Public Comments on draft FIPS 203

Comment period: August 24, 2023 – November 22, 2023

On August 24, 2023, NIST requested comments on the initial draft FIPS 203, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. The comments that NIST received during the comment period are collected below.

LIST OF COMMENTS

1. Comments from Simo Sorc, August 24, 2023

Hello,
I was checking reference number 4 and it leads to a not found page:
https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

Best Regards,
Simo.

--
Simo Sorce
RHEL Crypto Team
Red Hat, Inc

2. Comments from Mark G, August 24, 2023

What I would do or if possible is engineer a process code/ codes that would detect the attack or pending attack though various means. One such way is to have a false page that would lead those who would attack this subject with undisclosed codes or means which I do not have the expert knowledge of. Once the attack is underway by the AI computer it could be detected by the times it tries to attack if that is the way it works, if not then put in place a closed gate electronic barrier. But in the end send the attacking computer to attack itself, destroy not only itself but other networks on their system. This would cause an alarm and make them think twice about trying this again. Even if they figured it out, the codes are like a wheel that spins itself, so there is no way they could find out the next code or way in without doing never ending harm to their system.

I look back at history, what did the ancient countries do to keep the warriors of other countries from invading them. Those who did so kept changing their defensive means to survive lasted longer than others.

Mark G

3. Comments from Simon Hoerder, August 24, 2023

Hi,

Thanks for the draft standards, it is very good to have them. However, I notice that none of them contain a test vector. I believe it would be very useful to have at least a test vector included, maybe even at a suitably high level a worked example. This would help immensely with interoperability concerns.

Please note that I put the FIPS-20x-comments@nist.gov addresses in BCC to avoid list replies accidentally getting sent there.

Best regards,
Simon

4. Comments from Joe Hobo, August 24, 2023

- line 458: "copie" to "copy."
- line 715 ff.: Using SHAKE as a byte stream is well and good, but there is no specification for doing so in FIPS 202, i.e. "SHAKE128(M)" doesn't exist. Most if not all of the potential PQC will use SHAKE for pseudorandom streams, rejection sampling, etc. This should probably be addressed in FIPS 202.

5. Comments from Inverted 311, August 25, 2023

Hello. This algorithm line of defense is just not going to cut it. Now, fractals are infinite... Why not create a fractal line of defense by selecting a random location within the fractal? An equation of infinite possibilities is the best defense against a security breach. Older devices (like a 5 year old smart phone) for instance can run a fractal program with ease. The more simple the defense strategy is, the better results of our current systems capabilities. Growning in half life, current systems will become hybrid quantam devices that extend current technology's relevance even further. If this is implemented in theory, it would not matter how advanced quantum computers developed into, due to the fractal defense system's capability of hiding everything in infinite possible locations from the threat. This theory may result in defeating the security concerns, by utilizing the infinite possibilities that a quantum computer would have to solve is the optimal solution of defense tactics against them. Now these two options came up. Either keeping track of our signal inside a rotating fractal defense system that would nearly be impossible for an enemy to keep track of. Or by selecting a random coordinate inside the fractal defense system. That is Kept secure by our National Security Defenses. The odds would be of astronomical proportions to breach this type of defense. When our leaders that protected this information retire, a new coordinate would be selected and passed on to the replacement of thier predecessor. If it is possible to implement a neural network of diamonds and used in the processing potential of a crystalline tungsten shield for protection. This might open the door to develop a system even more advanced that in theory could be impenetrable. . God Bless you all and I pray this message finds someone far more inelegant than I. And is able to run with this theory and implement it.

6. Comments from A. Baksi, K. Jang, H. Kim, G. Song, H. Seo, A. Chattopadhyay, August 28, 2023

Dear NIST,

Regarding the recent FIPS reports (FIPS 203, FIPS 204 and FIPS 205), we have noticed that the estimates on the Grover's search complexity on AES-128/-192/-256 are taken from the Eurocrypt'20 paper by Jaques et al. However, their implementations contained some bugs (due to inherent issues with Q#), and consequently the estimates were erroneous. We have contacted the authors of this paper, and they agreed about the presence of the bug.

As we have shown in Table 11 of our paper ("Quantum Analysis of AES" by Jang et al.), the complexities with their bug-fixed code for AES-128/-192/-256 are respectively 2^{157.33}, 2^{222.76}, 2^{287.28} (we considered multiple bug-fixing and optimizations on top of their base codes and those were the best results).

Apart from that, our own implementations in the same paper achieve the complexity estimates of 2^{156.64}, 2^{221.99} and 2^{286.48} respectively; which are the currently best-known results to the best of our knowledge.

Therefore, we would be grateful if you kindly consider having a look at our work. We would be obliged to discuss with you, should there be any necessity.

Thanking you,
Anubhab Baksi (for, and on behalf of other authors),
Singapore
On behalf of: Kyungbae Jang, Hyunji Kim, Gyeongju Song, Hwajeong Seo, Anupam Chattopadhyay

7. Comments from Roberto Avanzi, August 25, 2023

Hello,

for some reason my name, Roberto Avanzi, is misspelled as "Robert Avanzi" in reference [2]. Please fix it.

best
Roberto

8. Comments from Simon Hoerder, August 28, 2023

Hi,

in the old Kyber spec
https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf
line 6 of Algorithm 5, KYBER.CPAPKE.Enc() specifies:
    Ahat_transposed[i][j] := Parse(XOF(rho,i,j))

In the new FIPS 203 draft for ML-KEM
(https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.ipd.pdf) line 6 of
Algorithm 13, K-PKE.Encrypt() specifies:
    Ahat[i][j] := SampleNTT(XOF(rho, i, j))

In both specs line 19, where Ahat_transposed is actually used, specifies:
    u := NTTinv(Ahat_transposed * rhat)) + e1

I'm wondering whether I'm missing something but this seems like a compatibility-breaking change that doesn't affect security and/or functionality. Unfortunately I'm a little too tired to trusted my own
research at the moment but as far as I can see:

* XOF is completely independent of the Ahat_transposed vs Ahat question.
* Parse() is the same as SampleNTT() so doesn't affect the Ahat_transposed vs Ahat question.
* Operator precedence in the original spec is (Ahat_transposed)[i][j], not (Ahat[i][j])_transposed.

Can someone clarify, please?

Thanks,
Simon

9. Comments from Elaine Barker, August 29, 2023

1. Line 55: The quotes are in the wrong font.
2. Line 56: I still would prefer "secret shared key" rather than "shared secret key" because of the use of "shared secret" in SP 800-56.
3. Line 56, "jointly by two parties": This sounds like key agreement, where each party contributes something that affects the value of the key. For the KEM, the entity performing ML-KEM. Encaps determines the key and sends it to the entity that performs the decapsulation process. Reword this sentence, perhaps to something like "A secret key is generated by one party (e.g., Party A) and shared with the other party (e.g., Party B) using a set of rules and parameters."
4. Lines 66-69: It would be useful to insert "plaintext" for clarity somewhere. Maybe reword to something like "Party B then generates a (plaintext) secret key, encapsulates that key using party A's encapsulation key to produce ciphertext and sends the ciphertext to Party A [don't say "same channel" here; at this point we can't presume what that will be]. Finally, Party A uses the ciphertext provided by Party B along with Party A's private decapsulation key to compute its copy of the (plaintext) secret key, i.e., the secret shared key."

   Also, consider swapping the roles of A and B.
5. Line 71-72: Please define "noisy linear equation" and briefly describe or reference an explanation of the MLWE problem.
6. Line 302-303: Does a 128-bit key need to be derived? Can the 256-bit key be used as two 128-bit keys? Or will this all be addressed in SP 800-227?
7. Line 319: The title has a format problem.
8. Section 2.1: Add definitions for randomness (see 90C), noisy linear equation, fresh (see 90c), and little endian.
9. Lines 414-415: Shouldn't the definition say something about the variables reversing but the coefficients don't?
10. Lines 431-432: How about something like "if x ≥ y + 1/2, then $\lceil x \rfloor$ = y+1; otherwise, $\lceil x \rfloor$ = y"?
11. Line 458: Misspelling of "copy".
12. Text below Algorithm 1: $\mathbb{Z}_m^l$ is not defined in Section 2.3. Does the "script L" mean that there are polynomials with 256 terms each?
13. Lines 559-562: K_A and K_B should be shown in the figure.
14. Lines 622-623 and Appendix A: I don't think that the explanation of the security categories belong in each FIPS; rather, they could be included in a NISTIR, SP 800-175B, or SP 800-57 Part 1.
15. Line 628: Insert "that" after "Provided".

16. Line 630: Remove the comma.
17. Line 632: The roles of Alice and Bob don't match the figure.
18. Line 680: Add NIST SP 800-227 (where the use of the KEM will be discussed.
19. Line 684, "fresh": define something like "a newly generated string that has never been used and will not be reused".
20. (4.1): ":=" not defined in Section 2.3.
21. Line 741: I'm a bit uncomfortable with claiming that SHAKE256 is a hash function when it has not been "officially" approved as such. Can we do something about this or change the title to include XOFs?
22. (4.4): How about something like "a || B <-- G(c) := SHA3-512(c)"?
23. Line 757: Capitalize "it".
24. Algorithm 2: How about "*Converts a bit string of a length that is a multiple*..."
25. Algorithm 7: Dη is not defined in Section2.3.
26. Line 820: Change to "On the input of a polynomial..."
27. Lines 832-833: Refer to the appropriate algorithm [(4.8) or (4.9)?]
28. Line 901: Refer to step 1 where the randomness is obtained.
29. Line 904, "XOF": Refer to (4.2).
30. Line 905, "seed": Is this referring to ρ shown in step 20? Insert "(ρ)" here.
31. Lines 906-907: Change to "Once **A**, **s**, and **e** are generated..."
32. Lines 913-914: How about, "takes an encryption key ek_{pke}, plaintext m, and randomness r as input and outputs..."?
33. Line 949, "$\mathbf{s}^T$": Not mentioned in Section 2.3, although $\mathbf{v}^T$ and $\mathbf{A}^T$ are there. Also, in steps 5 and 6, there is a hat over the **s**. Actually step 6 and this sentence don't quite "match".
34. Lines 962-963: The KEM implementer has no control over which parameter set will be selected. Reword this or remove it. The best you might be able to do is to say that an application must use the appropriate parameter set. I suppose that whoever sets up the application could select the proper parameter set.
35. Lines 1033-1036: I don't know from reading this what the application needs to do next. Does the calling application need to perform step 7 and compare against the returned value of K' or what?
36. Line 1065, "block cipher with a prescribed key size": The key needs to be generated using a method that supports the intended strength for the key. For AES, this means that the key needs to be generated using a method (e.g., using an RBG) that supports a security strength of 256 bits in order to provide that level of security.

37. Line 1083, "other parameter sets": This seems to imply that it may be OK to use ML-KEM-512 (for performance reasons) even though ML-KEM-1028 might be needed for security reasons. Either reword or remove this sentence.
38. Lines 1085-1086: Somehow indicate that SP 800-227 is in the works.

10. Comments from David Wittenberg, August 30, 2023

These are almost trivial, but clarity and consistency are important, so I'm reporting them:

Line 66-67: The statement: Party B then uses Party A's encapsulation key to generate one copy of a shared secret key along with an associated ciphertext. is confusing. It implies that the ciphertext and the secret key are unrelated. You should make clear that the ciphertext is the encrypted secret key. Also diagram at 555 shows only ciphertext being passed.

Similarly line 339 says: produces a shared secret key and an associated ciphertext as output.

Line 344: The encryption key can be made public. This will surprise people. Do you want to note that this is sometimes called a "public key"?

Line 351: defines KEM ciphertext but diagram at line 555 uses ciphertext. (also ciphertext is used at 339)

David "Doc" Wittenberg, PhD.
Chief Architect, HNJT
Hanscom Air Force Base

11. Comments from Andrew R. Huber, September 1, 2023

Several minor editorial comments on FIPS 203 (Draft)
Module-Lattice-based Key-encapsulation Mechanism Standard
1. “Copie” in lines 457 – 458:
“In other words, a copy of the inputs is created, and the subroutine is invoked with the copie.”
“copie” should be “copy”.

(OK, yes, there really is a word “copie” and it is used here in the correct sense, but using it instead of “copy” conveys nothing and it only confuses things and looks wrong.
It also seems inconsistent to use “copy” in the first instance and “copie” in the second.)

2. Section 2.3 Mathematical Symbols (lines 409 – 449)
Please alphabetize the entries in this section, as are Section 2.1 and 2.2.

Having to do a linear search on every lookup in this section is tedious and annoying.

3. Algorithm 5, line 3:
Here and in other pseudo-code algorithms the Greek Sigma appears for summation.

Please consider replacing these Sigmas with appropriate for loops, as that will be more code-like and make for easier reading and a less dense specification that will be clearer.

12. Comments from Manjula Sreenivasamurthy, September 3, 2023

Sir
Thank yiu giving the opportunity to review FIPS standard for public.

I am manjula sreenivasamurthy working senior advanced syber security architect-compliance, i am performing prouduct security risks, threats,vulnerabilities assessor activities as per NIST guidelines. working in Honeywell technology, bangalore, India.

I have reviewed the FIPS-203 draft standards, below are my review comments, how these can be handled as part of NIST standards.

1.How safe is the shared secret key that is shared by two parties over the public channels. Once the key is shared on public channel, it is highly vulnerable to cyber attacks.

2.How the two parties safe guard the secret keys against the disclosure of these values and secrecy of several values including randomnes used by 2 parties. Is there any methods, guidelines used to keep these value safe outof reach from the attackers.

3.There is no guarantee that the use of the product containing an implementation of ML-KEM safe gaurd the security of the overall system in which the product is used. What is the work around for this is not clearly articulated. How the products can implement this withnout any guarantee to safe gaurd the overall systems.

4. There needs to be guidelines for qualification of capsulation and decapsulation for every transaction from source to destination.

5. There needs to be guidelines for input validation and output validation.

6. There needs to be guidelines for validating the destruction of intermodules is needed.

7. There needs to be guidelines for security requirements and security testing to validate the ML-KEM modules algorithms as per regulatory requirements for defence, space and aerospace, and other critical sectors.

8.ML-KEM comes equipped with 3 parameter sets such as ML-KE-512, ML-KEM-768,ML-KEM-1024, beyong these parameter sets if the data transfer happenes how to handle such situation is not articulated in this standard.

9. Include the guidelines for assessing and validating these modules.

Hope my review comments helps.
Thank you
Manjula

13. Comments from Pierre-Agustin Berthet, September 6, 2023

Greetings,

In the ML-KEM Decapsulation (Algorithm 17 Section 6.3), the ciphertext c is concatenated with a secret element z, resulting in a 32*(du*k + dv + 1) bytes word which will be hashed (Algorithm 17 Line 7). From a side channel countermeasure point of view, this hash is sensitive and has to be protected, leading to a downgrade in performances. To circumvent this issue, we would like to suggest to pre-hash the ciphertext, as done in CRYSTALS-Kyber v3.02, as the ciphertext is not a sensitive data and thus the hash of the ciphertext does not need to be protected. This will significantly reduce the size of the data processed as input by the sensitive hash at Line 7 Algorithm 17 and thus improve its performances. With our tweak, this line would look like this: $\bar{K} \leftarrow J(z \| H(c), 32)$.
While we acknowledge that this suggestion has no use for unprotected implementations of ML-KEM and will downgrade them if put in the main standard, allowing this tweak as an alternate standard would improve performances of side channel protected implementations without changing the overall security of the algorithm and its compatibility with the current FIPS 203 draft as it only affects the "failure" part of the Fujisaki-Okamoto Transform.

Best regards,

Pierre-Augustin BERTHET, PhD Student at Hensoldt SAS France and at LTCI, Télécom Paris.

14. Comments from Peter Schwabe, September 14, 2023

Dear NIST PQC team,

At the Oxford PQC summit we had a rather extensive discussion about the modulus check for the public key in Kyber. Let's say that the input byte string encoding the polynomial is PK (ignore the public seed rho here), then we have three options for Encaps:

```
1.) pk = Decode(PK)
   if(ModulusCheck(pk) == fail): abort

2.) pk = Decode(PK)
   Proceed with pk and H(PK)

3.) pk = Decode(PK)
   Proceed with pk and H(Encode(pk))
```

Option 1.) is what the draft standard does.

Option 2.) reduces coefficients of the polynomials modulo q and uses the input byte string PK as input to the Hash. This is pretty much what (hopefully) existing implementations do.

