Network Working Group
Internet-Draft
Intended status: Informational
Expires: 8 January 2026

Q. Dang
NIST
S. Ehlen
S. Kousidis
BSI
J. Roth
F. Strenzke
MTG AG

7 July 2025

PQ/T Composite Schemes for OpenPGP using NIST and Brainpool Elliptic Curve Domain Parameters
draft-ehlen-openpgp-nist-bp-comp-02

Abstract

This document defines PQ/T composite schemes based on ML-KEM and ML-DSA combined with ECDH and ECDSA algorithms using the NIST and Brainpool domain parameters for the OpenPGP protocol.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ehlen-openpgp-nist-bp-comp/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:openpgp@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/openpgp/. Subscribe at https://www.ietf.org/mailman/listinfo/openpgp/.

Source for this draft and an issue tracker can be found at https://github.com/openpgp-pqc/draft-ehlen-openpgp-nist-bp-comp.

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/.

Copyright Notice

Table of Contents

1. Introduction

This document defines PQ/T composite schemes based on ML-KEM and ML-DSA combined with ECDH and ECDSA using the NIST and Brainpool domain parameters for the OpenPGP protocol. It is an extension of [I-D.draft-ietf-openpgp-pqc], which introduces post-quantum cryptography in OpenPGP using hybrid KEMs and digital signatures combining ML-KEM and ML-DSA with ECC algorithms based on the Edwards Curves defined in [RFC7748] and [RFC8032].

Due to their long-standing and wide deployment, there are well-tested, secure, and efficient implementations of ECDSA and ECDH with NIST-curves [SP800-186]. The same applies to Brainpool curves [RFC5639] which are recommended or required in certain regulatory domains, for instance in Germany [TR-03111]. The purpose of this document is to support users who would like to or have to use such hybrid KEMs and/or signatures with OpenPGP.

1.1. Conventions used in this Document

1.1.1. Terminology for Multi-Algorithm Schemes

The terminology in this document is oriented towards the definitions in [RFC9794]. Specifically, the terms "multi-algorithm", "composite" and "non-composite" are used in correspondence with the definitions therein. The abbreviation "PQ" is used for post-quantum schemes. To denote the combination of post-quantum and traditional schemes, the abbreviation "PQ/T" is used. The short form "PQ(/T)" stands for PQ or PQ/T.

1.2. Post-Quantum Cryptography

This section describes the individual post-quantum cryptographic schemes. All schemes listed here are believed to provide security in the presence of a cryptographically relevant quantum computer.

1.2.1. ML-KEM

ML-KEM [FIPS-203] is based on the hardness of solving the Learning with Errors problem in module lattices (MLWE). The scheme is believed to provide security against cryptanalytic attacks by classical as well as quantum computers. This specification defines ML-KEM only in composite combination with ECDH encryption schemes in order to provide a pre-quantum security fallback.

1.2.2. ML-DSA

ML-DSA [FIPS-204] is a signature scheme that, like ML-KEM, is based on the hardness of solving the Learning With Errors problem and a variant of the Short Integer Solution problem in module lattices (MLWE and SelfTargetMSIS). Accordingly, this specification only defines ML-DSA in composite combinaton with ECDSA signature schemes.

1.3. Elliptic Curve Cryptography

The ECDH encryption is defined here as a KEM.

All elliptic curves for the use in the composite combinations are taken from [RFC9580].

For interoperability this extension offers ML-* in composite combinations with the NIST curves P-256, P-384 defined in [SP800-186] and the Brainpool curves brainpoolP256r1, brainpoolP384r1 defined in [RFC5639].

1.4. Applicable Specifications for the use of PQC Algorithms in OpenPGP

This document is to be understood as an extension of [I-D.draft-ietf-openpgp-pqc], which introduced PQC in OpenPGP, in that it defines further algorithm code points. All general specifications in [I-D.draft-ietf-openpgp-pqc] that pertain to the ML-KEM and ML-DSA composite schemes or generally cryptographic schemes defined therein equally apply to the schemes specified in this document.

2. Preliminaries

This section provides some preliminaries for the definitions in the subsequent sections.

2.1. Elliptic curves

2.1.1. SEC1 EC Point Wire Format

Elliptic curve points of the generic prime curves are encoded using the SEC1 (uncompressed) format as the following octet string:

B = 04 || X || Y

where X and Y are coordinates of the elliptic curve point P = (X, Y), and each coordinate is encoded in the big-endian format and zero-padded to the adjusted underlying field size. The adjusted underlying field size is the underlying field size rounded up to the nearest 8-bit boundary, as noted in the "Field size" column in Table 3, Table 4, or Table 7. This encoding is compatible with the definition given in [SEC1].

2.1.2. Measures to Ensure Secure Implementations

In the following measures are described that ensure secure implementations according to existing best practices and standards defining the operations of Elliptic Curve Cryptography.

