Workgroup: LAMPS
Internet-Draft:
draft-ounsworth-pq-composite-kem-01
Published: 13 March 2023
Intended Status: Standards Track
Expires: 14 September 2023
Authors: M. Ounsworth J. Gray
Entrust Entrust

Composite KEM For Use In Internet PKI

Abstract

The migration to post-quantum cryptography is unique in the history of modern digital cryptography in that neither the old outgoing nor the new incoming algorithms are fully trusted to protect data for the required data lifetimes. The outgoing algorithms, such as RSA and elliptic curve, may fall to quantum cryptalanysis, while the incoming post-quantum algorithms face uncertainty about both the underlying mathematics as well as hardware and software implementations that have not had sufficient maturing time to rule out classical cryptanalytic attacks and implementation bugs.

Cautious implementers may wish to layer cryptographic algorithms such that an attacker would need to break all of them in order to compromise the data being protected using either a Post-Quantum / Traditional Hybrid, Post-Quantum / Post-Quantum Hybrid, or combinations thereof. This document, and its companions, defines a specific instantiation of hybrid paradigm called "composite" where multiple cryptographic algorithms are combined to form a single key, signature, or key encapsulation mechanism (KEM) such that they can be treated as a single atomic object at the protocol level.

This document defines the structure CompositeCiphertextValue which is a sequence of the respective ciphertexts for each component algorithm. Explicit pairings of algorithms are defined which should meet most Internet needs. The generic composite key type is also defined which allows arbitrary combinations of key types to be placed in the CompositePublicKey and CompositePrivateKey structures without needing the combination to be pre-registered or pre-agreed. For the purpose of combining KEMs, the combiner function from [I-D.ounsworth-cfrg-kem-combiners] is used.

This document is intended to be coupled with the composite keys structure define in [I-D.ounsworth-pq-composite-keys] and the CMS KEMRecipientInfo mechanism in [I-D.housley-lamps-cms-kemri].

Status of This Memo

Copyright Notice

Table of Contents

1. Changes in version -01

*Sycronized terminology with I-D.draft-driscoll-pqt-hybrid-terminology-01.

*Changed CompositeCiphertextValue from BIT STRING to OCTET STRING.

*Explicit composite combinations defined and ASN.1 module updated

2. Introduction

During the transition to post-quantum cryptography, there will be uncertainty as to the strength of cryptographic algorithms; we will no longer fully trust traditional cryptography such as RSA, Diffie-Hellman, DSA and their elliptic curve variants, while we may also not fully trust their post-quantum replacements until they have had sufficient scrutiny and time to discover and fix implementation bugs. Unlike previous cryptographic algorithm migrations, the choice of when to migrate and which algorithms to migrate to, is not so clear. Even after the migration period, it may be advantageous for an entity's cryptographic identity to be composed of multiple public-key algorithms.

The deployment of composite public keys and composite encryption using post-quantum algorithms will face two challenges

*Algorithm strength uncertainty: During the transition period, some post-quantum signature and encryption algorithms will not be fully trusted, while also the trust in legacy public key algorithms will start to erode. A relying party may learn some time after deployment that a public key algorithm has become

untrustworthy, but in the interim, they may not know which algorithm an adversary has compromised.

*Migration: During the transition period, systems will require mechanisms that allow for staged migrations from fully classical to fully post-quantum-aware cryptography.

This document provides a mechanism to address algorithm strength uncertainty by building on [I-D.ounsworth-pq-composite-keys] by providing the format and process for combining multiple cryptographic algorithms into a single key encapsulation operation. Backwards compatibility is not directly covered in this document, but is the subject of Appendix B.1.

This document is intended for general applicability anywhere that key establishment or enveloped content encryption is used within PKIX or CMS structures.

2.1. Algorithm Selection Criteria

The composite algorithm combinations defined in this document were chosen according to the following guidelines:

1. A single RSA combination is provided (but RSA modulus size not mandated), matched with NIST PQC Level 3 algorithms.

2. Elliptic curve algorithms are provided with combinations on each of the NIST [RFC6090], Brainpool [RFC5639], and Edwards [RFC7748] curves. NIST PQC Levels 1 - 3 algorithms are matched with 256-bit curves, while NIST levels 4 - 5 are matched with 384-bit elliptic curves. This provides a balance between matching classical security levels of post-quantum and traditional algorithms, and also selecting elliptic curves which already have wide adoption.

3. NIST level 1 candidates (Falcon512 and Kyber512) are provided, matched with 256-bit elliptic curves, intended for constrained use cases.

4. A single SPHINCS+ combination is provided for use cases that wish to put hash-based signatures into hybrid combination.

