Internet Draft
Document: draft-ietf-pkix-pkalgs-supp-01.txt
Expires: September 2002

Ari Singer, NTRU
William Whyte, NTRU
March 2002

Supplemental Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and CRL Profile

<draft-ietf-pkix-pkalgs-supp-01.txt>

Status of this Memo

Conventions used in this document

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].

Abstract

This document specifies algorithm identifiers and ASN.1 encoding formats for digital signatures and subject public keys, including NTRUSign digital signatures and NTRUEncrypt and NTRUSign subject public keys used in the Internet X.509 Public Key Infrastructure (PKI). Digital signatures are used to sign certificates and certificate revocation lists (CRLs). Certificates include the public key of the named subject. This document is intended to be a companion to draft-ietf-pkix-ipki-pkalgs-05.txt [PKIX-ALGS] and may be merged with that document in future revisions if approved by the PKIX working group.

Table of Contents

1. Overview

This document specifies algorithm identifiers and ASN.1 encoding formats for digital signatures and subject public keys used in the Internet X.509 Public Key Infrastructure (PKI). This specification supplements RFC 2459 [RFC2459], "Internet Public Key Infrastructure: X.509 Certificate and CRL Profile". Implementations of this specification must also conform to RFC 2459 [RFC2459]. This document is being written concurrently with the PKIX public key algorithms Internet Draft [PKIX-ALGS] (the latest version as of this writing is draft-ietf-pkix-ipki-pkalgs-05.txt). It is intended that when this document is completed and approved by the PKIX working group that it be merged with that document. The format of this document is written to approximately match the format of that Internet Draft.

This specification defines the contents of the signatureAlgorithm, signatureValue, signature and subjectPubliKeyInfo fields within Internet X.509 certificates and CRLs.

This document does not currently introduce any new one-way hash functions, but it specifies the use of SHA-256, SHA-384 and SHA-512 hash algorithms as defined in the draft of FIPS 180-2 [FIPS180-2] as well as the SHA-1 hash algorithm as defined in FIPS 180-1 [FIPS180-1] with the NTRUSign signature algorithm. It is anticipated that future revisions will include the algorithm identifiers and ASN.1 encoding of the FIPS 180-2 hash algorithms.

This specification describes the encoding of digital signatures generated with the following cryptographic algorithms;

* NTRUSign Signature Scheme (NTRUSign).

It is anticipated that future revisions of this document will include the extended version of the Digital Signature Algorithm (DSA) [FIPS186-2], which has not yet been published. In addition, it is anticipated that the document will include the algorithm identifiers and ASN.1 encoding of pre-existing algorithms (e.g. RSA) when used in conjunction with the FIPS 180-2 hash algorithms.

This document specifies the contents of the subjectPublicKeyInfo field in Internet X.509 certificates. For each algorithm, the appropriate alternatives for the keyUsage extension are provided. This specification describes encoding formats for public keys used with the following cryptographic algorithms:

* NTRUEncrypt Encryption Scheme (NTRUEncrypt)
* NTRUSign Signature Scheme (NTRUSign)

2. Algorithm Support

This section describes cryptographic algorithms that may be used with the Internet X.509 Certificate and CRL Profile. In particular, it describes the NTRUSign digital signature algorithm, which may be used to sign certificates and CRLs. In addition, this section identifies OIDs and ASN.1 encoding for NTRUSign and NTRUEncrypt

public keys contained in a certificate. It is anticipated that additional algorithms, such as the extended version of DSA, will be included in future revisions.

Conforming CAs and application are not required to support the algorithms or algorithm identifiers described in this section. However, conforming CAs and applications that use the algorithms identified here MUST support them as specified.

2.1 Signature Algorithms

Certificates and CRLs conforming to RFC 2459 [RFC2459] may be signed with any public key signature algorithm. The certificate or CRL indicates the algorithm through an algorithm identifier, which appears in the signatureAlgorithm field within the Certificate or CertificateList. An algorithm identifier consists of an OID and (optionally) associated parameters. This section describes OIDs and parameter encoding for NTRUSign.

Signature algorithms are always used in conjunction with a one-way hash function.

The data to be signed (e.g. the one-way hash function output value) is formatted for the signature algorithm to be used. Then, a private key operation (e.g. NTRUSign signature primitive) is performed to generate the signature value. This signature value is then ASN.1 encoded as a BIT STRING and included in the Certificate or CertificateList in the signature field.

2.1.1 NTRUSign Signature Algorithm

The NTRUSign signature algorithm was invented by Hoffstein, Howgrave-Graham, Pipher, Silverman and Whyte. It is defined in Efficient Embedded Security Standard (EESS) #1 [EESS#1]. This profile defines a single signature algorithm, the NTRUSign signature algorithm with the SHA-1, SHA-256, SHA-384 or SHA-512 one-way hash function.

The signature algorithm is implemented using the padding and encoding conventions described in EESS #1 [EESS#1]. The message digest is computed using the SHA-1 Hash Algorithm [FIPS180-1] or any of the SHA-2 algorithms [FIPS180-2] and the message digest is encoded using the MGF1 mask generation function as specified in Std IEEE 1363-2000 [IEEE1363].

