Hybrid PQ/T Key Encapsulation Mechanisms

draft-irtf-cfrg-hybrid-kems

https://datatracker.ietf.org/doc/draft-irtf-cfrg-hybrid-kems
https://github.com/cfrg/draft-irtf-cfrg-hybrid-kems

Timeline

- KEM Combiners Design Team delivered design goals at IETF 120, July 2024
- Rough[1] draft Nov 2024
- First draft uploaded to datatracker Jan 2025
- v01 added test vectors and code added thanks to Chris Wood, Feb 2025
- Generic keygen, labels/domain separators, added in v02
- Current version is v03

[1] Roughness indicated by commit message: '💀'

Shape

- One document
- Descriptions of cryptographic dependencies
- Two generic constructions and a common generic keygen design
- Three concrete instatiations of the above, two of which are FIPS-friendly and one which is more IETF-friendly
- Security properties sections for the whole document and each concrete construction
- Guidance on sampling scalars for elliptic curves
- Out of Scope Considerations
- Test vectors

Generic Constructions

§ 5.2 - ‘KitchenSink’ construction

```
def KitchenSink-KEM.SharedSecret(pq_SS, trad_SS, pq_CT, pq_PK, trad_CT,
                                 trad_PK, label):
    input = concat(pq_SS, trad_SS, pq_CT, pq_PK,
                   trad_CT, trad_PK, label)
    return KDF(input)
```

§ 5.2 - ‘KitchenSink’ construction

- Puts 'the whole transcript' through the KEM KDF. Because the entire hybrid KEM ciphertext and encapsulation key material are included in the KDF preimage, relies on the minimum security properties of its component algorithms at the cost of more bytes needing to be processed by the KDF, and is resilient against implementation errors in the component algorithms
- Simple
- Happens to ~match[1] KeyCombine□^{cca} from NIST draft SP 800-227

[1] TODO: move the order of encaps keys and ciphertexts around to exactly match

§ 5.3 - ‘QSF’ construction

```
def QSF-KEM.SharedSecret(pq_SS, trad_SS, pq_CT, pq_PK, trad_CT,
                         trad_PK, label):
    return KDF(concat(pq_SS, trad_SS, trad_CT, trad_PK, label))
```

§ 5.3 - ‘QSF’ construction

- Inspired by the generic QSF (Quantum Superiority Fighter) framework introduced and proven in X-Wing
- Leverages the security properties of a KEM like ML-KEM and an inlined instance of DH-KEM, to elide other public data like the PQ ciphertext and encapsulation key from the KDF input
- Requires more from the PQ KEM than just IND-CCA: ciphertext second preimage resistance (C2SPI, also known as chosen ciphertext resistance, CCR), equivalent to LEAK-BIND-K,PK-CT security

§ 5.1 - Generic key generation and derivation

```
def expandDecapsulationKey(sk):
  expanded = XOF(sk, Npqseed + Ntradseed)
  (pq_PK, pq_SK) = PQKEM.DeriveKey(expanded[..Npqseed])
  trad_SK = G.ScalarFromBytes(expanded[Npqseed..])
  trad_PK = G.SerializeElement(NominalGroup.ScalarMultBase(trad_SK))
  return (pq_SK, trad_SK, pq_PK, trad_PK)

def KeyGen():
  sk = random(Nseed)
  (pq_SK, trad_SK, pq_PK, trad_PK) = expandDecapsulationKey(sk)
  return sk, concat(pq_PK, trad_PK)
```

Similarly, `DeriveKey` works as follows:¶

```
def DeriveKey(seed):
  (pq_SK, trad_SK, pq_PK, trad_PK) = expandDecapsulationKey(seed)
  return sk, concat(pq_PK, trad_PK)
```

§ 5.1 - Generic key generation and derivation

- A common key generation and derivation design for both generic constructions
- X-Wing style, customizable by
- Gives strong binding properties and operational security ergonomics
- Felt dangerous to not provide generic keygen and derivation even with concrete ones
- Current signals from Dustin Moody indicate this will be a FIPS-supported mode with ML-KEM with the next update of SP 800-133

Concrete Constructions

Concrete Hybrid KEM Instances

`QSF-KEM(ML-KEM-768,P-256)-XOF(SHAKE256)-KDF(SHA3-256)`

- Using the QSF combiner with SHA3-256 as the hash function based on ML-KEM-768 and P-256, along with SHAKE256 as the key derivation XOF

`KitchenSink-KEM(ML-KEM-768,X25519)-XOF(SHAKE256)-KDF(HKDF-SHA-256)`

- Using the KitchenSink combiner based on ML-KEM-768 and X25519

`QSF-KEM(ML-KEM-1024,P-384)-XOF(SHAKE256)-KDF(SHA3-256)`

- Using the QSF combiner with SHA3-256 as the hash function based on ML-KEM-1024 and P-384, along with SHAKE256 as the key derivation XOF

Concrete Hybrid KEM Instances

- Needed a concrete instance of each generic construction (at least)
- Needed a NIST-friendly instance
- A !NIST/IETF-friendly instance
- X-Wing, but without X25519
- One with a KDF other than SHA3, therefore HKDF-SHA256
- Didn’t want too many instances!
- Any further concrete instances can be specified in further documents (HQC?)

Out of Scope

- More than two component KEMs (or N number of KEMs)
  - Most demand for one traditional, one post-quantum
- Parameterized output length
  - Fixed length outputs per concrete instance
- Protocol-specific labels / info
  - There is support for a label, that is per-concrete-instance, if you want a protocol-specific concrete instance that’s doable
- Other Component Primitives
  - Can be covered in subsequent documents

Next up

- HPKE KEM IANA Registration?
- Serialization nits
- More language on binding properties
- ???

Hybrid PQ/T Key Encapsulation Mechanisms

draft-irtf-cfrg-hybrid-kems

https://datatracker.ietf.org/doc/draft-irtf-cfrg-hybrid-kems
https://github.com/cfrg/draft-irtf-cfrg-hybrid-kems