Limited Additional Mechanisms for PKIX and SMIME
Internet-Draft
Intended status: Standards Track
Expires: 8 May 2026

J. Massimo
P. Kampanakis
AWS
S. Turner
sn3rd
B. E. Westerbaan
Cloudflare

4 November 2025

Internet X.509 Public Key Infrastructure -- Algorithm Identifiers for the Fast-Fourier Transform over NTRU-Lattice-Based Digital Signature Algorithm (FN-DSA)
draft-turner-lamps-fn-dsa-certificates-00

Abstract

Digital signatures are used within X.509 certificates and Certificate Revocation Lists (CRLs), and to sign messages. This document specifies the conventions for using, the forthcoming, FIPS 206, the Fast-Fourier Transform over NTRU-Lattice-Based Digital Signature Algorithm (FN-DSA), in Internet X.509 certificates and CRLs. The conventions for the associated signatures, subject public keys, and private key are also described.

About This Document

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

The latest revision of this draft can be found at https://seanturner.github.io/fn-dsa-certificates/draft-turner-lamps-fn-dsa-certificates.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-turner-lamps-fn-dsa-certificates/.

Discussion of this document takes place on the Limited Additional Mechanisms for PKIX and SMIME 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/seanturner/fn-dsa-certificates.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Copyright Notice

Table of Contents

1. Introduction

The Fast-Fourier Transform over NTRU-Lattice-Based Digital Signature Algorithm (FN-DSA) is a quantum-resistant digital signature scheme standardized by the US National Institute of Standards and Technology (NIST) PQC project [NIST-PQC] in, the forthcoming, [FIPS206]. This document specifies the use of the FN-DSA in Public Key Infrastructure X.509 (PKIX) certificates and Certificate Revocation Lists (CRLs) at two security levels: FN-DSA-512 and FN-DSA-1024.

Prior to standardization, FN-DSA was known as Falcon. FN-DSA and Falcon are not compatible.

[FIPS206] defines two variants of FL-DSA: pure and pre-hash. Only the former is specified in this document. See Section 8 for the rationale. The pure variant of FN-DSA supports the typical pre-hash flow.

1.1. Requirements Language

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

2. Identifiers

The AlgorithmIdentifier type is defined in [RFC5912] as follows:

```
 AlgorithmIdentifier{ALGORITHM-TYPE, ALGORITHM-TYPE:AlgorithmSet} ::=
   SEQUENCE {
     algorithm   ALGORITHM-TYPE.&id({AlgorithmSet}),
     parameters  ALGORITHM-TYPE.
                   &Params({AlgorithmSet}{@algorithm}) OPTIONAL
  }
```

> NOTE: The above syntax is from [RFC5912] and is compatible with the 2021 ASN.1 syntax [X680]. See [RFC5280] for the 1988 ASN.1 syntax.

The fields in AlgorithmIdentifier have the following meanings:

* algorithm identifies the cryptographic algorithm with an object identifier (OID).

* parameters, which are optional, are the associated parameters for the algorithm identifier in the algorithm field.

The NIST-registered OIDs [CSOR] are:

| NOTE: The OIDS, once registered by NIST, will be included
| below.

```
id-fn-dsa-512 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
            country(16) us(840) organization(1) gov(101) csor(3)
            nistAlgorithm(4) sigAlgs(3) id-fn-dsa-512(TBD) }

id-fn-dsa-1024 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
            country(16) us(840) organization(1) gov(101) csor(3)
            nistAlgorithm(4) sigAlgs(3) id-fn-dsa-1024(TBD) }
```

The contents of the parameters component for each algorithm MUST be absent.

3. FN-DSA Signatures in PKIX

FN-DSA is a lattice-based digital signature scheme based on the GPV hash-and-sign framework [GPV08], instantiated over NTRU (N-th Degree Truncated Polynomial Ring Unit) lattices with Fast Fourier sampling techniques [DP16]. The security is based upon the hardness of the underlying FN-DSA is the SIS (Short Integer Solution) problem over NTRU lattices. FN-DSA provides two parameter sets for the NIST PQC security categories 512 and 1024.

