CHANGES SINCE 116

- Major overhaul.
- Aligned alg combinations with OpenPGP WG, resulting in:
  - Signatures: 14 explicit + 3 generic pairings
    - Hash-then-sign versions of generics added.
  - KEMs: 10 explicit + 2 generic pairings
- ASN.1 modules (mostly) complete and compiling.
- KEM combiner function is academically-sound.
  - I published draft-ounsworth-cfrg-kem-combiners jointly with Aron Wussler (OpenPGP) and Stavros Kousidis (BSI), and I piggy-back on that multi-KDF construction here.
- KofN mode draft progressing (Max and Jan).

draft-ounsworth-pq-composite-keys/sigs/kem -- draft-pala-klaussner-composite-kofn
**Composite Drafts Ready for Adoption**

- **CFRG:**
  - draft-ounsworth-cfrg-kem-combiners-03

- **LAMPS:**
  - draft-ietf-lamps-cms-kemri

**CompositePublicKey**

```plaintext
::= SEQUENCE SIZE (2..MAX) OF SubjectPublicKeyInfo
```

**CompositeSignatureValue**

```plaintext
::= SEQUENCE SIZE (2..MAX) OF BIT STRING
```

**CompositeCiphertextValue**

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

- **draft-ounsworth-pq-composite-keys-04**
  - Uses

- **draft-ounsworth-pq-composite-sigs-08**
  - Uses

- **draft-pala-klaussner-composite-kofn-00**
  - Extension of

**draft-ounsworth-pq-composite-kem-01**

- Uses

Allows subset (k-threshold) signature validations.

**deprecated-algorithms** CRL / OCSP extension.

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

where

\[ k_i = H(ss_i || ct_i) \]
SIGNATURES (17)

- id-Dilithium3-RSA-PSS
- id-Dilithium3-RSA-PKCS15-SHA256
- id-Dilithium3-ECDSA-P256-SHA256
- id-Dilithium3-ECDSA-brainpoolP256r1-SHA256
- id-Dilithium3-Ed25519
- id-Dilithium5-ECDSA-P384-SHA384
- id-Dilithium5-ECDSA-brainpoolP384r1-SHA384
- id-Dilithium5-Ed448
- id-Falcon512-ECDSA-P256-SHA256
- id-Falcon512-ECDSA-brainpoolP256r1-SHA256
- id-Falcon512-Ed25519
- id-SPHINCSplusSHA256128sSimple-ECDSA-P256-SHA256
- id-SPHINCSplusSHA256128sSimple-ECDSA-brainpoolP256r1-SHA256
- id-SPHINCSplusSHA256128sSimple-Ed25519
- id-alg-composite
- id-alg-composite-sha256
- id-alg-composite-sha512

draft-ounsworth-pq-composite-keys/sigs/kem
KEMS (12)

- id-Kyber512-ECDH-P256-KMAC128
- id-Kyber512-ECDH-brainpoolP256r1-KMAC128
- id-Kyber512-X25519-KMAC128
- id-Kyber768-RSA-KMAC256
- id-Kyber768-ECDH-P256-KMAC256
- id-Kyber768-ECDH-brainpoolP256r1-KMAC256
- id-Kyber768-X25519-KMAC256
- id-Kyber1024-ECDH-P384-KMAC256
- id-Kyber1024-ECDH-brainpoolP384r1-KMAC256
- id-Kyber1024-X448-KMAC256
- id-composite-kem-KMAC128
- id-composite-kem-KMAC256
HASH-THEN-SIGN

Currently, PQ algorithms are used without pre-hashing.

There are situations where hash-then-sign is the preferred path
  - Performance over large data / large amount of signatures
  - Signing different data when using hybrid approaches

Informal conversations with NIST indicate
  - There are no security concerns over the extra hash dependency
  - SHA256 and SHA512 algorithms should be considered

We propose the definition of OIDs for hash-n-sign for PQC and Generic Composite

draft-ounsworth-pq-composite-keys/sigs/kem
K-OF-N

- Composite implements the “AND” paradigm.

- There are situations where devices might be able to validate K signatures out of the N present in a composite situation
  - Devices not fully upgraded (limitations)
  - Switch to a different subset of algorithms (cyclic migration)
  - Implementation issues for new algorithms (agile processes)

- During a transitioning period when the confidence in one or more specific algorithms might be still strong, K of N can provide the long-term confidence
  - Same structure as Composite Crypto, but a key parameter that is used to indicate K (Integer)
K-OF-N

Draft available for review


Open Questions (need help)

Specify only K?
Specify Key BitMask (i.e., BitMask w/ 1 or 0 to indicate MUST or MAY) ?
Specify K and optionally the Key BitMask?

Backward Compatibility considerations

Behaves like Composite if Key OptionalParams are absent
ALGORITHM REVOCATION: THE ISSUE

- When multiple algorithms are used in a PKI, there might be many situations where one or more algorithms should not be trusted anymore but others are still trusted
  - Mixed-Algorithm PKIs (e.g., Root, Intermediate, and End-Entities)
  - Hybrid PKIs (Composite or other)
- Single Algorithm PKIs are Out of Scope (crypto dependency)
- Differently from the normal revocation use-case (individual incidents)
  - algorithmic failure is a systemic issue
  - Scalability issue with the size of the certificate population
- We suggest a compact way to provide efficient mass-revocation

draft-pala-klaussner-composite-kofn
ALGORITHM REVOCATION: APPROACH

- A new system could be deployed to deal with algorithm revocation, or
- We can extend the revocation system in a backward compatible fashion
  - Add algorithm-based revocation alongside individual serial-number revocation
- An extension for CRLs and OCSP responses can be defined
  - A list of revoked algorithms identifiers, or
  - A list of revoked algorithms identifiers and associated start invalidity date, or
  - More complex structures
- No extra procedures needed for distributing the information
- Integrates in today’s processes and fit automation requirements

draft-pala-klaussner-composite-kofn
ALGORITHM REVOCATION: CALL TO ACTION

Scalable, Simple, and Cost-Effective Algorithm revocation procedures are needed to provide algorithm management over time.

The proposed approach provides a mass-revocation mechanism:
- Scalable – Independent from the certificates’ population size
- Simple – Integrates with current revocation checking procedures
- Cost-effective – Short crypto-periods suggests multiple migrations

Call to Actions
- Looking for Collaborators
- Looking for Use-Cases that might require more complex solution (did we miss something here?)

Looking forward to future discussions
OPEN DESIGN QUESTIONS

» Combine into one draft?

» 17 Sigs, 12 KEMs. Too many? Not enough? Debate! GO!

» ASN.1 problems (we need an adult):

1. How to carry EC P256 / brainpoolp256 params?

2. We need KEM wrapped versions of ECDH-ES (ex.: kema-ECDH, kema-x25519, etc), but they don’t exist. With more protocols only supporting KEM interfaces (ex.: HPKE RFC9180), it probably makes sense to define these separately from the composite stuff. I’m happy to co-author if someone else volunteers 😈

» Discussion on the details of hash-then-sign and K-of-N

draft-ounsworth-pq-composite-keys/sigs/kem
Adoption?