Workgroup: Internet Engineering Task Force
Individual Draft
draft-vredendaal-dilithium-00
Published: 23 October 2022
Intended Status: Informational
Expires: 26 April 2023
Authors: C. V. Vredendaal
NXP Semiconductors
S. Dragone
IBM Research GmbH
B. Hess
IBM Research GmbH
T. Visegrady
IBM Research GmbH
M. Osborne
IBM Research GmbH
D. Bong
Utimaco IS GmbH
J. Bos
NXP Semiconductors

Quantum Safe Cryptography Key Information for CRYSTALS-Dilithium

Abstract

This proposal defines key management approaches for the Quantum Safe Cryptographic (QSC) algorithm CRYSTALS-Dilithium which has been selected for standardization by the NIST Post Quantum Cryptography (PQC) process. This includes key identification, key serialization, and key compression. The purpose is to provide guidance such that the adoption of quantum safe algorithms is not hampered with the fragmented evolution of necessary key management standards. Early definition of key material standards will help expedite the adoption of new quantum safe algorithms and at the same time as improving interoperability between implementations and minimizing divergence across standards.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 26 April 2023.

Table of Contents

1. Introduction

QSC algorithms being standardized in the NIST PQC Process have evolved through several rounds and iterations. Keys are neither easily identifiable nor compatible across rounds. It is also expected that algorithms will evolve after final candidates have been selected. The lack of binary compatibility between algorithm versions and variants means that it is important to clearly identify key material. Parallel to the NIST process, industry is evaluating the impact of adopting new PQC algorithms, in particular key management. Here it is important to define and standardize key serialization and encoding formats. Finally, we have seen that many platforms and protocols are very constrained when it comes to the amount of memory or space available for key objects. This makes it important to define and standardize key compression formats. This proposal addresses aspects of key identification, key serialization, and key compression for the future primary NIST PQC Digital

Signature standard, CRYSTALS-Dilithium. For the other schemes, see draft-uni-qsckeys-kyber, draft-uni-qsckeys-falcon, draft-uni-qsckeys-sphincsplus and the previous Internet-Draft [draft-uni-qsckeys-01]. This proposal will be updated when the final NIST standard for CRYSTALS-Dilithium becomes available.

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] .

1.2. Algorithm Identification

Algorithm identification is important for several reasons:

- Managing a smooth transition from early adoption algorithm versions to production versions where there is no compatibility.
- Supporting different algorithm versions from different NIST rounds
- Identifying different key serialization strategies
- Identifying compressed and uncompressed keys

The current standardization of quantum safe algorithms does not address the definition of serialization structures for keys. As a result, it has become commonplace for the cryptographic community working on and with these algorithms to define their own approaches. This leads to proprietary and internal representations for key material. This has certain advantages in terms of ease of experimentation while focusing on finding the best-performing QSC algorithms. In terms of longer-term support where algorithm versions change this is a problem. This proposal defines in section 2 a long-term structured key representation format useful to address the goals outlined above.

1.3. Algorithm and Algorithm Parameter Object Identifier

Algorithm and algorithm parameter information shall have ASN.1 type AlgorithmIdentifier as given in [RFC5280] and shall be extended by an pqcAlgorithmParameterName type in the optional parameters field:

```
AlgorithmIdentifier ::=  SEQUENCE {
    algorithm  OBJECT IDENTIFIER, - OID: algorithm and algo parameter
    parameters pqcAlgorithmParameterName OPTIONAL
}
pqcAlgorithmParameterName ::= PrintableString
```

2. Overview of CRYSTALS-Dilithium and parameter OIDs

CRYSTALS-Dilithium consists of six parameter sets. This memo attributes a name and a placeholder for an OID to the different parameter sets of CRYSTALS-Dilithium. The following table gives an

overview of the possible OIDs in the algorithm field and possible parameters set names in the parameters field of the AlgorithmIdentifier type. Each name or OID represents a single parameter set of given security. Details can be found in the next section.