Signatures are used in a number of different ASN.1 structures. As shown in the ASN.1 equivalent to that in [RFC5280] below, in an X.509 certificate, a signature is encoded with an algorithm identifier in the signatureAlgorithm attribute and a signatureValue attribute that contains the actual signature.

```
Certificate  ::=  SIGNED{ TBSCertificate }

SIGNED{ToBeSigned} ::= SEQUENCE {
   toBeSigned           ToBeSigned,
   algorithmIdentifier  SEQUENCE {
       algorithm        SIGNATURE-ALGORITHM.
                          &id({SignatureAlgorithms}),
       parameters       SIGNATURE-ALGORITHM.
                          &Params({SignatureAlgorithms}
                            {@algorithmIdentifier.algorithm})
                              OPTIONAL
   },
   signature BIT STRING (CONTAINING SIGNATURE-ALGORITHM.&Value(
                            {SignatureAlgorithms}
                            {@algorithmIdentifier.algorithm}))
}
```

Signatures are also used in the CRL list ASN.1, the representation below is equivalent to that in [RFC5280]. In an X.509 CRL, a signature is encoded with an algorithm identifier in the signatureAlgorithm attribute and a signatureValue attribute that contains the actual signature.

```
CertificateList  ::=  SIGNED{ TBSCertList }
```

The following SIGNATURE-ALGORITHM ASN.1 classes are for FN-DSA-512 and FN-DSA-1024:

```
sa-fn-dsa-512 SIGNATURE-ALGORITHM ::= {
  IDENTIFIER id-fn-dsa-512
  PARAMS ARE absent
  PUBLIC-KEYS { pk-fn-dsa-512 }
  SMIME-CAPS { IDENTIFIED BY id-fn-dsa-512 }
  }

sa-fn-dsa-1024 SIGNATURE-ALGORITHM ::= {
  IDENTIFIER id-fn-dsa-1024
  PARAMS ARE absent
  PUBLIC-KEYS { pk-fn-dsa-1024 }
  SMIME-CAPS { IDENTIFIED BY id-fn-dsa-1024 }
  }
```

| NOTE: The above syntax is from [RFC5912] and is compatible with the 2021 ASN.1 syntax [X680].

The identifiers defined in Section 2 can be used as the AlgorithmIdentifier in the signatureAlgorithm field in the sequence Certificate/CertificateList and the signature field in the sequence

TBSCertificate/TBSCertList in certificates and CRLs, respectively, [RFC5280]. The parameters of these signature algorithms MUST be absent, as explained in Section 2. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, the OID id-fn-dsa-*, where * is 512 or 1024 -- see Section 2.

| TODO: Insert reference for context string (assuming there is one).

The signatureValue field contains the corresponding FN-DSA signature computed upon the ASN.1 DER-encoded TBSCertificate/TBSCertList [RFC5280]. The optional context string (ctx) parameter as defined in Section X of [FIPS206] is left to its default value: the empty string.

Conforming Certification Authority (CA) implementations MUST specify the algorithms explicitly by using the OIDs specified in Section 2 when encoding FN-DSA signatures in certificates and CRLs. Conforming client implementations that process certificates and CRLs using FN-DSA MUST recognize the corresponding OIDs. Encoding rules for FN-DSA signature values are specified in Section 2.

4. FN-DSA Public Keys in PKIX

In the X.509 certificate, the subjectPublicKeyInfo field has the SubjectPublicKeyInfo type, which has the following ASN.1 syntax:

```
SubjectPublicKeyInfo {PUBLIC-KEY: IOSet} ::= SEQUENCE {
    algorithm        AlgorithmIdentifier {PUBLIC-KEY, {IOSet}},
    subjectPublicKey BIT STRING
}
```

| NOTE: The above syntax is from [RFC5912] and is compatible with the 2021 ASN.1 syntax [X680]. See [RFC5280] for the 1988 ASN.1 syntax.