Unlike previously defined public-key signature algorithms, the object identifier for the NTRUSign signature algorithm does not specify the hash function. Rather, the parameter field in the AlgorithmIdentifier contains an indication of the hash function as well as the encoding methods that are to be used.

The ASN.1 object identifier used to identify this signature algorithm is named id-ntru-EESS1v1-NTRUSign and is given by the following ASN.1:

```
ntru OBJECT IDENTIFIER ::=
        {iso(1) identified-organization(3) dod(6) internet(1)
        private(4) enterprises(1) ntruCryptosystems (8342) }

id-eess1  OBJECT IDENTIFIER ::= {ntru eess(1) 1}

id-eess1-algs  OBJECT IDENTIFIER ::= {id-eess1 1}

id-ntru-EESS1v1-NTRUSign  OBJECT IDENTIFIER ::=
        {id-eess1-algs 3}
```

When this OID appears in the signatureAlgorithm field or the signature field of an X.509 certificate, the encoding SHALL omit the parameters field. That is, the AlgorithmIdentifier shall be a SEQUENCE of one component: the OBJECT IDENTIFIER id-ntru-EESS1v1-SVSSA.

The NTRUSign parameters in the subjectPublicKeyInfo field of the certificate of the issuer shall apply to the verification of the signature.

When signing, the NTRUSign algorithm generates a signature polynomial. This polynomial SHALL be encoded as an OCTET STRING as described in EESS #1 [EESS#1]. The signature SHALL be ASN.1 encoded using the following ASN.1 structure:

```
NTRUSignSignedData ::= NTRUPublicVector

NTRUPublicVector ::= CHOICE {
     modQVector        [0] IMPLICIT ModQVector,
     packedModQVector  [1] IMPLICIT PackedModQVector,
     ...}

ModQVector ::= OCTET STRING

PackedModQVector ::= OCTET STRING
```

The field choices of type NTRUPublicVector have the following meanings:

modQVector is the representation of the NTRUPublicVector in unpacked form. For a polynomial of degree N-1 with coefficients reduced mod q, each of the N bytes of the OCTET STRING represent integers x in the range 0 <= x < q corresponding to the coefficient values of the polynomial from lowest degree to highest.

packedModQVector is the representation of the NTRUPublicVector in packed form. For a polynomial of degree N-1 with

coefficients reduced mod q, each log_2(q) bits of the OCTET STRING represent integers x in the range 0 <= x < q corresponding to the coefficient values of the polynomial from lowest degree to highest. The values are concatenated bitwise,

without any intermediate padding, and irrespective of the byte boundaries. If necessary, zero bits are appended to the packed data in order to make the length a multiple of 8 bits.

Implementations that sign certificates using NTRUSign SHOULD encode the signature as a ModQVector.

2.2 Subject Public Key Algorithms

Certificates conforming to RFC 2459 [RFC2459] may convey a public key for any public key algorithm. The certificate indicates the algorithm through an algorithm identifier. This algorithm identifier consists of an OID and optionally associated parameters.

This section identifies preferred OIDs and parameters for the NTRUEncrypt and NTRUSign algorithms. Conforming CAs MUST use the identified OIDs when issuing certificates containing public keys for these algorithms. Conforming applications supporting any of these algorithms MUST, at a minimum, recognize the OIDs identified in this section.

2.2.1 NTRUEncrypt Keys

This section identifies the preferred OID and parameter encoding for the inclusion of an NTRUEncrypt public key in a certificate. The NTRUEncrypt encryption algorithm is defined in EESS #1 [EESS#1].

The OID id-ntru-EESS1v1-SVES identifies NTRUEncrypt public keys.

```
id-ntru-EESS1v1-SVES  OBJECT IDENTIFIER ::= {id-eess1-algs 1}
```

The id-ntru-EESS1v1-SVES OID is intended to be used in the algorithm field of a value of type AlgorithmIdentifier. NTRUEncrypt requires use of certain parameters with the public key. The parameters may be implied by context, implicitly included through reference of a degree, implicitly included through reference of a standard parameter set or explicitly included in the certificate. The parameters associated with id-ntru-EESS1v1-SVES are EESS1v1-SVES-Parameters.

```
EESS1v1-SVES-Parameters ::= CHOICE {
            degree                  Degree,
            standardNTRUParameters  StandardNTRUParameters,
            explicitNTRUParameters  ExplicitNTRUParameters,
            externalParameters      NULL
            }
```

When the parameters are implied by context, the parameters field SHALL contain externalParameters, which is a value of the ASN.1 type NULL.