Option 3.) reduces mod q and uses the re-encoded reduced polynomial as input to H. There was solid agreement that option 3.) is a bad idea.

The main argument for option 1.) is the philosophy that you don't compute on inputs that you know to be malformed or "fishy". So, this check would be somewhat in line with, for example, the "point-on-curve"
check during encryption in HPKE.

The main arguments for option 2.) are

* In languages that don't support returning exceptions (like C), you'd have to fail by returning a non-zero value. As some callers will ignore that return value, you'd also have to specify how to fill the return buffers in that case, which makes a full specification and possibly also implementation of option 1.) more complex. Generally, forcing the caller to deal with non-zero return codes also adds complexity for the caller.

* There are many ways for a bad implementation of key generation to generate bad keys (the most obvious example is the all-zero key). Checking just one possible way to screw up is a bit arbitrary.

In particular the first of these two arguments makes us lean towards recommending option 2.).

All the best,
Peter

Sent by Peter Schwabe, summarizing a group discussion at the Oxford PQC Workshop.

15. Comments from Stephan Mueller, September 29, 2023

Dear ladies and gentlemen,

I would like to share the following comments to the FIPS 203 draft from August 2023.

# KDFs

Please allow SP800-56C KDFs to the section 3.3, paragraph "Approved usage of the shared secret key."

The reason is that those KDFs are allowed for key agreement mechanisms and Kyber is intended as a replacement for key agreement (e.g. DH and ECDH).
Considering that existing frameworks using DH and ECDH may be adjusted to use Kyber, it may aid the acceptance of Kyber if the existing KDF implementations compliant to SP800-56C are also approved to be used with Kyber.

Thanks a lot
Stephan

16. Comments from Robin Larrieu, October 20, 2023

Dear NIST,

The PQC-forum group had a discussion about the generation of the public matrix A in ML-KEM that does not match what was done in the Kyber submission.
See the thread "[pqc-forum] Question about Ahat_transposed in ML-KEM / Kyber".
Recall that in ML-KEM, the matrix A is generated in NTT representation by deterministically expanding a pseudo-random string from a seed and the row/column indices. Specifically, the coefficient in row i and
column j of the matrix A, denoted Ahat[i,j] is generated:
- from the output of XOF(rho, i, j) in the FIPS-203 draft
- from the output of XOF(rho, j, i) in the Kyber submission

Notice that the row/column indices in the input of XOF are swapped in
FIPS-203 compared to the Kyber submission. According to Dang, Quynh H.
(Fed) <quynh.dang@nist.gov>, this mismatch is unintentional, and the underlying PKE should have been the same in ML-KEM and Kyber.
- Assuming this is the case, there are a few changes to do in the description of algorithms 12 -- K-PKE.KeyGen() and 13 -- K-PKE.Encrypt() as proposed below.
- If this change was in fact intentional, the difference should be mentioned in section "1.3 Differences From the CRYSTALS-KYBER Submission" (and the rest of this email can be ignored).

# Changes required in algorithm 12 -- K-PKE.KeyGen():

Indices i,j must be swapped in the input of XOF, that is rewrite line 6 as
  Ahat[i,j] <- SampleNTT(XOF(ρ, j, i))
instead of
  Ahat[i,j] <- SampleNTT(XOF(ρ, i, j))

# Changes required in algorithm 13 -- K-PKE.Encrypt()

For correctness of the scheme, generation of matrix Ahat must be done in a compatible way with what is done in KeyGen. Due to the multiplication by Ahat_transpose in line 19, there are several possible presentations that make sense, each with their own advantages and disadvantages. Here are the main possibilities that I can think of

1) Simply swap the indices as in KeyGen, that is
- rewrite line 6 as Ahat[i,j] <- SampleNTT(XOF(ρ, j, i))

2) Sample directly the transpose of Ahat, using either of the two options:

2.a) as in the Kyber specification
- rewrite line 6 as Ahat_transpose[i,j] <- SampleNTT(XOF(ρ, i, j))

2.b) using an auxiliary notation Bhat for Ahat_transpose:
- rewrite the comment in line 4 as "re-generate matrix A in [snip], by
its transpose Bhat=Ahat_transpose"
- rewrite line 6 as: Bhat[i,j] <- SampleNTT(XOF(ρ, i, j))
- rewrite line 19 as: u <- NTT^(-1)(* r) + e1

3) Use left-multiplication in line 19
- rewrite line 6 as Ahat[i,j] <- SampleNTT(XOF(ρ, j, i))
- rewrite line 19 as: u <- NTT^(-1)(r_hat * Ahat) + e1
- rewrite line 21 as: v <- NTT^(-1)(r_hat * t_hat) + e2 + mu

# Advantages and disadvantages of each option (personal opinion)

Option 1) makes it clear that generation of matrix Ahat is done in the same way in KeyGen and Encrypt.
On the other hand, this means implementers must be careful of the transpose in line 19 (u <- NTT^(-1)(Ahat_transpose * r) + e1). Recall that page 10 gives the formula for a multiplication of the form yhat =
Ahat_transpose * uhat, so this is well defined.

Option 2) corresponds to a straightforward implementation trick of generating A directly in its transposed form, as used in the Kyber reference implementation. In particular, option 2.a) matches exactly the Kyber specification, but it may be confusing due to the multiple appearances of Ahat_transpose. Option 2.b) avoids most risk of confusion by removing all matrix transposition in the pseudo-code, at the cost of introducing a new notation.

Option 3) visually echoes what happens in KeyGen and Decrypt which could give a better intuition on how the scheme works as a whole and why it is correct. It also makes it clear that generation of matrix Ahat is done in the same way in KeyGen and Encrypt, while also removing all matrix transposition to avoid confusion. It would however require a more extensive editing work in the rest of the document to make all equations consistent.

I do not have a strong opinion on which option is the best, but if I had to choose one I would recommend option 1.

Best regards,
Robin Larrieu
CryptoNext Security

17. Comments from Stiepan Aurélien Kovac, October 24, 2023

Dear NIST PQC reviewing team,

Following the publication of the FIPS-203 draft and various discussions at the global level in both private, semi-public and public (Internet) settings, it occurs that the security levels allowed on page 33, table 2 of that draft are incoherent with those of CNSA2.0, which mandates 256 bit strength or equivalent (here, the 1024 parameter of ML-KEM/Kyber) min, aka "level V".

Source: https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF .

This is also coherent with NIST IR 8105, which highlights the need for "larger key sizes" in the case of AES, which serves here as the benchmark for asymmetric PQC strength.

Being that CNSA2.0 regards commercial cryptography recommendations for all classification levels (not only TOP S CRET as used to be the case for AES-256), we fail to see any rationale grounded in US law and processes to lower security levels below the new (2022) CNSA minimum of 256 bits or its asymmetric PQC equivalent. On another hand, it is true that "commercial cryptography" under PR China's law means cryptography the (Chinese) state can easily break as opposed to "state cryptography".
Yet even in the remote possibility of the USA following the PRC law's letter on that matter, it should still follow its own national security agency's guidelines as to parameters.

We therefore strongly encourage NIST to remove levels 1 and 3 (I and III in Roman letters) from their set of allowed PQC parameters, and to think about the possibility of introducing higher key sizes as well in the near future, in the spirit of its IR 8105.

Note that our company Quantum Resistant Cryptography (US) offers security levels that start at 256 bits (AES equivalent) and go up to 512-bit both for symmetric and asymmetric (Kyber and classic McEliece, NTRU also supported). A separate submission of the analysis of our enhancements to AES and its modes of operation, in response to the announced revision of the same by NIST, has been made by our company in September.

Best Regards,
Stiepan Aurélien Kovac
CEO, Quantum Resistant Cryptography (US, global)

P.s.: our chairman, quantum physics Professor Davor Pavuna from the EPFL, former US presidential advisor under both US Presidents Bush and Obama, receives us in Cc.

18. Comments from Danny Niu, October 24, 2023

In step 7 of ML-KEM.Decaps, it says:

> K^bar <- J(z||c,32)

Since J is already defined as a fixed-length function, the length specifier 32 is un-needed.

19. Comments from Filippo Valsorda, October 30, 2023

Hello,

The comment on line 6 of Algorithm 14 (K-PKE.Decrypt) reads

▷ NTT^{-1} and NTT invoked *k* times

I believe that is incorrect. NTT is indeed run *k* times on the vector **u**, but NTT^{-1} is run only once on *v′*, which is the lone polynomial result of a dot product of the 1×k matrix $\mathbf{s}^{\intercal}$ and the k×1 matrix NTT(**u**). That's confirmed by the fact that the result of the NTT^{-1} operation is used in a subtraction with *v* to yield *w*, both polynomials.

Cheers,
Filippo

20. Comments from Gustavo Banegas, October 31, 2023

Dear NIST,
Regarding the choice of XOF used in ML-DSA and ML-KEM (Dilithium and Kyber), if we take the example of SPHINCS+, it is possible to implement it with different hash functions. However, this choice seems to have already been firmly decided upon for Kyber and Dilithium.

An alternative could be an XOF based on a symmetric primitive (e.g. ASCON). For security levels beyond 128 bits of security, it's worth exploring the potential of upcoming, yet-to-be-specified variants of ASCON or a different secure XOF besides SHAKE, there were previous discussions concerning the performance of SHAKE/SHA-3.

Additionally, with regard to performance optimization, we would like to advocate for the implementation of randomization in the context of Dilithium. This topic was previously deliberated on May 5th and other occasions this has been discussed into modify the XOF, and we agree with the previous discussion in this topic that it would be beneficial to replace SHAKE to produce pseudo-randomness with an SP 800-90C certified DRBG.

Do you plan to give any comment or modifications regarding this matter?

All the best,
Gustavo

21. Comments from Duke Abbaddon, November 11, 2023

I have read your PQC posts a little and the constant name calling that
goes on in it, However this is not the main issue! the fact that :
"the cost of breaking Kyber512, Kyber768, and Kyber1024, using known
techniques, is higher than the cost of breaking respectively AES128,
AES192, and AES256"

Now in my view if you are creating a certificate you need to stick
with the 2048Bit Public 256Bit RSA Standards at a minimum; In the case
of Dilithium that is a truth that we need,

Kyber being set like ECC AES as a random key with refresh & not the
verifiable main certificate?

Most probably sound!

Kyber at AES Standard 2048Bit with public key at 384Bit, no problem..

In these terms Kyber & Dilithium both make profoundly good certificates!

Small key AES 128 Bit keys are for ECC Elliptic curves & not verification...

Personally i hold with these principles regarding Kyber & Dilithium :
Certificate Key 2048Bit Public 256Bit/384Bit : RS

22. Comments from P. Kampanakis, R. Chapman, J. Massimo, November 11, 2023

Dear Dustin, NIST PQ Project team,

We appreciate NIST's leadership in standardizing schemes that the whole world can use. We know the process is tough and can sometimes be heavily scrutinized. The new schemes will be with us for years, and we can understand the impact this whole project led by NIST will have.

We would like to provide feedback on the ML-KEM FIPS-203 draft:

First, we want to bring up the security level mappings to ML-KEM parameters. It is unclear how the ML-KEM parameters are mapped to the MAXDEPTH levels. For example, we are not sure how to reach to that ML-KEM-768 maps to about "*2^221/MAXDEPTH quantum gates or 2^207 classical gates* ". This could be done in Section 7 or Appendix A, but we think you want to reference section 5.2 of the Kyber Round 3 submission which quantifies the gate count about Kyber512. A better analysis of Kyber-768 and Kyber-1024 should also be provided how they can be mapped in the security levels potentially with the Kyber team's help, as the Kyber Round 3 submission only focused on Kyber512.

We want to suggest addressing the keypair vs pseudorandom value storage issue. This has come up previously in other fora. The encapsulation and decapsulation keys can be reconstructed by storing the two 32 random byte values (d and z). In some cases parties may prefer to store these only and reconstruct the keypair when they need it. Using these values of 64 bytes instead of a few kilobytes for (ek,dk) could be briefly discussed as an option for the keypair owner in Section 6.1. SP800-227 could be another document where this could be discussed too.

Mandating the KeyGen and Encaps functions to be randomized, i.e., generate the key pair and produce the shared secret key internally in a randomized fashion, makes implementing other crypto primitives that use the KEM as well as known answer tests for the KEM challenging in some cases. PQ-Crystals implementation already implements derandomized versions of KeyGen and Encaps. It would be good to address this issue in the standard.

Additionally, a few technical comments and minor suggestions

- Line 268: We suggest to add a reference to NIST SP 800-227 when it is available.
- Line 409 / Section 2.3: It would help a reader if these definitions appeared in declaration-before-use order. For example, on lines 416 and 417 (the definition of f-hat), we find references to Rq and Tq, but these are not defined until later on lines 440 and 448 respectively.
- Line 458: Typo in "*copie*". It should be "copy".
- Line 675 - 680: Approved usage of the shared secret key. KEMs differ from currently used key exchange mechanisms in that encapsulation creates a shared secret key, rather than a shared secret from which a key can be derived. As such, more clarity should be

added to the approved usage of the shared secret key. At first read this paragraph was hard to parse. In particular the sentence “When key derivation is needed, the final symmetric key(s) shall be derived...” - may be misleading, perhaps changing the “When key derivation..” to “If further key derivation”, to make it seem less like it is inevitable, and instead optional. It may also be useful to point out to readers the limitations of the shared secret key produced (via. a reference to other NIST documents on this), or to note that additional entropy can be mixed in by using an approved KDF (in the case of constrained devices with shared entropy pools).

- Line 699: “Destruction of intermediate values”. This could be more specific. All intermediate values or only those that can be used in an attack? The way I read the paragraph is that we should destroy only the values that can be used in an attack, but then the standard should specify these values.
- Line 762-763, in the Definition (4.5) of the Compress_d function. It appears that this definition implicitly requires a “modulo 2**d” operator to be applied after the “round to nearest” function is applied. Consider the case where d = 10. If we evaluate using pure rational arithmetic, then Compress10(3328) = 1024, which cannot be right since we require an answer in 0 .. 1023. If there is an implicit “modulo 1024” on the end, then are we required to return Compress10(3328) = 0? Perhaps the “round to nearest” operator is wrong, but should be “floor” so that Compress10(3328) = 1023? Which is right? Either way, this should be clarified.
- Lines 803-804. In the specification of Algorithm 7 (SamplePolyCBD), on lines 3 and 4. Is the Summation (Sigma) operator intended to sum bits as pure unbounded integers, or as bits modulo 2? This is not clear. Put another way - what types should we infer for the local variables x and y?
- Line 905 (Algorithm 12) and line 923 (Algorithm 13) - We note that the order of the actual parameters in to call to XOF() has changed from the Kyber Round 3 specification. Specifically, line 6 of both algorithms has XOF(rho, i,j) in FIPS-203, while it was XOF(rho, j, i) in Kyber. This change is not mentioned in Section 1.3 either. Is this change a mistake, or is there an omission from section 1.3?
- Line 923 - the specification of Algorithm 13 (K-PKE.Encrypt). Line 2 does not strictly type-check. With k = 3, line 2 passes 1152 bytes to ByteDecode12, but the specification of ByteDecode12 clearly states that its input is a fixed array of 384 bytes. I assume the intent here is to split the 1152 bytes of input into K * 384 byte arrays, apply ByteDecode12 three times, and produce a vector of 3 polynomials as a result. Is that correct? This could be clarified.
- Line 984. Similarly, the application of ByteDecode12(EK_Tilde) cannot be correct, since EK_Tilde is 1184 bytes when K = 3, which is not an integer multiple of 384. Perhaps ByteDecode12(EK_Tilde[0:384k]) is intended (as in line 2 of Algorithm 13) so that the final 32 bytes are eliminated?
- Line 1038, Algorithm 17. Line 7. We note that K__Bar is always computed even though the value may not be needed. We assume this is in order to achieve implicit rejection. A

well-meaning implementor might be tempted to move that assignment inside the "if" statement on line 10 in order to improve efficiency. An explicit note here to forbid that optimization might be wise. If both branches of this code are intended to be executed in constant-time, them this should also be stated, or the "if" statement should be replaced with a conditional move or swap procedure.

- Line 1080: Given that ML-KEM-1024 is still very efficient, much more than ECDHE with P521, you could use the size as an adverse effect example instead of "*e.g., the algorithm may be unacceptably slow* ".
- Line 1213: The text says "*(the approximate number of gates that presently envisioned quantum computing architectures are expected to serially perform in a year)*". Are there any references to substantiate that which could be added here?