The fields in SubjectPublicKeyInfo have the following meaning:

* algorithm is the algorithm identifier and parameters for the public key (see above).

* subjectPublicKey contains the public key.

| TODO: Include reference to FIPS's section.

Section XX of [FIPS206] defines the raw byte string encoding of an FN-DSA public key. When used in a SubjectPublicKeyInfo type, the subjectPublicKey BIT STRING contains this raw byte string encoding of

the public key. When an FN-DSA public key appears outside of a SubjectPublicKeyInfo type in an environment that uses ASN.1 encoding, it could be encoded as an OCTET STRING by using the FN-DSA-512-PublicKey and FN-DSA-1024-PublicKey types corresponding to the correct key size defined below.

The PUBLIC-KEY ASN.1 types for FN-DSA are defined here:

| TODO: Include key sizes below.

```
pk-fn-dsa-512 PUBLIC-KEY ::= {
  IDENTIFIER id-fn-dsa-512
  -- KEY no ASN.1 wrapping --
  CERT-KEY-USAGE
    { digitalSignature, nonRepudiation, keyCertSign, cRLSign }
  -- PRIVATE-KEY no ASN.1 wrapping; YYYY octets --
}

pk-fn-dsa-87 PUBLIC-KEY ::= {
  IDENTIFIER id-fn-dsa-1024
  -- KEY no ASN.1 wrapping --
  CERT-KEY-USAGE
    { digitalSignature, nonRepudiation, keyCertSign, cRLSign }
  -- PRIVATE-KEY no ASN.1 wrapping; YYYY octets --
}

FN-DSA-512-PublicKey ::= OCTET STRING (SIZE (897))

FN-DSA-1024-PublicKey ::= OCTET STRING (SIZE (1793))

FN-DSA-PrivateKey ::= OCTET STRING (SIZE (32))
```

| NOTE: The above syntax is from [RFC5912] and is compatible with the 2021 ASN.1 syntax [X680].

[RFC5958] specifies the Asymmetric Key Package's OneAsymmetricKey type for encoding asymmetric keypairs. When an FN-DSA private key or keypair is encoded as a OneAsymmetricKey, it follows the description in Section 6.

When the FN-DSA private key appears outside of an Asymmetric Key Package in an environment that uses ASN.1 encoding, it can be encoded using FN-DSA-PrivateKey.

Appendix C contains example FN-DSA public keys encoded using the textual encoding defined in [RFC7468].

5. Key Usage Bits

The intended application for the key is indicated in the keyUsage certificate extension; see Section 4.2.1.3 of [RFC5280]. If the keyUsage extension is present in a certificate that includes id-fn-dsa-* (where * is 512 or 1024 -- see Section 2) in the SubjectPublicKeyInfo, then the subject public key can only be used for verifying digital signatures on certificates or CRLs, or those used in an entity authentication service, a data origin authentication service, an integrity service, and/or a non-repudiation service that protects against the signing entity falsely denying some action. This means that the keyUsage extension MUST have at least one of the following bits set:

* digitalSignature
* nonRepudiation
* keyCertSign
* cRLSign

FN-DSA subject public keys cannot be used to establish keys or encrypt data, so the keyUsage extension MUST NOT have any of following bits set:

* keyEncipherment
* dataEncipherment
* keyAgreement
* encipherOnly
* decipherOnly

Requirements about the keyUsage extension bits defined in [RFC5280] still apply.

6. Private Key Format

> | NOTE: Hope the following is true!

[FIPS206] specifies an FN-DSA private key as a 32-octet seed (両) (GREEK SMALL LETTER XI, U+03BE).