5. A generic composite algorithm is provided for implementers who wish to use combinations not listed here, without the overhead of defining new OIDs. Caution should be exercised to avoid issues with compatibility and complex cryptographic policy mechanisms.

The authors wish to note that although all the composite structures defined in this and the companion composite signatures

[I-D.ounsworth-pq-composite-sigs] and composite signatures [I-D.ounsworth-pq-composite-sigs] specifications are defined in such a way as to easily allow 3 or more component algorithms, it was decided to only specify explicit pairs. The generic composite algorithm allows for an arbitrary number of components. This also does not preclude future specification of explicit combinations with three or more components.

2.2. Terminology

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.

This document is consistent with all terminology from [I-D.driscoll-pqt-hybrid-terminology].

In addition, the following terms are used in this document:

BER: Basic Encoding Rules (BER) as defined in [X.690].

CLIENT: Any software that is making use of a cryptographic key. This includes a signer, verifier, encrypter, decrypter.

COMBINER: A combiner specifies how multiple shared secrets are combined into a single shared secret. DER: Distinguished Encoding Rules as defined in [X.690].

KEM: A key encapsulation mechanism as defined in Section 3.1.

PKI: Public Key Infrastructure, as defined in [RFC5280].

SHARED SECRET: A value established between two communicating parties for use as cryptographic key material, but which cannot be learned by an active or passive adversary. This document is concerned with shared secrets established via public key cryptagraphic operations.

3. Composite KEM Structures

3.1. Key Encapsulation Mechanisms (KEMs)

We borrow here the definition of a key encapsulation mechanism (KEM) from [I-D.ietf-tls-hybrid-design], in which a KEM is a cryptographic primitive that consists of three algorithms:

* KeyGen() -> (pk, sk): A probabilistic key generation algorithm, which generates a public key pk and a secret key sk.

*Encaps(pk) -> (ct, ss): A probabilistic encapsulation algorithm, which takes as input a public key pk and outputs a ciphertext ct and shared secret ss.

*Decaps(sk, ct) -> ss: A decapsulation algorithm, which takes as input a secret key sk and ciphertext ct and outputs a shared secret ss, or in some cases a distinguished error value.

This document is not concerned with the KeyGen() algorithm of a KEM, but it is included above for completeness.

The KEM interface defined above differs from both traditional key transport mechanism (for example for use with KeyTransRecipientInfo defined in [RFC5652]), and key agreement (for example for use with KeyAgreeRecipientInfo defined in [RFC5652]).

The KEM interface was chosen as the interface for a composite key exchange because it allows for arbitrary combinations of component algorithm types since both key transport and key agreement mechanisms can be promoted into KEMs in the following ways:

A key transport mechanism can be transformed into a KEM.Encaps(pk) by generating a random shared secret ss and performing KeyTrans.Encrypt(pk, ss) -> ct; and into a KEM.Decaps(sk, ct) by KeyTrans.Decrypt(sk, ct) -> ss. This follows the pattern of RSA-KEM [RFC5990].

A key agreement mechanism can be transformed into a KEM.Encaps(pk) by generating an ephemeral key pair (pk_e, sk_e), and performing KeyAgree(pk, sk_e) -> (ss, pk_e); and into a KEM.Decaps(sk, ct) by completing the key agreement as KeyAgree(pk_e, sk) -> ss.

A composite KEM allows two or more underlying key transport, key agreement, or KEM algorithms to be combined into a single cryptographic operations by performing each operation, transformed to a KEM as outline above, and using a specified combiner function to combine the two or more component shared secrets into a single shared secret.

The main security property for KEMs is indistinguishability under adaptive chosen ciphertext attack (IND-CCA2), which means that shared secret values should be indistinguishable from random strings even given the ability to have other arbitrary ciphertexts decapsulated. By using the KEM combiner defined in [I-D.ounsworth-cfrg-kem-combiners], the composite KEMs defined in this document inherit the IND-CCA2 security from the general combiner.

TODO: needs more formal analysis that the methods of transforming KeyTrans and KeyAgree meet this.

