAttribute

By definition, Attributes within a Certification Signing Request are typed as ATTRIBUTE. This attribute definition contains one or more attestation statements of a type "AttestStatement".

```
id-aa-attestStatement OBJECT IDENTIFIER ::= { id-aa (TBDAA2) }

AttestAttribute ATTRIBUTE ::= {
  TYPE AttestStatement
  IDENTIFIED BY id-aa-attestStatement
}
```

3.4. AttestCertsAttribute

The "AttestCertsAttribute" contains a sequence of certificates that may be needed to validate the contents of an attestation statement contained in an attestAttribute. By convention, the first element of the SEQUENCE **SHOULD** contain the object that contains the public key needed to directly validate the attestAttribute. The remaining elements should chain that data back to an agreed upon root of trust for the attestation.

```
id-aa-attestChainCerts OBJECT IDENTIFIER ::= { id-aa (TBDAA1) }

attestCertsAttribute ATTRIBUTE ::= {
  TYPE SEQUENCE OF CertificateChoice
  COUNTS MAX 1
  IDENTIFIED BY id-aa-attestChainCerts
}
```

3.5. AttestStatement

An AttestStatement is an object of class ATTEST-STATEMENT encoded as a sequence fields, of which the type of the "value" field is controlled by the value of the "type" field, similar to an Attribute definition.

```
ATTEST-STATEMENT ::= CLASS {
  &id                 OBJECT IDENTIFIER UNIQUE,
  &Type,                  -- NOT optional
  &algidPresent       ParamOptions DEFAULT absent,
  &sigPresent         ParamOptions DEFAULT absent,
  &ancillaryPresent   ParamOptions DEFAULT absent,
  &sigType            DEFAULT OCTET STRING
  &ancillaryType      DEFAULT OCTET STRING

} WITH SYNTAX {
  TYPE  &Type
  IDENTIFIED BY &id
  [ALGID IS &algidPresent]
  [SIGNATURE [TYPE &sigType] IS &sigPresent]
  [ANCILLARY [TYPE &ancillaryType] IS &ancillaryPresent]
}

AttestStatement { ATTEST-STATEMENT:IOSet}  ::= SEQUENCE
  {
    type          ATTEST-STATEMENT.&id({IOSet}),
    value         ATTEST-STATEMENT.&Type({IOSet}{@type}),
    algId         [0] IMPLICIT  AlgorithmIdentifier OPTIONAL,
    signature     [1] ATTEST-STATEMENT.&sigType OPTIONAL -- NOT implicit
    ancillaryData [2] ATTEST-STATEMENT.&ancillaryType OPTIONAL
  }
```

Depending on whether the "value" field contains an entire signed attestation, or only the toBeSigned portion, the algId field may or may not be present. If present it contains the AlgorithmIdentifier of the signature algorithm used to sign the attestation statement. If absent, either the value field contains an indication of the signature algorithm, or the signature algorithm is fixed for that specific type of AttestStatement.

Similarly for the "signature" field, if the "value" field contains only the toBeSigned portion of the attestation statement, this field **SHOULD** be present. The "signature" field may by typed as any valid ASN.1 type. Opaque signature types **SHOULD** specify the use of sub-typed OCTET STRING. For example:

```
MyOpaqueSignature ::= OCTET STRING
```

If possible, the ATTEST-STATEMENT **SHOULD** specify an un-wrapped representation of a signature, rather than an OCTET STRING or BIT STRING wrapped ASN.1 structure. I.e., by specifying ECDSA-Sig-Value from PKIXAlgs-2009 (see [RFC5912]) to encode an ECDSA signature.

```
ECDSA-Sig-Value ::= SEQUENCE {
  r  INTEGER,
  s  INTEGER
}
```

The ancillaryData field contains data provided externally to the attestation engine,and/or data that may be needed to relate the attestation to other PKIX elements. The format or content of the externally provided data is not under the control of the attestation engine. For example, this field might contain a freshness nonce generated by the relying party, a signed time stamp, or even a hash of protocol data or nonce data. See below for a few different examples.

4. IANA Considerations

The IANA is requested to open one new registrie, allocate a value from the "SMI Security for PKIX Module Identifier" registry for the included ASN.1 module, and allocate values from "SMI Security for S/MIME Attributes" to identify two Attributes defined within.

