

COSE FN-DSA and SLH-DSA

draft-ietf-cose-falcon and draft-ietf-cose-sphincs-plus

Michael Prorock, Ori Steele, Hannes Tschofenig

Status Update

- Prototype implementation of COSE FN-DSA and SLH-DSA
 - Uses t_cose as a foundation
 - Implementation also offers ML-DSA
 - Code available at https://github.com/hannestschofenig/t_cose/tree/pqc
- Updates to examples in both drafts (for COSE part)
 - COSE_Key and COSE_Sign1 representation
 - For COSE SLH-DSA there is currently only a single example available (due to size reasons)
 - Example can be found in the appendix

Re-Evaluating Design Decisions

How many algorithm variants do we want to register?

Should COSE also register HashSLH-DSA algorithms, analogous to RFC 9909?

- RFC 9909 registers both pure SLH-DSA and HashSLH-DSA for X.509/PKIX.
- The current COSE SLH-DSA draft only covers a small set of pure SLH-DSA algorithms.
- The decision point is whether COSE should follow the PKIX model or keep a smaller profile.

Why RFC 9909 includes HashSLH-DSA?

- RFC 9909 addresses X.509 certificates and CRLs, where object sizes can be large.
- It explicitly considers operational constraints, such as the use of HSMs.
- HashSLH-DSA reduces the amount of data that must be buffered, transferred, or processed by the signer.
- In PKIX, this was considered valuable enough to standardize both pure and pre-hash variants.

COSE has a different design context

- COSE has generally favored smaller, more interoperable algorithm profiles.
- The existing PQC COSE drafts do not try to register every possible FIPS variant.
- Hash Envelope offers a COSE-native alternative. It allows signing a digest instead of the original payload.
- This helps with large objects, detached content, remote signing, and constrained devices.
- It can provide many of the same operational benefits that motivated HashSLH-DSA in RFC 9909.
- It does so without creating a new family of PQC algorithm registrations.

Why Hash Envelope is not the same as HashSLH-DSA?

- HashSLH-DSA is a standardized pre-hash signature algorithm.
- Hash Envelope is a protocol construction that signs a digest as content.
- With Hash Envelope, security depends on the application profile requiring recomputation and comparison of the digest against the original content.
- So Hash Envelope is operationally similar, but not semantically identical.

Arguments for registering HashSLH-DSA in COSE

- Align COSE with RFC 9909 and FIPS 205 more closely.
- Provide a standardized algorithm-level pre-hash mode, not just a protocol workaround.
- Help applications that want explicit cryptographic semantics rather than a signed-digest construction.
- Reduce ambiguity for implementers who expect both pure and pre-hash variants to exist everywhere.
- Hash Envelope is only defined for COSE – not for JOSE.

Arguments against registering HashSLH-DSA in COSE

- Adds more algorithm identifiers and more implementation complexity.
- Duplicates functionality that Hash Envelope already addresses at the protocol layer (for COSE).
- Weakens the case for keeping COSE algorithm registries small and focused.
- May solve a PKIX-specific problem in a place where COSE already has another tool.

Practical Options

- **Option 1: Do not register HashSLH-DSA in COSE**
 - Keep COSE focused on pure SLH-DSA.
 - Use Hash Envelope when large-object or remote-signing workflows are needed.
- **Option 2: Register HashSLH-DSA now**
 - Align with RFC 9909.
 - Offer explicit algorithm-level pre-hash semantics in COSE.
- **Option 3: Defer registration**
 - First gain implementation and deployment experience with Hash Envelope.
 - Revisit HashSLH-DSA only if clear interoperability or deployment gaps remain.

SLH-DSA: RFC 9909 vs. COSE/JOSE Draft

- SLH-DSA defines parameter sets for security levels 128, 192, and 256.
- The 192 and 256 variants provide higher security margins than the 128 variants.
- They are mainly relevant for deployments with very long security lifetimes, conservative security policies, or regulatory requirements.
- The tradeoff is cost: larger keys, larger signatures, higher computation cost, and more implementation complexity.
- That is why some protocols start with 128-level variants only: they cover the main interoperability case while keeping the initial algorithm set small.

Why Some Specifications Omit 192/256

- Omitting 192 and 256 does not mean they are weak or undesirable.
- It usually reflects a profiling decision: standardize a small mandatory-to-implement core first.
- A smaller initial set reduces the number of code paths, test cases, and interoperability combinations.
- This is especially attractive in early PQC deployments, where simplicity and implementation experience matter.
- The downside is reduced alignment with the full FIPS 205 parameter space and with standards such as RFC 9814 and RFC 9909.

Meaning of s and f

- In SLH-DSA, s means small signatures.
- In SLH-DSA, f means fast signatures.
- These are two optimization points within the same security category.
- The s variants reduce signature size, but signing is typically slower.
- The f variants speed up signing, but signatures are typically larger.

Why s/f Exists

- The s and f variants let implementers choose between bandwidth/storage efficiency and signing performance.
- This is useful because deployment constraints differ:
 - constrained links or storage-heavy environments may prefer s
 - high-throughput signing environments may prefer f
- Keeping both gives implementers a meaningful choice without changing the basic algorithm family.
- In practice, s/f is a performance and size tradeoff, while 128/192/256 is a security-level choice.

Next Steps

- Address discussion from this IETF meeting.
- There is no hurry with COSE FN-DSA (as discussed in earlier meetings) – COSE/JOSE SLH-DSA could be finalized soon.
- Interest in collecting test artifacts in the PKIX interop test group.
- Reach out to me if you are planning to deploy these algorithms for COSE & JOSE.