3.2. kema-CompositeKEM

The ASN.1 algorithm object for a composite KEM is:

```
kema-CompositeKEM KEM-ALGORITHM ::= {
    IDENTIFIER TYPE OBJECT IDENTIFIER
    VALUE CompositeCiphertextValue
    PARAMS TYPE CompositeKemParams ARE required
    PUBLIC-KEYS { pk-Composite }
    SMIME-CAPS { IDENTIFIED BY id-alg-composite } }
```

The following is an explanation how KEM-ALGORITHM elements are used to create Composite KEMs:

SIGNATURE-ALGORITHM element	**Definition**
IDENTIFIER	The Object ID used to identify the composite Signature Algorithm
VALUE	The Sequence of BIT STRINGS for each component signature value
PARAMS	Parameters of type CompositeKemParams may be provided when required
PUBLIC-KEYS	The composite key required to produce the composite signature
SMIME_CAPS	Not needed for composite

Table 1

3.3. Composite Keys

A composite KEM MAY be associated with a composite public key as defined in [I-D.ounsworth-pq-composite-keys], but MAY also be associated with multiple public keys from different sources, for example multiple X.509 certificates, or multiple cryptographic modules. In the latter case, composite KEMs MAY be used as the mechanism for carrying multiple ciphertexts in a non-composite hybrid encryption equivalent of those described for digital signatures in [I-D.becker-guthrie-noncomposite-hybrid-auth].

3.3.1. Key Usage Bits

When using composite KEM keys in a structure which defines a key usage (such as in an X509Certificate as defined in [RFC5280]), the following key usage MUST be used.

keyEncipherment

Additional key usages SHOULD not be used.

3.4. CompositeCiphertextValue

The compositeCipherTextValue is a concatenation of the ciphertexts of the underlying component algorithms. It is represented in ASN.1 as follows:

```
CompositeCiphertextValue ::= SEQUENCE SIZE (2..MAX) OF OCTET STRING
```

3.5. CompositKemParameters

Composite KEM parameters are defined as follows and MAY be included when a composite KEM algorithm is used with an AlgorithmIdentifier:

```
CompositeKemParams ::= SEQUENCE SIZE (2..MAX) OF AlgorithmIdentifier{
    KEM-ALGORITHM, {KEMAlgSet} }
```

The KEM's CompositeKemParams sequence MUST contain the same component algorithms listed in the same order as in the associated CompositePublicKey.

Generic composite algorithms must carry the list of component KEM algorithms so that the reciever knows which algorithms to use. For explicit composite algorithms, it is required in cases where one or both of the components themselves have parameters that need to be carried, however the authors have chosen to always carry it in order to simplify parsers. Implementation SHOULD NOT rely directly on the algorithmIDs contained in the CompositeKemParams and SHOULD verify that they match the algorithms expected from the overall composite AlgorithmIdentifier.

3.6. Encoding Rules

Many protocol specifications will require that composite KEM data structures be represented by an octet string or bit string.

When an octet string is required, the DER encoding of the composite data structure SHALL be used directly.

EDNOTE: will this definition include an ASN.1 tag and length byte inside the OCTET STRING object? If so, that's probably an extra unnecessary layer.

When a bit string is required, the octets of the DER encoded composite data structure SHALL be used as the bits of the bit string, with the most significant bit of the first octet becoming the first bit, and so on, ending with the least significant bit of the last octet becoming the last bit of the bit string.

In the interests of simplicity and avoiding compatibility issues, implementations that parse these structures MAY accept both BER and DER.

3.7. KEM Combiner

TODO: as per https://www.enisa.europa.eu/publications/post-quantum-cryptography-integration-study section 4.2, might need to specify behaviour in light of KEMs with a non-zero failure probility.

This document follows the construction of [I-D.ounsworth-cfrg-kem-combiners], which is repeated here for clarity:

```
KDF(counter || k_1 || ... || k_n || fixedInfo, outputBits)

where
k_i = H(ss_i || ct_i)
```

where:

- KDF and H, and outputBits represent a hash functions suitable to the chosen KEMs,
- fixedInfo any additional context string provided by the protocol,
- counter is fixed to the 32-bit value 0x00000001,
- || represents concatenation.

Each registered composite KEM algorithm must specify the exact KEM combiner construction that is to be used.

For convenience we define the following KMAC-basid intantiations of KEM combiner:

KEM Combiner	KDF	H	outputBits
KMAC128/256	KMAC128	SHA3-256	256
KMAC256/384	KMAC256	SHA3-512	384
KMAC256/512	KMAC256	SHA3-512	512

Table 2: KEM Combiners

KMAC is defined in NIST SP 800-185 [SP800-185]. The KMAC(K, X, L, S) parameters are instantiated as follows:

- K: the ASCI value of the name of the Kem Type OID.
- X: the value "0x00000001 || k_1 || ... || k_n || fixedInfo", where k_i = H(ss_i || ct_i), as defined above.

*L: integer representation of outputBits.