Even though the zero point, also called the point at infinity, may occur as a result of arithmetic operations on points of an elliptic curve, it MUST NOT appear in any ECC data structure defined in this document.

Furthermore, when performing the explicitly listed operations in Section 5.1.1.1 it is REQUIRED to follow the specification and security advisory mandated from the respective elliptic curve specification.

3. Supported Public Key Algorithms

This section specifies the composite ML-KEM + ECDH and ML-DSA + ECDSA schemes. All of these schemes are fully specified via their algorithm ID, that is, they are not parametrized.

3.1. Algorithm Specifications

For encryption, the following composite KEM schemes are specified:

ID	Algorithm	Requirement	Definition
TBD	ML-KEM-512+ECDH-NIST-P-256	MAY	Section 5.2
TBD	ML-KEM-768+ECDH-NIST-P-384	MAY	Section 5.2
TBD	ML-KEM-1024+ECDH-NIST-P-384	MAY	Section 5.2
TBD	ML-KEM-768+ECDH-brainpoolP256r1	MAY	Section 5.2
TBD	ML-KEM-1024+ECDH-brainpoolP384r1	MAY	Section 5.2

Table 1: KEM algorithm specifications

For signatures, the following (composite) signature schemes are specified:

ID	Algorithm	Requirement	Definition
TBD	ML-DSA-44+ECDSA-NIST-P-256	MAY	Section 6.2
TBD	ML-DSA-65+ECDSA-NIST-P-384	MAY	Section 6.2
TBD	ML-DSA-87+ECDSA-NIST-P-384	MAY	Section 6.2
TBD	ML-DSA-65+ECDSA-brainpoolP256r1	MAY	Section 6.2
TBD	ML-DSA-87+ECDSA-brainpoolP384r1	MAY	Section 6.2

Table 2: Signature algorithm specifications

3.1.1. Experimental Codepoints for Interop Testing

[Note: this section to be removed before publication]

The algorithms in this draft are not assigned a codepoint in the current state of the draft since there are not enough private/experimental code points available to cover all newly introduced public-key algorithm identifiers.

The use of private/experimental codepoints during development are intended to be used in non-released software only, for experimentation and interop testing purposes only. An OpenPGP implementation MUST NOT produce a formal release using these experimental codepoints. This draft will not be sent to IANA without every listed algorithm having a non-experimental codepoint.

4. Algorithm Combinations

4.1. Composite KEMs

The ML-KEM + ECDH public-key encryption involves both the ML-KEM and an ECDH KEM in a non-separable manner. This is achieved via KEM combination, i.e. both key encapsulations/decapsulations are performed in parallel, and the resulting key shares are fed into a key combiner to produce a single shared secret for message encryption.

4.2. Composite Signatures

The ML-DSA + ECDSA signature consists of independent ML-DSA and ECDSA signatures, and an implementation MUST successfully validate both signatures to state that the ML-DSA + ECDSA signature is valid.

4.3. Key Version Binding

All PQ(/T) asymmetric algorithms are to be used only in v6 (and newer) keys and certificates, with the single exception of ML-KEM-768+X25519 (algorithm ID 35), which is also allowed in v4 encryption-capable subkeys.

5. Composite KEM Schemes

5.1. Building Blocks

5.1.1. ECDH KEM

In this section we define the encryption, decryption, and data formats for the ECDH component of the composite algorithms.

Table 3 and Table 4 describe the ECDH KEM parameters and artifact lengths.

	NIST P-256	NIST P-384
Algorithm ID reference	TBD (ML-KEM-512+ECDH-NIST-P-256)	TBD (ML-KEM-768+ECDH-NIST-P-384, ML-KEM-1024+ECDH-NIST-P-384,)
Field size	32 octets	48 octets
ECDH KEM	ECDH-KEM Section 5.1.1.1	ECDH-KEM Section 5.1.1.1
ECDH public key	65 octets of SEC1-encoded public point	97 octets of SEC1-encoded public point
ECDH secret key	32 octets big-endian encoded secret scalar	48 octets big-endian encoded secret scalar
ECDH ephemeral	65 octets of SEC1-encoded ephemeral point	97 octets of SEC1-encoded ephemeral point
ECDH key share	32 octets	48 octets

Table 3: NIST curves parameters and artifact lengths

	brainpoolP256r1	brainpoolP384r1
Algorithm ID reference	TBD (ML-KEM-768+ECDH-brainpoolP256r1)	TBD (ML-KEM-1024+ECDH-brainpoolP384r1)
Field size	32 octets	48 octets
ECDH KEM	ECDH-KEM Section 5.1.1.1	ECDH-KEM Section 5.1.1.1
ECDH public key	65 octets of SEC1-encoded public point	97 octets of SEC1-encoded public point
ECDH secret key	32 octets big-endian encoded secret scalar	48 octets big-endian encoded secret scalar
ECDH ephemeral	65 octets of SEC1-encoded ephemeral point	97 octets of SEC1-encoded ephemeral point
ECDH key share	32 octets	48 octets

Table 4: Brainpool curves parameters and artifact lengths

The SEC1 format for point encoding is defined in Section 2.1.1.

