Public Comments on draft FIPS 205

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

On August 24, 2023 NIST requested comments on the initial draft FIPS 205, Stateless Hash-Based Digital Signature Standard. The comments that NIST received during the comment period are collected below.

LIST OF COMMENTS

1. Comments from Simon Hoerder, August 24, 2023

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

2. Comments from Scott Fluhrer, August 25, 2023

This comment is not on the cryptography of the proposed standard, but how it is explained.

The current text gives the formatting of the ADRS structure for all the various use cases, and then gives one paragraph that explains how to convert from an ADRS structure to the compressed $ADRS^c$ format.

My suggestion would be to explicitly list both the format of the ADRS and the $ADRS^c$ structures for all of the use cases. This obviously increases the length of the text; however, it would make how things are laid out in the SHA-2 case more obvious.

3. Comments from Anubhab Baksi, 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

4. Comments from the Netherlands National Communications Security Agency, September 6, 2023

I am very grateful for the many clarifications that NIST added to the draft Specifications that were released on august 24.

The ML-KEM and ML-DSA specifications are much clearer than the Kyber and Dilithium specifications and I am glad to see it.

Especially the specification of the NTT and inverse NTT algorithms makes the ML algorithms much easier to implement.

I have also seen that the SLH-DSA specification is a great improvment over the SPHINCS+ specification.

However, I find that there are still some minor issues with the SLH-DSA specification.

1. It is unclear to me whether ADRS function arguments should pass by value or by reference.

2. In my opinion, it would be better to remove the lines 1, 2, and 3 (If i + s >= w, ...) in Algorithm 4, and replace it with a comment like "The chaining function requires i + s < w".
I can imagine a developer writing "assert i + s < w" during development. But in production I would leave it out. A correct implementation will never call chain with i + s >= w, this is easy enough to prove.

3. In algorithm 8 there is a similar issue. I would replace lines 1, 2, and 3 with a comment.

4. In algorithm 6: wots_sign it is somewhat unclear if "skADRS <- ADRS" means "make a copy".
In algorithm 5: wots_PKgen, line 10, a comment indicates that a copy should be made.
A developer could interpret the lack of a similar comment in algorithm 6 to mean that making a copy is not needed.

5. Algorithm 7: wots_PKFromSig, line 15 has a similar issue.

We hope our input is useful to your team, and we wish you the best of luck in finishing the specs.

5. Comments from Elaine Barker, September 8, 2023

Attached are my comments on FIPS 205; I can put them in a template if you like. I also attached a powerpoint presentation that I attempted to develop to better understand what is going on (just the generation and signature processes, so far). Something along these lines might be useful for the readers (with more work, of course).

Elaine

Line 43	800-208 specifies the stateful hash-based signature. The names are awfully close.
Line 97	Are the allowed algorithms in FIPS 140-3 necessarily in an approved status for govt. use?
Line 267	Consider not including sections 1.2 and 1.3 in the final FIPS, but providing the content in 1.2, 1.3, and Appendix A in a separate document (e.g., a NISTIR).
Line 290	The SHA-2 family of hash functions
Line 308	the SHA-2 family
Line 355-356	Would it helpful to identify what is sufficiently long here? If I'm reading 202 right, the length needs to exceed (r+c)/2 = 800 bits (Appendix A.1 in FIPS 202)
Line 449-450	Would it be useful to include that the leftmost bits are 0s?
Line 451-452	Would it be useful to include that the rightmost bits are 0s?
Line 458-459	Since "must" is essentially a requirement word, should "is" replace "mist be" here, with the actual requirement stated in the main body, say in 3.1?
Line 462-463	This is hard to parse. How about "numbering" the schemes, e.g., "1) a few-time signature scheme. forest of random subsets (FORS) and 2) a multi-time signature scheme, the eXtended Merkle Signature Scheme (XMSS)." [assuming I understand this is what is intended]