*S: empty string.

~~~ BEGIN EDNOTE ~~~

these choices are somewhat arbitrary but aiming to match security level of the input KEMs. Feedback welcome.

*Kyber512: KMAC128/256

*Kyber768: KMAC256/384

*Kyber1024 KMAC256/512

~~~ END EDNOTE ~~~

For example, the KEM combiner instantiation of the first entry of Table 3 would be:

```
ss = KMAC128("id-Kyber512-ECDH-P256-KMAC128",
     0x00000001 || SHA3-256(ss_1 || ct_1) || SHA3-256(ss_2 || ct_2) || fi
     256, "")
```

4. Algorithm Identifiers

This table summarizes the list of explicit composite Signature algorithms by the key and signature OID and the two component algorithms which make up the explicit composite algorithm. These are denoted by First Signature Alg, and Second Signature Alg.

The OID referenced are TBD and MUST be used only for prototyping and replaced with the final IANA-assigned OIDS. The following prefix is used for each: replace <CompKEM> with the String "2.16.840.1.114027.80.5.2"

Therefore <CompKEM>.1 is equal to 2.16.840.1.114027.80.5.2.1

The "KEM Combiner" column refers to the definitions in Section 3.7.

KEM Type OID	OID	First Algorithm	Second Algorithm	KEM Combiner
id-Kyber512-ECDH-P256-KMAC128	<CompKEM>.1	Kyber512	ECDH-P256	KMAC128/256
id-Kyber512-ECDH-brainpoolP256r1-KMAC128	<CompKEM>.2	Kyber512	ECDH-brainpoolp256r1	KMAC128/256

KEM Type OID	OID	First Algorithm	Second Algorithm	KEM Combiner
id-Kyber512-X25519-KMAC128	<CompKEM>.3	Kyber512	X25519	KMAC128/256
id-Kyber768-RSA-KMAC256	<CompKEM>.4	Kyber768	RSA-KEM	KMAC256/384
id-Kyber768-ECDH-P256-KMAC256	<CompKEM>.5	Kyber768	ECDH-P256	KMAC256/384
id-Kyber768-ECDH-brainpoolP256r1-KMAC256	<CompKEM>.6	Kyber768	ECDH-brainpoolp256r1	KMAC256/384
id-Kyber768-X25519-KMAC256	<CompKEM>.7	Kyber768	X25519	KMAC256/384
id-Kyber1024-ECDH-P384-KMAC256	<CompKEM>.8	Kyber1024	ECDH-P384	KMAC256/512
id-Kyber1024-ECDH-brainpoolP384r1-KMAC256	<CompKEM>.9	Kyber1024	ECDH-brainpoolP384r1	KMAC256/512
id-Kyber1024-X448-KMAC256	<CompKEM>.10	Kyber1024	X448	KMAC256/512
id-composite-kem-KMAC128	<CompKEM>.11	Any	Any	KMAC128/256
id-composite-kem-KMAC256	<CompKEM>.12	Any	Any	KMAC256/512

Table 3: Composite KEM key types

The table above contains everything needed to implement the listed explicit composite algorithms, with the exception of some special notes found below in this section. See the ASN.1 module in section Section 5 for the explicit definitions of the above Composite signature algorithms.

Full specifications for the referenced algorithms can be found as follows:

* *ECDH*: There does not appear to be a single IETF definition of ECDH, so we refer to the following:

  - *ECDH NIST*: SHALL be Elliptic Curve Cryptography Cofactor Diffie-Hellman (ECC CDH) as defined in section 5.7.1.2 of [SP.800-56Ar3].

  - *ECDH BSI / brainpool*: SHALL be Elliptic Curve Key Agreement algorithm (ECKA) as defined in section 4.3.1 of [BSI-ECC]

*Kyber: [I-D.ietf-lamps-kyber-certificates]

*RSA-KEM: [RFC5990]

*X25519 / X448: [RFC8410]

EDNOTE: I believe that [SP.800-56Ar3] and [BSI-ECC] give equivalent and interoperable algorithms, so maybe this is extranuous detail to include?

4.1. Notes on id-Kyber768-RSA-KMAC256

Use of RSA-KEM [RFC5990] deserves a special explanation.

GenericHybridParameters is defined in [RFC5990], repeated here for clarity:

```
GenericHybridParameters ::= {
    kem  KeyEncapsulationMechanism,
    dem  DataEncapsulationMechanism
}
```

The GenericHybridParameters.kem MUST be id-kem-rsa as defined in [RFC5990]:

```
id-kem-rsa OID ::= {
    is18033-2 key-encapsulation-mechanism(2) rsa(4)
}
```

The associated parameters for id-kem-rsa have type RsaKemParameters:

```
RsaKemParameters ::= {
    keyDerivationFunction  KeyDerivationFunction,
    keyLength              KeyLength
}
```

For use with id-Kyber768-RSA-KMAC256, the keyDerivationFunction SHALL be id-sha3-384 and keyLength SHALL be 384.