When the parameters are specified by degree, the values are restricted to 251, 347 and 503. For the three permitted choices, the parameters are defined to be ees251ep1, ees347ep1 and ees503ep1

respectively as defined in EESS #1 [EESS#1]. Specifying the degree is the preferred way for transmitting parameter information for the scheme when the parameters are not implied by context.

```
Degree ::= INTEGER  (251 | 347 | 503, ...)
```

When the parameters are specified by reference of a standard, the parameters shall consist of an OID chosen from the list NTRUParameters. The current list of NTRUParameters OIDs is:

```
StandardNTRUParameters ::= OIDS.&id({NTRUParameters})

NTRUParameters  OIDS ::= {
        { OID id-ees251ep1 }|
        { OID id-ees347ep1 }|
        { OID id-ees503ep1 },
        ...}
```

The above object identifiers are specified by:

```
id-eess1-params  OBJECT IDENTIFIER ::= {id-eess1 2}

id-ees251ep1  OBJECT IDENTIFIER ::= {id-eess1-params 1}
id-ees347ep1  OBJECT IDENTIFIER ::= {id-eess1-params 2}
id-ees503ep1  OBJECT IDENTIFIER ::= {id-eess1-params 3}
```

When the parameters are explicitly included, they SHALL be encoded in the ASN.1 structure ExplicitNTRUParameters:

```
ExplicitNTRUParameters ::= SEQUENCE {
        version         Version,
        degree          INTEGER,
        bigModulus      INTEGER,
        smallModulus    SmallModulus,
        mrgm            NTRUMRGMAlgorithmIdentifier,
        db              INTEGER,
        bvgm            NTRUBVGMAlgorithmIdentifier,
        ...}

Version ::= INTEGER { v0(0) } (v0, ...)

SmallModulus ::= CHOICE {
        integerValue    INTEGER,
        polynomialValue NTRUGeneralPolynomial
        }

NTRUGeneralPolynomial ::= SEQUENCE {
        numberOfEntries         INTEGER,
        modulus                 INTEGER,
        coefficients            GeneralVector
```

```
            }

     GeneralVector ::= OCTET STRING
```

The fields of type NTRUGeneralPolynomial have the following meanings:

numberOfEntries is the number of coefficients used to represent the polynomial - this number is equal to the degree of the polynomial plus 1.

modulus is an upper bound on the value of the coefficients.

coefficients is the list of numberOfEntries coefficients, represented in order from lowest degree to highest degree. If modulus < 257, each coefficient is stored in a single byte. If modulus > 256 and modulus < 2^16, each coefficient is stored in two bytes.

The fields of type SmallModulus have the following meanings:

integerValue is the value of p if p is an integer.

polynomialValue is the value of p if p is a polynomial.

The fields of type ExplicitNTRUParameters have the following meanings:

version is the version number, for compatibility with future revisions of this document. It SHALL be 0 for this version of the document.

degree is the value N.

bigModulus is the value q. q will be 256 or less.

smallModulus is the value p. It SHALL be represented with the SmallModulus type.

mrgm identifies the message representative generation method using an allowed AlgorithmIdentifier.

db is the size of the random component.

bvgm identifies the blinding value generation method using an allowed AlgorithmIdentifier.

The ASN.1 for the mrgm used in ExplicitNTRUParameters is specified below.

```
NTRUMRGMAlgorithmIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1MRGMs}}

NTRUEESS1v1MRGMs ALGORITHM ::= {
```

```
            {OID id-mrgm-ntru-1 PARMS NTRUMRGM1-params},
            ...}

     id-eess1-encodingMethods  OBJECT IDENTIFIER ::= {id-eess1 3}
```

```
id-mrgm-ntru-1  OBJECT IDENTIFIER ::=
        {id-eess1-encodingMethods 1}

NTRUMRGM1-params ::= NTRUHashAlgorithmIdentifier

NTRUHashAlgorithmIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1Hashes}}
```

The identifier id-mrgm-ntru-1 identifies the message representative generation method MRGM-NTRU1, defined in EESS #1 [EESS#1]. The parameters identify the hashing mechanism using an allowed AlgorithmIdentifier.

```
NTRUEESS1v1Hashes ALGORITHM ::= {
        {OID id-sha1    PARMS NULL}|
        {OID id-sha256  PARMS NULL }|
        {OID id-sha384  PARMS NULL }|
        {OID id-sha512  PARMS NULL },
        ...}
```

These identifiers identify the one-way hash algorithms SHA-1 [FIPS180-1] and SHA-2 [TBD].

The ASN.1 for the bvgm used in ExplicitNTRUParameters is specified below.

```
NTRUBVGMAlgorithmIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1BVGMs}}

NTRUEESS1v1BVGMs ALGORITHM ::= {
        {OID id-bvgm-ntru-1 PARMS NTRUBVGM1-params}|
        {OID id-bvgm-ntru-2 PARMS NTRUBVGM2-params},
        ...}

id-bvgm-ntru-1  OBJECT IDENTIFIER ::=
        {id-eess1-encodingMethods 2}

NTRUBVGM1-params ::= SEQUENCE {
        c       INTEGER,
        prng    NTRUPRNGAlgorithmIdentifier,
        dr      INTEGER
        }

id-bvgm-ntru-2  OBJECT IDENTIFIER ::=
        {id-eess1-encodingMethods 3}

NTRUBVGM2-params ::= SEQUENCE {
        c       INTEGER,
        prng    NTRUPRNGAlgorithmIdentifier,
```

```
            dr1     INTEGER,
            dr2     INTEGER,
            dr3     INTEGER
            }
```

The identifier id-bvgm-ntru-1 identifies blinding value generation method BVGM-NTRU1, defined in EESS #1 [EESS#1]. The identifier id-bvgm-ntru-2 identifies blinding value generation method BVGM-NTRU2, defined in EESS #1 [EESS#1].

