FIPS 204 STATUS UPDATE

Ray Perlner
Computer Security Division
NIST
FIPS 204 specifies ML-DSA Based on the Selected NIST PQC submission CRYSTALS-Dilithium

- A lattice-based signature scheme
- In the Fiat-Shamir with aborts paradigm
- Uses modules over an NTT friendly ring $F_q[x]/\langle x^{256} + 1 \rangle$

ML-DSA is expected to become the main NIST Approved signature scheme for general use

- Relatively small signatures and keys
- Fast KeyGen, Signing, and Verification
- Not as small as Falcon/FN-DSA, but doesn’t need floating point arithmetic
FIPS 204 COMMENT PERIOD

• Draft FIPS 204 was posted August 24, 2023 on the NIST website: https://csrc.nist.gov/pubs/fips/204/ipd

• In the 90 day comment period we had 37 commenters give feedback (80 pages)

• Lots of pqc-forum discussion, both before and after
1. Expand secret key \((s_1, s_2, \ldots)\) using skDecode

2. Expand matrix \(A\) using ExpandA

3. Create message representative: \(\mu \leftarrow H(tr | M)\)

4. Perform Rejection Sampling loop until a valid signature \((\tilde{c}, z, h)\) is produced
   
   1. \(y \leftarrow \text{ExpandMask(“Per-Sig-Random”, “Counter”)}\)
   2. \(\tilde{c} \leftarrow H(\text{HighBits}(Ay), \mu)\)
   3. \(c \leftarrow \text{SampleInBall}(\tilde{c})\)
   4. \(z \leftarrow y + c \cdot s_1\)
   5. \(h \leftarrow \text{MakeHint(...)}\)

5. Pack Signature using SigEncode (which calls HintBitPack)
Draft FIPS 204 introduced a few changes from version 3.1 of the Dilithium Spec


- The lengths of $t_r$ and $\tilde{c}$ were increased aiming to increase security strength for BUFF properties
- The default way to generate “Per-Sig-Random” was changed from fully deterministic to “hedged”
- (Unintentional) the pseudocode for HintBitUnpack was missing a check that was present in the Dilithium reference implementation
Change SampleInBall to take all of $\tilde{c}$, rather than just the first 256 bits

- We don’t think this makes a security difference, but the new way is cleaner, and several commenters requested it

Change ExpandMask to use SHAKE output from the beginning rather than at an offset

- As pointed out by Vadim Lyubashevsky, offset not necessary to prevent SHAKE output bits from being reused

Fix missing check in HintBitUnpack

- Check is necessary for Strong Unforgeability (SUF-CMA)
- Thanks to Mike Hamburg for pointing this out and to Sönke Jendral for confirming the security impact

Domain Separated Pure and Pre-hash variants

- Similar change planned for FIPS 205 (Except, for ML-DSA, no SHAKE256 pre-hash — would be redundant)
- To be discussed in upcoming panel
Fixed lengths of private key and signature in tables and algorithm Input/Output description
  • Several commenters noted these did not match the pseudocode

Use of SHAKE with indeterminant output length described with “Streaming Interface”
  • Similar change planned for FIPS 203

Treat hash functions as inputting/outputting byte strings (except when hashing message -- which may be a bit string)

Removed an unnecessary check for the weight of the hint in Verification
  • Hint Unpacking already guarantees the weight of the hint is small enough (pointed out by Beat Heeb)

Explicitly allow implementations to limit iterations in while loops
  • Provide minimum number of implementations such that hitting limit (without bug) will be cryptographically rare
  • Similar change planned for FIPS 203

Lower level “derandomized API”
  • For testing, random values can be treated as inputs to inner keygen and signing functions
  • Similar change planned for FIPS 203 and FIPS 205
Some suggestions from the public comments we do NOT currently plan to accept:

- Replace XOF with DRBG during sampling procedures (ExpandA, ExpandMask)
- Replace XOF with RNG during sampling procedures
- Replace all hashing using SHAKE with SHA2
- Swap the order of tr and M in computing the message representative
- Increase the size of the private random seed during keygen from 32 bytes

Generally, we defaulted to not making a change when the case seemed borderline
We welcome your comments/Questions!

Also feel free to send comments via email:

• Send comments to pqc-comments@nist.gov
• Public discussions: pqc-forum@list.nist.gov

THANK YOU!