EDNOTE: I'm borrowing id-sha3-384 from draft-turner-lamps-adding-sha3-to-pkix-00, which looks ilke was abandoned. Do we have PKIX OIDs for SHA3?

EDNOTE: Since the crypto is fixed, we could omit the parameters entirely and expect implementations to re-constitute the params structures as necessary in order to call into lower-level crypto libraries.

TODO: there must be a way to put all this the ASN.1 Module rather than just specifying it as text?

4.2. Notes on Generic Composite

The id-alg-composite-kem-KMAC128 and id-alg-composite-kem-KMAC256 object identifiers are used for identifying a generic composite KEM algorithm. This allows arbitrary combinations of component key transport, key agreement and KEM algorithms without needing the combination to be pre-registered or standardized. Thes generic KEM composite algorithms use KMAC128 and KMAC256-based KEM combiners and so are intended for use with component KEM algorithms that target the 128 bit or 256 bit security levels respectively.

When the id-alg-composite-kem-KMAC128 or id-alg-composite-kem-KMAC256 object identifiers are used with an AlgorithmIdentifier, the AlgorithmIdentifier.parameters MUST be of type CompositeKemParams containing an AlgorithmIdentifier for each component algorithm in the same order as the ciphertexts appear in the corresponding CompositeCiphertextValue`.

