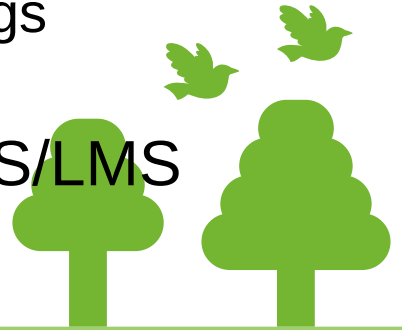
draft-josefsson-cfrg-mothma-00

Mothma: Generic Instantiated Post-Quantum/Traditional Hybrid Signatures

Motivation

- Signed artifacts stay around essentially forever (30+ year use)
- Current signature schemes (EdDSA, ECDSA, RSA) are vulnerable to quantum attackers
- Migrating to new algorithms (ML-DSA, SLH-DSA, XMSS, LMS) without hedging would come with risks
  - Obvious risks: ML-DSA: lattices, SLH-DSA: implementation bugs, XMSS/LMS: state compromise and implementation bugs
- Therefor: hybrid signature schemes combining EdDSA/ECDSA/RSA with ML-DSA/SLH-DSA/XMSS/LMS



Motivation^2

- Why the generic instantiated model?
- Generic:
  - A conservative generalized way to combine two algorithms allows a re-usable pattern and separate security analysis of the construct.
- Instantiated:
  - Engineers do not want to work with generic frameworks: provide safe-to-use hybrid signature schemes as a black box.
- Inspired by the Chempat Hybrid KEM combiner

Mothma: sign

The signed message is (s2,s1,r,h,m) where

m = the message being signed,
r = H(fresh randomness chosen during signing),
h = H(r,H(hybridpk),hybridsigname,appname,appcontext,m),
s1 = traditional signature of (r,h),
s2 = post-quantum signature of (s1,r,h),
H = SHA3-256.

The hash function SHA3-256 is defined in [NIST.FIPS.202].

Mothma: verify

Signed message is (s2,s1,r,h,m)

h' = H(r,H(hybridpk),hybridsigname,appname,appcontext,m),

v1 = Ed25519 verification of s1 on message (r,h),

v2 = ML-DSA-65 verification of s2 on message (s1,r,h),

verify = h == h' && v1 && v2

Comparison/Properties

- Mothma is SUF-CMA if the outer PQ alg is SUF-CMA
- draft-ietf-lamps-pq-composite-sigs
  - Specific to ML-DSA - no generic construct
  - Concatenate signatures vs inner/outside hashing – no SUF-CMA
- Mothma is compatible with signed-message APIs: concatenation designs requires detached signatures
- Signatures with message recovery?
  - Is anyone interested in this? Possible to support.

Obscure usage modes

- It will be possible to find environments in which this signature scheme is not optimal
- The big problem to solve is rolling out PQ-safe signatures without risking disaster if the PQ algorithm/implementation is vulnerable
- Ergo: don’t let perfect stand in the way of progress