FILIPPO VALSORDA

# Comments on the draft versions of FIPS 203

I am a maintainer of the cryptography libraries distributed with the Go language, and these are my personal comments, but they are informed by and oriented towards implementing ML-KEM for the Go ecosystem.

The specification is welcome and well-written. I was able to implement the whole scheme based on it without referring to existing implementations, even without a formal background in mathematics. I especially appreciated the discussion of the various types in Section 2.4, and the consistent use of formatting to denote types. I also appreciated the guidance on key naming (lines 644–652) and the requirement not to expose K-PKE (lines 657–661).

Regarding the changes from the Kyber submission, **I support the restriction on the output key (lines 299–303), and the removal of the randomness hashing step (lines 309–314)**. The former helps with API simplicity. The latter because in practice it's hard to imagine a real-world system that can survive—as a whole—compromise of its RBG just because its KEM hashes the RBG output.

I am neutral on whether the encapsulation key is required to decode to integers already reduced modulo q (lines 315–318), or reduced in the process of decoding. In practice, our APIs will have to be able to return errors anyway for keys of incorrect length, for example. It's important for NIST to provide test vectors exercising these edge cases.

However, **I suggest rolling back the change to the Fujisaki-Okamoto transform** (lines 304–308), reintroducing the hash of the ciphertext in the shared secret derivation. The native transcript hash was for me one of the most welcome features of Kyber. It's virtually certain that there will be protocols in the future that won't implement transcript hashing appropriately and will inadvertently rely on the contributory properties of ML-KEM, just like it happened in the past with non-contributory ECDH. Those protocols will be incorrect, but blaming those designers is a suboptimal solution when we have the option of making the primitive robust to such misuse. This change removed a meaningful, even if not

hi@filippo.io | https://filippo.io

strictly required, security property. There's also some risk of confusion in downgrading the security properties of later versions of the (approximately) same scheme. Note how there are two threads in the IRTF CFRG about this already: the recent posts under "[CFRG] draft-westerbaan-cfrg-hpke-xyber768d00" and "[CFRG] HPKE-xyber will need to commit to ct when you update it to ML-KEM".

At this stage, we are probably going to implement only ML-KEM-768, but **I support the specification of the whole range of parameters**. It's unfortunate that an application targeting 128 bits of security strength but choosing ML-KEM-768 for extra margin against ML cryptanalysis progress is forced to use an approved RBG with a security strength of at least 192 bits (lines 687–689). **I suggest that NIST approve the use of a 128-bit RBG with ML-KEM-768 if targeting Security Category 1.** This would make it easier for applications to adopt the NIST recommendation for ML-KEM-768 (line 1081) without having to replace other components, or having to downgrade to ML-KEM-512 even if the performance budget allowed the use of ML-KEM-768.

**Compression and decompression in Section 4.2.1 could use some extra implementation guidance** for how to perform those operations with bitwise operators. Implementing these in constant time is not trivial, and requiring each implementor to devise (or copy) a strategy risks introducing needless bugs or timing side channels.

The specification almost never reuses or overloads variable names. This consistent global lexical scope helps avoid confusion in discussions or code, and makes test vectors such as those at c2sp.org/CCTV/ML-KEM easier to use. The one exception is that **"r" is reused for the 32-byte K-PKE.Encrypt input and for the vector of polynomials sampled from it. Renaming one of the two would make the property complete.**

**The comment on line 6 of Algorithm 14 (K-PKE.Decrypt) is incorrect**: NTT−1 is invoked only once.

Some paragraphs are missing line numbers, such as between lines 475 and 476 or between 530 and 531.