5. ASN.1 Module

<CODE STARTS>

```
Composite-KEM-2023
           {iso(1) identified-organization(3) dod(6) internet(1) securit
       mechanisms(5) pkix(7) id-mod(0) id-mod-composite-kems(999)}

DEFINITIONS IMPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS

AlgorithmIdentifier{}
    FROM AlgorithmInformation-2009  -- RFC 5912 [X509ASN1]
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-algorithmInformation-02(58) }

KEM-ALGORITHM, KEMAlgSet
    FROM KEMAlgorithmInformation-2023
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
          id-mod-kemAlgorithmInformation-2023(99) }

id-rsa-kem, GenericHybridParameters
    FROM CMS-RSA-KEM
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        pkcs-9(9) smime(16) modules(0) cms-rsa-kem(21) }

id-RSASSA-PSS, RSASSA-PSS-Params
    FROM PKCS-1 {
       iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1)
       modules(0) pkcs-1(1)
   }

id-composite-key,
pk-composite-kem,
id-Kyber512-ECDH-P256-KMAC128,
pk-Kyber512-ECDH-P256-KMAC128,
id-Kyber512-ECDH-brainpoolP256r1-KMAC128,
pk-Kyber512-ECDH-brainpoolP256r1-KMAC128,
id-Kyber512-X25519-KMAC128,
pk-Kyber512-X25519-KMAC128,
id-Kyber768-RSA-KMAC256,
pk-Kyber768-RSA-KMAC256,
id-Kyber768-ECDH-P256-KMAC256,
pk-Kyber768-ECDH-P256-KMAC256,
id-Kyber768-ECDH-brainpoolP256r1-KMAC256,
pk-Kyber768-ECDH-brainpoolP256r1-KMAC256,
```

```
id-Kyber768-X25519-KMAC256,
pk-Kyber768-X25519-KMAC256,
id-Kyber1024-ECDH-P384-KMAC256,
pk-Kyber1024-ECDH-P384-KMAC256,
id-Kyber1024-ECDH-brainpoolP384r1-KMAC256,
pk-Kyber1024-ECDH-brainpoolP384r1-KMAC256,
id-Kyber1024-X448-KMAC256,
pk-Kyber1024-X448-KMAC256
    FROM CompositeKeys-2023
           {iso(1) identified-organization(3) dod(6) internet(1) securit
       mechanisms(5) pkix(7) id-mod(0) id-mod-composite-keys(98)};



--
-- Composite structures
--

CompositeCiphertextValue ::= SEQUENCE SIZE (2..MAX) OF OCTET STRING

CompositeKemParams ::= SEQUENCE SIZE (2..MAX) OF AlgorithmIdentifier{
    KEM-ALGORITHM, {KEMAlgSet} }

ExplicitCompositeKemParams{KEM-ALGORITHM:FirstKemAlg,
   KEM-ALGORITHM:SecondKemAlg}  ::=
      SEQUENCE {
        kemAlgorithm1   AlgorithmIdentifier
                        { KEM-ALGORITHM, {FirstKemAlg}},
            kemAlgorithm2   AlgorithmIdentifier
                        { KEM-ALGORITHM, {SecondKemAlg}} }


kema-explicitCompositeKEM{OBJECT IDENTIFIER:id, PUBLIC-KEY:publicKeyObje
    CompositeKemParams} KEM-ALGORITHM ::=  {
         IDENTIFIER id
         VALUE CompositeCiphertextValue
         PARAMS TYPE CompositeKemParams ARE required
         PUBLIC-KEYS {publicKeyObject} }



-- Generic Composite KEM


-- TODO: To be replaced by IANA
id-composite-kem-KMAC128 OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) entrust(114027)
  algorithm(80) tbd(98) }

kema-Composite-kem-KMAC128 KEM-ALGORITHM ::= {
```

```
    IDENTIFIER id-composite-kem-KMAC128
    VALUE CompositeCiphertextValue
    PARAMS TYPE CompositeKemParams ARE required
    PUBLIC-KEYS {publicKeyObject} }


-- TODO: To be replaced by IANA
id-composite-kem-KMAC256 OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) entrust(114027)
  algorithm(80) tbd(99) }

kema-Composite-kem-KMAC256 KEM-ALGORITHM ::= {
    IDENTIFIER id-composite-kem-KMAC256
    VALUE CompositeCiphertextValue
    PARAMS TYPE CompositeKemParams ARE required
    PUBLIC-KEYS {publicKeyObject} }




-- Explicit Composite KEMs


kema-Kyber512-ECDH-P256-KMAC128 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber512-ECDH-P256-KMAC128,
    pk-Kyber512-ECDH-P256-KMAC128,
    ExplicitCompositeKemParams{{kema-Kyber512TBD}, {kema-ecdh-p256}} }

--TODO: `kema-ecdh-p256` does not actually exist yet.


kema-Kyber512-ECDH-brainpoolP256r1-KMAC128 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber512-ECDH-brainpoolP256r1-KMAC128,
    pk-Kyber512-ECDH-brainpoolP256r1-KMAC128,
    ExplicitCompositeKemParams{{kema-Kyber512TBD}, {kema-ecdh-brainpoolp

--TODO: `kema-ecdh-brainpoolp256r1` does not actually exist yet.


kema-Kyber512-X25519-KMAC128 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber512-X25519-KMAC128,
    pk-Kyber512-X25519-KMAC128,
    ExplicitCompositeKemParams{{kema-Kyber512TBD}, {kema-x25519}} }

--TODO: `kema-x25519` does not actually exist yet.


kema-Kyber768-RSA-KMAC256 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber768-RSA-KMAC256,
    pk-Kyber768-RSA-KMAC256,
```

```
    ExplicitCompositeKemParams{{kema-Kyber512TBD}, {kema-kem-rsa}} }


kema-Kyber768-ECDH-P256-KMAC256 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber768-ECDH-P256-KMAC256,
    pk-Kyber768-ECDH-P256-KMAC256,
    ExplicitCompositeKemParams{{kema-Kyber768TBD}, {kema-ecdh-p256}} }

--TODO: `kema-ecdh-p256` does not actually exist yet.



kema-Kyber768-ECDH-brainpoolP256r1-KMAC256 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber768-ECDH-brainpoolP256r1-KMAC256,
    pk-Kyber768-ECDH-brainpoolP256r1-KMAC256,
    ExplicitCompositeKemParams{{kema-Kyber768TBD}, {kema-ecdh-brainpoolp

--TODO: `kema-ecdh-brainpoolp256r1` does not actually exist yet.


kema-Kyber768-X25519-KMAC256 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber768-X25519-KMAC256,
    pk-Kyber768-X25519-KMAC256,
    ExplicitCompositeKemParams{{kema-Kyber768TBD}, {kema-x25519}} }

--TODO: `kema-x25519` does not actually exist yet.


kema-Kyber1024-ECDH-P384-KMAC256 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber1024-ECDH-P384-KMAC256,
    pk-Kyber1024-ECDH-P384-KMAC256,
    ExplicitCompositeKemParams{{kema-Kyber1024TBD}, {kema-ecdh-p384}} }

--TODO: `kema-ecdh-p384` does not actually exist yet.


kema-Kyber1024-ECDH-brainpoolP384r1-KMAC256 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber1024-ECDH-brainpoolP384r1-KMAC256,
    pk-Kyber1024-ECDH-brainpoolP384r1-KMAC256,
    ExplicitCompositeKemParams{{kema-Kyber1024TBD}, {kema-ecdh-brainpool

--TODO: `kema-ecdh-brainpoolp384r1` does not actually exist yet.


kema-Kyber1024-X448-KMAC256 KEM-ALGORITHM ::=
    kema-explicitCompositeKEM{id-Kyber1024-X448-KMAC256,
    pk-Kyber1024-X448-KMAC256,
    ExplicitCompositeKemParams{{kema-Kyber1024TBD}, {kema-x448}} }

--TODO: `kema-x448` does not actually exist yet.
```

```
END
```

<CODE ENDS>

6. IANA Considerations

The following need to be assigned by IANA:

*The OID for the ASN.1 module Composite-KEM-2023,

*OIDs for id-composite-kem-KMAC128, id-composite-kem-KMAC256

7. Security Considerations

7.1. Policy for Deprecated and Acceptable Algorithms

Traditionally, a public key, certificate, or signature contains a single cryptographic algorithm. If and when an algorithm becomes deprecated (for example, RSA-512, or SHA1), it is obvious that structures using that algorithm are implicitly revoked.