The fields of type NTRUBVGM1-params have the following meanings:

c is the random polynomial generation constant used to select the polynomial r.

prng identifies the pseudo-random number generation algorithm using an allowed AlgorithmIdentifier.

dr is the number of 1s in the blinding value r.

The fields of type NTRUBVGM2-params have the following meanings:

c is the random polynomial generation constant used to select the polynomial r.

prng identifies the pseudo-random number generation algorithm using an allowed AlgorithmIdentifier.

dr1 is the number of 1s in the blinding value component r1.

dr2 is the number of 1s in the blinding value component r2.

dr3 is the number of 1s in the blinding value component r3.

The allowed pseudo-random number generation algorithms are defined by:

```
NTRUPRNGAlgorithmIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1PRNGs}}

NTRUEESS1v1PRNGs ALGORITHM ::= {
        NTRUMGFAlgorithms,
        ...}
```

This identifies the pseudo-random number generation algorithm to be used when generating blinding values. The only allowed algorithms are MGF1 (see [IEEE 1363]) using SHA-1 [FIPS180-1] or SHA-2 [FIPS180-2].

```
NTRUMGFAlgorithms ALGORITHM ::= {
        {OID id-mgf1 PARMS MGF1Parameters},
        ...}
```

```
pkcs-1  OBJECT IDENTIFIER ::=
        { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        1}
```

```
id-mgf1  OBJECT IDENTIFIER ::= {pkcs-1 8}

MGF1Parameters ::= AlgorithmIdentifier {{NTRUEESS1v1Hashes}}
```

The NTRUEncrypt public key MUST be encoded using the ASN.1 type NTRUPublicKey.

```
NTRUPublicKey ::= SEQUENCE {
        publicKeyVector         NTRUPublicVector,  -- h
        ntruKeyExtensions       NTRUKeyExtensions OPTIONAL
        }

NTRUKeyExtensions ::=
        SEQUENCE SIZE(1..MAX) OF NTRUKeyExtension

NTRUKeyExtension ::= CHOICE {
        keyID           [0] IMPLICIT INTEGER,
        ...}
```

The fields of the type NTRUPublicKey have the following meanings:

publicKeyVector is the polynomial h. If the NTRUPublicVector is a ModQVector, each coefficient will be represented by one byte starting with the lowest degree and going to the highest. If the NTRUPublicVector is a PackedModQVector, this is the OCTET STRING representing h obtained using RE2BSP and then BS2OSP as defined in EESS #1 [EESS#1]. All coefficients up to X^(N-1) SHALL be explicitly included in publicKeyVector. Representing the NTRUEncrypt public key as a ModQVector is the preferred method.

ntruKeyExtensions is provided for future extensibility. Only one extension is currently defined.

The fields of the type NTRUKeyExtension have the following meanings:

keyID can be used to associate a unique key identifier with the key.

If the keyUsage extension is present in an end entity certificate that conveys an NTRUEncrypt public key, any combination of the following values MAY be present:

keyEncipherment;
dataEncipherment;

If the keyUsage extension is present in a CA certificate that conveys an NTRUEncrypt public key, any combination of the following values MAY be present:

keyEncipherment; and
dataEncipherment.

2.2.2 NTRUSign Keys

This section identifies the preferred OID and parameter encoding for the inclusion of an NTRUSign public key in a certificate. The NTRUSign signature algorithm is defined in EESS #1 [EESS#1].

The OID id-ntru-EESS1v1-SVSSA identifies NTRUSign public keys.

```
id-ntru-EESS1v1-NTRUSign  OBJECT IDENTIFIER ::=
        {id-eess1-algs 3}
```

The id-ntru-EESS1v1-NTRUSign OID is intended to be used in the algorithm field of a value of type AlgorithmIdentifier. NTRUSign requires use of certain parameters with the public key. The parameters may be implied by context (e.g. they may be inherited from the issuer), implicitly included through reference of a degree, implicitly included through reference of a standard parameter set or explicitly included in the certificate. The parameters associated with id-ntru-EESS1v1-NTRUSign are EESS1v1-NTRUSign-Parameters.

```
EESS1v1-NTRUSign-Parameters ::= CHOICE {
        degree                  Degree,
        standardNTRUSignParameters
                                StandardNTRUSignParameters,
        explicitNTRUSignParameters
                                ExplicitNTRUSignParameters,
        externalParameters      NULL
        }
```

When the parameters are implied by context, the parameters field SHALL contain externalParameters, which is the ASN.1 value NULL.