The various procedures to perform the operations of an ECDH KEM are defined in the following subsections. Specifically, each of these subsections defines the instances of the following operations:

(ecdhCipherText, ecdhKeyShare) <- ECDH-KEM.Encaps(ecdhPublicKey)

and

(ecdhKeyShare) <- ECDH-KEM.Decaps(ecdhCipherText, ecdhSecretKey)

To instantiate ECDH-KEM, one must select a parameter set from Table 3 or Table 4.

5.1.1.1. ECDH-KEM

The operation ECDH-KEM.Encaps() is defined as follows:

1. Generate an ephemeral key pair {v, V=vG} as defined in [SP800-186] or [RFC5639] where v is a random scalar with 0 < v < n, n being the base point order of the elliptic curve domain parameters

2. Compute the shared point S = vR, where R is the recipient’s public key ecdhPublicKey, according to [SP800-186] or [RFC5639]

3. Extract the X coordinate from the SEC1 encoded point S = 04 || X || Y as defined in section Section 2.1.1

4. Set the output ecdhCipherText to the SEC1 encoding of V

5. Set the output ecdhKeyShare to X

The operation ECDH-KEM.Decaps() is defined as follows:

1. Compute the shared Point S as rV, where r is the ecdhSecretKey and V is the ecdhCipherText, according to [SP800-186] or [RFC5639]

2. Extract the X coordinate from the SEC1 encoded point S = 04 || X || Y as defined in section Section 2.1.1

3. Set the output ecdhKeyShare to X

5.1.2. ML-KEM

ML-KEM features the following operations:

(mlkemCipherText, mlkemKeyShare) <- ML-KEM.Encaps(mlkemPublicKey)

and

(mlkemKeyShare) <- ML-KEM.Decaps(mlkemCipherText, mlkemSecretKey)

The above are the operations ML-KEM.Encaps and ML-KEM.Decaps defined in [FIPS-203]. Note that mlkemPublicKey is the encapsulation and mlkemSecretKey is the decapsulation key.

ML-KEM has the parametrization with the corresponding artifact lengths in octets as given in Table 5. All artifacts are encoded as defined in [FIPS-203].

	ML-KEM-512	ML-KEM-768	ML-KEM-1024
Algorithm ID reference	TBD	TBD	TBD
Public key	800 octets	1184 octets	1568 octets
Secret key	64 octets	64 octets	64 octets
Ciphertext	768 octets	1088 octets	1568 octets
Key share	32 octets	32 octets	32 octets

Table 5: ML-KEM parameters and artifact lengths

To instantiate ML-KEM, one must select a parameter set from the column "ML-KEM" of Table 5.

5.2. Composite Encryption Schemes with ML-KEM

Table 1 specifies the following ML-KEM + ECDH composite public-key encryption schemes:

Algorithm ID reference	ML-KEM	ECDH-KEM curve
TBD (ML-KEM-512+ECDH-NIST-P-256)	ML-KEM-512	NIST P-256
TBD (ML-KEM-768+ECDH-NIST-P-384)	ML-KEM-768	NIST P-384
TBD (ML-KEM-1024+ECDH-NIST-P-384)	ML-KEM-1024	NIST P-384
TBD (ML-KEM-768+ECDH-brainpoolP256r1)	ML-KEM-768	brainpoolP256r1
TBD (ML-KEM-1024+ECDH-brainpoolP384r1)	ML-KEM-1024	brainpoolP384r1

Table 6: ML-KEM + ECDH composite schemes

The ML-KEM + ECDH composite public-key encryption schemes are built according to the following principal design:

* The ML-KEM encapsulation algorithm is invoked to create an ML-KEM ciphertext together with an ML-KEM symmetric key share.

* The encapsulation algorithm of an ECDH KEM is invoked to create an ECDH ciphertext together with an ECDH symmetric key share.

* A Key-Encryption-Key (KEK) is computed as the output of a key combiner that receives as input both of the above created symmetric key shares, the ECDH ciphertext, the ECDH public key, and the protocol binding information.

* The session key for content encryption is then wrapped as described in [RFC3394] using AES-256 as algorithm and the KEK as key.

* The PKESK packet’s algorithm-specific parts are made up of the ML-KEM ciphertext, the ECDH ciphertext, and the wrapped session key.

5.2.1. Key Combiner

For the composite KEM schemes defined in this document the procedure multiKeyCombine that is defined in Section 4.2.1 of [I-D.draft-ietf-openpgp-pqc] MUST be used to compute the KEK that wraps a session key.