In the composite model this is less obvious since implementers may decide that certain cryptographic algorithms have complementary security properties and are acceptable in combination even though one or both algorithms are deprecated for individual use. As such, a single composite public key, certificate, signature, or ciphertext may contain a mixture of deprecated and non-deprecated algorithms.

Specifying behaviour in these cases is beyond the scope of this document, but should be considered by Implementers and potentially in additional standards.

EDNOTE: Max is working on a CRL mechanism to accomplish this.

7.2. OR Modes

TODO -- we'll need security consideration analysis of whatever OR modes we choose.

7.3. KEM Combiner

This document uses directly the KEM Combiner defined in [I-D.ounsworth-cfrg-kem-combiners] and therefore inherits all of its security considerations, which the authors believe have all been addressed in the concrete choices for both explicit and generic composites.

8. References

8.1. Normative References

[BSI-ECC] Federal Office for Information Security (BSI), "Technical Guideline BSI TR-03111: Elliptic Curve Cryptography. Version 2.10", 1 June 2018.

[I-D.housley-lamps-cms-kemri] Housley, R., Gray, J., and T. Okubo, "Using Key Encapsulation Mechanism (KEM) Algorithms in the Cryptographic Message Syntax (CMS)", Work in Progress, Internet-Draft, draft-housley-lamps-cms-kemri-02, 20 February 2023, <https://datatracker.ietf.org/doc/html/draft-housley-lamps-cms-kemri-02>.

[I-D.ietf-lamps-kyber-certificates] Turner, S., Kampanakis, P., Massimo, J., and B. Westerbaan, "Internet X.509 Public Key Infrastructure - Algorithm Identifiers for Kyber", Work in Progress, Internet-Draft, draft-ietf-lamps-kyber-certificates-00, 26 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-kyber-certificates-00>.

[I-D.ounsworth-pq-composite-keys] Ounsworth, M., Gray, J., Pala, M., and J. Klaussner, "Composite Public and Private Keys For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-keys-04, 13 March 2023, <https://datatracker.ietf.org/api/v1/doc/document/draft-ounsworth-pq-composite-keys/>.

[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>.

[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/info/rfc5280>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5990] Randall, J., Kaliski, B., Brainard, J., and S. Turner, "Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS)", RFC 5990, DOI 10.17487/RFC5990, September 2010, <https://www.rfc-editor.org/info/rfc5990>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[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/info/rfc8174>.

[RFC8410] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/info/rfc8410>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[SHA3] National Institute of Standards and Technology (NIST), "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202, DOI 10.6028/NIST.FIPS.202", August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf\>.

[SP.800-56Ar3] National Institute of Standards and Technology (NIST), "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography", April 2018, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf>.

[SP800-185] National Institute of Standards and Technology (NIST), "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash", December 2016, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf>.

[X.690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:2015, November 2015.

8.2. Informative References

[I-D.driscoll-pqt-hybrid-terminology] D, F., "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-driscoll-pqt-hybrid-terminology-01, 20 October 2022,

<https://datatracker.ietf.org/doc/html/draft-driscoll-pqt-hybrid-terminology-01>.

[I-D.ietf-tls-hybrid-design] Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid key exchange in TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-hybrid-design-04, 11 January 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-04>.

[I-D.ounsworth-cfrg-kem-combiners] Ounsworth, M., Wussler, A., and S. Kousidis, "Combiner function for hybrid key encapsulation mechanisms (Hybrid KEMs)", Work in Progress, Internet-Draft, draft-ounsworth-cfrg-kem-combiners-03, 13 March 2023, <https://datatracker.ietf.org/doc/html/draft-ounsworth-cfrg-kem-combiners-03>.

[I-D.ounsworth-pq-composite-sigs] Ounsworth, M., Gray, J., and M. Pala, "Composite Signatures For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-sigs-08, 13 March 2023, <https://datatracker.ietf.org/api/v1/doc/document/draft-ounsworth-pq-composite-sigs/>.

[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/info/rfc5639>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <https://www.rfc-editor.org/info/rfc6090>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[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/info/rfc7748>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version

4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

Appendix A. Samples

TBD

Appendix B. Implementation Considerations

This section addresses practical issues of how this draft affects other protocols and standards.