When the parameters are specified by degree, the value is restricted to 251. For the permitted choice, the parameters are defined to be ees251sp2 as defined in EESS #1 [EESS#1]. Specifying the degree is the preferred way for transmitting parameter information for the scheme when the parameters are not implied by context.

When the parameters are specified by reference of a standard, the parameters shall consist of an OID chosen from the list NTRUSignParameters. The current list of NTRUSignParameters OIDs is:

```
StandardNTRUSignParameters ::= OIDS.&id({NTRUSignParameters})

NTRUSignParameters  OIDS ::= {
        { OID id-ees251sp2 },
        ...}
```

The above object identifier is specified by:

```
id-ees251sp2  OBJECT IDENTIFIER ::= {id-eess1-params 7}
```

When the parameters are explicitly included, they SHALL be encoded in the ASN.1 structure ExplicitNTRUSignParameters:

```
ExplicitNTRUSignParameters ::= SEQUENCE {
        version                 Version,
        degree                  INTEGER,
        bigModulus              INTEGER,
        normBound               INTEGER,
        messageRandLength       INTEGER,
        hash                    NTRUSignHashAlgIdentifier,
        mrgm                    NTRUSignMRGMAlgIdentifier,
        ...}
```

The fields of type ExplicitNTRUSignParameters have the following meanings:

version is the version number, for compatibility with future revisions of this document. It SHALL be 0 for this version of the document.

degree is the value N.

bigModulus is the value q. q will be 256 or less.

normBound is the maximum norm of the signature

messageRandLength is the length of the randomization padding appended to the message digest before generating the message representative

hash identifies the hash algorithm used using an allowed AlgorithmIdentifier.

mrgm identifies the message representative generation method using an allowed AlgorithmIdentifier.

The AlgorithmIdentifiers for the field hash of ExplicitNTRUSignParameters are chosen from the set NTRUEESS1v1Hashes, which is defined in section 2.2.1.

```
NTRUSignHashAlgIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1Hashes}}
```

The AlgorithmIdentifiers for the field mrgm of ExplicitNTRUSignParameters are specified below.

```
NTRUSignMRGMAlgIdentifier ::=
                AlgorithmIdentifier {{NTRUSignEESS1v1MRGMs}}

NTRUSignEESS1v1MRGMs ALGORITHM ::= {
        {OID id-mrgm-ntrusign-1 PARMS NTRUSignMRGM1-params}|
```

```
            {OID id-mrgm-ntrusign-2 PARMS NTRUSignMRGM2-params},
            ...}

    id-mrgm-ntrusign-1  OBJECT IDENTIFIER ::=
```

```
        {id-eess1-encodingMethods 6}

    NTRUSignMRGM1-params ::= NTRUSignPRNGAlgIdentifier

    id-mrgm-ntrusign-2  OBJECT IDENTIFIER ::=
            {id-eess1-encodingMethods 7}

    NTRUSignMRGM2-params ::= SEQUENCE {
            c               INTEGER,
            numGroups       INTEGER,
            numElements     INTEGER,
            prng            NTRUSignPRNGAlgIdentifier
            }

    NTRUSignPRNGAlgIdentifier ::=
                    AlgorithmIdentifier {{NTRUEESS1v1PRNGs}}
```

The identifier id-mrgm-ntrusign-2 identifies the message representative generation method MRGM-NTRUSign1, defined in EESS #1 [EESS#1]. The identifier id-mrgm-ntrusign-2 identifies the message representative generation method MRGM-NTRUSign2, defined in EESS #1 [EESS#1].

The fields of type NTRUSignMRGM1-params have the following meanings:

NTRUSignPRNGAlgIdentifier is the pseudo-random number generation method using an allowed AlgorithmIdentifier

The fields of type NTRUSignMRGM2-params have the following meanings:

c is the random polynomial generation constant used to select the message representative.

numGroups is the number of factors combined to form the message representative.

numElements is the number of non-zero coefficients in each factor of the message representative

prng identifies the pseudo-random number generation method using an allowed AlgorithmIdentifier.

The allowed pseudo-random number generation algorithms are chosen from the set NTRUEESS1v1PRNGs, which is defined in section 2.2.1.

The NTRUSign public key MUST be encoded using the ASN.1 type NTRUSignPublicKey.

```
    NTRUSignPublicKey ::= SEQUENCE {
            publicKeyVector         NTRUPublicVector,  -- h
```

```
            ntruSignKeyExtensions  NTRUSignKeyExtensions OPTIONAL
            }

     NTRUSignKeyExtensions ::=
```

```
            SEQUENCE SIZE(1..MAX) OF NTRUSignKeyExtension

    NTRUSignKeyExtension ::= CHOICE {
            keyID           [0] IMPLICIT INTEGER,
            ...}
```

The fields of the type NTRUSignPublicKey have the following meanings:

publicKeyVector is the polynomial h. If the NTRUPublicVector is a ModQVector, each coefficient will be represented by one byte starting with the lowest degree and going to the highest. If the NTRUPublicVector is a PackedModQVector, this is the OCTET STRING representing h obtained using RE2BSP and then BS2OSP as defined in EESS #1 [EESS#1]. All coefficients up to X^(N-1) SHALL be explicitly included in publicKeyVector. Representing the NTRUSign public key as a ModQVector is the preferred method.

ntruSignKeyExternsions is provided for future extensibility. Only one extension is currently defined.