It is probably in NIST's plans already, but it is essential to timely adoption that we can FIPS certify implementations shortly after standardization, and this will require an updated FIPS 140-3 Implementation Guidance (IG) document. We want to reiterate that the IG will need to be updated to include mentions of FIPS 203, 4, 5. There are a few references already of Stateful HBS (SP 800-208) in the IG. And SPs like SP 800-56C or FIPS 186-5 will probably be updated to also refer to FIPS 203, 4, 5. But the IG has mentions of elliptic curves algorithms, and signatures and LMS/XMSS used in protocols like TLS of self-test. The IG needs to be updated to properly refer to the new algorithms for the relevant uses and also to clarify that the hybrid key exchange is approved as per SP 800-56C (Section 2) in contexts like TLS or SSH.

Regards,
Panos Kampanakis, Rod Chapman, Jake Massimo
Amazon Web Services (AWS)

23. Comments from D. J. Bernstein, November 16, 2023

1. I recommend adding the following general requirement: "Any upgrade from pre-quantum encryption to this KEM shall retain the pre-quantum encryption and add this KEM as an extra layer of defense, rather than removing the pre-quantum encryption."

This produces a negligible increase in marketwide TCO; is in line with statements by ANSI, ANSSI, and BSI; and is in line with industry practice by Google, Cloudflare, and OpenSSH. Systematic use of hybrids
is the only reason that user sessions encrypted by CECPQ2b were not exposed to today's attackers by fast attacks that are now publicly known.

2. There have been more and more advances in algorithms to break lattice systems. As an illustration of the magnitude of the advances, lattice dimension 512 is threatened by attacks known today, whereas a 2010 paper by Lindner and Peikert stated that lattice dimension 256 would achieve "security levels matching those of AES-128". I am skeptical about the long-term security of lattice dimension 1024.

Beyond general lattice risks, Kyber incurs further risks from its use of cyclotomic lattices. Attacks against structured lattices, especially cyclotomic lattices, have broken through one claimed barrier after another. See https://ntruprime.cr.yp.to/latticerisks-20211031.pdf Section 1, for a survey and references through 2021. Newer speedups include, e.g., my paper https://cr.yp.to/papers.html#abeliannorms.

The risk of today's attackers already being able to break Kyber-512 is high enough to justify classifying standardization of Kyber-512 as a substantial danger to public safety. Adding a "use only in emergencies" warning label does not address the danger: it is much more common for the public to simply ask what has been standardized than to check for warnings, recommendations, etc.

NIST claimed in pqc-forum email dated 30 Nov 2022 12:25:47 +0000 that Kyber-512 is "unlikely" to be easier to break than AES-128 by known attacks. This claim has three basic problems.

First, the claim comes directly from a calculation error by NIST, namely multiplying the "real cost of memory access" by total costs in "the RAM model" rather than just by the number of memory accesses. See https://blog.cr.yp.to/20231003-countcorrectly.html and https://blog.cr.yp.to/20231023-clumping.html. There are legitimate uncertainties regarding the underlying numbers for lattice attacks, but multiplying the numbers that NIST multiplied is wrong in any case.

Second, NIST's announced evaluation criteria specified AES-128 as a "floor" for the security levels in the NIST Post-Quantum Cryptography Standardization Project. It's not good enough to merely say that a cryptosystem is "unlikely" to be below the floor.

Third, focusing on known attacks is dangerous. One also has to account for the risks of better attacks. From this perspective, it is deeply concerning to see the levels of complication and instability in the
recent literature on lattice attacks.

I recommend terminating standardization of Kyber-512.

3. For reasons explained in my pqc-forum messages dated 8 Nov 2023 14:58:29 +0100 and 9 Nov 2023 17:02:54 +0100, I recommend standardizing the 2020 version of Kyber (for whichever parameter sets NIST ends up standardizing), rather than NIST's new version of Kyber.

This recommendation includes undoing NIST's accidental changes such as
(i,j) vs. (j,i); undoing NIST's changes to the FO layer; undoing NIST's "validation" changes; putting back the RNG hashing; and making sure to use SHA-3/SHAKE in exactly the same way as the 2020 submission.

I recommend, before releasing the final specification, making sure to reimplement Kyber from the specification and checking that the resulting software produces the same test vectors as the 2020 submission.

Under no circumstances should NIST use any of the changes publicly proposed by NSA in October 2023. NIST should also immediately release the records of all private input from NSA to NIST, and should allow time for the public to review and comment on that input.

---D. J. Bernstein

24. Comments from Graham Costa, November 16, 2023

Hi,

Thank you for the opportunity to review this standard. Our team has reviewed the standard from the perspective of a module developer who regularly submits modules to CMVP for validation. With that in mind, our main focus is on how requirements and statements in the standard may be interpreted by developers, test labs and certifiers.

Comment 1:

Line 678-680: We would strongly propose the inclusion of SP 800-56Cr2 as an alternative key derivation scheme. As ML-KEM will be a replacement for existing Key Establishment schemes defined in SP800-56Ar3 and SP 800-56Br2, modules implementing this scheme will currently support key derivation using SP 800-56Cr2. To minimize changes to modules, we'd strongly propose that this existing NIST standard for expanding key material is included to allow minimal changes to existing implementations as part of adopting ML-KEM.

Our preferred fix is to add an allowance to use SP 800-56Cr2 KDF on the output of ML-KEM in all circumstances. If there is a legitimate concern with adding this allowance an alternative proposal would be to add support for the use of SP 800-56Cr2 alongside SP 800-108 in situations where keys output from ML-KEM are ephemeral and short-lived (i.e. when used as part of comment IETF protocols such as IPSec, TLS and MACsec).

Comment 2:

Line 685-687: To avoid potential confusion with available options to an 'Approved RBG' we'd propose explicitly stating here that "random bytes shall be generated using output from an approved RBG using one of the architectures defined in SP 800-90C." That standard in turn references SP 800-90A and SP 800-90B which don't need to be included in this standard.

There is potential for confusion by referencing all three standards. In particular, whether outputs from an SP 800-90B noise source could be used without a DRBG and/or whether a platform supporting an approved DRBG must also include its own noise source in addition to a DRBG. SP 800-90C makes it clear that all architectures require the use of a DRBG and separately makes it clear that DRBG may be seeded using external noise sources.

Comment 3:

Without access to SP 800-227 (planned for 2024) it is difficult to assess if in combination both standards contain sufficient information to avoid common security problems when deploying a KEM as part of a communication protocol. In particular, in isolation guidance provided in ML-KEM is not sufficient to mitigate the risk of man-in-the-middle attacks during the exchange between Alice and Bob set out in section 3.1.

In the absence of SP 800-227 and noting the companies will start to use ML-KEM as soon as the standard is complete, we'd suggest explicitly adding a warning to this standard that should Alice not be able to authenticate Bob using a post quantum-safe method, it is possible for an adversary (i.e. Eve) to man in the middle the exchange (i.e. Alice establishes a shared secret with Eve and separately Bob establishes a shared secret with Eve).

This is outlined in equivalent standards such as SP 800-56Ar3 where detail is added to the standard to make it clear that public keys need to come from a trusted source.

Should you have any further questions, please do not hesitate to contact us.

Graham COSTA (he/him)
Security and Certifications Manager
Digital Identity and Security
Thales

25. Comments from Beat Heeb, November 17, 2023

Minor Issue in equation (4.7) on line 771:
The left side of the inequality should be the absolute value of the given term.

Regards
Beat Heeb
Oberon microsystems

26. Comments from Martin Strand, November 19, 2023

Dear NIST -- thanks for all the hard work you're doing.

This almost feels petty, but here goes: Line 20 in Algorithm 13 appears to have a superfluous closing parenthesis.

Regards,

Martin Strand

27. Comments from John Preuß Mattsson, November 20, 2023

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents. Please find attached our comments on FIPS 203, 204, and 205.

Best Regards,
John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

Comment attached.

ERICSSON

Date: November 20, 2023

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on the draft versions of FIPS 203 (ML-KEM), FIPS 204 (ML-DSA), and FIPS 205 (SLH-DSA)

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents. Please find below our comments on FIPS 203 (Draft), FIPS 204 (Draft), and FIPS 205 (Draft).

General comments on all three draft specifications:

– *"secure even against adversaries who possess a quantum computer"*
  "including after the advent of quantum computers"
  "If large-scale quantum computers are realized"
  "resistance to attacks from a large-scale quantum computer"
  "adversaries in possession of a large-scale quantum computer"

  A lot of adversaries already have small, error prone, and currently quite useless quantum computers. "Large-scale" is better but is also not a good term as in addition to being large, the quantum computer must have a sufficiently low error rate to be relevant. We suggest that NIST uses the established and excellent term Cryptographically (or Cryptanalytically) Relevant Quantum Computer (CRQC). This aligns with CNSA 2.0 [1].

– "A number associated with the security strength of a post-quantum cryptographic algorithm"

  We approve NIST sparingly using the term "post-quantum" at all in the draft standards. It is a quite bad term as quantum-resistant algorithms need to be deployed before the advent of CRQCs. We suggest that NIST removes the last few "post-quantum" and replace them with the established and excellent term "quantum-resistant". This aligns with CNSA 2.0 [1].

– "Key search on block cipher with 128-bit key"

  Attacking any sort of 128-bit symmetric cryptography (block cipher or not) requires a drastic number of computational resources comparable to attacking AES-128. As concluded in [2–3], at least cubic (n^3) or quartic (n^4) speedups are required for a practical quantum advantage. The viability of quantum advantage with cubic speedups is still ambiguous [3]. Algorithms with quadratic (n^2) speedup like Grover's algorithm (which is proven to be optimal) will not provide any practical quantum advantage for breaking symmetric cryptography or any other problems.

NIST will soon standardize Ascon-128 [4] which is quantum-resistant but not a block cipher. Stream ciphers like SNOW 3G and sponge-based key derivation and authentication functions like TUAK with 128-bit keys used for protection in 4G and 5G mobile networks are also quantum-resistant. In fact, we would expect that key search on SNOW 3G requires more gates than attacking AES-128. But the exact gate counts are not very important practically. While SNOW 3G and TUAK are not NIST algorithms, we think NIST has an important role in educating the general public that also these types of algorithms are and will remain quantum-resistant.

We suggest that NIST changes the definition of security category 1 to "Key search on cipher with 128-bit key". This aligns with the statement from UK NCSC [5]:

"the security of symmetric cryptography is not significantly impacted by quantum computers, and existing symmetric algorithms with at least 128-bit keys (such as AES) can continue to be used. The security of hash functions such as SHA-256 is also not significantly affected, and secure hash functions can also continue to be used."

We also suggest that NIST provides a statement in some SP or FIPS document estimating for how many decades security category 1 and 128-bit symmetric cryptography will be allowed. The current text in SP 800-57 just states that security strengths 128, 192, and 256 are acceptable beyond 2030. When RSA-1024 was disallowed in 2010 it could almost be broken by the world's fastest supercomputer [6–7]. When RSA-2048 is disallowed in 2030 it is expected to take another 30 years until it can be broken by a supercomputer [6–7]. Using similar security margins (0–30 years), security category 1 should be allowed until 2060–2090. A suggested very conservative statement would be "security category 1 can be used at least until 2060". That would give some needed guidance for industry, but also enable NIST to allow security category 1 to be used longer if Moore's law slows down as many people predict.

– We think it is excellent that ML-KEM and ML-DSA only use SHA-3/Keccak. As stated by Mike Hamburg "SHAKE has a more appropriate interface, comparable or better performance, and is easier to make side-channel resistant" [8]. Hash functions should be designed to provide indifferentiability from a random oracle [9]. Looking at hash performance figures for small output strings it is easy to think that SHA-2 is slightly faster on some platforms, but this is often completely negated by the fact that to use SHA-2 in a secure you need a lot of complex and heavy constructions only designed to overcome the severe shortcomings of SHA-2 such as HMAC, HKDF, MGF1, HASH_DRBG, some function to get short output (NIST defines several ways for SHA-2), and often a mix of SHA-256 and SHA-512. Doing anything secure with SHA-2 is very complex. SHA-2 is not robust.

– We strongly think NIST should produce negative test vectors for all algorithms. Negative test vectors are very important for catching bugs that might have security implications. We think all future algorithm and protocols standards should be accompanied with negative test vectors. It is often claimed that security agencies participate in standardization and production of cryptographic modules with the explicit goal of sabotaging security to enhance their surveillance capabilities. Taking a strong stance on finding security threatening implementation bugs would increase the trust in NIST as a global SDO for cryptography. Two functions that require negative test vectors are ML-KEM.Encaps and ML-KEM.Decaps. FIPS validation shall not be achievable without input validation.

– *"At present, ML-KEM is believed to be secure even against adversaries who possess a quantum computer."*
  "ML-DSA is believed to be secure even against adversaries in possession of a large-scale quantum computer."
  "SLH-DSA is expected to provide resistance to attacks from a large-scale quantum computer."
  "ML-DSA is designed to be strongly existentially unforgeable"

  We suggest removing "At present", which is not suitable for a document that will live for many years. The terms believed to, expected to, and designed to are all used when talking about security properties. We suggest removing "believed". A huge amount cryptanalytic effort targeting lattice-based cryptography has been done before and during the NIST PQC project and US government is planning to protect all national security systems using lattice-based cryptography. Maybe only use the term "designed to", which seems to be the most common in FIPS 203—205 and other NIST specifications.

– "1.3 Differences From the … Submission"

  These sections are probably better suited as appendixes in the final standards.

Comments on FIPS 203 (Draft):

– We strongly disagree with suggestions that NIST should remove ML-KEM-512 based on a heavily contested claim regarding the gate count in one theoretical memory model [10]. We agree with NIST that the cost of breaking ML-KEM-512 is higher than the cost to break AES-128. We think that it is excellent that NIST has specified ML-KEM-512. If ML-KEM-512 is slightly above or below the theoretical security level of AES-128 in one theoretical model is practically completely irrelevant. The important thing practically is that ML-KEM-512 is approximately as hard to break as AES-128. We believe ML-KEM-512 offer a significant security margin for many applications, especially if used in hybrid mode with Curve25519. As stated by UK NCSC [5], ML-KEM-512 provides an acceptable level of security for personal, enterprise, and government information.

  The maximum transmission unit (MTU) on the Internet is typically just around 1300 bytes. The encapsulation key and ciphertext are 800 and 768 bytes in ML-KEM-512 versus 1184 and 1088 bytes in ML-KEM-768. The size difference means that when using ML-KEM-512, a lot more additional information can be sent in the same packet. This significantly reduces latency, which is very important in many applications. We believe that the availability of ML-KEM-512 will increase the adoption rate of quantum-resistant cryptography. We believe most implementations will support all of the security levels so applications should be able to change the security level quickly if needed.

- We are strongly against replacing SHA-3/Keccak in ML-KEM with SHA-2 as suggested in [8]. Such a big change would risk introducing various kinds of security problems, decrease trust in ML-KEM and NIST, lower performance on most/all platforms, and significantly delay deployment of ML-KEM. Using Keccak should not be a problem for organizations that have invested in cryptographic agility.

- "makes the encapsulation key available to Party B. Party B then uses Party A's encapsulation key to generate one copy of a shared secret key along with an associated ciphertext. Party B then sends the ciphertext to Party A over the same channel"

  It does not have to be the same channel. It is quite common that a different channel is used.

- "used by two parties to establish a shared secret key over a public channel"
  "A shared secret key is computed jointly by two parties (e.g., Party A and Party B)"
  "randomness used by the two parties"

  ML-KEM is also very useful for quantum-resistant protection of data at rest using for example Hybrid Public Key Encryption (HPKE) [11]. In such use cases the party encapsulating and decapsulation may be one and the same, i.e., there is only one party. We think this should be mentioned in the specification.

- "As a result, ML-KEM is believed to satisfy so-called IND-CCA security"

  We think it should be described that the encapsulation key can be used several times. That this follows from the IND-CCA security is likely not obvious to most readers. We think this information should be mentioned in FIPS 203 and not just in the future SP 800-227.

- We think it would be good if FIPS 204 also discusses additional security properties. E.g., is ML-KEM believed to have key commitment or not? How does reusing the encapsulation key affect the security bounds. Can anything be said about multi-key security?

- "The scheme K-PKE is not sufficiently secure"

  We suggest that NIST explains is some detail why NIST believes that K-PKE is not sufficiently secure.

Comments on FIPS 204 (Draft):

- Very good that NIST embraced the suggestion to include hedged signatures [12] and made it the default mode. We believe that making this mode the default will increase the practical security in deployed systems.