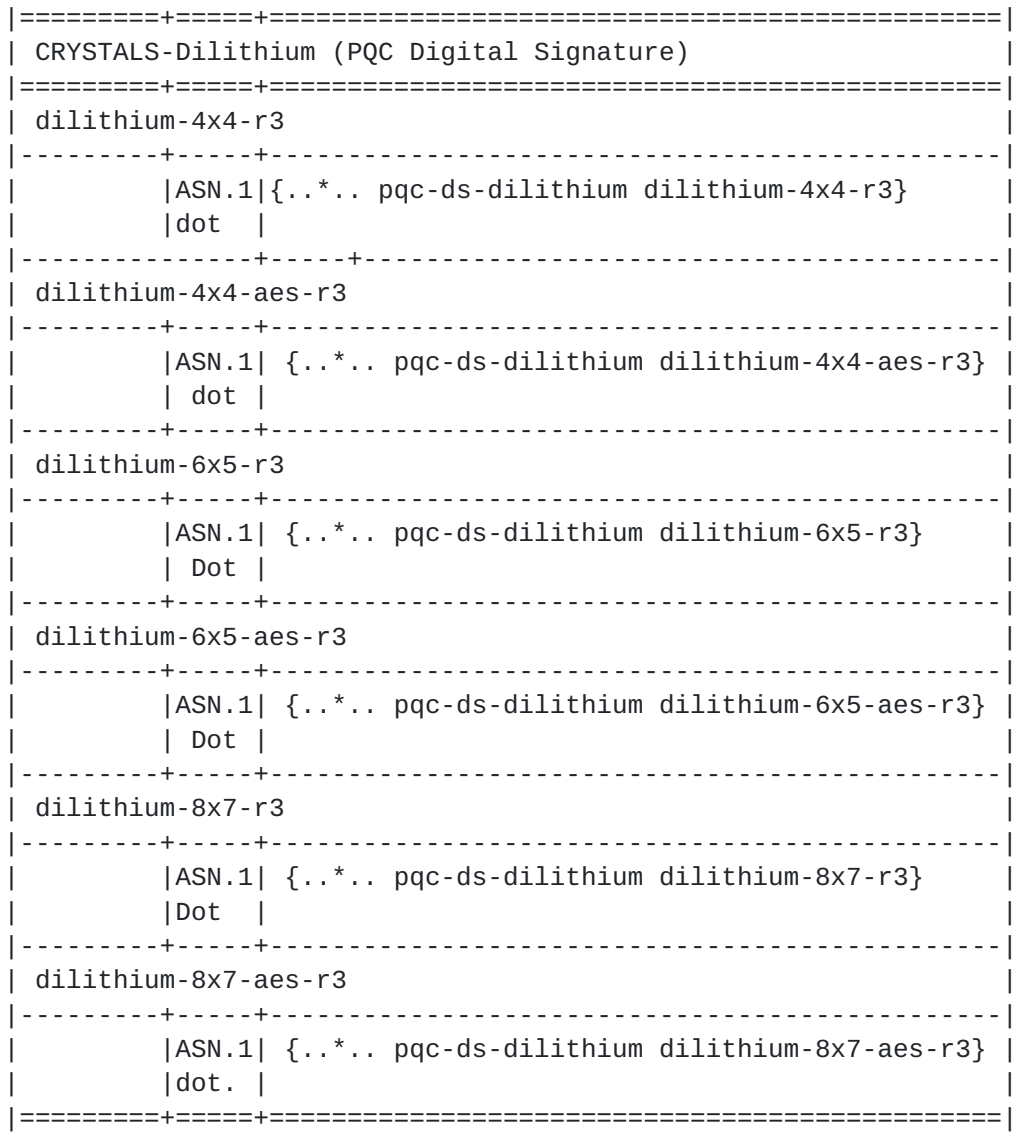

Figure 1

2.1. Key Formats

The private key format defined is from PKCS#8 [RFC5208] . PKCS#8 PrivateKeyInfo is defined as:

```
PrivateKeyInfo ::=  SEQUENCE {
    version               INTEGER             -- PKCS#8 syntax ver
    privateKeyAlgorithm   AlgorithmIdentifier -- see chapter above
    privateKey            OCTET STRING,       -- see chapter below
    attributes            [0]  IMPLICIT Attributes OPTIONAL
}
```

Distributing a PQC private key requires a PKCS#8 PrivateKeyInfo with a joined PQC algorithm and algorithm parameter OID in the algorithm field of AlgorithmIdentifier and a PQC algorithm specific private key object in the privateKey field of PrivateKeyInfo. Both objects are defined in the specific algorithm sections of this document. For an overview see tables above and below.