The fields of the type NTRUSignKeyExtension have the following meanings:

keyID can be used to associate a unique key identifier with the key.

If the keyUsage extension is present in an end entity certificate that conveys an NTRUSign public key, any combination of the following values MAY be present:

digitalSignature;
nonRepudiation;

If the keyUsage extension is present in a CA certificate that conveys an NTRUSign public key, any combination of the following values MAY be present:

digitalSignature;
nonRepudiation;
keyCertSign; and
cRLSign.

3. ASN.1 Module

```
PKIXAlgorithmOIDTBD -- {TBD} --

DEFINITIONS EXPLICIT TAGS ::= BEGIN
```

```
-- EXPORTS ALL; --

-- IMPORTS None; --
```

```
   -- Supporting definitions

        AlgorithmIdentifier { ALGORITHM: IOSet } ::= SEQUENCE {
                algorithm       ALGORITHM.&id({IOSet}),
                parameters      ALGORITHM.&Type({IOSet}{@algorithm})
                                OPTIONAL
                }

        ALGORITHM ::= CLASS {
                &id     OBJECT IDENTIFIER  UNIQUE,
                &Type   OPTIONAL
                }
                WITH SYNTAX { OID &id [PARMS &Type] }

        OIDS ::= ALGORITHM

   -- Informational object identifiers

        pkcs-1  OBJECT IDENTIFIER ::=
                {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                1}

        id-mgf1  OBJECT IDENTIFIER ::= {pkcs-1 8}

        id-sha1         OBJECT IDENTIFIER ::=
                {iso(1) identified-organization(3) oiw(14) secsig(3)
                algorithms(2) 26}

        id-sha256       OBJECT IDENTIFIER ::=
                {joint-iso-itu-t(2) country(16) us(840) organization(1)
                gov(101) csor(3) nistalgorithm(4) hashalgs(2) 1}

        id-sha384       OBJECT IDENTIFIER ::=
                {joint-iso-itu-t(2) country(16) us(840) organization(1)
                gov(101) csor(3) nistalgorithm(4) hashalgs(2) 2}

        id-sha512       OBJECT IDENTIFIER ::=
{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
csor(3) nistalgorithm(4) hashalgs(2) 3}

   -- NTRU Object Identifiers

        ntru OBJECT IDENTIFIER ::=
                {iso(1) identified-organization(3) dod(6) internet(1)
                private(4) enterprises(1) ntruCryptosystems (8342) }

        id-eess1  OBJECT IDENTIFIER ::= {ntru eess(1) 1}

        id-eess1-algs  OBJECT IDENTIFIER ::= {id-eess1 1}
        id-eess1-params  OBJECT IDENTIFIER ::= {id-eess1 2}
```

```
     id-eess1-encodingMethods  OBJECT IDENTIFIER ::= {id-eess1 3}

-- OID for NTRUSign Algorithm and Public Key
```

```
     id-ntru-EESS1v1-NTRUSign  OBJECT IDENTIFIER ::=
             {id-eess1-algs 3}

   -- OID for NTRUSign Parameter Set

     id-ees251sp2  OBJECT IDENTIFIER ::= {id-eess1-params 7}

   -- OIDs for NTRUSign Encoding Methods

     id-mrgm-ntrusign-1  OBJECT IDENTIFIER ::=
             {id-eess1-encodingMethods 6}

     id-mrgm-ntrusign-2  OBJECT IDENTIFIER ::=
             {id-eess1-encodingMethods 7}

   -- OID for NTRUEncrypt Algorithm and Public Key

     id-ntru-EESS1v1-SVES  OBJECT IDENTIFIER ::=
          {id-eess1-algs 1}

   -- OIDs for NTRUEncrypt Parameter Sets

     id-ees251ep1  OBJECT IDENTIFIER ::= {id-eess1-params 1}
     id-ees347ep1  OBJECT IDENTIFIER ::= {id-eess1-params 2}
     id-ees503ep1  OBJECT IDENTIFIER ::= {id-eess1-params 3}

   -- OIDs for NTRUEncrypt Encoding Methods

     id-mrgm-ntru-1  OBJECT IDENTIFIER ::=
             {id-eess1-encodingMethods 1}

     id-bvgm-ntru-1  OBJECT IDENTIFIER ::=
             {id-eess1-encodingMethods 2}

     id-bvgm-ntru-2  OBJECT IDENTIFIER ::=
             {id-eess1-encodingMethods 3}

-- General Types

     NTRUPublicVector ::= CHOICE {
             modQVector        [0] IMPLICIT ModQVector,
             packedModQVector  [1] IMPLICIT PackedModQVector,
             ...}

     ModQVector ::= OCTET STRING

     PackedModQVector ::= OCTET STRING

     NTRUGeneralPolynomial ::= SEQUENCE {
             numberOfEntries         INTEGER,
```

```
            modulus                 INTEGER,
            coefficients            GeneralVector
            }
```

```
     GeneralVector ::= OCTET STRING

     SmallModulus ::= CHOICE {
             integerValue    INTEGER,
             polynomialValue NTRUGeneralPolynomial
             }

     Degree ::= INTEGER  (251 | 347 | 503, ...)

     Version ::= INTEGER { v0(0) } (v0, ...)

     NTRUEESS1v1Hashes ALGORITHM ::= {
             {OID id-sha1    PARMS NULL}|
             {OID id-sha256  PARMS NULL }|
             {OID id-sha384  PARMS NULL }|