5.2.2. Key Generation Procedure

The implementation MUST generate the ML-KEM and the ECDH component keys independently. ML-KEM key generation follows the specification in [FIPS-203], and the artifacts are encoded as fixed-length octet strings whose sizes are listed Section 5.1.2. ECDH key generation follows the specification in [SP800-186] or [RFC5639], and the artifacts are encoded as fixed-length octet strings whose sizes and format are listed in Table 3 or Table 4.

5.2.3. Encryption Procedure

The procedure to perform public-key encryption with an ML-KEM + ECDH composite scheme is as follows:

1. Take the recipient’s authenticated public-key packet pkComposite and sessionKey as input

2. Parse the algorithm ID from pkComposite and set it as algId

3. Extract the ecdhPublicKey and mlkemPublicKey component from the algorithm specific data encoded in pkComposite with the format specified in Section 5.3.2.

4. Instantiate the ECDH-KEM and the ML-KEM depending on the algorithm ID according to Table 6

5. Compute (ecdhCipherText, ecdhKeyShare) = ECDH-KEM.Encaps(ecdhPublicKey)

6. Compute (mlkemCipherText, mlkemKeyShare) = ML-KEM.Encaps(mlkemPublicKey)

7. Compute KEK = multiKeyCombine(mlkemKeyShare, ecdhKeyShare, ecdhCipherText, ecdhPublicKey, algId) as defined in Section 5.2.1

8. Compute C = AESKeyWrap(KEK, sessionKey) with AES-256 as per [RFC3394] that includes a 64 bit integrity check

9. Output the algorithm specific part of the PKESK as ecdhCipherText || mlkemCipherText || len(C, symAlgId) (|| symAlgId) || C, where both symAlgId and len(C, symAlgId) are single octet fields, symAlgId denotes the symmetric algorithm ID used and is present only for a v3 PKESK, and len(C, symAlgId) denotes the combined octet length of the fields specified as the arguments.

5.2.4. Decryption Procedure

The procedure to perform public-key decryption with an ML-KEM + ECDH composite scheme is as follows:

1. Take the matching PKESK and own secret key packet as input

2. From the PKESK extract the algorithm ID as algId and the wrapped session key as encryptedKey

3. Check that the own and the extracted algorithm ID match

4. Parse the ecdhSecretKey and mlkemSecretKey from the algorithm specific data of the own secret key encoded in the format specified in Section 5.3.2

5. Instantiate the ECDH-KEM and the ML-KEM depending on the algorithm ID according to Table 6

6. Parse ecdhCipherText, mlkemCipherText, and C from encryptedKey encoded as ecdhCipherText || mlkemCipherText || len(C, symAlgId) (|| symAlgId) || C as specified in Section 5.3.1, where symAlgId is present only in the case of a v3 PKESK.

7. Compute (ecdhKeyShare) = ECDH-KEM.Decaps(ecdhCipherText, ecdhSecretKey)

8. Compute (mlkemKeyShare) = ML-KEM.Decaps(mlkemCipherText, mlkemSecretKey)

9. Compute KEK = multiKeyCombine(mlkemKeyShare, ecdhKeyShare, ecdhCipherText, ecdhPublicKey, algId) as defined in Section 5.2.1

10. Compute sessionKey = AESKeyUnwrap(KEK, C) with AES-256 as per [RFC3394], aborting if the 64 bit integrity check fails

11. Output sessionKey

5.3. Packet Specifications

5.3.1. Public-Key Encrypted Session Key Packets (Packet Type ID 1)

The algorithm-specific fields consist of the output of the encryption procedure described in Section 5.2.3:

* A fixed-length octet string representing an ECDH ephemeral public key in the format associated with the curve as specified in Section 5.1.1.

* A fixed-length octet string of the ML-KEM ciphertext, whose length depends on the algorithm ID as specified in Table 5.

* A one-octet size of the following fields.

* Only in the case of a v3 PKESK packet: a one-octet symmetric algorithm identifier.

* The wrapped session key represented as an octet string.

Note that like in the case of the algorithms X25519 and X448 specified in [RFC9580], for the ML-KEM + ECDH composite schemes, in the case of a v3 PKESK packet, the symmetric algorithm identifier is not encrypted. Instead, it is placed in plaintext after the mlkemCipherText and before the length octet preceding the wrapped session key. In the case of v3 PKESK packets for ML-KEM composite schemes, the symmetric algorithm used MUST be AES-128, AES-192 or AES-256 (algorithm ID 7, 8 or 9).

In the case of a v3 PKESK, a receiving implementation MUST check if the length of the unwrapped symmetric key matches the symmetric algorithm identifier, and abort if this is not the case.

Implementations MUST NOT use the obsolete Symmetrically Encrypted Data packet (Packet Type ID 9) to encrypt data protected with the algorithms described in this document.