- Rejection sampling is new to most users of digital signatures. FIPS 203 has an excellent table showing decapsulation failure rate. We think FIPS 204 should have a similar table showing the probability for one or more rejections in the signing algorithm. This is not trivial for most readers to calculate. Users will want to know if the variable signing time is something they need to care about or if the probabilities are so low that the variable signing time can be ignored.

– "ML-DSA is designed to be strongly existentially unforgeable under chosen message attack"

  We suggest also adding the abbreviation SUF-CMA, i.e., "ML-DSA is designed to be strongly existentially unforgeable under chosen message attack (SUF-CMA)". This allows the reader to search for "SUF-CMA" or "CMA".

– "ML-DSA is also designed to satisfy additional security properties beyond unforgeability, which are described in [6]"

  We suggest that FIPS 204 lists the additional security properties that ML-DSA is designed to satisfy. The current text does not say if ML-DSA satisfies all or a subset of the properties, and the paper [13] analyzes a non-standardized version of ML-DSA. It is not clear to most readers if the analysis is still valid for ML-DSA.

Comments on FIPS 205 (Draft):

– *"The 12 parameter sets included in Table 1 were designed to meet certain security strength categories defined by NIST in its original Call for Proposals [21] with respect to existential unforgeability under chosen message attack (EUF-CMA)".*

  We think this is a good selection of parameters. Sections 10.1, 10.2, and 10.3 provide a clear illustration of how much easier it is to work with SHAKE instead of SHA2. The SHA-2 versions are downright inelegant and complexity like this often leads to specification and implementation bugs. We think NIST should also mention if SLH-DSA is (believed to be) SUF-CMA or not. i.e., given a number of different message-signature pairs (m_i, σ) can an attacker create a new signature (m_i, σ') for an already signed message m_i.

– For ML-DSA, hedged signatures are the default, the value rnd should be generated by an approved RBG, and the deterministic mode should not be used on platforms where side-channel attacks are a concern. For SLH-DSA, hedged signatures are not the default, opt_rand does not require use of an approved RBG, and for devices that are vulnerable to side-channel attacks opt_rand may be set to a random value. We suggest that NIST aligns the SLH-DSA specification with use the stronger ML-DSA requirements alternatively explain why it is acceptable for SLH-DSA to have much weaker requirements.

– We think FIPS 205 should describe how the security depends on the number of times the SLH-DSA private key is used to generate signatures. We think it would be helpful for the reader to understand if there are no practical limits for the number of signatures that can be generated or if systems producing a very large number of signatures should change the SLH-DSA private key periodically to keep a high security level. In ECDSA the collision probability can be ignored while AES-256-GCM with r random IVs only provide $\approx 97 - \log_2 r$ bits of security due to the collision probability [14].

- "finding such a collision would be expected to require fewer computational resources than specified for the parameter sets' claimed security levels in all cases except SLH-DSA-SHA2-128f and SLH-DSA-SHAKE-128f."

  We suggest that FIPS 205 describes how much fewer resources would be needed. An application might require different security levels for different properties. It would also be good if NIST stated that it is unknown if SLH-DSA provides the properties exclusive ownership and non re-signability [13].

- "Don't support component use."
  "cryptographic modules should not make interfaces to these components available to applications"

  We would suggest that this is softened or rewritten. That some hardware implementations do not support important building blocks like the AES round function and the KECCAK-p permutation has turned out to be very limiting for innovation, significantly decreasing performance (or security) of future standards like ML-KEM and meaning that the acceleration cannot be used for algorithms like AEGIS [15], Rocca-S [16], Snow 5G [17], and Simpira [18] that make use the AES round function. We think it is very important that many types of hardware implementations do support component use to enable future innovation and standards. In general, we strongly think that NIST should encourage hardware implementations such as CPUs to have flexible APIs supporting component use.

Best Regards,
John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

[1] NSA, "Announcing the Commercial National Security Algorithm Suite 2.0"
https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF

[2] Hoefler, Häner, Troyer, "Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage"
https://cacm.acm.org/magazines/2023/5/272276-disentangling-hype-from-practicality-on-realistically-achieving-quantum-advantage/fulltext

[3] Babbush, McClean, Newman, Gidney, Boixo , Neven, "Focus beyond Quadratic Speedups for Error-Corrected Quantum Advantage"
https://arxiv.org/pdf/2011.04149.pdf

[4] NIST, "NIST Selects 'Lightweight Cryptography' Algorithms to Protect Small Devices"
https://www.nist.gov/news-events/news/2023/02/nist-selects-lightweight-cryptography-algorithms-protect-small-devices

[5] UK NCSC, "Next steps in preparing for post-quantum cryptography"
https://www.ncsc.gov.uk/whitepaper/next-steps-preparing-for-post-quantum-cryptography

[6] CRYPTEC, "Cryptographic Technology Evaluation Committee Activity Report"
https://www.cryptrec.go.jp/symposium/2023_cryptrec-eval.pdf

[7] CRYPTEC, "Japan CRYPTREC Activities on PQC"
https://events.btq.li/Japan_CRYPTREC_Activities_on_PQC_Shiho_Moriai.pdf

[8] NIST PQC Forum, "Comments on FIPS 203/204"
https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/SPTpYEP7vRg

[9] Maurer, Renner, Holenstein, "Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology"
https://eprint.iacr.org/2003/161

[10] NIST PQC Forum, "Kyber security level?"
https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/W2VOzy0wz_E

[11] IETF RFC 9180, "Hybrid Public Key Encryption"
https://www.rfc-editor.org/rfc/rfc9180.html

[12] Preuß Mattsson, "OFFICIAL COMMENT: CRYSTALS-Dilithium"
https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/1mQjngj_2Po/m/-p4RKXGQAwAJ

[13] Cremers, Düzlü, Fiedler, Fischlin, Janson, "BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures"
https://eprint.iacr.org/2020/1525.pdf

[14] Preuß Mattsson, Smeets, Thormarker, “Proposals for Standardization of Encryption Schemes“
https://csrc.nist.gov/csrc/media/Events/2023/third-workshop-on-block-cipher-modes-of-operation/documents/accepted-papers/Proposals%20for%20Standardization%20of%20Encryption%20Schemes%20Final.pdf

[15] IRTF, “The AEGIS Family of Authenticated Encryption Algorithms“
https://datatracker.ietf.org/doc/draft-irtf-cfrg-aegis-aead/

[16] IRTF, “Encryption algorithm Rocca-S“
https://datatracker.ietf.org/doc/draft-nakano-rocca-s/

[17] Ekdahl, Johansson, Maximov, Yang, “SNOW-Vi: an extreme performance variant of SNOW-V for lower grade CPUs“
https://eprint.iacr.org/2021/236

[18] Gueron, Mouha, “Simpira v2: A Family of Efficient Permutations Using the AES Round Function“
https://eprint.iacr.org/2016/122

28. Comments from the National Security Agency, November 20, 2023

As part of our review process, NSA Cybersecurity had a team programming up the NIST draft standards, including software engineers who were chosen specifically as non-experts. The idea was that experts would likely implement everything correctly, but we wanted to make sure average software engineers could also follow the standards. This process led to some mis-implementations and potential security issues. Many of the comments below suggest clarifications and minor tweaks to the draft standards document to hopefully make the standards easier to securely and correctly implement and to avoid some of the issues we saw during our study. We also include comments which would help increase usability and adoption rate for Federal systems. Note that some of these issues have already been addressed in the PQC forum since NIST released the drafts on August 24, 2023, but we still include them here for completeness. Comments which relate to both FIPS.203 and FIPS.204 will be included in both places.

The following are directly related to issues our prototypers had in their experiments, where we believe clarification could help prevent mis-implementation.

- **One issue some of our implementers had was which mod to use.**
  - In general, there was some confusion amongst implementers of **mod± versus mod**, especially in parts of FIPS.204 which don't specify which to use. Suggest making this more explicit in FIPS.204 and removing mod± in FIPS.203 since it is used only one place (4.7) where an interval would have worked instead. Having two separate mod functions ended up leading to a lot of errors/problems in our tests.
- **Bits versus bytes was perhaps the most consistent source of initial errors in our study.**
  - **Bit Strings/padding** – we had multiple implementers who ended up writing out a bit string as a byte string where each byte was padded to 8 bits. We suggest adding some wordage of bit strings versus byte strings and to clarify that bits are in adjacent positions without padding. This could be in the definition section of FIPS.204 and ideally should be the same in FIPS.203.
  - **Hash input** - In FIPS.203 4.1, the input is in bytes, but the referenced standards are in bits. As a result, all SHAKE inputs if given to a FIPS-validated module may give a correctly formatted output that is not the intended output. In addition, in section 2.1 of FIPS.203, hash inputs are specified to be bits. Suggest clarifying this, increasing the constant 64 to 512 in Equation 4.1 to correct this, and correct lines 730-731 accordingly. Similarly in Equation 4.3 replacing 32 with 512.
- **The final common type of error we came across related to Pass-by-reference**
  - FIPS.203 says to assume no pass-by-reference in the pseudo-code (FIPS.204 doesn't specify) but in practice pass-by-reference will be used in real implementations. We suggest either repeating this where it matters to remind readers or tweaking pseudo-code to be safe with either method, for example with explicit copies. As an example of the danger, the BytetoBits() function in

FIPS.203 will zeroize the encryption key if you use pass-by-reference with the pseudo code.

o

The following are comments which we felt were important but didn't fall into the previous category.

- **Harmonization of FIPS.203 and FIPS.204.** By design, the background information and many functions of Kyber and Dilithium are mathematically the same. It would help developers if the standards lined up more closely. Some possible changes/tweaks:
    - At minimum, the level 1,3,5 parameter names should have the same naming conventions ; using the security target for naming would be one way to simplify rather than ML-DSA-87 vs ML-KEM-1024.
    - The organization of FIPS.203 and FIPS.204 could be closer – for example putting together Auxiliary functions in the same portions of the documents.
    - Although the NTT functions are slightly different, most of the mathematical discussion and background information could be the same in both FIPS.203 and FIPS.204.
    - Some functions like ExpandA in FIPS.204 and SampleNTT in FIPS.203 are doing the exact same thing (they are both making the matrix Ahat in KeyGen) but the pseudocode is very different, with it broken up into multiple functions in FIPS.204. A lot of the bit/byte-manipulation and sampling functions could be the same or very similar in both standards.
- **Hash choices.**
    - In the ML-KEM specification, the cryptographic functions are defined in an early section (4.1 Cryptographic Functions) which makes the different instantiations of hashes easier to keep track of than in ML-DSA. We strongly suggest ML-DSA adopt a similar section and the language be consistent between the two. As such, the remainder of these comments apply to both ML-DSA and ML-KEM.
    - In ML-KEM section 4.1, the function XOF is defined as SHAKE and comments suggest that one can generate bits as needed by "squeezing" the sponge function to output more bits. While mathematically this is true, it is not quite consistent as written with FIPS 202 which defines the SHA3 and SHAKE functions. In section 6.2 of FIPS 202, SHAKE is explicitly defined as taking a digest length d in addition to a message m, and so FIPS implementations cannot necessarily be used in the way described. In the appendix A.2 they make it clear that SHAKE(m,d) and SHAKE(m,d+e) will share the same first d bits of output, the property being used in ML-KEM. We offer two options to fix this:
        - a) Keep SHAKE as the XOF. One can select a very large d to call for SHAKE and if more than d bits of random are needed, have the algorithm return failure. An appendix can then note that in practice for the sake of efficiency fewer bits can be called and more can be squeezed out as needed. Alternatively, have verbiage to the effect that if you need e additional bits, call SHAKE(m,d+e) and discard the first d bits.

    - b) If one wants to keep the functionality of generating bits as you need them, that is much closer to the description of the 800-90B standards for DRBG. Given that ML-KEM relies heavily elsewhere on secure hashes, we would suggest defining Hash_DRBG as the XOF used in ML-KEM/ML-DSA.
  - It can be impractical to call multiple hash functions when one is taking advantage of previously-validated FIPS modules because frequently modules only implement a subset of the FIPS they are tested against. Hence, we suggest only one hash primitive be used. If SHAKE continues to be chosen for the XOF then it should be used for the hash as well. If Hash_DRBG is used, then the hash primitive chosen for Hash_DRBG should be the same hash primitive chosen to construct all of PRFs and hashes in the standard.
  - Further, we suggest that the hash function being invoked in the above discussion be SHA2 with a size appropriate to the security level. The reasons to select SHA2 are manifold:
    - SHA2 family algorithms are ubiquitous in the federal space and SHA-384/SHA-512 are the algorithms used for National Security Systems as specified in CNSA. Because SHA2 is used so often in low-level parts of the computing trust infrastructure, from today's commercial and government signing infrastructure, to HMAC, to Hash_DRBG, nearly any FIPS-compliant device in use by the government that utilizes hardware acceleration for cryptography will need to have SHA2 present for the next decade. For example, Federal PKIs (including the DoD PKI) have transitioned to SHA2 and so FIPS-validated SHA2 modules are on all PKI-enabled devices in the federal government as well as any device being purchased in the medium term. SHA3/SHAKE, on the other hand, has very limited uptake in this space.
    - Due to this ubiquity, SHA2 will generally not require additional hardware if one is upgrading a system to ML-KEM/ML-DSA while the use of SHA3 may. On many near-term systems, SHA3 and SHAKE require more hardware space than SHA2 because it is an add-on.
    - If the module is implemented in software, SHA2 is substantially faster than SHA3 or SHAKE.

It should be noted that we believe that hash functions are well enough understood and that SHA2 is ubiquitous enough, that a change from SHA3 to SHA2 would be a minor change. Our non-expert implementers were able to swap hash functions in less than an hour with a few lines of code and we have already seen in SPHINCS+ that the community has high confidence in both SHA2 and SHA3 for PQ solutions. For why we think this overall comment is important to make for faster federal adoption, we would like to provide some context for how dominant SHA2 is over SHA3 when it comes to market penetration in the government sector. We consider as a proxy the number of cryptographic hardware products that have active NIST Cryptographic Module Validation Program (CMVP) validation credentials in the

CMVP public database. This hardware represents the acceleration available to federal agencies as they try to adopt FIPS 203 and 204 over the next several years.

In a search performed on 10/13/23, there were 429 hardware modules implementing FIPS 180-4 (SHA2) while 34 have FIPS 202 (SHA3 or SHAKE). This order-of-magnitude difference is not solely the result of the longer life of FIPS 180-4: if we restrict to hardware modules validated in the last calendar year, 10 have FIPS 202 while 102 contain FIPS 180-4. Further, of those 10, only a single one of the hardware modules containing FIPS 202 actually had a FIPS-validated SHAKE. This dominance also extends to digital signatures. A similar search of the Cryptographic Algorithm Validation Program (CAVP) database shows that in 2023 there are 284 CAVP validated implementations of FIPS 186 signature schemes using SHA2, 2 using SHAKE, and none using SHA3.

The ubiquity of SHA2 applies to internet protocols as well. For example, most TLS, IPSec, and SSH implementations will make use of SHA2 to generate session keys. Further, these protocols often use SHA2 as part of the integrity function alongside a non-AEAD cipher. For either use case, the supporting IETF RFCs do not define SHA3 as an option - the use of SHA3 (or SHAKE) is primarily limited to signatures, which as seen above, are typically not FIPS-compliant.

While several products will likely be able to transition to ML-KEM regardless of the hash used (such as software defined services on general purpose devices), constrained or embedded form factors as well as high performance gear may delay their transition until hardware acceleration is available. That acceleration is presently ubiquitous for SHA2, in limited availability for SHA3, and nearly unavailable in SHAKE.

To summarize, in order to ease transition for large commercial enterprises, reduce Size/Weight/Power requirements, and make it more likely that the whole of government can comply to the greatest extent possible with the requirements in National Security Memo 10 we suggest that both ML-KEM and ML-DSA adopt SHA2. For the largest parameter sets (and potentially for all) this could be something similar to:

- XOF(p,i,j):=sequential bits of SHA512_DRBG(p||i||j) as instantiated by Hash_DRBG_Generate_algorithm.
- H(s) := truncated(SHA-512(s), 256)
- J(s) := truncated(SHA-512(s), 256)
- G(c) := SHA-512(c)
- Domain separation if desired can be achieved by prepending a domain-specific value, i.e. G(c)=SHA-512('G'||c).