             {OID id-sha512  PARMS NULL },
             ...}

     NTRUEESS1v1PRNGs ALGORITHM ::= {
             NTRUMGFAlgorithms,
             ...}

     NTRUMGFAlgorithms ALGORITHM ::= {
             {OID id-mgf1 PARMS MGF1Parameters},
             ...}

             MGF1Parameters ::= AlgorithmIdentifier
     {{NTRUEESS1v1Hashes}}

-- Encoding for NTRUSign Signatures

     NTRUSignSignedData ::= NTRUPublicVector

-- Encoding for NTRUSign Public Keys

     NTRUSignPublicKey ::= SEQUENCE {
             publicKeyVector         NTRUPublicVector,  -- h
             ntruSignKeyExtensions   NTRUSignKeyExtensions OPTIONAL
             }

     NTRUSignKeyExtensions ::=
             SEQUENCE SIZE(1..MAX) OF NTRUSignKeyExtension

     NTRUSignKeyExtension ::= CHOICE {
             keyID           [0] IMPLICIT INTEGER,
             ...}

     EESS1v1-NTRUSign-Parameters ::= CHOICE {
             degree                  Degree,
             standardNTRUSignParameters
```

```
                              StandardNTRUSignParameters,
    explicitNTRUSignParameters
                              ExplicitNTRUSignParameters,
    externalParameters        NULL
```

```
        }

    StandardNTRUSignParameters ::= OIDS.&id({NTRUSignParameters})

    NTRUSignParameters  OIDS ::= {
        { OID id-ees251sp2 },
        ...}

    ExplicitNTRUSignParameters ::= SEQUENCE {
        version                 Version,
        degree                  INTEGER,
        bigModulus              INTEGER,
        normBound               INTEGER,
        messageRandLength       INTEGER,
        hash                    NTRUSignHashAlgIdentifier,
        mrgm                    NTRUSignMRGMAlgIdentifier,
        ...}

    NTRUSignHashAlgIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1Hashes}}

    NTRUSignMRGMAlgIdentifier ::=
                AlgorithmIdentifier {{NTRUSignEESS1v1MRGMs}}

    NTRUSignEESS1v1MRGMs ALGORITHM ::= {
        {OID id-mrgm-ntrusign-1 PARMS NTRUSignMRGM1-params}|
        {OID id-mrgm-ntrusign-2 PARMS NTRUSignMRGM2-params},
        ...}

    NTRUSignMRGM1-params ::= NTRUSignPRNGAlgIdentifier

    NTRUSignMRGM2-params ::= SEQUENCE {
        c               INTEGER,
        numGroups       INTEGER,
        numElements     INTEGER,
        prng            NTRUSignPRNGAlgIdentifier
        }

    NTRUSignPRNGAlgIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1PRNGs}}

-- Encoding for NTRUEncrypt Public Keys

    NTRUPublicKey ::= SEQUENCE {
        publicKeyVector         NTRUPublicVector,  -- h
        ntruKeyExtensions       NTRUKeyExtensions OPTIONAL
        }

    NTRUKeyExtensions ::=
        SEQUENCE SIZE(1..MAX) OF NTRUKeyExtension
```

```
     NTRUKeyExtension ::= CHOICE {
             keyID           [0] IMPLICIT INTEGER,
             ...}
```

```
EESS1v1-SVES-Parameters ::= CHOICE {
        degree                  Degree,
        standardNTRUParameters  StandardNTRUParameters,
        explicitNTRUParameters  ExplicitNTRUParameters,
        externalParameters      NULL
        }

StandardNTRUParameters ::= OIDS.&id({NTRUParameters})

NTRUParameters  OIDS ::= {
        { OID id-ees251ep1 }|
        { OID id-ees347ep1 }|
        { OID id-ees503ep1 },
        ...}

ExplicitNTRUParameters ::= SEQUENCE {
        version         Version,
        degree          INTEGER,
        bigModulus      INTEGER,
        smallModulus    SmallModulus,
        mrgm           NTRUMRGMAlgorithmIdentifier,
        db             INTEGER,
        bvgm           NTRUBVGMAlgorithmIdentifier,
        ...}

NTRUMRGMAlgorithmIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1MRGMs}}

NTRUBVGMAlgorithmIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1BVGMs}}

NTRUEESS1v1MRGMs ALGORITHM ::= {
        {OID id-mrgm-ntru-1 PARMS NTRUMRGM1-params},
        ...}

NTRUMRGM1-params ::= NTRUHashAlgorithmIdentifier

NTRUHashAlgorithmIdentifier ::=
                AlgorithmIdentifier {{NTRUEESS1v1Hashes}}

NTRUEESS1v1BVGMs ALGORITHM ::= {
        {OID id-bvgm-ntru-1 PARMS NTRUBVGM1-params}|
        {OID id-bvgm-ntru-2 PARMS NTRUBVGM2-params},
        ...}

NTRUBVGM1-params ::= SEQUENCE {
        c       INTEGER,
        prng    NTRUPRNGAlgorithmIdentifier,
        dr      INTEGER
```

```
            }

     NTRUBVGM2-params ::= SEQUENCE {
            c       INTEGER,
```

```
    prng    NTRUPRNGAlgorithmIdentifier,
    dr1     INTEGER,
    dr2     INTEGER,
    dr3     INTEGER
    }

NTRUPRNGAlgorithmIdentifier ::= AlgorithmIdentifier
                                {{NTRUEESS1v1PRNGs}}

END -- PKIXAlgorithmOIDTBD --
```