"Asymmetric Key Packages" [RFC5958] specifies how to encode a private key in a structure that both identifies what algorithm the private key is for and allows for the public key and additional attributes about the key to be included as well. For illustration, the ASN.1 structure OneAsymmetricKey is replicated below.

```
  OneAsymmetricKey ::= SEQUENCE {
    version                   Version,
    privateKeyAlgorithm       SEQUENCE {
    algorithm                 PUBLIC-KEY.&id({PublicKeySet}),
    parameters                PUBLIC-KEY.&Params({PublicKeySet}
                                {@privateKeyAlgorithm.algorithm})
                                   OPTIONAL}
    privateKey                OCTET STRING (CONTAINING
                                PUBLIC-KEY.&PrivateKey({PublicKeySet}
                                  {@privateKeyAlgorithm.algorithm})),
    attributes           [0] Attributes OPTIONAL,
    ...,
    [[2: publicKey       [1] BIT STRING (CONTAINING
                                PUBLIC-KEY.&Params({PublicKeySet}
                                  {@privateKeyAlgorithm.algorithm})
                                  OPTIONAL ]],
    ...
  }
```

> NOTE: The above syntax is from [RFC5958] and is compatible with the 2021 ASN.1 syntax [X680].

For FN-DSA private keys, the privateKey field in OneAsymmetricKey contains raw octet string encoding of the 32-octet seed.

Appendix C contains example FN-DSA private keys encoded using the textual encoding defined in [RFC7468].

7. IANA Considerations

For the ASN.1 module in Appendix A, IANA [is requested/has assigned] the following object identifier (OID) in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0):

Decimal	Description	Reference
TBD	id-mod-x509-fn-dsa-2026	This RFC

Table 1: Object Identifier Assignments

8. Operational Considerations

8.1. Rationale for Disallowing HashFN-DSA

| TODO: Get section reference for HashFN-DSA.

The HashFN-DSA mode defined in Section X.X of [FIPS206] MUST NOT be used; in other words, public keys identified by id-hash-fn-dsa-512-with-sha512 and id-hash-fn-dsa-1024-with-sha512 MUST NOT be in X.509 certificates used for CRLs, OCSP, certificate issuance, and related PKIX protocols. This restriction is primarily to increase interoperability.

FN-DSA and HashFN-DSA are incompatible algorithms that require different Verify() routines. This introduces the complexity of informing the verifier whether to use FN-DSA.Verify() or HashFN-DSA.Verify(). Additionally, since the same OIDs are used to identify the FN-DSA public keys and FN-DSA signature algorithms, an implementation would need to commit a given public key to be either of type FN-DSA or HashFN-DSA at the time of certificate creation. This is anticipated to cause operational issues in contexts where the operator does not know whether the key will need to produce pure or pre-hashed signatures at key-generation time.

9. Security Considerations

| TODO: Most copied from RFC 9881. Dropped the bit about
| Gaussian sampling. Also, FN-DSA is only going to support
| randomized sigs, I figured we could use the text about why they
| didn’t pick deterministic to introduce floating-point issues.

The Security Considerations section of [RFC5280] applies to this specification as well.

| TODO: Verify EUF-CMA. Get #s for chosen messages

The FN-DSA signature scheme is unforgeable under chosen message attacks (EUF-CMA). For the purpose of estimating security strength, it has been assumed that the attacker has access to signatures for no more than 2^{XX} chosen messages.

| TODO: Get section reference.

FN-DSA depends on high quality random numbers that are suitable for use in cryptography. The use of inadequate pseudo-random number generators (PRNGs) to generate such values can significantly undermine various security properties. For instance, using an inadequate PRNG for key generation, might allow an attacker to

efficiently recover the private key by trying a small set of possibilities, rather than brute force search the whole keyspace. The generation of random numbers of a sufficient level of quality for use in cryptography is difficult; see Section X.X.X of [FIPS206] for some additional information.

In the design of FN-DSA, care has been taken to make side-channel resilience easier to achieve. Implementations must still take great care not to leak information via various side channels. While deliberate design decisions such as these can help to deliver a greater ease of secure implementation - particularly against side-channel attacks - it does not necessarily provide resistance to more powerful attacks such as differential power analysis. Some amount of side-channel leakage has been demonstrated in parts of the signing algorithm (specifically the bit-unpacking function), from which a demonstration of key recovery has been made over a large sample of signatures. Masking countermeasures exist for FN-DSA, but come with a performance overhead.

| TODO: Expand the following to also talk about floating point implementation challenges.