4.1. Object Identifier Allocations

4.1.1. Module Registration - SMI Security for PKIX Module Identifer

*Decimal: IANA Assigned - Replace TBDMOD
*Description: Attest-2023 - id-mod-pkix-attest-01
*References: This Document

4.1.2. Object Identifier Registrations - SMI Security for S/MIME Attributes

Attest Statement

*Decimal: IANA Assigned - Replace TBDAA2
*Description: id-aa-attestStatement
*References: This Document

Attest Certificate Chain

*Decimal: IANA Assigned - Replace TBDAA1
*Description: id-aa-attestChainCerts
*References: This Document

4.2. "SMI Security for PKIX Attestation Statement Formats" Registry

Please open up a registry for Attestation Statement Formats within the SMI-numbers registry, allocating an assignment from id-pkix ("SMI Security for PKIX" Registry) for the purpose.

- *Decimal: IANA Assigned - replace TBD1
- *Description: id-ata
- *References: This document
- *Initial contents: None
- *Registration Regime: Specification Required. Document must specify an ATTEST-STATEMENT definition to which this Object Identifier shall be bound.

Columns:

- *Decimal: The subcomponent under id-ata
- *Description: Begins with id-ata
- *References: RFC or other document

5. Security Considerations

The attributes and structures defined in this document are primarily meant to be used as additional Attributes for a PKCS10 Certification Signing Request (CSR). As such, it's up to the receiving/relying party to place as much or as little trust in the contents of these attributes as necessary to satisfy its own policies.

A relying party will need either a specification defining how an attestation type was formed and how to validate that type, or a trusted method of verifying the attestation. In the former case, a relying party should consider the information available from any certificate chain covering the attesting key when deciding to accept the attestation.

Most attestations will need to provide a method to convert the attested key representation into the equivalent SubjectPublicKey info structure and the attested key **MUST** be compared for equivalence to the public key provided in the CSR before accepting the attestation.

The relying party, as always, is responsible for setting the rules for what it will accept. The presence of an AttestAttribute is not required by any current standard, but such attribute may provide the relying party with additional assurance as a prerequisite to issuing certificates or other credentials. That acceptance criteria is out of scope for this document. Whether to require an AttestAttribute or its contents in any specific use case is out-of-scope for this document.

6. References

6.1. 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>.

6.2. Informative References

[EQMV] Certicom Research, "Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV)", Standards for Efficient Cryptography SEC-4, January 2013, <https://www.secg.org/sec4-1.0.pdf>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[TPM20] Trusted Computing Group, "Trusted Platform Module Library - Part 1: Architecture", TPM 2.0 Module Library Part1, 00-01.59, November 2019.
Trusted Computing Group, "Trusted Platform Module Library - Part 2: Structures", TPM 2.0 Module Library Part2, 00-01.59, November 2019.
Trusted Computing Group, "Trusted Platform Module Library - Part 2: Commands", TPM 2.0 Module Library Part2, 00-01.59, November 2019.

[TR-03110-3]
Federal Office for Information Security, "Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token - Part 3 Common Specifications V2.21", Federal Republic of Germany, Technical Guideline TR-03110, December 2016, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI_TR-03110_Part-3-V2_2.pdf?__blob=publicationFile&v=1>.

Appendix A. Examples

A.1. Simple Attestation Example

This is a fragment of ASN.1 meant to demonstrate an absolute minimal definition of an ATTEST-STATEMENT. A similar fragment could be used to define an ATTEST-STATEMENT for an opaque HSM vendor specific atterstation model.

```
-- This OCTET STRING is not like any other OCTET STRING
-- Please see https://example.com/simple-attest.txt,
-- Structure labled "Mike's simple attest" for the
-- structure of this field and how to verify the attestation

MikesSimpleAttestData ::= OCTET STRING

mikesSimpleAttestOid OBJECT IDENTIFIER ::= { id-mikes-root 1 }

MikesSimpleAttest ATTEST-STATEMENT ::= {
  TYPE MikesSimpleAttestData
  IDENTIFIED BY mikesSimpleAttestOid
  -- These are all implied
  -- ALGID IS absent
  -- SIGNATURE is absent
  -- ANCILLARY is absent
}
```

