Hybrid KEMs

draft-irtf-cfrg-hybrid-kems-05
draft-irtf-cfrg-concrete-hybrid-kems-00

CFRG @ IETF123

Agenda

The Brief

The Current Drafts

The Authors' Assessment of Current State

Request for Feedback

The Brief

CFRG email 2024-07-10

Also:

- Provide guidance that lets non-cryptographers make safe hybrid KEMs
- Provide some targets for the ecosystem to converge on
- Alignment with LAMPS, OpenPGP, X-Wing

(A) Identify which KEM security properties are IETF-relevant, and provide a terse overview of those security properties (eg. IND-CCA, LEAK-BIND-K-PK, HON-BIND-K-CT, etc), as well as security properties unique to hybrid KEMs (component key material reuse between hybrid and non-hybrid uses or between multiple hybrids, one component is malicious while the other is honest, etc) with reference to literature, and put into context with real-world attacks. From that, give guidance on a sensible baseline.

(B) Provide a terse overview of well-reviewed techniques that are options to safely produce the concrete combinations in (C), and which security properties are achieved given those of the constituents.

(C) Provide an initial number of explicit PQ/T hybrid KEMs using techniques from (B) that reach the baseline set in (A), and should include:

(I) a hybrid of P-256 and ML-KEM-768,
(II) a hybrid of X25519 and ML-KEM-768, and,
(III) a hybrid of P-384 and ML-KEM-1024.

These hybrids should be accompanied by pseudocode and test vectors.

The Current Drafts

draft-irtf-cfrg-hybrid-kems (“the generic doc”)

- Discusses security properties relevant for hybrid KEMs for IETF use
- Three frameworks for building hybrid KEMs
- Security requirements for inputs to building hybrid KEMs

draft-irtf-cfrg-hybrid-kems (“the concrete doc”)

- Three explicit hybrid KEMs instantiated according to the generic document

The Security Goals and Requirements

Goals for the Hybrid KEM

- **IND-CCA**, even if the T broken by QC, or PQ has a vuln
- Binding properties, mainly {**LEAK**, **HON**}-BIND-**K**-Q

Requirements

- Nominal Groups: secure in the normal DH sense(s)
- KEMs: IND-CCA; C2PRI in order to use QSF
- KDFs: RO
- PRG: PRG (not PRNG!)

The Hybrid KEM Schemes

GHP = H(ss_PQ, ss_T, ct_PQ, ct_T, ek_PQ, ek_T, label)

- Simple and generic; usable with a pair of IND-CCA KEMs
- Based on [GHP18]

PRE = H(ss_PQ, ss_T, ct_PQ, ct_T, **H(**ek_PQ, ek_T**)**, label)

- Pre-hash public keys for perf when keys are (a) reused and (b) large
- Based on Chempat [draft-josefsson-chempat]

QSF = H(ss_PQ, ss_T, ~~**ct_PQ**~~, ct_T, ~~**ek_PQ**~~, ek_T, label)

- Performance optimization when PQ KEM has C2PRI
- Based on [BCD+24]

Do we really need three?

Encap Time (ms, M1 Pro)	GHP	PRE1		PREN		QSF	
X25519 + FrodoKEM-976	**3.12**	3.12	-0.17%	3.12	-0.20%	3.08	-1.31%
X25519 + Classic McEliece 6960119f	1.32	1.32	0.08%	**0.13**	**-90.25%**	0.13	-89.95%
X25519 + ML-KEM-768	0.80	0.78	-2.23%	0.77	-3.26%	**0.76**	**-5.28%**

Performance differences are measurable, but small / conditional

Some differences in level of vetting among the techniques

The Concrete Hybrid KEMs

Currently, three instantiations:

- QSF-MLKEM768-P256-SHA3256-SHAKE256
- QSF-MLKEM768-X25519-SHA3256-SHAKE256
- QSF-MLKEM1024-P384-SHA3256-SHAKE256

Not in there currently, but maybe should be?

- Other combinations, e.g., to demonstrate techniques other than QSF
- General guidance on how to choose between GHP / PRE / QSF

The Authors' Assessment of Current State

"Engineering" parts are pretty solid

- Clearly specified, implementable
- Techniques seem well-supported by the literature
- Main decisions are fairly minor, e.g., what to pre-hash

"Research" parts could use some work

- Filling in gaps / clarifying links between specified techniques and literature
- Arguing for why these constructions are the right set for the Internet
- Ensuring that requirements on constituents are clear and correct
- Explaining how the constituents in the concrete doc meet the requirements

Validating our assumptions

1. It is reasonable to expect designers of concrete instances to verify C2PRI before using QSF
    - Pro: Faster with a compatible KEM
    - Con: Newer / less commonly-analyzed property
2. Pre-hashing public keys provides noticeable performance win and **seems** compatible with published security proofs
    - H(ss_PQ, ss_T, **H(**ct_PQ, ct_T**)?**, **H(**ek_PQ, ek_T**)?**, label)
        - Pre-hash nothing (== GHP)
        - Pre-hash public keys (== PRE)
        - Pre-hash ciphertexts
        - Pre-hash both (== Chempat)
    - Pro: Faster when the hashed thing is large and reused
    - Con: Complexity; proof misalignment; hashing CT would impact IND-CCA proofs

Let’s Get This Done

Do folks agree that this draft is generally on the right track?

What are the important gaps that need to be filled to finalize?

Should we have a virtual interim in ~September to discuss?

Work in HPKE and MLS WGs is blocked on us finishing this document