We want to be clear that this comment is proposing a minor change to section 4.1, and not an addition to it. Under no circumstance should there be multiple versions of ML-KEM of a particular security level that differ only by the choice of XOF. These would not interoperate with each other and would only confuse customers as they try to comply with mandates to transition to a particular version. While several different approaches can be used for e.g. key generation in RSA or ECDH, ML-KEM is unique in that due to the FO transform a change in key generation breaks interoperability with other implementations.

Other clarifying suggestions which we think would help developers:

- **Polynomial multiplication over Rings.** FIPS.204 mentions NTT(ab) = NTT(a) ◦NTT(b) which then leads developers to want to use this as a test. For both FIPS.203 and FIPS.204, we suggest explaining how polynomial multiplication works in an appendix or explicitly calling out a reference.
- **Explicit functions**. Several auxiliary functions (brv in FIPS.204, bitrev7, Compress, decompress in FIPS.203) are given pseudocode but not explicit functions. We suggest giving them full functions.
- **NTT tables**. Algorithm 35 in FIPS.204 mentions that usually zeta is pre-computed. Suggest adding the pre-computed table to FIPS.203 and FIPS.204 since this is one of the more complicated operations in the algorithm and also acts as an extra validation test.

Typos and Minor comments in FIPS.203:

- 203:
  - Line 34: Remove “so-called”
  - Line 40: Suggestion to add quantum-resistant, key-establishment, asymmetric cryptography to keywords
  - Line 58: Replace “adversaries” with “any other parties”
  - Line 62: Remove “particular”
  - Line 113: “share a desire, in the public interest, that the licensed patents by freely…” -> either “share a desire, in the public interest, for the licensed patents to be freely” or “share a desire, in the public interest, for the licensed patents to be freely”
  - Line 257: Remove “it is well-known that”
  - Line 262: NIST PQ standardization "project" should probably be "process."
  - Line 271: “non-classified” -> “unclassified”
  - Line 282: “first four” This could be reworded- it sounds like the first four out of 82 algorithms were standardized, instead of that a first batch of 4 was selected, and more may be selected in the future.
  - Line 288: Remove “so-called”
  - Line 394: “encapsulation” -> “Encapsulation”
  - Line 458: "copie" should be "copy”
  - Line 531: seems like it should reference 4.3.1, algorithm 10 rather than 2.2.
  - Line 531: Sentence says “above” twice.

- Line 546: Remove “or” before KEM
- Line 571: Suggestion to remove “presumed” and “so-called.” Note that “presumed” is already use (probably more appropriately) later in the paragraph (line 574).
- Line 585: Remove “with”, change “replaced” to “replaces”, change “constructed” to “, which is constructed”
- 766-767: "Informally, Compress discards low-order bits” – this led to some confusion for developers; since q is not near a power of 2, the results are comparatively far from just dropping bits. Suggest a word like approximately or just mention dropping bits is what would happen if q was a power of 2.
- 768: Should say “y in \mathbb{Z}_{2^d}” since can’t be true for all q.
- Line 988: “none of them” -> “neither”
- Line 989: “can be” -> “is”
- Line 1005: Algorithm 16 should be Algorithm 17 (Algorithm 16 is Encaps)
- Line 1171: “classification” -> “categorization”
- Algorithm 4 lines 4-5: Computes floor(a/2) very differently than algorithm 3 does.
- Algorithm 13-2: input to ByteDecode not consistent with function definition (appears to implicitly want to call many copies of it in a vector). Line 22 does something similar with ByteEncode. Similar issue in 6.2.
- Algorithm 14-6: only NTT is invoked K times, not NTT_inverse.
- Algorithm 17 line 7: J(z||c,32) should just be J(z||c) by definition of J.

29. Comments from Joe Hobo, November 20, 2023

The FO transform isn't great for various reasons (e.g. side-channels basically destroy IND-CCA), but my comment is on implicit rejection. While from a theoretical point of view this is "hiding" whether or not the ciphertext comparison failed, from a practical point of view this is completely useless. No matter what the specifications say, some form of error will have to be returned, if only "hey this channel is trash, restart the protocol." Since this is going to happen, wouldn't it be better if NIST specified how this is handled?

30. Comments from the Infineon PQC team, November 21, 2023

Dear NIST PQ team,
we have the following comments on the current FIPS203 draft, we omit points that were already publicly discussed on the mailing list.

- Input Validation (line 984): according to the draft, the modulus check needs to be performed on the entire encapsulation key ek. However, the modulus check can only be applied to the component t of ek, not to the random seed rho.
- Input Validation (line 1013): the length check of dk should be removed. The decapsulation key could be stored in a multitude of different formats, depending on the implementation requirements. For example, one might only store the seed or store the secret component s in two shares. This is explicitly mentioned in the FIPS204 draft (line 651ff), a similar explanation should be added to FIPS203.

Best regards,
the Infineon PQC team

31. Comments from C. Majenz, L. Noer, November 21, 2023

Dear NIST,

Please find attached a few comments on the FIPS 203.

Best regards,

Louise Noer
MSc Student
DTU Compute

Technical University of Denmark
and
Christian Majenz (CC)
Associate Professor
DTU Compute

Comment attached.

Comments on ML-KEM Draft Standard

Louise Noer Kolborg and Christian Majenz
Technical University of Denmark

November 2023

P. 18 line 768:
The decompression function is defined with domain $\mathbb{Z}_{2_d}$ but line 768 on page 18 says,

$$Compress_d(Decompress_d(y)) = y \quad \forall y \in \mathbb{Z}_q$$

P. 18 eq. 4.7:
Equation 4.7 says,

$$\Big[Decompress_d(Compress_d(x)) - x\Big] mod^{\pm} q \leq \left\lceil \frac{q}{2^{d+1}} \right\rfloor$$

But the rounding error can be both positive and negative, so it would be more precise to say either,

$$-\left\lceil \frac{q}{2^{d+1}} \right\rfloor \leq \Big[Decompress_d(Compress_d(x)) - x\Big] mod^{\pm} q \leq \left\lceil \frac{q}{2^{d+1}} \right\rfloor$$

or,

$$\Big|\Big[Decompress_d(Compress_d(x)) - x\Big] mod^{\pm} q\Big| \leq \left\lceil \frac{q}{2^{d+1}} \right\rfloor$$

P. 28 Algorithm 14:
In the decryption algorithm, the $\mathbf{u}$ and v values are not quite the same as the $\mathbf{u}$ and v values in the encryption algorithm. A rounding error must have been added to each. Suggest naming them $\mathbf{u}'$ and v' (or similar) in the decryption algorithm to avoid confusion. Same is also true for m.

P. 32 Algorithm 17:
On page 17, equation 4.3, function J is defined as having one input. In algorithm 17 line 7, $J(z||c, 32)$ should therefore be changed to $J(z||c)$. (Shake output length should be removed.)

32. Comments from Gideon Samid, November 21, 2023

Dear NIST officials,
Your invitation to the public to speak on this important matter is much appreciated. The wisdom of crowds. It is the best antidote against groupthink. Cyber security today is based on the assumption that the attacker is not smarter than expected. It is a tenuous assumption as history indicates. It calls for a broader examination of the challenge ahead. I have come to cryptography from the field of innovation appraisal, specializing in estimating the innovation load associated with an innovation challenge. Cracking a cipher may be represented as an innovation challenge. This off-side approach to cryptography has led me to the following observation.

The very process managed by NIST to identify a post quantum algorithm points to its inherent flaw. Selection of algorithms is based on absence of a published breach. The longer an algorithm stays in the public domain without a public breach, the more it is deemed fit. This selection process suffers from two critical flaws expressed in the two following prospective scenarios: (i) a public breach may surface in the near or far future, (ii) an undetected private breach will undermine the purpose of the selection process.
The first scenario is well accounted for by NIST, preparing standby alternatives, and managing a modular configuration to ease the process of cipher replacement. The second scenario is (i) more likely, (ii) more damaging, and (iii) leading to more narrow adoption. All in all, the high likelihood of a private breach of any selected algorithm presents a strong need to re-imagine our cyber defense.

Ahead in this comment, I will elaborate on the unacceptable risk of the private breach scenario, and present a cryptographic alternative that responds to the challenge of quantum computers even when combined with the challenge of a smarter mathematician. This alternative approach is based on pattern-devoid cryptography, which has peer-reviewed accounts (including a book to be published next month), is expressed in a few dozen US patents, and has been vetted by the German national institute of standard, TÜV.

The motivation of the hundreds, perhaps thousands, of cryptographers to publish a breach of any selected algorithm, is personal reputation, which indeed keeps very smart people awake many a night. These mathematicians work with very limited computing power. By contrast, would be private breachers are all the countries in the world, to name a subset of interested breach parties. *The ideal state for any country in the world today is for it to have a secret private breach for an algorithm that no lesser than the US NIST declares secure*. Carefully handled such a secret breach will remain hidden for a long time, and offer substantial political and economical advantages to the breach holder -- in times of peace. In times of war the breach holder will impress deep confusion and paralytic mayhem on its adversary. Especially if the adversary is the United States that is disproportionally cyber reliant. All countries in the world, therefore, will regard the prospect of identifying a private breach to a NIST declared secure PQC, as a prime (secret) national objective. Countries will assemble large teams of their best mathematicians and provide them with powerful computing machines, to which academic cryptographers are not privy. We will never know if any of the algorithms NIST lists as qualified has been privately breached by one or more of the countries of the world bent on so doing. This specter will make the entire NIST operation a supportive tool for US adversaries.

Looking from the opposite side, countries of the world will suspect that NIST selected algorithms have all privately been breached by the NSA, and therefore be apprehensive of adoption.

The reason that all the proposed algorithms are breach-open is that they are all based on mathematical complexity. The more complex a mathematical challenge, the more avenues for mathematical shortcuts are associated with it. Obviously, such shortcuts that simplify the math and enable a breach, lie far off on the territory that is traversed with human imagination. Otherwise, these simplifications would have been spotted right away. Indeed, these mathematical breach procedures may require so much mathematical imagination that no one, no country will have what it takes to extract them in a timely fashion. Namely these algorithms will serve their purpose.

Both the existence of such mathematical shortcuts as well as the measure of mathematical imagination needed for their extraction, are matters that pose a great challenge for being credibly estimated. Using a methodology I started to develop in my PhD dissertation at the Technion in Israel the results for mathematical complexities of the NIST candidates are alarming. Transforming flat (Cartesian) lattice representation to an unbound geometry (non metric space) may enable a much simpler computational dynamics leading to an effective breach. It is therefore that we should strive to achieve security with ciphers that are mathematically so simple that there is no room for a shortcut. Security then is constructed through lavish use of randomness. Pattern devoid cryptography accounts for ciphers which limit their cryptanalyst to brute force attack, and then deny the attacker success by using a dynamic key where size and content grow with use, and by deploying AI-guided use of ciphertext dilution that is readily reversed by the intended recipient, but remains inherently confusing to the omnipotent attacker. Pattern devoid cryptography is less elegant. It is using larger, secret size, secret geometry keys, and is communicating extra-long ciphertexts.   But what is gained by this inelegance is mathematically proven security.

As described pattern devoid ciphers don't fit neatly into modern day cyber dynamics, the way this NIST candidate algorithm does, but given the unrelenting risk of private breach, it is well advised to install a pattern devoid cipher at least in the mode of "Lifeboats on the Titanic" --

namely for the eventuality when the elegant cryptography fails. The pattern devoid, clumsy cipher will kick in and save the day.

Reference:

1. "Tesla Cryptography:" Powering Up Security with Other Than Mathematical Complexity https://eprint.iacr.org/2023/803
2. “Pattern Devoid Cryptography” https://eprint.iacr.org/2021/1510
3. “Artificial Intelligence Assisted Innovation” https://www.intechopen.com/chapters/75159
4. “AI Resistant (AIR) Cryptography” https://eprint.iacr.org/2023/524
5. “The UnEnding CyberWar” https://www.amazon.com/Unending-Cyberwar-Gideon-Samid/dp/0963522043
6. “The Cipher Who Came in from the Cold” https://www.amazon.com/dp/B0B8PFGZSB/

Gideon Samid

33. Comments from the Canadian Centre for Cyber Security, November 22, 2023

Hello,

Please find attached our comments for the FIPS 203, Module-Lattice-Based Key-Encapsulation Mechanism Standard. If posted publicly, please attribute our comments to the "Canadian Centre for Cyber Security", and refrain from publishing my name.

If you have any questions, please don't hesitate to reach out to me directly.

Comment attached.

FIPS 203: ML-KEM Comments

Small edits:

The prefix to each comment is in the format **Section / Page / Line numbers** (or Equation, or Algorithm where relevant)

- Preamble / Foreword / 20: "Federal Information Processing Standards Publication (FIPS) Series" should read "Federal Information Processing Standards (FIPS) Publication Series".
- Preamble / Foreword / 26: "James A. St Pierre" is spelled without a period after "St", which is inconsistent with the spelling in the ML-DSA and SLH-DSA drafts.
- Preamble / i / 50: The "Name of Standard" field here is inconsistent with the other two draft standards. Unlike the others, it includes the abbreviation "(ML-KEM)" in the name, and has "PUB" in "(FIPS PUB 203)".
- Preamble / ii / 113: "the licensed patents" should be "that the licensed patents".
- Preamble / ii / 116: "KYBER" here is in all-caps, instead of small caps as it is in the rest of the standard.
- Preamble / iv / 169: "does not currently intend holding" should be "does not currently intend to hold".
- 2.4 / 6 / 458: "copie" should be spelled "copy".
- 4.2.1 / 17 / 757: "it" starts a sentence, but is not capitalized.
- 4.2.1 / 18 / 768: The guarantees are stated to apply to all integers modulo q, this should be all integers modulo 2^d.
- 4.2.1 / 18 / Equation 4.7: The left-hand side should be in absolute value.
- 4.3 / 21 / 838: The equivalency symbol is used, which was not defined in the mathematical symbols glossary.
- 5.2 / 28 / 936: "all of which are sampled from the centered binomial distribution" should be "all of which are sampled from centered binomial distributions" (or "a centered binomial distribution") since the distributions may be different (as is the case for ML-KEM-512).
- 6.1 / 29 / 973: the last component of the decapsulation key is described as a "pseudorandom", but in fact should be random as defined in Section 3.3.
- 6.3 / 32 / Algorithm 17: The input to Algorithm 17 is in the order '(c, dk)' where as the input to Algorithm 14 is in the order '(dk_pke, c)'.
- 6.3 / 32 / Algorithm 17, line 7: Here, the hash length of 32 is provided as a second input to 'J', whereas when 'J' is defined in Equation 4.3 it takes in only one argument (and outputs a fixed-length digest).
- 7 / 33 / 1042-1043: "is comprised of" is arguably ungrammatical, and can be replaced with either "comprises" or "is composed of".
- 7 / 33 / 1046-1047: The parameter k is stated to determine the dimension of the vector e_2, but this is not the case. Additionally, k does determine the dimension of t and u, and this may be stated here.

- References / 35 / 1098-1099: The link to the round 3 submissions is broken.
- References / 35 / 1118 (reference 10): "page 84–93" should be "pages 84–93".
- References / 36 / 1157 (reference 21): "page 212–219" should be "pages 212–219".
- On page 39, footnote 4, the text says “security of SHA” but “quantum security of AES”. Presumably both should say “quantum security”.

Suggested changes to presentation:

- Pseudocode is written using latex symbols instead of ASCII names to represent both variables and parameters. Occasionally this has the potential for confusion or even collisions when names are rendered in ASCII in a straightforward manner. For example, in Algorithm 13, "\mathbf{r}" represents the secret vector to be multiplied by the matrix A^T and "r" represents the encryption randomness. In the NTT and NTT^-1 Algorithms 8 and 9, the symbol "\zeta" represents a system constant set to 17 and "zeta" is an intermediate variable. Furthermore, the use of ASCII names can help clarify the role of different variables. For example, renaming ρ and σ into something like publicseed and privateseed helps implementers understand both the role of these variables and how they should be handled with respect to the "shall" directive on destroying intermediate variables. The names given to the input of functions can also be clarified; for example the input to PRF may be changed to 'seed' and 'index', rather than 's' and 'b'.
- The value \zeta is set be a constant, 17, for all parameter sets. This should be listed along with q and n as a system parameter that does not change between parameter sets.
- The standard tends to leave q as symbolic, while writing out n as 256. Keeping these values as symbolic throughout may help with clarity and consistency.
- There are occasional consistency issues with terminology. The draft sometimes refers to "Alice" and "Bob" as the participants in the protocol, and sometimes to "Party A" and "Party B". While the glossary does note the equivalency of these terms, it is preferable if only one convention is used throughout. Similarly, section 4.3 refers to both the "direct sum" and the "product" of rings.
- The phrasing "so-called" is occasionally used in this draft when referring to technical terms (e.g., "so-called M-LWE problem", "so-called Fujisaki-Okamoto (FO) transform"). This suggests ambiguity or disagreement in terminology where none really exists. In all cases we recommend the phrase "so-called" be removed for clarity.
- In parts of the specification for which there is a high degree of overlap with ML-DSA, some notation could be made more consistent. For example, ML-DSA uses 'brv' to denote bit reversal of a byte and ML-KEM uses 'BitRev_7' to denote reversal of seven bits.
- In the glossary of terms, it would be beneficial to include the definition of key generation to emphasize that there are two usages of the term, for PKE and KEM (in the same way that "key pair" is defined).