4. Security Considerations

This document is entirely concerned with security mechanisms. It is based on the Internet X.509 Public Key Infrastructure Certificate and CRL Profile [RFC 2459], IEEE P1363.1 [P1363.1] and EESS #1 [EESS#1] and the appropriate security considerations from those documents apply.

5. Intellectual Property Rights

NTRU Cryptosystems, Inc. has been granted U.S. Patent No. 6,081,597, which covers aspects of the NTRUEncrypt public-key encryption scheme, and has applied for a patent (or patents) that covers the NTRUSign public-key signature scheme. In addition, NTRU Cryptosystems may have applied for additional patent coverage on implementation techniques related to the use of NTRUEncrypt or NTRUSign. This and any additional patent information will be sent to the IETF.

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights, which may cover technology that may be required to implement the techniques in this document. Please address the information to the IETF Executive Director.

6. Acknowledgements

The authors would like to thank Phil Griffin for his considerable aid in the formulation of the ASN.1 structures for this document.

7. References

[EESS#1] Efficient Embedded Security Standards (EESS) #1: Implementation Aspects of NTRU and NTRUSign, Draft Version 4, March 2002, Consortium for Efficient Embedded Security Standards, Available at http://www.ceesstandards.org.

[FIPS180-1] FIPS PUB 180-1, Secure Hash Standard, Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/National Institute of Standards and Technology, National Technical Information Service, Springfield, Virginia, April 17, 1995 (supersedes FIPS PUB 180). Available at http://www.itl.nist.gov/div897/pubs/fip180-1.htm.

[FIPS180-2] Draft FIPS PUB 180-2, Secure Hash Standard, Federal Information Processing Standards Publication 180-2, U.S. Department of Commerce/National Institute of Standards and Technology, National Technical Information Service, Springfield, Virginia, May 30, 2001 (draft available at http://csrc.nist.gov/encryption/shs/dfips-180-2.pdf)

[FIPS186-2] FIPS PUB 186-2, Digital Signature Standard, Federal Information Processing Standards Publication 186-2, U.S. Department of Commerce/National Institute of Standards and Technology, National Technical Information Service, Springfield, Virginia, 2000. Available at http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf

[IEEE1363] IEEE Std 1363-2000: IEEE Standard Specifications for Public-Key Cryptography, IEEE Computer Society, New York, NY, August 2000, Institute of Electrical and Electronics Engineers

[P1363.1] IEEE Draft Standard P1363.1 D2: IEEE Standard Specifications for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, Draft 2, May 2001, Available at http://grouper.ieee.org/groups/1363.

[PKIX-ALGS] L. Bassham, R. Housley, W. Polk, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and CRL Profile", draft-ietf-pkix-pkalgs-05.txt, October 2001

[RFC2026] S. Bradner, "The Internet Standards Process", IETF RFC 2026, October 1996

[RFC2119] S. Bradner, "Key Words for Use in RFCs to Indicate Requirement Levels", IETF RFC 2119, March 1997

[RFC2459] R. Housley, W. Ford, W. Polk and D. Solo, "Internet X.509

Public Key Infrastructure Certificate and CRL Profile", IETF RFC 2459, January 1999

Authors' Addresses

Ari Singer
NTRU
5 Burlington Woods Phone: 1-781-418-2500
Burlington, MA 01803, USA Email: asinger@ntru.com

William Whyte
NTRU
5 Burlington Woods Phone: 1-781-418-2500
Burlington, MA 01803, USA Email: wwhyte@ntru.com