2.2. Public Key Format based on *[RFC5280]*

RFC5280 subjectPublicKeyInfo is defined in as:

```
SubjectPublicKeyInfo := SEQUENCE {
    algorithm          AlgorithmIdentifier  -- see chapter above
    subjectPublicKey   BIT STRING           -- see chapter below
}
```

Distributing a PQC public key requires a [RFC5480] subjectPublicKeyInfo with a joined PQC algorithm and algorithm parameter OID in the algorithm field of AlgorithmIdentifier and a PQC algorithm specific public key object in the subjectPublicKey field of subjectPublicKeyInfo. Both objects are defined in the specific algorithm sections of this document. For an overview see tables above and below.

2.3. Overview of Memo Definitions - PQC Key Formats

The privateKey field in the PrivateKeyInfo type [RFC5480] is an OCTET STRING whose contents are the value of the private key. The interpretation of the content differs from PQC algorithm to algorithm. The subjectPublicKey field in the subjectPublicKeyInfo type [RFC5480] is a BIT STRING whose contents are the value of the public key. Here also the interpretation of the content differs from PQC algorithm to algorithm.

3. CRYSTALS-Dilithium

CRYSTALS-Dilithium is a digital signature scheme that is based on the hardness of lattice problems over module lattices.

*Project Website: https://pq-crystals.org/dilithium/index.shtml
*NIST Round 3 Submission (version 3.1): https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/

submissions/Dilithium-Round3.zip, https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

3.1. Algorithm Parameter Identifiers

CRYSTALS-Dilithium uses OIDs to identify parameters sets.

dilithium-4x4-r3	
Parameter OID	{..*.. dilithium-4x4-r3} <.>
NIST Level Security	Level 2
Parameters	Polynomial Ring Zq[x]/(x^n+1) Dimension/Degree n=256 Modulus q=8380417 Dropped bits from t: d=13 # of +-1's in c: tau=39 challenge entropy=192 gamma coefficient range: gamma1=2^17 low-order rounding range: gamma2=(q- 1)/88 Private key Range eta=2 Dimensions of A: (k,l)=(4,4) Max # of 1's in the hint h: w=80 Repetitions=4.25

dilithium-4x4-aes-r3	
Parameter OID	{..*.. dilithium-4x4-aes-r3} <.>
NIST Level Security	Level 2
Parameters	Polynomial Ring Zq[x]/(x^n + 1) Dimension/Degree n=256 Modulus q=8380417 Dropped bits from t: d=13 # of +-1's in c: tau=39 challenge entropy=192 y coefficient range: gamma1=2^17 low-order rounding range:gamma2=(q- -1)/88 Private key Range eta=2 Dimensions of A: (k,l)=(4,4) Max # of 1's in the hint h: w=80 Repetitions=4.25

dilithium-6x5-r3	
Parameter OID	{..*.. dilithium-6x5-r3} <.>
NIST Level Security	Level 3
Parameters	Polynomial Ring Zq[x]/(x^n + 1) Dimension/Degree n=256 Modulus q=8380417 Dropped bits from t: d=13 # of +-1's in c: tau=49

	challenge entropy=225
	y coefficient range: gamma1=2^19
	low-order rounding range:gamma2=(q-
	-1)/32
	Private key Range eta=4
	Dimensions of A: (k,l)=(6,5)
	Max # of 1's in the hint h: w=55
	Repetitions=5.1
dilithium-6x5-aes-r3	
Parameter OID	{..*.. dilithium-6x5-aes-r3}
	<.>
NIST Level Security	Level 3
Parameters	Polynomial Ring Zq[x]/(x^n +1)
	Dimension/Degree n=256
	Modulus q=8380417
	Dropped bits from t: d=13
	# of +-1's in c: tau=49
	challenge entropy=225
	y coefficient range: gamma1=2^19
	low-order rounding range:gamma2=(q-
	-1)/32
	Private key Range eta=4
	Dimensions of A: (k,l)=(6,5)
	Max # of 1's in the hint h: w=55
	Repetitions=5.1
dilithium-8x7-r3	
Parameter OID	{..*.. dilithium-8x7-r3}
	<.>
NIST Level Security	Level 5
Parameters	Polynomial Ring Zq[x]/(x^n + 1)
	Dimension/Degree n=256
	Modulus q=8380417
	Dropped bits from t: d=13
	# of +-1's in c: tau=60
	challenge entropy=257
	y coefficient range: gamma1=2^19
	low-order rounding range:gamma2=(q-
	-1)/32
	Private key Range eta=2
	Dimensions of A: (k,l)=(8,7)
	Max # of 1's in the hint h: w=75
	Repetitions=3.85
dilithium-8x7-aes-r3	
Parameter OID	{..*.. dilithium-8x7-aes-r3}
	<.>