FN-DSA only offers randomized signing. Deterministic signing could be dangerous as mistakes in floating-point implementation could cause different signatures for same hash.

| TODO: Get reference.

A security property also associated with digital signatures is non-repudiation. Non-repudiation refers to the assurance that the owner of a signature key pair that was capable of generating an existing signature corresponding to certain data cannot convincingly deny having signed the data, unless its private key was compromised. The digital signature scheme FN-DSA possess three security properties beyond unforgeability, that are associated with non-repudiation. These are exclusive ownership, message-bound signatures, and non-resignability. These properties are based tightly on the assumed collision resistance of the hash function used (in this case SHAKE-256). A full discussion of these properties in FN-DSA can be found at XXXX.

10. References

10.1. Normative References

[CSOR] NIST, "Computer Security Objects Register", 20 August 2024, <https://csrc.nist.gov/projects/computer-security-objects-register/algorithm-registration>.

[FIPS206] "Fast Fourier Transform over NTRU-Lattice-Based Digital Signature Algorithm", n.d., <https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>.

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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[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/rfc/rfc5912>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/rfc/rfc5958>.

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

[X680] ITU-T, "Information Technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680>.

[X690] ITU-T, "Information Technology -- ASN.1: ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690>.

10.2. Informative References

[DP16] Ducas, L. and T. Prest, "Fast Fourier Orthogonalization", Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation (ISSAC ’16), pp. 191198 , 2016, <https://doi.org/10.1145/2930889.2930923>.

[GPV08] Gentry, C., Peikert, C., and V. Vaikuntanathan, "Trapdoors for Hard Lattices and New Cryptographic Constructions", Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08), pp. 197206 , 2008, <https://doi.org/10.1145/1374376.1374407>.

[NIST-PQC] National Institute of Standards and Technology (NIST), "Post-Quantum Cryptography Project", 20 December 2016, <https://csrc.nist.gov/Projects/post-quantum-cryptography>.

[RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", RFC 3647, DOI 10.17487/RFC3647, November 2003, <https://www.rfc-editor.org/rfc/rfc3647>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/rfc/rfc7468>.

Appendix A. ASN.1 Module

This appendix includes the ASN.1 module [X680] for the FN-DSA. Note that as per [RFC5280], certificates use the Distinguished Encoding Rules; see [X690]. This module imports objects from [RFC5912].

```
<CODE BEGINS>
X509-FN-DSA-2026
{ iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-x509-fn-dsa-2026(TBD1) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS

 PUBLIC-KEY, SIGNATURE-ALGORITHM
   FROM AlgorithmInformation-2009  -- [RFC 5912]
     { iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) pkix(7) id-mod(0)
       id-mod-algorithmInformation-02(58) } ;

--
-- FN-DSA Identifiers
--
```

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

   sigAlgs OBJECT IDENTIFIER ::= { nistAlgorithms 3 }

   id-fn-dsa-512 OBJECT IDENTIFIER ::= { sigAlgs XX }

   id-fn-dsa-1024 OBJECT IDENTIFIER ::= { sigAlgs XX }

   --
   -- Public Key Algorithms
   --

   PublicKeys PUBLIC-KEY ::= {
     -- This expands PublicKeys from [RFC 5912]
     pk-fn-dsa-512 |
     pk-fn-dsa-1024,
     ...
   }

   --
   -- FN-DSA Public Keys
   --

   pk-fn-dsa-512 PUBLIC-KEY ::= {
     IDENTIFIER id-fn-dsa-512
     -- KEY no ASN.1 wrapping; XXXX octets --
     PARAMS ARE absent
     CERT-KEY-USAGE { digitalSignature, nonRepudiation,
                      keyCertSign, cRLSign }
     -- PRIVATE-KEY no ASN.1 wrapping; YYYY octets --
     }

   pk-fn-dsa-1024 PUBLIC-KEY ::= {
     IDENTIFIER id-fn-dsa-1024
     -- KEY no ASN.1 wrapping; XXXX octets --
     PARAMS ARE absent
     CERT-KEY-USAGE { digitalSignature, nonRepudiation,
                      keyCertSign, cRLSign }
     -- PRIVATE-KEY no ASN.1 wrapping; YYYY octets --
     }