5.3.2. Key Material Packets

The composite ML-KEM + ECDH schemes defined in this specification MUST be used only with v6 keys, as defined in [RFC9580], or newer versions defined by updates of that document.

5.3.2.1. Public Key Packets (Packet Type IDs 6 and 14)

The algorithm-specific public key is this series of values:

* A fixed-length octet string representing an ECC public key, in the point format associated with the curve specified in Section 5.1.1.

* A fixed-length octet string containing the ML-KEM public key, whose length depends on the algorithm ID as specified in Table 5.

5.3.2.2. Secret Key Packets (Packet Type IDs 5 and 7)

The algorithm-specific secret key is these two values:

* A fixed-length octet string of the encoded ECDH secret key, whose encoding and length depend on the algorithm ID as specified in Section 5.1.1.

* A fixed-length octet string containing the ML-KEM secret key in seed format, whose length is 64 octets (compare Table 5). The seed format is defined in accordance with Section 3.3 of [FIPS-203]. Namely, the secret key is given by the concatenation of the values of d and z, generated in steps 1 and 2 of ML-KEM.KeyGen [FIPS-203], each of a length of 32 octets. Upon parsing the secret key format, or before using the secret key, for the expansion of the key, the function ML-KEM.KeyGen_internal [FIPS-203] has to be invoked with the parsed values of d and z as input.

6. Composite Signature Schemes

6.1. Building Blocks

6.1.1. ECDSA-Based Signatures

To sign and verify with ECDSA the following operations are defined:

```
(ecdsaSignatureR, ecdsaSignatureS) <- ECDSA.Sign(ecdsaSecretKey,
                                                 dataDigest)
```

and

```
(verified) <- ECDSA.Verify(ecdsaPublicKey, dataDigest,
                           ecdsaSignatureR, ecdsaSignatureS)
```

Here, the operation ECDSA.Sign() is defined as the algorithm in Section "6.4.1 ECDSA Signature Generation Algorithm" of [SP800-186-5], however, excluding Step 1: H = Hash(M) in that algorithm specification, as in this specification the message digest H is a direct input to the operation ECDSA.Sign(). Equivalently, the operation ECDSA.Sign() can be understood as representing the algorithm under Section "4.2.1.1. Signature Algorithm" in [TR-03111], again with the difference that in this specification the message digest H_Tau(M) appearing in Step 5 of the algorithm specification is the direct input to the operation ECDSA.Sign() and thus the hash computation is not carried out. The same statement holds for the definition of the verification operation ECDSA.Verify(): it is given either through the algorithm defined in Section "6.4.2 ECDSA Signature Verification Algorithm" of [SP800-186-5] omitting the message digest computation in Step 2 or by the algorithm in Section "4.2.1.2. Verification Algorithm" of [TR-03111] omitting the message digest computation in Step 3.

The public keys MUST be encoded in SEC1 format as defined in section Section 2.1.1. The secret key, as well as both values R and S of the signature MUST each be encoded as a big-endian integer in a fixed-length octet string of the specified size.

The following table describes the ECDSA parameters and artifact lengths:

	NIST-P-256	NIST P-384	brainpoolP256r1	brainpoolP384r1
Algorithm reference	TBD (ML-DSA-44+ECDSA-NIST-P-256)	TBD (ML-DSA-65+ECDSA-NIST-P-384,ML-DSA-87+ECDSA-NIST-P-384	TBD (ML-DSA-65+ECDSA-brainpoolP256r1)	TBD (ML-DSA-87+ECDSA-brainpoolP384r1)
Field size	32 octets	48 octets	32 octets	48 octets
Public key	65 octets	97 octets	65 octets	97 octets
Secret key	32 octets	48 octets	32 octets	48 octets
Signature value R	32 octets	48 octets	32 octets	48 octets
Signature value S	32 octets	48 octets	32 octets	48 octets

Table 7: ECDSA parameters and artifact lengths

6.1.2. ML-DSA Signatures

Throughout this specification ML-DSA refers to the default pure and hedged version of ML-DSA defined in [FIPS-204].

ML-DSA signature generation is performed using the default hedged version of the ML-DSA.Sign algorithm, as specified in [FIPS-204], with an empty context string ctx. That is, to sign with ML-DSA the following operation is defined:

(mldsaSignature) <- ML-DSA.Sign(mldsaSecretKey, dataDigest)

ML-DSA signature verification is performed using the ML-DSA.Verify algorithm, as specified in [FIPS-204], with an empty context string ctx. That is, to verify with ML-DSA the following operation is defined:

(verified) <- ML-DSA.Verify(mldsaPublicKey, dataDigest, mldsaSignature)

ML-DSA has the parametrization with the corresponding artifact lengths in octets as given in Table 8. All artifacts are encoded as defined in [FIPS-204].