NIST Level Security	Level 5
Parameters	Polynomial Ring Zq[x]/(x^n + 1)
	Dimension/Degree n=256
	Modulus q=8380417
	Dropped bits from t: d=13
	# of +-1's in c: tau=60
	challenge entropy=257
	y coefficient range: gamma1=2^19
	low-order rounding range:gamma2=(q--1)/32
	Private key Range eta=2
	Dimensions of A: (k,l)=(8,7)
	Max # of 1's in the hint h: w=75
	Repetitions=3.85

Figure 2

The AES variants listed above differ from the other variants in that they use AES, rather than SHAKE internally to expand the key parameters from an initial seed. While the parameters listed in the table are the same, the key-pairs will not be compatible with the 'aes' variants.

3.2. Key Details

Public key. The public-key consists of two parameters:

- *rho: nonce
- *t1: a vector encoded in 320*k bytes

The size necessary to hold all public key elements accounts to 32+320*k bytes.

Private key. The private key consists of 6 parameters:

- *rho: nonce
- *K: a key/seed/D
- *tr: PRF bytes
- *s1: vector (L)
- *s2: vector (K)
- *t0: k polynomials

If the private key is fully populated, it consists of 6 parameters. The size necessary to hold all private key elements accounts to 32+32+32+32*[(k+l)*ceiling(log(2*eta+1))+13*k] bytes. The resulting public key and private key sizes can be found in the table below.

Algorithm	Public Key Length	Private Key SK Length	Partial SK (V1) Length	Partial SK (V2) Length
dilithium-4x4-r3	1312	2528	64	32
dilithium-4x4-aes-r3	1312	2528	64	32
dilithium-6x5-r3	1952	4000	64	32
dilithium-6x5-aes-r3	1952	4000	64	32
dilithium-8x7-r3	2592	4864	64	32
dilithium-8x7-aes-r3	2592	4864	64	32

Figure 3

3.3. Private Key Full Encoding

Encoding a CRYSTALS-Dilithium private key with PKCS#8 must include the following two fields:

- *dilithium-(kxl)-r3 in the algorithm field of AlgorithmIdentifier
- *DilithiumPrivateKey in the privateKey field, which is an OCTET STRING.

CRYSTALS-Dilithium public keys are optionally distributed in the PublicKey field of the PrivateKeyInfo structure.

ASN.1 Encoding for a CRYSTALS-Dilithium private key when fully populated:

```
DilithiumPrivateKey ::= SEQUENCE {
    version     INTEGER {v0(0)}     -- version (round 3)
    nonce       BIT STRING,         -- rho
    key         BIT STRING,         -- key/seed/D
    tr          BIT STRING,         -- PRF bytes (CRH in spec)
    s1          BIT STRING,         -- vector(L)
    s2          BIT STRING,         -- vector(K)
    t0          BIT STRING,
    publicKey  [0] IMPLICIT DilithiumPublicKey OPTIONAL
                                    -- see next section
}
```

3.4. Private Key Partial Encoding Option 1

In option 1 ofCRYSTALS-Dilithium partial encoding the rho (nonce) and the seed (key) are used to regenerate the full key. Note: There are a number of alternative ways to encode a partially filled structure that include defining fields as optional and defining fields as 'EMPTY'. As an example partial RSA keys are encoded using EMPTY fields. It can be argued that defining fields as EMPTY significantly simplifies the implementation of parsing ASN.1 frames. The ASN.1 format for the partially populated versions is the same as for the fully populated version. The ASN.1 encoding for the first variant (rho and seed) is defined as follows:

```
DilithiumPrivateKey ::= SEQUENCE {
    version     INTEGER {v0(0)}     -- version (round 3)
    nonce       BIT STRING,         -- rho
    key         BIT STRING,         -- key/seed/D
    tr          BIT STRING,         -- EMPTY
    s1          BIT STRING,         -- EMPTY
    s2          BIT STRING,         -- EMPTY
    t0          BIT STRING,         -- EMPTY
    publicKey   [0] IMPLICIT DilithiumPublicKey OPTIONAL
                                    -- see next section
}
```