- On line 522, A is used as an example of an input to NTT, but the actual A matrix in the scheme is sampled directly in the NTT domain and thus is never used as an input. Changing A in the example to another value here may reduce confusion.
- The comments in the pseudocode that state how many times a function is employed are very helpful. A similar comment could be made for Compress and Decompress in Algorithms 13 and 14. On line 540, it can also be reiterated that Compress and Decompress act on each component of each ring element of a vector.
- On line 762, an example of little-endian ordering could be helpful, as the term usually means byte-order in a multi-byte integer. For example, the byte representing 134 decimal corresponds to the bit sequence 01100001.
- On line 938, it is written that v \leftarrow t^T r + e_2, and then states that the encoding of the input message is added to this component to prepare the v value before compression. In order to (even briefly) avoid stating that v = t^T r + e_2 it may be beneficial to note that \mu is added right away.

Suggested clarifications:

- We recommend the discussion on "equivalent implementations" and "equivalent processes" as it applies to algorithms that use randomness (K-PKE.KeyGen, ML-KEM.KeyGen, and ML-KEM.Encaps) be clarified. The glossary (lines 345-347) states that processes are equivalent when the same input produces the same output and suggests that "values made available during the process" are considered input in this context. If this means the randomness used by K-PKE.KeyGen, ML-KEM.KeyGen, and ML-KEM.Encaps, this should be plainly stated so. The notion of "equivalent implementations" later discussed (lines 663-673) is somewhat at odds with the notion in the glossary. First, it states that ML-KEM.Encaps takes only one byte array as input (the public key), and does not here seem to consider the randomness as part of the input. The example provided later only states that the distribution of output arrays must be identical. If an equivalent implementation should have the same output when the same input and seed randomness are provided, this should be plainly stated to be so.
- The destruction of intermediate values is given a "shall" directive, emphasizing its importance. To help an implementer follow this directive, we recommend which intermediate data needs deletion at which point should be clarified, as the guidance only states "as soon as it is no longer needed".
- For the purposes of FIPS validation and patent protection, we recommend NIST clarifies whether alternative private key formats are allowed. For example, in TLS one may wish to skip encoding / decoding s, or defer the hashing of the public key to decapsulation. Currently the description of equivalent process suggests that all of these values must be entirely computed at the conclusion of key generation.
- On line 68 it is stated that Party B sends the ciphertext "over the same channel". Presumably this means the same channel that was used to send the encapsulation key, but this is not necessary or desired in many instances.
- On line 599, it is stated that the transformation is necessary for "full security". Clarifying that this means "full (IND-CCA) security" might help convey what "full" means.

- On lines 675-680 (or in NIST SP 800-227, which has yet to be published), we recommend NIST clarifies or provides guidance on how the 256-bit shared secret K may be used when only a 128- or 192-bit symmetric key (e.g. for AES-128 or -192) is desired.
- On lines 880-883 it is suggested that it must be the case that input validation occurs prior to invoking the K-PKE subroutines. But the discussion of equivalent processes in 6.2 (lines 993 - 997) and 3.3 allow input validation to occur during these subroutines. We recommend language in Section 5 be rewritten to note that while the pseudocode there does not contain input validation, it could occur here because of the definition of equivalent process.
- On lines 703-705 it is stated that floating point arithmetic should not be used. A suggestion on how to avoid floating point arithmetic could be helpful to implementers (e.g., via integer floor and remainder operations).
- On lines 775-776 we recommend the language be clarified: 'ByteEncode' and 'ByteDecode' serialize and deserialize arrays of length n = 256, whereas the serialized lengths (i.e., in bytes) differ. Moreover, since these operations are applied component-wise to vectors, those arrays will have length 'k * n'.

Functional change:

- 5.1 / 26 / Algorithm 12: Change the key generation seed d to at least 40 bytes to prevent a multi-target attack. (Note that we have the same comment for ML-DSA about the seed used in Key generation).

34. Comments from Dev Null, November 22, 2023

This letter contains comments on the chapter "Appendix A — Security Strength Categories" identical among the 3 proposed FIPS standards NIST FIPS 203 ipd (Lines 1159 to 1238), NIST FIPS 204 ipd (Lines 1014 to 1088), NIST FIPS 205 ipd (Lines 1134 to 1213).

Definitions
To avoid confusion and distinguish between:
A) the Security Strength categories listed in FIPS ipd 203, 204, 205
B) the Security Strength method of using bits of security from "Recommendation for Key Management" SP 800-57 rev. 5 and earlier.
(https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final)

The following terms will be used:
"Strength Categories" – The new proposed method in FIPS ipd 203, 204, 205.
"Bits of Security Method" – The established method in SP 800-57.
Since both of them are called "Security Strength" in their respective documents.

Table of content:

Dev Point 1 – The problem of using sliding scales as a base of measurements

The Security Strength Categories are units that change over time with advances in technology and crypto-analysis.
Using AES-256 today (pre-quantum) has a higher security level (in respect to effort and money needed to compromise a key)
than AES-256 will have post-quantum plus years of Moore's law making computing power cheaper.

Therefore, I propose a Security Category scale of 10 with room for future improvements or reverting back to the Bits of Security Method for a more granular and uncapped approach.

Using specific algorithms as metering sticks is like having one person's arm being the base of all measurements of lengths.
If that reference person then has a sword accident shortening their arm, everything using that reference scale need to change values, to compare with the new arm length. With everything increasing in numerical length because of the new shortened measurement base. But everything (apart from the arm) has not changed size, yet increase in length because of the reference getting shorter.
If a new vulnerability is found in AES, all the algorithms in Strength Categories 1, 3 and 5 not affected by that vulnerability need re-evaluation because they suddenly are stronger in

comparison to AES and may change to a higher Strength Category, yet in practice those other algorithms have had no increase in security.
Science moved towards physical constants for units of measurements in order to avoid constant changes as updating a basic measurement unit affects all derivatives of that unit. Hence metrologists generally try to keep units of measurement stable.
Please keep the standard for how to measure the security of cryptographic algorithms as stable as possible, we use these high-quality standards all over the world; they are crucial for the global economy.

Dev Point 2 – The future will have lower security than what is available today

The proposed Strength Category method is capped at level 5, which is how secure AES 256 is today.
NISTIR 8105 http://dx.doi.org/10.6028/NIST.IR.8105 Table 1 explicitly mentions a reduction in security level for AES and SHA-3 as an impact of a large-scale quantum computer.
Furthermore the advancement in computing technology will make computing resources cheaper each year, making all the Strength Categories weaker year by year when looking at the capital investment needed to break the probabilistic security strength. Breakthroughs in crypto-analysis will decrease the cost even further.

In order to reach the same security level as pre-quantum, there needs to be a Security Strength Category that in the future post-quantum world will display the equivalent of the current day strength of AES-256.

Dev Point 3 – The world needs a standard for something better than AES-256

I know of companies that are trying to make home-brewed AES-512 but their implementations often end up only being AES-257 because of an incorrect understanding of bits of security.
An official AES-512 would be a good first step to establish a post-quantum security level that is at least as good or better than our current pre-quantum AES-256.

Here is a good reference on how to count bits of security by cryptographer Daniel J. Bernstein October 2023:
https://blog.cr.yp.to/20231003-countcorrectly.html
An article also touching on the Strength Categories method.

I am not an expert on Grover's algorithm, meaning that I have to rely on sources like "NIST IR 8105 - Report on Post-Quantum Cryptography", that on page 8 states:

"Grover's algorithm provides a quadratic speed-up for quantum search algorithms in comparison with search algorithms on classical computers. We don't know that Grover's algorithm will ever be practically relevant, but if it is, doubling the key size will be sufficient to preserve security."

Following this quoted recommendation a current pre-quantum system using AES-128 can “just” upgrade to using AES-256 post-quantum and maintain the same or higher level of security (using the Bits of Security Method to count security strength and ignoring Moore’s law for an easier analogy).

But a current pre-quantum system using AES-256 has no way to stay at the same level of security post-quantum. The security level will just be decreased with the amount a practical implementation of Grover’s algorithm will compromise it with.

While I am no expert on Grover’s algorithm, I am an expert on cybersecurity risk management and in many safety-critical systems (IEC 61508 provides guidance) a decrease in safety level is not allowed when proposing a new implementation or updating a system.
The probability of failure must be the same or lower than the existing system (in some countries).
Many safety-critical systems and critical infrastructure (defined in EU Directive 2555 from 2022; NIS2) rely on cryptography to provide safety.
A remotely controlled rail switch or drinking water pumps are good examples.

But as noted, there is no such option to keep the same security/safety level post-quantum for existing systems reliant on using AES-256.
Furthermore, as noted in Dev Point 2 the advancement in computing technology and crypto-analysis will decrease the price of the needed computing resources each year, making all the Strength Categories weaker year by year when looking at the capital investment needed to break the probabilistic security strength.

This conflicts with the risk management strategy of always striving for equal or better safety and security over time.

As AES-128 with time needs to be replaced with AES-256.
&
RSA and ECC needs to be replaced or combined with PQC (like in X25519Kyber768Draft00).
&
Hashing functions need larger outputs.

The IT industry is already moving towards increased modularity in all crypto systems, from software over MFA tokens to HSMs.
Many systems are more ready than ever to handle change to new cryptographic algorithms without downtime. By some called the Crypto Agility approach.
This change to a more flexible approach in the IT sector also opens up for the option that organisations wanting a stable risk level can steadily increase the key and hash output lengths if there are algorithms that allow for it.
But with the current suggested cap of Strength Category level 5 and the level in practice being decreased with advances in technology and crypto-analysis it is hard (if not even impossible) for an organisation to keep a stable risk level in respect to the usage of cryptographic algorithms

detailed in NIST standards.

I appreciate the huge work done by NIST, scientists and the industry in the making of these PQC standards, it has been a great journey, my thanks go out to everyone involved.

Kind Regards
-Dev Null

November 22nd, 2023
Dev Null
Quantum Key Distribution Network Engineer
at the Technical University of Denmark
www.fysik.dtu.dk/english/research/qpit
Open Researcher and Contributor ID (ORCID) identifier 0009-0005-0121-3627
https://orcid.org/0009-0005-0121-3627

I have labelled the major themes "Dev Point" 1 to 3 in order to make them easier to reference in talks, meetings and written materials.
Hopefully no one else named Dev will label their comments in a similar manner.

35. Comments from L. Noer, C. Majenz, November 22, 2023

Dear NIST,

We would like to add one more comment regarding the upper bounds on failure rates (table 1, page 14).

We assume these bounds originate from the third Kyber-Crystals proposal, which (according to the proposal) were computed using the software available here: https://github.com/pq-crystals/security-estimates/tree/master.

The version of Kyber that is encoded in the software, is different from the Kyber presented in the standard.

In the code, vector t (from the public key) is compressed and decompressed, giving rise to an additional rounding error. In the code, the rounding error is named rqk (see the Kyber.py file, line 16) and the error is included in the computation of the failure rate (Kyber_failure.py, line 14).
Additionally, the parameter set for Kyber_512 in the code is slightly different to the parameter set of the standard: eta_2 has value 3 in the code, but 2 in the standard.

It seems a little unnatural to include the extra rounding error still (presumably the compression and decompression is a remnant from previous versions of Kyber). The bounds may be worth revisiting.

Best regards,
Louise Noer
MSc Student
DTU Compute
Technical University of Denmark
and
Christian Majenz (CC)
Associate Professor
DTU Compute

36. Comments from the NXP PQC team, November 22, 2023

Dear NIST team,

The NXP PQC team would like to make the following comments on the FIPS 203 draft:

1a) We think it would be preferable to specify a key format that excludes the public encapsulation key from the private decapsulation key. Storage of an additional 800, 1184, 1568 bytes without need is a significant challenge for very embedded devices. This format can be useful even without changing the Decaps API, to have a clear format for internal storage where the encapsulation key could be separated from the secret data. The same argument could be made for H(ek), but that is small enough that it has negligible impact.

1b) If exclusion of the public encapsulation key is not possible, we think it is preferable to change the format of the decapsulation key to (dk_PKE, z, H(ek), ek) to allow easier splitting between secret and public data in the decapsulation key, and to allow easier parsing for alternative key storage.

2) We believe it can be useful to have an ephemeral only version of Kyber for use cases where authentication is not required (or done by other means), as it is faster, simpler, and more secure. Could NIST please elaborate why the K-PKE was excluded as stand-alone, or why a direct ephemeral key exchange is not included?

3) We would like to swap the order of hashing of z and c in Line 7 of ML-KEM.Decaps. It is preferred to hash public data c first, so that the Keccak invocations on c do not require masking. In case z is hashed first, the data would contain secret information and all subsequent permutations would require masking.

4) We would like NIST to confirm whether explicit rejection in the Decaps routine is considered compliant with the spec, or not. Even if there is benefit to the API in case of implicit rejection, having the option to fall back to explicit rejection can be beneficial for many applications as the protocol can be aborted earlier without unnecessary additional computation. In most practical situations, the implicit rejection is not retained as virtually no protocol will act in constant-time (or even deterministically) depending on the decapsulation result, let alone protect the result from side-channel attacks.

Thanks for all the hard work.

Kind regards,
NXP PQC team

37. Comments from Cloudflare, November 22, 2023

Please find our public comments on FIPS IPD 203, 204, and 205 attached.

Best,

Bas

Comment attached.

To: fips-203-comments@nist.gov; fips-204-comments@nist.gov; fips-205-comments@nist.gov
Subject: Comments on FIPS 203, 204, and 205

A submission from Cloudflare, Inc., in response to the National Institute of Standards and Technology's (NIST) 24 August 2023 request for public comments on initial public drafts of
FIPS 203 "Module-Lattice-Based Key-Encapsulation Mechanism Standard",
FIPS 204 "Module-Lattice-Based Digital Signature Standard", and
FIPS 205 "Stateless Hash-Based Signature Standard".

Cloudflare appreciates this opportunity to comment on the National Institute of Standards and Technology's (NIST) request for public comments on the initial public drafts of FIPS 203, 204, and 205. Cloudflare submits the following comments, which will address our own experience with post-quantum cryptography and protocol design, our view on measured cryptographic agility as it applies to the present drafts, and specific comments for each.

Introduction and Cloudflare Background

Cloudflare is a leading connectivity cloud company. It empowers organizations to make their employees, applications, and networks faster and more secure everywhere, while reducing complexity and cost. Cloudflare's connectivity cloud delivers a full-featured, unified platform of cloud-native products and developer tools, so any organization can gain the control they need to work, develop, and accelerate their business.

Powered by one of the world's largest and most interconnected networks, Cloudflare blocks billions of threats online for its customers every day. It is trusted by millions of organizations – from the largest brands to entrepreneurs and small businesses to nonprofits, humanitarian groups, and governments across the globe. Cloudflare is used by nearly 20% of all Internet websites.[1]

At Cloudflare we have helped increase security and privacy on the Internet by pushing the envelope on cryptographic design and deployment, by contributing to among others TLS 1.3, MLS, DNS-over-HTTPS, Encrypted ClientHello. From 2019 onwards we have executed several internal and external large-scale experiments to determine the real-world deployability of post-quantum cryptography.[2] This has allowed us to deploy an early version ML-KEM (FIPS 203) in production, which at the time of writing, is used by 1.7% of all our inbound TLS 1.3 connections.

---

[1] See W^{3}Techs, Usage Statistics and Market Share of Reverse Proxy Services for Websites, https://w3techs.com/technologies/overview/proxy.
[2]https://blog.cloudflare.com/the-tls-post-quantum-experiment/
https://blog.cloudflare.com/sizing-up-post-quantum-signatures/
https://blog.cloudflare.com/post-quantum-to-origins/

CLOUDFLARE®

Measured cryptographic agility

The need for cryptographic agility

We believe it is crucial that protocols for use on the Internet are designed with the flexibility to replace its underlying cryptographic primitives gradually and securely in case of compromise. The threat of quantum computers is the obvious example, and we applaud NIST's efforts to standardize post-quantum cryptography.