	ML-DSA-44	ML-DSA-65	ML-DSA-87
Algorithm ID reference	TBD	TBD	TBD
Public key	1312 octets	1952 octets	2592 octets
Secret key	32 octets	32 octets	32 octets
Signature	2420 octets	3309 octets	4627 octets

Table 8: ML-DSA parameters and artifact lengths

6.2. Composite Signature Schemes with ML-DSA

6.2.1. Key Generation Procedure

The implementation MUST generate the ML-DSA and the ECDSA component keys independently. ML-DSA key generation follows the specification in [FIPS-204] and the artifacts are encoded as fixed-length octet strings whose sizes are listed in Section 6.1.2. ECDSA key generation follows the specification in [SP800-186] or [RFC5639], and the artifacts are encoded as fixed-length octet strings whose sizes are listed in Section 6.1.1.

6.2.2. Signature Generation

To sign a message M with ML-DSA + ECDSA the following sequence of operations has to be performed:

1. Generate dataDigest according to Section 5.2.4 of [RFC9580]

2. Create the ECDSA signature over dataDigest with ECDSA.Sign() from Section 6.1.1

3. Create the ML-DSA signature over dataDigest with ML-DSA.Sign() from Section 6.1.2

4. Encode the ECDSA and ML-DSA signatures according to the packet structure given in Section 6.3.1

6.2.3. Signature Verification

To verify an ML-DSA + ECDSA signature the following sequence of operations has to be performed:

1. Verify the ECDSA signature with ECDSA.Verify() from Section 6.1.1

2. Verify the ML-DSA signature with ML-DSA.Verify() from Section 6.1.2

As specified in Section 4.2 an implementation MUST validate both signatures, that is, ECDSA and ML-DSA, successfully to state that a composite ML-DSA + ECDSA signature is valid.

6.3. Packet Specifications

6.3.1. Signature Packet (Packet Type ID 2)

The composite ML-DSA + ECDSA schemes MUST be used only with v6 signatures, as defined in [RFC9580], or newer versions defined by updates of that document.

The algorithm-specific v6 signature parameters for ML-DSA + ECDSA signatures consist of:

* A fixed-length octet string of the big-endian encoded ECDSA value R, whose length depends on the algorithm ID as specified in Table 7.

* A fixed-length octet string of the big-endian encoded ECDSA value S, whose length depends on the algorithm ID as specified in Table 7.

* A fixed-length octet string of the ML-DSA signature value, whose length depends on the algorithm ID as specified in Table 8.

A composite ML-DSA + ECDSA signature MUST use a hash algorithm with a digest size of at least 256 bits for the computation of the message digest. A verifying implementation MUST reject any composite ML-DSA + ECDSA signature that uses a hash algorithm with a smaller digest size.

6.3.2. Key Material Packets

The composite ML-DSA + ECDSA schemes MUST be used only with v6 keys, as defined in [RFC9580], or newer versions defined by updates of that document.

6.3.2.1. Public Key Packets (Packet Type IDs 6 and 14)

The algorithm-specific public key for ML-DSA + ECDSA keys is this series of values:

* A fixed-length octet string representing the ECDSA public key in SEC1 format, as specified in section Section 2.1.1, whose length depends on the algorithm ID as specified in Table 7.

* A fixed-length octet string containing the ML-DSA public key, whose length depends on the algorithm ID as specified in Table 8.

6.3.2.2. Secret Key Packets (Packet Type IDs 5 and 7)

The algorithm-specific secret key for ML-DSA + ECDSA keys is this series of values:

* A fixed-length octet string representing the ECDSA secret key as a big-endian encoded integer, whose length depends on the algorithm ID as specified in Table 7.

* A fixed-length octet string containing the ML-DSA secret key in seed format, whose length is 32 octets (compare Table 8). The seed format is defined in accordance with Section 3.6.3 of [FIPS-204]. Namely, the secret key is given by the value xi generated in step 1 of ML-DSA.KeyGen [FIPS-204]. Upon parsing the secret key format, or before using the secret key, for the expansion of the key, the function ML-DSA.KeyGen_internal [FIPS-204] has to be invoked with the parsed value of xi as input.

7. Security Considerations

The following security considerations given in [I-D.draft-ietf-openpgp-pqc] equally apply to this document:

* the security aspects of composite signatures (Section 9.1 in [I-D.draft-ietf-openpgp-pqc]),

* the arguments for the security features of the KEM combiner given in Section 9.2 of [I-D.draft-ietf-openpgp-pqc], as also the NIST and Brainpool curves represent nominal groups according to [ABH_21],

* the considerations regarding domain separation and context binding for the KEM combiner (Section 9.2.1 in [I-D.draft-ietf-openpgp-pqc]),

* the use of the hedged variant of ML-DSA (Section 9.3 in [I-D.draft-ietf-openpgp-pqc]),

* the minimum digest size for PQ/T signatures (Section 9.4 in [I-D.draft-ietf-openpgp-pqc]),

* the use of symmetric encryption in SEIPD packets (Section 9.5 in [I-D.draft-ietf-openpgp-pqc]),

* and key generation for composite schemes (Section 9.6 in [I-D.draft-ietf-openpgp-pqc]).