A.2. Example TPM V2.0 Attestation Attribute - Non Normative

What follows is a fragment of an ASN.1 module that might be used to define an attestation statment attribute to carry a TPM V2.0 key attestation - i.e., the output of the TPM2_Certify command. This is an example and NOT a registered definition. It's provided simply to give an example of how to write an ATTEST-STATEMENT definition and module.

```
-- IMPORT these.
-- PKI normal form for an ECDSA signature
ECDSA-Sig-Value ::= SEQUENCE {
  r INTEGER,
  s INTEGER
  }

-- Octet string of size n/8 where n is the
-- bit size of the public modulus
RSASignature ::= OCTET STRING

-- One or the other of these depending on the value in TPMT_SIGNATURE
TpmSignature CHOICE ::= {
  ecSig [0] IMPLICIT ECDSA-Sig-Value,
  rsaSig [1] IMPLICIT RSASignature
  }

-- The TPM form of the public key being attested.
-- Needed to verify the attestation - this is the TPMT_PUBLIC structure.
TpmtPublic ::= OCTET STRING

-- The TPMS_ATTEST structure as defined in TPM2.0
-- Unwrapped from the TPM2B_ATTEST provided
-- by the TPM2_Certify command.
TpmsAttest ::= OCTET STRING

-- The qualifying data provided to a TPM2_Certify call, may be absent
-- This is the contents of data field of the TPM2B_DATA structure.
QualifyingData ::= OCTET STRING

TpmAncillary ::= SEQUENCE {
   toBeAttestedPublic TpmtPublic,
   qualifyingData QualifyingData OPTIONAL
   }

-- This represents a maximally unwrapped TPM V2.0 attestation.  The
-- output of TPM2_Certify is a TPM2B_ATTEST and a TPMT_SIGNATURE.
-- The former is unwrapped into a TPMS_ATTEST and the latter is
-- decomposed to provide the contents of the algId and signature fields.

--
-- This attestation statement can be verified by:
-- Signature siggy = Signature.getInstance (stmt.algId);
-- siggy.init (attestPublicKey, VERIFY);
-- siggy.update ((short)stmt.value.length) // todo: big or little endian
-- siggy.update (stmt.value.data)
-- bool verified = siggy.verify (getSigData(stmt.signature)); //
unwrap the signature
--
```

```
TpmV2AttestStatement ATTEST STATEMENT ::= {
   TYPE TpmsAttest
   IDENTIFIED BY id-ata-tpmv20-1
   ALGID IS present
   SIGNATURE TYPE TpmSignature IS present
   ANCILLARY TYPE TpmAncillary IS present
   }
```

This attestation is the result of executing a TPM2_Certify command over a TPM key. See [TPM20] for more details.

The data portion of the value field encoded as OCTET STRING is the attestationData from the TPM2B_ATTEST produced by the TPM. In other words, strip off the TPM2B_ATTEST "size" field and place the TPMS_ATTEST encoded structure in the OCTET STRING data field.

The algId is derived from the "sigAlg" field of the TPMT_SIGNATURE structure.

The signature field is a TpmSignature, created by transforming the TPMU_SIGNATURE field to the appropriate structure given the signature type.

The ancillary field contains a structure with the TPMT_PUBLIC structure that contains the TPM's format of the key to be attested. The attestation statement data contains a hash of this structure, and not the key itself, so the hash of this structure needs to be compared to the value in the attestation attestation statement. If that passes, the key needs to be transformed into a PKIX style key and compared to the key in the certificate signing request to complete the attestation verification.

The ancillary field also contains an optional OCTET STRING which is used if the TPM2_Certify command is called with a non-zero length "qualifyingData" argument to contain that data.

An AttestCertChain attribute **MUST** be present if this attribute is used as part of a certificate signing request.

Appendix B. ASN.1 Module for Attestation

The following module imports definitions from modules defined in [RFC5912] and [RFC8551].