We also feel it has been a prudent choice of NIST to standardize SHA3, so that it can replace SHA2 in case a weakness is found (for which there is no indication).

The cost of cryptographic agility

However, we also believe that due restraint should be exercised in adopting new primitives in *existing* protocols. The case for post-quantum cryptography is clear, but, for instance, *at the moment* there is no point replacing SHA2 with SHA3 for the key schedule of TLS 1.3: there is no indication that SHA2 is weak, and adopting SHA3 comes with a significant cost. If both are available, some users will end up enabling one, and disabling the other. Those that want to be compatible with all, will need to deploy both. This incurs a significant cost in development, testing, and increased surface for implementation mistakes. The situation is more pertinent for constrained use cases, such as embedded devices.

That does not mean SHA3 will see no deployment. To the contrary: for new use cases, we prefer SHA3, or to be more precise, SHAKE, as it is a more versatile primitive that is easier to use correctly (no need for HMAC, HKDF, or MGF1). We are pleased to see SHAKE used in these drafts.

Call for continued restraint

We appreciate the choices NIST has made so far, showing restraint where differences between variants are minimal, but allowing different options where it matters:

- NIST has removed many variants in the present drafts as compared to the submissions: the AES-based variants have been removed, leaving only the SHA3-based variants for ML-KEM and ML-DSA.
- NIST has picked only a single KEM for now. We understand a second KEM might be picked from the fourth round, in case cryptanalysis against structured lattices improves.
- Three signature schemes have been chosen for standardization, of which standards have been drafted for two. NIST is looking to standardize more in an on-ramp. Considering the widely varying performance and security characteristics of the available schemes, we feel it was the right choice.

We see one opportunity to reduce the number of variants:

- We do not see the need for both a SHAKE and a SHA2-based variants of SLH-DSA.

We ask NIST to continue exercising restraint, and reject calls to standardize *additional* variants using different symmetric primitives.

CLOUDFLARE

12-round SHAKE / SHA3

For all schemes, not just SLH-DSA, hashing accounts for a significant, if not often the majority, of the computation.

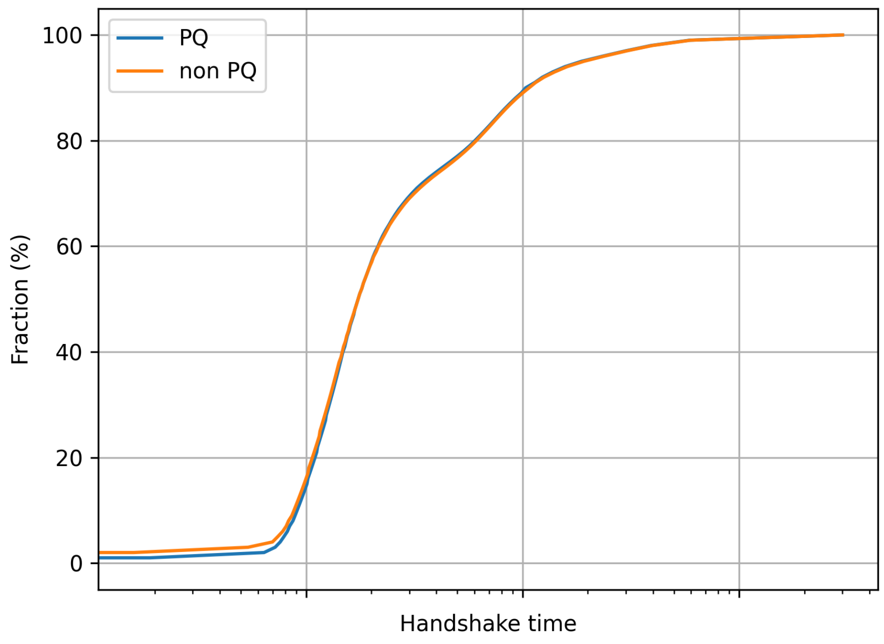
The designers of SHAKE / SHA3 have reiterated their 2016 proposal to use a variant of 12 rounds as was originally proposed, which has now been dubbed "TurboSHAKE". We are comfortable with the wide security margin of the 12-round variant, and would welcome NIST *replacing* SHAKE / SHA3 in these drafts with their "Turbo" variants.

Specific comments on FIPS 203 (ML-KEM)

Of the three drafts, we have the most hands-on experience with ML-KEM, which is now deployed in production worldwide.

We expect ML-KEM to be a practical replacement for classical key agreement on the Internet. We like its simple design and great performance on most platforms. Informed by our 2019 experiment with Google, we were concerned that a significant portion of connections would break because of the large key sizes of ML-KEM. Fortunately, this fraction has decreased dramatically since then, and it does not seem to be a blocker.

At the time of writing, 7% of all Chrome 118+ TLS 1.3 connections to Cloudflare hosted websites are using *X25519Kyber768Draft00*, a hybrid of the classical key exchange X25519 and an early version of ML-KEM. The following graph shows the cumulative distribution function of TLS handshake times of those connections (PQ) compared to those Chrome 118+ connections that did not use Kyber (non PQ).



CLOUDFLARE

This shows excellent performance *on average*. Although promising, we cannot yet conclude that performance is great for all users.

Specific comments on FIPS 204 (ML-DSA)

None of the signature schemes submitted to the competition are an ideal drop-in replacement for classical signature schemes. ML-DSA's drawback is the size of its public key and signatures. This makes it impractical for some applications, and unfavorable in many. On the other hand, ML-DSA is reasonably easy to implement, and is computationally relatively cheap.

Despite its drawbacks, standardization of ML-DSA is a good choice and absolutely necessary, so that those that need to adopt early can move forward despite the costs.

We applaud NIST's ongoing efforts to standardize additional signature schemes that fit use cases for which ML-DSA is ill-suited.

Specific comments on FIPS 205 (SLH-DSA)

(No specific comments.)

38. Comments from Filippo Valsorda, November 22, 2023

Dear NIST,

I am attaching my comments on FIPS 203.

Regards,
Filippo

Comment attached.

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 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.

39. Comments from Concerned Cryptographer, November 22, 2023

I am speaking pseudonymously for myself alone and the opinions expressed herein do not reflect the opinions of any other person or organization.

I am concerned with the semi-documented changes to the spec for what is now called ML-KEM and is based on Kyber. The Kyber spec called for hashing of all entropy before being used by the function: ML-KEM drops this entropy becomes e = (e), instead of Kyber's e = h(e). It also drops hashing of the ciphertext inside of the Fujisaki-Okamoto transform, so that M = h(m|k) becomes M = (m|k), so predictable entropy can work its way all the way to the output. When combined by an attacker with intentionally triggered decryption errors, this could substantially lower the security claim of the system, especially at greater dimensionalities.

Neither of these changes breaks the security of the system in itself if one believes the provided security reductions to be valid, but combined, the two changes put too much faith in implementers. The spec calls for ML-KEM to only be used in concert with a NIST SP 800-90 series entropy source and CSPRNG. There exists a long list of cryptographic compromise via poisoned, incorrectly seeded, or incorrectly used entropy. But will developers always follow the spec? No. Nor do we have guarantees that the NIST RNGs will remain immutable and unbroken. Hashing entropy mitigates this risk with no real-world performance penalty.

Kyber had guardrails against this, in hashing entropy before use and hashing the ciphertext internal to the FO transform, that I do not find. Assuming ad arguendum that developers will only use NIST-approved entropy sources and that those entropy sources are good, developers will still need to be aware of the context to hash the ciphertext themselves for applications requiring it.

These changes put too much faith in the developers, since we're only human and don't have an error rate of below 1 in 2^{118} (the lower bound of ML-KEM's security claim given credence, ad arguendum, to claims made in recent debate), let alone 1 in 2^{143} or higher.

As a result ML-KEM is a weaker, more brittle cryptosystem than the Kyber it is based on, and one that puts too much faith in the hands of implementers to implement correctly instead of baking them in to the spec.

No performance is gained by dropping the entropy hash or the FO hash in any real-life application that can run even the smallest instantiation of ML-KEM-512.

As a result I recommend reverting to the entropy hashing and hashing of the ciphertext internal to the FO transform that was present in the Kyber spec, but is not present in FIPS 203 ML-KEM.

I'd also like to see negative as well as positive test vectors published in the final FIPS 203.

Thank you.

40. Comments from Jim Goodman, November 22, 2023

Hello,
In general I thought this was a well-written document that structured things in a logical fashion and order. Well done!

Some minor comments:

Line(s)	**Comment(s)**
112	Change "selection of public-key" to "selection of **the** public-key".
113	Change "desire, in the public interest, the licensed" to "desire, in the public interest, **that** the licensed".
125	First time they reference a document that does not yet exist (NIST SP800-227), this is done several times throughout the document.
130	Change "that any module that" to "that any **instantiation** that".
290	Why diff against the round 3 specification when a more recent one exists that incorporates a number of the identified changes? It seems more reasonable to use the most recent "official" publication, no?
268	Second time they reference a document that does not yet exist (NIST SP800-227).
290	In ML-DSA you quoted specific version numbers in the specifications whereas here you don't and that could lead to confusion.
456	Change "copie" to "copies".
545	Third time they reference a document that does not yet exist (NIST SP800-227).
565	Fourth time they reference a document that does not yet exist (NIST SP800-227).
655	Fifth time they reference a document that does not yet exist (NIST SP800-227).
677	Sixth, and final, time they reference a document that does not yet exist (NIST SP800-227).
Figure 1	Should label left side as "Bob" and right side as "Alice"
Figure 1	Should include K_A and K_B in the figure as well since it is called out in the following text.
757	Change "it also specifies" to "It also specifies"
774	Dilithium round 3 specification included diagrams showing the different byte packing schemes which made it very clear and easy to follow. This section could benefit from a similar approach.
Algorithm 6, Line 4	Consider adding comment stating range of d_1 is [0, 4096)
Algorithm 6, Line 5	Consider adding comment stating range of d_2 is [0, 4096)
Algorithm 7, Line 3	Consider adding comment stating range of x is [0, 2^eta)
Algorithm 7, Line 4	Consider adding comment stating range of y is [0, 2^eta)
820	Change "input a polynomial" to "input of a polynomial"
Equation 4.10	Consider adding the operator symbol to section 2.3
Algorithm 14, Line 6	NTT^-1 is only invoked one time.

Take care.
Jim

41. Comments from Jacob Appelbaum, November 22, 2023

To Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology, the Secretary of Commerce Gina Raimondo, any relevant Inspector General(s), the NIST post-quantum team, and to you dear future historians,

I am an American postdoctoral researcher working [0] on the topic of post-quantum cryptography in a research group at a university in the Netherlands and I speak only for myself with this letter. Thank you in advance for taking the time to read my comments.

In summary NIST should at a minimum undo two late-in-the-game unilateral changes to Kyber as found in the current proposed ML-KEM standard:

- NIST removed a step in the Fujisaki-Okamoto (FO) transform that would hash the ciphertext into the shared secret computation; see page 2, 304 to 308. The ciphertext should be hashed into the shared secret. NIST should restore the hash of the ciphertext into the shared secret computation.

- NIST removed the step to hash system randomness; see page 2, lines 309 to 314. The raw random value should be hashed. NIST should restore the hash over the random data.

These two last minute changes could reasonably be described as dangerous and could be leveraged by an adversary to seriously damage security.

Additionally, NIST should continue ignore the NSA's recent suggestion to use a different hash function. NSA did not adequately explore length extension vulnerability considerations in their recommendation.

A historically relevant analysis should show that defense in depth must consider that decisions by NIST itself, such as the above, may be the root cause of future cryptographic failures. Changes such as the two changes above in the last few months of a multi-year competition are dangerous. The changes risk invalidating years of otherwise positive security analysis work.

On page 2, lines 315 to 318 NIST describes a change for input validation steps. This change should be carefully analyzed, and justified in public by NIST in detail.

Generally for the overall pqc process, I offer the following:

NIST plays an active role in the pqc competition beyond simply impartially overseeing a fair and transparent post-quantum cryptography competition. NIST has influenced key aspects of the cryptographic primitives under discussion in various ways, and some of those ways appear unsafe as mentioned in my earlier summary.

NIST employees have generally engaged in discussion, produced interesting public research, and they have performed public engagement such as by attending academic cryptography conferences.

NIST has influenced discussions in the pqc-forum through both selective non-participation and with surprising, sometimes seemingly overtly hostile, selective participation. Former NIST employee participants appear to receive different treatment when engaging in behavior fairly perceived as hostile in the pqc-forum than non-NIST alumni participants. NIST admonishes and shames people by name. NIST should answer questions in a direct, reasonable, and timely manner.

NIST appears to have not always acted in an impartial or a fair manner, nor have they provided clear explanations if the party asking may be portrayed by NIST as problematic. NIST has repeatedly done the latter to high profile scientists participating in the pqc process. NIST has taken actions that have discouraged public participation and indeed, several of my professional academic colleagues expressed a complete refusal to engage in the NIST process given the hostility from official U.S. Government officials in their treatment of scientists on the pqc-forum email list. For myself, I debated drafting this email until the very last moments of the final deadline for comment. A deadline that could have clearly been a date and time with UTC offset, but as is often the NIST pattern, it was expressed in a colloquial English that also disadvantages non-native speakers. NIST should say a time and a date and provide a timezone for each milestone, and/or deadline.

NIST sets a goal of being transparent while ignoring that researchers have had to file Freedom of Information Act (FOIA) requests to gain access to relevant documents that could have been published proactively. NIST should only hold open forums and should publish all related documents unredacted, and proactively so. As much communication as possible should be published by NIST. NIST should proactively request an Inspector General audit of all relevant processes and material. NIST should proactively request such reviews as one way to begin to restore trust that has been lost in relation to their involvement in cryptographic weaknesses, cryptographic sabotage, backdoors, and other project BULLRUN related issues.

A consistent, fair engagement process from NIST for all participants has been lacking. A review of the practices involved in the post-quantum cryptography standardization process should be performed by the appropriate Inspector General, the Secretary of Commerce herself, and/or any other appropriately cleared non-NIST personnel. Future national competitions should integrate the findings of that review and the review should be made public without redactions.

The NSA has repeatedly been shown that their signals intelligence (SIGINT) mission through project BULLRUN style cryptographic sabotage is more important than their IAD defense mission. The relationship between NIST and NSA should be examined to understand if specific individuals at
NIST are witting or unwitting accomplices in the cryptographic sabotage performed by NSA on cryptographic standards published by NIST. NIST should draft and adopt a policy to report

undue pressure or unofficial requests to weaken security by any agency in public, immediately, and to
any relevant Inspector General.

NSA communications with NIST beyond participation in the pqc-forum email list have not been fully and transparently communicated to participants or the public. A full accounting of these communications is important to restore the public trust in NIST that was severely damaged by the NIST standard Dual_EC_DRBG which we now understand was purposeful cryptographic sabotage by NSA.

Corporations appear to have outsized influence in cryptographic design changes asserted by NIST. NIST does not always clearly explain or fully clarify their decisions after corporate representatives have provided feedback. While on the one hand this could appear to demonstrate that NIST is indeed being responsive to some concerns, on the other hand, usually we must infer that read of NIST's behavior. NIST does not clearly explain the rationale for each decision or their decision making processes. NIST should always clearly explain their decisions and their motivations for decisions.

More specifically, I offer the following:

- Backdoor accommodations: NIST unilaterally informed the pqc-form that they had decided to remove a very important Kyber security feature which consists of a single hash over the direct system DRBG output as previously mentioned. This security feature was designed to mitigate the risk of cryptographic sabotage presented by known flawed systems such as the NIST standardized backdoor known as Dual_EC DRBG. The removal of the single hash over the raw random bits from Kyber accommodates a DRBG backdoor such as Dual_EC_DRBG. The removal of this security feature was not explained excepting that NIST acknowledged that there exist pro-hash and the anti-hash scientific factions. It seems that the anti-hash faction at NIST has won the day, and decided but surprisingly they did so without p roviding a clear and convincing motivation. The very short sentence provided in the draft standard is not clear nor is it convincing given the history. Passing the raw RNG/DRBG output without
hashing is not a defense-in-depth posture. It is not reasonable to say that since the DRBG must a NIST approved DRBG, there need not be defense in depth. Other RNG/DRBG "SIGINT enabled" backdoors exist, sometimes in software, and sometimes in hardware. A single hash is a negligible cost, especially if the lack of a hash may cause the entire system to catastrophically fail. However, the removal of the single hash has a very high cost both technically and socially: This change certainly raises serious questions about the explanations around NIST's role and knowledge in cryptographic sabotage generally. It is not unlike the issues surrounding Dual_EC_DRBG. It may invalidate the cryptography
entirely in certain deployments, and it erodes the already extremely fragile public trust. It also discourages security analysis. In essence, this change has happened at the eleventh hour, and the full security ramifications of this change are difficult to fully explore with limited time remaining.