When implementing or using any of the algorithms defined in this specification, the above referenced security considerations should be noted.

8. IANA Considerations

IANA is requested to add the algorithm IDs defined in Table 9 to the existing registry OpenPGP Public Key Algorithms. The field specifications enclosed in brackets for the ML-KEM + ECDH composite algorithms denote fields that are only conditionally contained in the data structure.

[Note: Once the working group has agreed on the actual algorithm choice, the following table with the requested IANA updates will be filled out.]

ID	Algorithm	Public Key Format	Secret Key Format	Signature Format	PKESK Format	Reference
TBD	ML-DSA-44+ECDSA-NIST-P-256	65 octets ECDSA public key (Table 7), 1312 octets ML-DSA-44 public key (Table 8)	32 octets ECDSA secret key (Table 7), 32 octets ML-DSA-44 secret key (Table 8)	64 octets ECDSA signature Table 7 , 2420 octets ML-DSA-44 signature (Table 8)	N/A	Section 6.2

TBD	ML-DSA-65+ECDSA-NIST-P-384	97 octets ECDSA public key (Table 7), 1952 octets ML-DSA-65 public key (Table 8)	48 octets ECDSA secret key (Table 7), 32 octets ML-DSA-65 secret key (Table 8)	96 octets ECDSA signature Table 7 , 3309 octets ML-DSA-65 signature (Table 8)	N/A	Section 6.2
TBD	ML-DSA-87+ECDSA-NIST-P-384	97 octets ECDSA public key (Table 7), 2592 octets ML-DSA-87 public key (Table 8)	48 octets ECDSA secret key (Table 7), 32 octets ML-DSA-87 secret key (Table 8)	96 octets ECDSA signature Table 7 , 4627 octets ML-DSA-87 signature (Table 8)	N/A	Section 6.2
TBD	ML-DSA-65+ECDSA-brainpoolP256r1	65 octets ECDSA public key (Table 7), 1952 octets ML-DSA-65 public key (Table 8)	32 octets ECDSA secret key (Table 7), 32 octets ML-DSA-65 secret key (Table 8)	64 octets ECDSA signature Table 7 , 3309 octets ML-DSA-65 signature (Table 8)	N/A	Section 6.2

TBD	ML-DSA-87+ECDSA-brainpoolP384r1	97 octets ECDSA public key (Table 7), 2592 octets ML-DSA-87 public key (Table 8)	48 octets ECDSA secret key (Table 7), 32 octets ML-DSA-87 secret key (Table 8)	96 octets ECDSA signature Table 7 , 4627 octets ML-DSA-87 signature (Table 8)	N/A	Section 6.2
TBD	ML-KEM-512+ECDH-NIST-P-256	65 octets ECDH public key (Table 7), 800 octets ML-KEM-512 public key (Table 5)	32 octets ECDH secret key (Table 7), 64 octets ML-KEM-512 secret key (Table 5)	N/A	65 octets ECDH ciphertext, 768 octets ML-KEM-512 ciphertext, 1 octet remaining length, [1 octet algorithm ID in case of v3 PKESK,] n octets wrapped session key (Section 5.3.1)	Section 5.2
TBD	ML-KEM-768+ECDH-NIST-P-384	97 octets ECDH public key (Table 7), 1184 octets ML-KEM-768 public key	48 octets ECDH secret key (Table 7), 64 octets ML-KEM-768 secret key	N/A	97 octets ECDH ciphertext, 1088 octets ML-KEM-768 ciphertext, 1 octet remaining length, [1 octet algorithm ID in case of	Section 5.2

		key (Table 5)	(Table 5)		v3 PKESK,] n octets wrapped session key (Section 5.3.1)	
TBD	ML-KEM-1024+ECDH-NIST-P-384	97 octets ECDH public key (Table 7), 1568 octets ML-KEM -768 public key (Table 5)	48 octets ECDH secret key (Table 7), 64 octets ML-KEM -1024 secret key (Table 5)	N/A	97 octets ECDH ciphertext, 1568 octets ML-KEM-1024 ciphertext, 1 octet remaining length, [1 octet algorithm ID in case of v3 PKESK,] n octets wrapped session key (Section 5.3.1)	Section 5.2
TBD	ML-KEM-768+ECDH-brainpoolP256r1	65 octets ECDH public key (Table 7), 1184 octets ML-KEM -768 public key (Table 5)	32 octets ECDH secret key (Table 7), 64 octets ML-KEM -768 secret key (Table 5)	N/A	65 octets ECDH ciphertext, 1088 octets ML-KEM-768 ciphertext, 1 octet remaining length, [1 octet algorithm ID in case of v3 PKESK,] n octets wrapped session key (Section 5.3.1)	Section 5.2
TBD	ML-KEM-1024+ECDH-brainpoolP384r1	97 octets ECDH	48 octets ECDH	N/A	97 octets ECDH ciphertext,	Section 5.2