   FN-DSA-512-PublicKey ::= OCTET STRING (SIZE (897))

   FN-DSA-1024-PublicKey ::= OCTET STRING (SIZE (1793))

   FN-DSA-PrivateKey ::= OCTET STRING (SIZE (32))
```

```
   --
   -- Signature Algorithms
   --

   SignatureAlgorithms SIGNATURE-ALGORITHM ::= {
     -- This expands SignatureAlgorithms from [RFC 5912]
     sa-fn-dsa-512 |
     sa-fn-dsa-1024,
     ... }

   --
   -- ML-DSA Signature Algorithm Identifiers
   --

   sa-fn-dsa-512 SIGNATURE-ALGORITHM ::= {
     IDENTIFIER id-fn-dsa-512
     PARAMS ARE absent
     PUBLIC-KEYS { pk-fn-dsa-512 }
     SMIME-CAPS { IDENTIFIED BY id-fn-dsa-512 }
     }

   sa-fn-dsa-1024 SIGNATURE-ALGORITHM ::= {
     IDENTIFIER id-fn-dsa-1024
     PARAMS ARE absent
     PUBLIC-KEYS { pk-fn-dsa-1024 }
     SMIME-CAPS { IDENTIFIED BY id-fn-dsa-1024 }
     }

   END
   <CODE ENDS>
```

Appendix B. Security Strengths

| TODO

Appendix C. Examples

This appendix contains examples of FN-DSA private keys, public keys, certificates, and inconsistent seed and expanded private keys.

C.1. Example Private Keys

The following examples show FN-DSA private keys in different formats, all derived from the same seed 000102...1e1f. For each security level, we show the seed-only format (using a context-specific [0] primitive tag with an implicit encoding of OCTET STRING), the expanded format, and both formats together.

NOTE: All examples use the same seed value, showing how the same seed produces different expanded private keys for each security level.

C.1.1. FN-DSA-512 Private Key Examples

Each of the examples includes the textual encoding [RFC7468] followed by the so-called "pretty print"; the private keys are the same.

| TODO

C.1.2. FN-DSA-1024 Private Key Examples

Each of the examples includes the textual encoding [RFC7468] followed by the so-called "pretty print"; the private keys are the same.

| TODO

C.2. Example Public Keys

The following is the FN-DSA-512 public key corresponding to the private key in the previous section. The textual encoding [RFC7468] is followed by the so-called "pretty print"; the public keys are the same.

| TODO

The following is the FN-DSA-1024 public key corresponding to the private key in the previous section. The textual encoding [RFC7468] is followed by the so-called "pretty print"; the public keys are the same.

| TODO

C.3. Example Certificates

| NOTE: The example certificates in this section have key usage
| bits set to digitalSignature, keyCertSign, and cRLSign to
| lessen the number of examples, i.e., brevity. Certificate
| Policies (CPs) [RFC3647] for production CAs should consider
| whether this combination is appropriate.

The following is a self-signed certificate for the FN-DSA-512 public key in the previous section. The textual encoding [RFC7468] is followed by the so-called "pretty print"; the certificates are the same.

| TODO

The following is a self-signed certificate for the FN-DSA-1024 public key in the previous section. The textual encoding [RFC7468] is followed by the so-called "pretty print"; the certificates are the same.

| TODO

Acknowledgments

| TODO

Authors' Addresses

Jake Massimo
AWS
United States of America
Email: jakemas@amazon.com

Panos Kampanakis
AWS
United States of America
Email: kpanos@amazon.com

Sean Turner
sn3rd
Email: sean@sn3rd.com

Bas Westerbaan
Cloudflare
Email: bas@cloudflare.com