EDNOTE 10: Possible topics to address:

- *The size of these certs and cert chains.
- *In particular, implications for (large) composite keys / signatures / certs on the handshake stages of TLS and IKEv2.
- *If a cert in the chain is a composite cert then does the whole chain need to be of composite Certs?
- *We could also explain that the root CA cert does not have to be of the same algorithms. The root cert SHOULD NOT be transferred in the authentication exchange to save transport overhead and thus it can be different than the intermediate and leaf certs.
- *We could talk about overhead (size and processing).
- *We could also discuss backwards compatibility.
- *We could include a subsection about implementation considerations.

B.1. Backwards Compatibility

As noted in the introduction, the post-quantum cryptographic migration will face challenges in both ensuring cryptographic strength against adversaries of unknown capabilities, as well as providing ease of migration. The composite mechanisms defined in this document primarily address cryptographic strength, however this section contains notes on how backwards compatibility may be obtained.

The term "ease of migration" is used here to mean that existing systems can be gracefully transitioned to the new technology without requiring large service disruptions or expensive upgrades. The term "backwards compatibility" is used here to mean something more

specific; that existing systems as they are deployed today can interoperate with the upgraded systems of the future.

These migration and interoperability concerns need to be thought about in the context of various types of protocols that make use of X.509 and PKIX with relation to key establishment and content encryption, from online negotiated protocols such as TLS 1.3 [RFC8446] and IKEv2 [RFC7296], to non-negotiated asynchronous protocols such as S/MIME signed email [RFC8551], as well as myriad other standardized and proprietary protocols and applications that leverage CMS [RFC5652] encrypted structures.

B.1.1. K-of-N modes

~~~ BEGIN EDNOTE ~~~ In the context of encryption, K-of-N modes could mean one of two things:

Type 1: sender uses a subset

This would mean the sender (encrypter) uses a subset of K the N component keys within the receiver's public key. The obvious way to combine them is with skipping the unused keys / algorithms and emitting a NULL ciphertext in their place. This mechanism is straight-forward and allows ease of migration where a sender encounters a composite encryption public key where it does not support all component algorithms. It also supports performance optimization where, for example, a receiver can be issued a key with many component keys and a sender can choose the highest-performance subset that are still considered safe.

Type 2: receiver uses a subset

This would mean that the sender (encrypter) uses all N of the component keys within the receiver's public key in such a way that the receiver (decrypter) only needs to use K private keys to decrypt the message. This implies the need for some kind of Shamir's-like secret splitting scheme. This is a reasonably complex mechanism and it's currently unclear if there are any use-cases that require such a mechanism.

~~~ END EDNOTE ~~~

B.1.2. Parallel PKIs

We present the term "Parallel PKI" to refer to the setup where a PKI end entity possesses two or more distinct public keys or certificates for the same identity (name), but containing keys for different cryptographic algorithms. One could imagine a set of parallel PKIs where an existing PKI using legacy algorithms (RSA, ECC) is left operational during the post-quantum migration but is

shadowed by one or more parallel PKIs using pure post quantum algorithms or composite algorithms (legacy and post-quantum).

Equipped with a set of parallel public keys in this way, a client would have the flexibility to choose which public key(s) or certificate(s) to use in a given signature operation.

For negotiated protocols, the client could choose which public key(s) or certificate(s) to use based on the negotiated algorithms.

For non-negotiated protocols, the details for obtaining backwards compatibility will vary by protocol, but for example in CMS [RFC5652].

EDNOTE: I copied and pruned this text from [I-D.ounsworth-pq-composite-sigs]. It probably needs to be fleshed out more as we better understand the implementation concerns around composite encryption.

Appendix C. Intellectual Property Considerations

The following IPR Disclosure relates to this draft:

https://datatracker.ietf.org/ipr/3588/

EDNOTE: I don't think this applies to this draft.

Appendix D. Contributors and Acknowledgements

This document incorporates contributions and comments from a large group of experts. The Editors would especially like to acknowledge the expertise and tireless dedication of the following people, who attended many long meetings and generated millions of bytes of electronic mail and VOIP traffic over the past year in pursuit of this document:

Serge Mister (Entrust), Ali Noman (Entrust), Scott Fluhrer (Cisco), Jan Klaussner (D-Trust), Max Pala (CableLabs), and Douglas Stebila (University of Waterloo).

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

D.1. Making contributions

Additional contributions to this draft are welcome. Please see the working copy of this draft at, as well as open issues at:

https://github.com/EntrustCorporation/draft-composite-kem/

Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada

Email: mike.ounsworth@entrust.com

John Gray
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada

Email: john.gray@entrust.com