3.5. Private key Partial Encoding Option 2

In option 2 of CRYSTALS-Dilithium partial encoding only zeta (nonce) is used to regenerate the full key. The ASN.1 encoding for this is defined as follows:

```
DilithiumPrivateKey ::= SEQUENCE {
    version     INTEGER {v0(0)}     -- version (round 3)
    nonce       BIT STRING,         -- zeta
    key         BIT STRING,         -- EMPTY
    tr          BIT STRING,         -- EMPTY
    s1          BIT STRING,         -- EMPTY
    s2          BIT STRING,         -- EMPTY
    t0          BIT STRING,         -- EMPTY
    publicKey   [0] IMPLICIT DilithiumPublicKey OPTIONAL
                                  -- see next section
}
```

3.6. Public Key Full Encoding

Components are individual OCTET STRINGs, without unused bits, encoded with the exact size. There is no removal of leading zeroes.

```
DilithiumPublicKey ::= SEQUENCE {
    rho         OCTET STRING,
    t1          OCTET STRING
}
```

4. Acknowledgements

This template was derived from an initial version written by Pekka Savola and contributed by him to the xml2rfc project.

This document is part of a plan to make xml2rfc indispensable.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

Any processing of the ASN.1 private key structures, such as base64 en/decoding shall be performed in "constant-time", meaning without secret-dependent control flow and table lookups. The ASN.1 structures in this document are defined with fixed tag-lengths. The purpose is to prevent side-channel leakage of variable lengths during DER parsing. Any DER parsing of the private key ASN.1 key structures shall be performed with these fixed lengths.

7. References

7.1. 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>.

[RFC5208] Kaliski, B., "Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2", BCP 14, RFC 5208, DOI 10.17487/RFC5208, May 2008, <hhttps://www.rfc-editor.org/info/rfc5208>.

[RFC5280] Cooper, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", BCP 14, RFC RFC5280, DOI 10.17487/RFC5280, May 2008, <hhttps://www.rfc-editor.org/info/rfcRFC5280>.

[RFC5480] Turner, S., "Elliptic Curve Cryptography Subject Public Key Information", BCP 14, RFC RFC5480, DOI 10.17487/RFC5480, May 2009, <hhttps://www.rfc-editor.org/info/rfc5480>.

7.2. Informative References

[draft-uni-qsckeys-01] Vredendaal, C. V., Dragone, S., Hess, B., Visegrady, T., Osborne, M., Bong, D., and J. Bos, "Quantum Safe Cryptography Key Information", Work in Progress, Internet-Draft, draft-uni-qsckeys-01, 12 May 2022, <https://www.ietf.org/archive/id/draft-uni-qsckeys-01.txt>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <https://www.rfc-editor.org/info/rfc2629>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI

10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Christine van Vredendaal
NXP Semiconductors
High Tech Campus 60
5656 AE Eindhoven
Netherlands

Email: cvvrede@gmail.com

Silvio Dragone
IBM Research GmbH
Saeumerstrasse 4
CH-8803 Rueschlikon
Switzerland

Email: sid@zurich.ibm.com

Basil Hess
IBM Research GmbH
Saeumerstrasse 4
CH-8803 Rueschlikon
Switzerland

Email: bhe@zurich.ibm.com

Tamas Visegrady
IBM Research GmbH
Saeumerstrasse 4
CH-8803 Rueschlikon
Switzerland

Email: tvi@zurich.ibm.com

Michael Osborne
IBM Research GmbH
Saeumerstrasse 4
CH-8803 Rueschlikon
Switzerland

Email: osb@zurich.ibm.com

Dieter Bong
Utimaco IS GmbH
Germanusstrasse 4
52080 Aachen
Germany

Email: dieter.bong@utimaco.com

Joppe Bos
NXP Semiconductors
High Tech Campus 60
5656 AE Eindhoven
Netherlands

Email: joppe.bos@nxp.com