IANA Note: Please replace TBDMOD, TBD1 and TBD2 with assigned values.

```
-- This module provides a definition for two attributes thay may be
-- used to carry key attestation information within a
-- CertificationSigningRequest (aka PKCS10), or for other purposes.

-- IANA - Value needed
Attest-2023
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix-attest-01(TBDMOD) }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS

Attribute, SingleAttribute, id-pkix, Certificate
FROM PKIX1Explicit-2009
      {iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-explicit-02(51)}

ATTRIBUTE,AttributeSet
FROM PKIX-CommonTypes-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
     mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57)}

ParamChoice
FROM AlgorithmInformation-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58)}

id-aa
FROM SecureMimeMessageV3dot1
     { iso(1) member-body(2) us(840) rsadsi(113549)
        pkcs(1) pkcs-9(9) smime(16) modules(0) msg-v3dot1(21) }

-- Repeated here for easy reference.
--   id-aa OBJECT IDENTIFIER ::= {iso(1) member-body(2) usa(840)
--      rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) attributes(2)}


-- IANA - Values needed
-- Branch for attestation statement types
id-ata OBJECT IDENTIFIER ::= { id-pkix (TBD1) }

-- A general comment is that a certificate is a signed binding between
-- public key and some identifying info. Below "cert" is an X.509
-- "Certificate". "opaqueCert" is just string of bytes that the
-- receiving CA must know how to parse given information not carried
-- in this object.  "typedCert" and "typedFlatCert" both use an OID to
-- identify their types, but differ in that the encoding for typedCert
-- is always valid ASN1, whereas the typedFlatCert is just a string of
-- bytes that must be interpreted according to the type.  Note that a
-- typedFlatCert MAY contain an encapsulated ASN1 object, but this is
-- not the best use of the type and is hereby discouraged.

CertificateChoice ::=
   CHOICE {
      cert Certificate, -- typical X.509 cert
      opaqueCert [0] IMPLICIT OCTET STRING,
      typedCert [1] IMPLICIT TypedCert, -- not ASN1 parseable

      typedFlatCert [2] IMPLICIT TypedFlatCert
   }

-- Cribbed from definition of CONTENT-TYPE
-- Alternately as TypedCert ::= SingleAttribute
--
TYPED-CERT ::= TYPE-IDENTIFIER -- object id and a matching ASN1
                               -- structure encoded as a sequence
CertType ::= TYPED-CERT.&id

TypedCert ::= SEQUENCE {
              certType     TYPED-CERT.&id({TypedCertSet}),
              content      TYPED-CERT.&Type ({TypedCertSet}{@certType})
          }

TypedCertSet TYPED-CERT ::=
             ... -- Empty for now,
             }


-- The receiving entity is expected to be able to parse the certBody
-- field given the value of the certType field.  This differs from
-- TypedCert in that the contents of the certBody field are not
-- necessarily well formed ASN1 in this case the certType tells you
-- how to parse the body of the OCTET STRING,

TypedFlatCert ::= SEQUENCE {
    certType OBJECT IDENTIFIER,
    certBody OCTET STRING
}


-- A sequence of certificates used to validate an attestation chain.
-- By convention, the first certificate in the chain is the one that
-- contains the public key used to verify the attestation.  If the
-- related attestStatementAttribute contains more than a single
-- attestation, this attribute is expected to contain all of the
-- certificates needed to validate all attestations

id-aa-attestChainCerts OBJECT IDENTIFIER ::= { id-aa (TBDAA1) }


attestCertCertsAttribute ATTRIBUTE ::= {
        TYPE SEQUENCE OF CertificateChoice
        COUNTS MAX 1
        IDENTIFIED BY id-aa-attestChainCerts
    }

-- If the signature is provided separately, the value field need not
-- contain the signature.  Note that some attestation methods include
-- a signature method in the part signed by the signature and some do
-- not.

ATTEST-STATEMENT ::= CLASS {
  &id                 OBJECT IDENTIFIER UNIQUE,
  &Type,                  -- NOT optional
  &algidPresent       ParamOptions DEFAULT absent,
  &sigPresent         ParamOptions DEFAULT absent,
  &sigType            DEFAULT OCTET STRING
  &ancillaryPresent   ParamOptions DEFAULT absent,
  &ancillaryType      DEFAULT OCTET STRING

} WITH SYNTAX {
  TYPE  &Type
  IDENTIFIED BY &id
  [ALGID IS &algidPresent]
  [SIGNATURE [TYPE &sigType] IS &sigPresent]
  [ANCILLARY [TYPE &ancillaryType] IS &ancillaryPresent]
}

AttestStatement { ATTEST-STATEMENT:IOSet}  ::= SEQUENCE
  {
    type          ATTEST-STATEMENT.&id({IOSet}),
    value         ATTEST-STATEMENT.&Type({IOSet}{@type}),
    algId         [0] IMPLICIT  AlgorithmIdentifier OPTIONAL,
    signature     [1] ATTEST-STATEMENT.&sigType OPTIONAL -- NOT implicit
    ancillaryData [2] ATTEST-STATEMENT.&ancillaryType OPTIONAL
  }

-- An attribute that contains a attestation statement.

id-aa-attestStatement OBJECT IDENTIFIER ::= { id-aa (TBDAA2) }

attestAttribute ATTRIBUTE ::= {
        TYPE AttestStatement
        IDENTIFIED BY id-aa-attestStatement
    }
```

END

Acknowledgements

Thanks to Russ Housley for a first and useful pass over the original ASN.1. Thanks to Mike Ounsworth for not complaining too much when I wrote this. Placeholder here for people who spend time reviewing the draft!

Author's Address

Michael StJohns
NthPermutation Security LLC
Germantown, MD 20874
United States of America

Email: msj@nthpermutation.com