		public key (Table 7), 1568 octets ML-KEM -768 public key (Table 5)	secret key (Table 7), 64 octets ML-KEM -1024 secret key (Table 5)	1568 octets ML-KEM-1024 ciphertext, 1 octet remaining length, [1 octet algorithm ID in case of v3 PKESK,] n octets wrapped session key (Section 5.3.1)		

Table 9: IANA updates for registry 'OpenPGP Public Key Algorithms'

IANA is asked to add the following note to this registry:

> The field specifications enclosed in square brackets for PKESK Format represent fields that may or may not be present, depending on the PKESK version.

9. Changelog

This section gives the history of changes in the respective document versions. The order is newest first.

9.1. draft-ehlen-openpgp-nist-bp-comp-02

* Completed the IANA table.
* Added "Security Considerations" section.
* Alignment of various technical details to [I-D.draft-ietf-openpgp-pqc].
* Various editorial alignments to [I-D.draft-ietf-openpgp-pqc].

9.2. draft-ehlen-openpgp-nist-bp-comp-01

* Replaced the explicit description of the KEM combiner with a reference to [I-D.draft-ietf-openpgp-pqc].

10. Contributors

11. References

11.1. Normative References

[FIPS-203] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard", August 2024, <https://doi.org/10.6028/NIST.FIPS.203>.

[FIPS-204] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard", August 2024, <https://doi.org/10.6028/NIST.FIPS.204>.

[I-D.draft-ietf-openpgp-pqc] Kousidis, S., Roth, J., Strenzke, F., and A. Wussler, "Post-Quantum Cryptography in OpenPGP", Work in Progress, Internet-Draft, draft-ietf-openpgp-pqc-12, 17 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-pqc-12>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/rfc/rfc3394>.

[RFC5639] Lochter, M. and J. Merkle, "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation", RFC 5639, DOI 10.17487/RFC5639, March 2010, <https://www.rfc-editor.org/rfc/rfc5639>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/rfc/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/rfc/rfc8032>.

[RFC9580] Wouters, P., Ed., Huigens, D., Winter, J., and Y. Niibe, "OpenPGP", RFC 9580, DOI 10.17487/RFC9580, July 2024, <https://www.rfc-editor.org/rfc/rfc9580>.

[SEC1] Standards for Efficient Cryptography Group, "Standards for Efficient Cryptography 1 (SEC 1)", May 2009, <https://secg.org/sec1-v2.pdf>.

[SP800-186] Chen, L., Moody, D., Regenscheid, A., and K. Randall, "Recommendations for Discrete Logarithm-Based

Cryptography: Elliptic Curve Domain Parameters", NIST Special Publication 800-186 , February 2023, <https://doi.org/10.6028/NIST.SP.800-186>.

[SP800-186-5] Information Technology Laboratory, National Institute of Standards and Technology, "Digital Signature Standard (DSS)", NIST Special Publication 800-186 , February 2023, <https://doi.org/10.6028/NIST.FIPS.186-5>.

11.2. Informative References

[ABH_21] Alwen, J., Blanchet, B., Hauck, E., Kiltz, E., Lipp, B., and D. Riepel, "Analysing the HPKE Standard", 2021, <https://doi.org/10.1007/978-3-030-77870-5_4>.

[RFC9794] Driscoll, F., Parsons, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", RFC 9794, DOI 10.17487/RFC9794, June 2025, <https://www.rfc-editor.org/rfc/rfc9794>.

[TR-03111] Federal Office for Information Security, Germany, "Technical Guideline BSI TR-03111 Elliptic Curve Cryptography, Version 2.1", June 2018, <https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03111/TR-03111_node.html>.

Appendix A. Test Vectors

TBD

A.1. Sample v6 PQC Subkey Artifacts

TBD ## V4 PQC Subkey Artifacts

TBD

Acknowledgments

Authors' Addresses

Quynh Dang
NIST
United States of America
Email: quynh.dang@nist.gov

Stephan Ehlen
BSI
Germany
Email: stephan.ehlen@bsi.bund.de

Stavros Kousidis
BSI
Germany
Email: kousidis.ietf@gmail.com

Johannes Roth
MTG AG
Germany
Email: johannes.roth@mtg.de

Falko Strenzke
MTG AG
Germany
Email: falko.strenzke@mtg.de