Metrics: Various security metrics over the course of the post-quantum standardization process were and remain unclear. Attempts by various participants to increase confidence in any particular interpretation
through discussion on the pqc-forum list about the NIST stated metrics has been lacking in terms of NIST's response.

- Hybrid constructions: In addition to restoring the two hash steps, defense in depth at the overall cryptographic primitive level is an important goal. NIST appears to be unwilling to engage in standard risk management where a new unproven cryptographic system is required to be combined with an established, trusted, well understood, and practically proven cryptographic system. Kyber should be assumed to be broken and it should be combined with something currently deployed such as Elliptic-curve cryptography (ECC) cryptography. As an example, any hybrid construction should be as secure as the most secure of either the post-quantum primitive or the contemporary cryptographic primitive such
as ECC. While complaints about energy consumption or other notions of efficiency may be raised, preventing failure of the security of the entire system is a critical goal. Hybrid cryptography is an important defense against hubris. NIST should standardize and recommend hybrid cryptography as other European agencies have done.

- Patents: NIST should not adopt patented cryptography. Nevertheless, NIST appears to have licensed some, but surprisingly, perhaps not all possible patents that are likely apply to Kyber. This appears to be a positive outcome to correct the initial mistake of considering patented cryptography in the first place until one considers that the license does not appear to allow for variations. If for example, the pro-hash
faction wishes to deploy Kyber with the hash over the system randomness, the developer/user would need to seek their own patent licenses. If this patent analysis is incorrect, NIST should explicitly endorse variants as being covered under their patent license and encourage anyone who disagrees to settle it with NIST. This would increase confidence that the risk of patents could then entirely be borne by NIST. This sets aside the unnecessary waste in acquiring licensing for patents in the first case when patent free cryptographic alternatives also exist. NIST should ensure a patent license isn't used to restrict use in a way that will weaken security.

Risky security margins: rough comparisons to AES-128 is not conservative enough as a long term security goal in the quantum setting and bit security is only part of the picture. Related-key attacks against AES
are generally ignored in the security analysis. NIST has not clearly explained the AES security comparison process or the metrics in a way that promotes cryptanalysis. The confusion and disagreement about gate analysis on the pqc-forum show a large gap in any shared, scientific understanding. NIST should require much more conservative security margins, and any comparison between a post-quantum primitive and AES by NIST should be made public.

Delays: Other countries have already completed standardization of post-quantum cryptography. A lack of absolutely clear security metrics for security evaluation and a very slow standardization process are problematic. The delay in standardization has slowed and even stalled deployment of post-quantum cryptography. Mass surveillance adversaries have access to many more ciphertexts that will forever remain unprotected in the quantum adversary model. These delays are avoidable by having clear guidelines for all metrics, including public explanations or calculations of any NIST provided results, scheduling fixed dates in advance for key events over shorter periods of time, and publishing specific guidelines to assist with encouraging further cryptanalysis.

Little to no public accountability: NIST has not yet provided sufficient information to the American public for their participation in their last known standardized backdoor. At a time when trust in institutions is
reportedly very low, any further sabotage by NIST, witting or unwitting, will damage the national security far beyond embarrassment. As an example, Dual_EC_DRBG parameters have reportedly been changed in deployments of security sensitive American systems by non-American adversaries to the advantage of those same adversaries. NIST should provide full and total transparency on all matters related to cryptographic design and especially cryptographic sabotage issues.

I sincerely hope that NIST is not recreating another national shame as NSA and NIST did together with Dual_EC_DRBG. I also hope that this time NIST will listen to cryptographers and the larger security community who have urged a much more conservative approach to security, including a rejection of backdoors, or intentional weaknesses in critical national standards. If any NIST or NSA insiders have concerns about the standardization process such as misleading the public or regarding cryptographic sabotage, please consider blowing the whistle and drawing attention to the issues immediately. Courage is contagious, after all.

These comments are my own and I speak only for myself. Thank you for taking the time to read my comments. Please feel free to contact me directly for clarification on any of the above points.

Kind regards,
Dr. Jacob Appelbaum

[0]
https://research.tue.nl/en/publications/communication-in-a-world-of-pervasive-surveillance-sources-and-me

42. Comments from Logan, Cyberstorm, November 21, 2023

For ML-KEM, we don't believe in having SHA-2 and instead would prefer to have Keccak/SHAKE as Kyber-90s appears not to be actively maintained anymore. Please see https://github.com/open-quantum-safe/liboqs/pull/1465.

43. Comments from Michael Ravnitzky, November 22, 2023

Comment attached.

Comments on FIPS 203: ML-KEM

FIPS 203 presents the Module Learning with errors Key Encapsulation Mechanism (MLWE-KEM), a key-encapsulation mechanism (KEM) situated within the post-quantum cryptography domain, engineered to resist the potential onslaught of quantum computer attacks. A KEM is instrumental in generating and exchanging secret keys, pivotal for encrypting and decrypting messages. Post-quantum cryptographic algorithms are designed to maintain security against the formidable computational capabilities of both classical and quantum computers. Notably, quantum computers threaten the integrity of prevalent cryptographic algorithms such as RSA and ECC, which are predicated on mathematical problems that quantum computing could solve with relative ease.

The National Institute of Standards and Technology (NIST) deserves recognition for their meticulous work on FIPS 203, which specifies the Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM). This standard is a testament to NIST's dedication to enhancing secure communications in the quantum era. By focusing on key encapsulation mechanisms that are believed to be secure against quantum computer attacks, NIST is ensuring a path toward maintaining a robust and trustworthy cryptographic infrastructure, despite technological changes. Their thoughtful approach in developing these standards is invaluable for maintaining the confidentiality and integrity of sensitive information in the digital age.

The following observations aim to highlight areas of consideration for (and potential enhancement to) the FIPS 203 document.

Indistinguishability under IND-CCA Security

The document lacks a detailed explanation of how ML-KEM achieves indistinguishability under chosen ciphertext attack (IND-CCA) security, a critical security property for KEMs. IND-CCA security implies that an adversary despite having the capability to choose and alter ciphertexts, cannot differentiate between the shared secrets produced by the KEM, safeguarding the secrecy of keys and messages. The document should elucidate how ML-KEM achieves IND-CCA security, drawing from the KYBER scheme's security proofs and properties, and delineate the distinctions between KYBER and other schemes like FrodoKEM.

Impact of Parameter Choices

The document omits a discussion on the implications of parameter selection for ML-KEM's security and efficiency. It presents three parameter sets—ML-KEM-512, ML-KEM-768, and ML-KEM-1024—each representing varying security levels and trade-offs. These sets are sanctioned for securing sensitive, non-classified U.S. Federal

Government communication systems. However, the rationale behind these parameter choices, including the target security level against quantum threats, estimated quantum attack complexity, quantum hardware expectations, and the security margin's confidence interval, is not provided. The document should offer a comprehensive analysis of the parameter selection process and its impact on ML-KEM. Additionally, it should include comparative data, possibly in tables or graphs, benchmarking ML-KEM against other post-quantum KEM candidates and classical KEM schemes, focusing on key size, ciphertext size, computation time, memory usage, and security level.

Potential Vulnerabilities/Limitations in Practice

The document overlooks the potential vulnerabilities and limitations of ML-KEM when applied in practical settings. While it delves into the theoretical underpinnings of the algorithm—its mathematical foundation, description, and security analysis—it falls short of addressing real-world implementation challenges. Notably absent is a discussion on side-channel attacks and their countermeasures, interoperability with existing standards, testing protocols, and the legal ramifications of deployment. The document also omits consideration of insider threat risks. It would be beneficial for the document to explore the practical deployment of ML-KEM, detailing how potential issues might be mitigated. Furthermore, the document could benefit from an examination of ML-KEM's application across various use cases, such as secure communication channels, cloud computing environments, and blockchain technologies.

Resistance to Key-Recovery Attacks

The document fails to address ML-KEM's resistance to key-recovery attacks—attacks aimed at extracting the secret key from the public key or ciphertext. Unlike decryption attacks, which target the shared secret, key-recovery attacks threaten the security of both future and past communications using the compromised key. ML-KEM currently lacks explicit mechanisms to thwart such attacks, such as the incorporation of random nonces or hash functions in its algorithms. A vulnerability in the MLWE problem or ML-KEM's structure could potentially allow adversaries to retrieve the secret key, undermining the entire security framework. The document should provide evidence or reasoning to support ML-KEM's resilience against key-recovery attacks or suggest potential modifications to bolster its defenses.

Forward Secrecy

The document does not address the concept of forward secrecy, a critical feature that ensures the security of past sessions remains intact even if a long-term secret key is later compromised. This property is particularly vital in safeguarding data confidentiality against future quantum attacks. ML-KEM's current design, which directly utilizes the secret key for decapsulation, does not inherently support forward secrecy. As a result, a compromised secret key could potentially allow an adversary to decrypt historical ciphertexts. The document should articulate the reasons behind the absence of forward secrecy in ML-KEM or propose methodologies to integrate this property, thereby enhancing the long-term security of the system.

Characterization of the Attributes of the MLWE Problem

The document's reliance on the hardness of the MLWE problem, a relatively nascent and unproven challenge in cryptography, warrants scrutiny. While MLWE extends LWE—a well-examined problem—there may exist nuanced differences or characteristics that potentially simplify solving MLWE. For instance, MLWE's incorporation of matrices and modules could introduce exploitable structural or algebraic nuances. Additionally, emerging quantum algorithms might outpace classical counterparts in solving MLWE, thereby narrowing ML-KEM's security margin. The document should substantiate MLWE's difficulty relative to LWE and affirm the absence of efficient algorithms—classical or quantum—capable of solving MLWE. Furthermore, it would be prudent for the document to (periodically) refresh ML-KEM's security assessments to reflect ongoing advancements in MLWE and post-quantum cryptography research.

Limitations of Fixed Set of Parameters for Each Security Level

The document's adherence to a static parameter set for each security level may not align with the diverse requirements of various scenarios and applications. For example, ML-KEM-512 offers a smaller ciphertext size but lower quantum security, whereas ML-KEM-1024 ensures higher quantum security at the expense of increased ciphertext size. Certain contexts may necessitate a distinct balance between ciphertext size and quantum security. The current ML-KEM framework lacks the flexibility to tailor parameters, potentially constraining its applicability. The document should offer guidance on selecting the most fitting parameter set for specific scenarios and applications and consider introducing parameter set variations to cater to a broader spectrum of needs and preferences.

Impact of Error Correction on Functionality

The document seems to overlook the ramifications of error correction on ML-KEM's security and operational efficiency. Error correction enables the decapsulation algorithm to retrieve the accurate shared secret from a noisy ciphertext, which may be error-laden due to the inherent randomness or noise in the MLWE problem. Nonetheless, this technique adds complexity and overhead, potentially impinging on performance and security. For instance, error correction could enlarge the ciphertext, prolong decapsulation computation time, or elevate the likelihood of decryption failure. It might also diminish the shared secret's entropy, rendering it more susceptible to compromise. The document should delve into the impact of error correction on ML-KEM, examining the balance it strikes between security and efficiency. Additionally, the document should justify the selected error correction method and parameters, elucidating their rationale.

Impact of Parameter Generation on Security and Randomness

The document overlooks the significance of parameter generation on ML-KEM's security and randomness. This process, which establishes the module dimension, ring dimension, modulus, and error distribution, is pivotal for ML-KEM's integrity and security, influencing the MLWE problem's complexity, public key and ciphertext sizes, and decapsulation success rate. Yet, parameter generation is not without its assumptions and choices that could potentially inject bias or uncertainty into the algorithm. For instance, the quality of a random number generator, estimations of security levels, attack complexities, or the calibration of error distribution, all play a role in parameter generation. The document would benefit from a thorough exposition on the rationale behind parameter selection and its impact on ML-KEM's security and randomness. Additionally, it should elucidate the specifics of the parameter generation methodology and its sources. This would provide a clearer understanding of the underpinnings of ML-KEM's security model and its robustness against potential threats.

Potential Unorthodox Attacks on ML-KEM

Exploitation of Algebraic Structure: ML-KEM, based on the MLWE problem, may be susceptible to attacks leveraging the algebraic structure of module lattices. Quantum algorithms capable of linear algebra operations on quantum states could potentially exploit the additional properties and relations of module lattices, such as their decomposability into sub-lattices and inherent symmetries. This could lead to the recovery of the secret key or shared secret, thus compromising ML-KEM's security. Although such an attack is currently unlikely due to the complexity of performing linear algebra on quantum states and the novelty of exploiting module lattice structure, it cannot be ruled out as impossible.

Statistical Analysis of Error Correction: The error correction technique used in ML-KEM's decapsulation algorithm could be targeted by statistical analysis methods that differentiate between valid and invalid ciphertexts. This could facilitate a chosen ciphertext attack (CCA), threatening the IND-CCA security of ML-KEM. The error correction technique might inadvertently act as a distinguisher, providing insights into the shared secret or secret key. While this attack requires sophisticated statistical analysis and numerous ciphertext queries, making it difficult and uncertain, it remains a theoretical possibility.

Parameter Optimization Attacks: The fixed parameter sets for each security level in ML-KEM might be suboptimal for certain applications. A parameter optimization technique could potentially identify more suitable parameters for the MLWE problem and ML-KEM algorithm, enhancing performance or security. The ability to adjust parameters could lead to smaller, faster, or more secure ciphertexts or shared secrets, potentially breaching ML-KEM's security framework.

Quantum Computing Challenges

Brute Force Quantum Search: A hypothetical large-scale quantum computer performing quantum search operations poses a brute force threat to ML-KEM. Such a computer could search through all possible values compatible with the MLWE problem to find the secret key or shared secret. Although this method is highly improbable due to the current technological limitations and the complexity of the MLWE problem, it exemplifies the need to consider future quantum computing capabilities when assessing ML-KEM's security.

Advanced Quantum Cryptanalysis: As quantum computing research progresses, the development of advanced quantum cryptanalytic techniques could pose a significant threat to cryptographic algorithms. It is essential for FIPS 203 to consider not only the brute force quantum search but also the potential for quantum algorithms tailored to exploit specific vulnerabilities in the MLWE problem. The document should address the need for continuous monitoring of quantum algorithmic advancements and their implications for ML-KEM's security.

Quantum Decoherence and Error Rates: Quantum computers face practical challenges such as decoherence and high error rates, which could impact the effectiveness of quantum attacks. FIPS 203 might wish to discuss the current limitations of quantum computing and how they might delay the threat to cryptographic systems like ML-KEM. However, it should also emphasize the importance of preparing for future advancements that may overcome these hurdles.

Transition to Post-Quantum Algorithms: The transition from classical cryptographic systems to post-quantum algorithms will be a complex process. FIPS 203 should outline strategies for a smooth transition, including the adoption of hybrid cryptographic systems that combine classical and quantum-resistant algorithms. This ensures backward compatibility and security during the transitional period.

Standardization of Quantum-Resistant Protocols: The standardization of quantum-resistant protocols is crucial for long-term security. FIPS 203 should advocate for the development and standardization of such protocols, ensuring that infrastructure is ready to withstand quantum computing threats. This includes not only key encapsulation mechanisms like ML-KEM but also other cryptographic primitives.

In summary, while FIPS 203 lays a solid foundation, and is based on a thorough and transparent evolution process conducted by NIST, it is imperative that the document evolves to address the practical implementation challenges, potential unorthodox attacks, and the ever-changing landscape of quantum computing threats. The inclusion of detailed discussions on parameter generation, error correction, and the transition to quantum-resistant protocols would enhance the robustness and longevity of the standard. It is recommended that continuous reassessment and updates be incorporated into the standard to ensure that it remains resilient against both current and future cryptographic threats. The collaborative effort of the cryptographic community in identifying and mitigating these challenges will be crucial in safeguarding the security of sensitive communications in the post-quantum era.

Michael Ravnitzky
Silver Spring, Maryland