Line 460	Shouldn't the key and signature sizes be included (as they are in NISTIR 8413)?
Line 473-481	This needs a more detailed figure and explanation with examples of just one Merkle/XMSS tree. Maybe in an appendix? The value of h' is in Table 1. The example could be of h'=3, so there would be 6 WOTS+ keys at the bottom of the tree. Show how the path works (as is done in 800-208).
Line 482-491	Use another figure to show the hypertree. The value of d also comes from Table 1 (I think). show how this works and explain in text.
Line 492-497	Include a diagram showing the structure and contents of the public and private keys.
Fig 1 Caption	So where does the signature come out?
Line 510-517	This is VERY complicated!
Line 521	Remove the comma (after ‘take into consideration’)
Line 536	Should this be "private key"?
Line 541	Insert a comma after "required"
Line 549	Don't think these are necessarily the approved SHA-3 hash functions
Line 550-553	explain this notation in Secion 2.3
Line 573	Remove the comma (after ‘4 bytes long’)
Line 568	Change to "...input: PRF, T_subl H, and F. ADRS is structure for the parameters to be provided to each function." (at least that's what it appears to be to me.)
Line 568-569	Change to something like "Each ADRS indicates the position in a hypertree whose value is being computed by the function."
Line 575	How about using "position" here; using "height \" get confusing.

Line 576	How about "hypertree position" here?
Line 576	How a bout "hypertree position" here as well?
Line 589-591	This needs further explanation and probably an example. Is this referring to the paths indicated in Figure 1 for a single XMSS tree?
Line 597-598	Need to explain this (e.g., when explaining the XMSS tree, as suggested above)
Line 598-599	Ditto
Line 616-620	Consider including figures for these as well.
Line 942	Have we made a decision to allow a XOF? Are there specific output lengths to use? Would OIDS be needed for negotiation?
Line 963	Specify SHA-512 for categories 3 and 5 as discussed in 1.2? (Table 1 Caption)
Line 964-967	Referring to 800-56B may not be the best idea, since we intend to transition away from it. Better. to include the MGF here.
Line 971	Consider using another document (e.g., a NISTIR) for this so they are not duplicated for each of the PQ FIPS.
Line 973	approved? not all the block ciphers are equally secure. You might check 800-57 Part 1 to see if the description there would be useful
Line 1011	Why require SHA-256 and SHA-512 in the same implementation? Make them all SHA-512?

6. Comments from Ben Livelsberger, September 21, 2023

As I've been implementing the pseudorandom and hash functions defined in sections 10.1 - 10.3 of the draft FIPS, it struck me that the definition of F() in section 10.3 may be incorrect. In sections 10.1 - 10.2, F() = H() = Tl(). In section 10.3, H() = Tl(), but F() is its own unique function (the same function as in 10.2). I wondered if this might be a typo and whether it should be the case that F() = H() = Tl() for section 10.3? Or is the definition for F() in section 10.3 correct as it stands in the draft?

Thank you!

Ben

7. Comments from Lingyun Li, October 12, 2023

Dear NIST PQC Team,

Please find attached my comment on the "FIPS 205 (Initial Public Draft) Stateless Hash-Based Digital Signature Standard".

I appreciate the opportunity to provide feedback on FIPS 205, and look forward to the further development of this standard. Thank you for your time and consideration.

Sincerely,
Lingyun Li
Institute of Information Engineering,
Chinese Academy of Sciences

Comment attached.

Comment on "FIPS 205: Stateless Hash-Based Digital Signature Standard"

In section 10.2 and 10.3 of FIPS 205, when instantiating the four underlying functions PRF, F, H and T_l for multiple invocations using SHA-2, padding is applied to PK.seed to a full block to reduce redundant calculations. I would like to propose an alternative design approach of a Hash function based on the SHA-2 function, which is named $\text{SHA2}^{\text{SLH-DSA}}$. The objectives of designing $\text{SHA2}^{\text{SLH-DSA}}$ function are as follows: 1. Expanding the range of constant inputs: including SK.seed as constant input parameter like PK.seed. 2. Opening up the compression function of the Hash function, using the number of computational steps within the compression function as the fundamental unit to measure the performance of the SLH-DSA scheme, aiming to enhance performance while minimizing the number of input blocks or reducing redundant calculations within the underlying compression function. 3. Ensuring that $\text{SHA2}^{\text{SLH-DSA}}$ provides a comparable level of security to the original SHA-2 function.

- **The design methodology for $\text{SHA2}^{\text{SLH-DSA}}$ function**

Compared to the design of SHA-2, the differences of $\text{SHA2}^{\text{SLH-DSA}}$ are as follows when instantiating the four underlying functions for the SLH-DSA scheme:

1. Segmentation of Function Inputs. The function's inputs are divided into constant parameters and variable parameters. PK.seed and SK.seed are treated as constant input parameters, while other inputs are considered variable parameters.

2. Elimination of PK.seed Padding. Padding for PK.seed is removed.

3. Modification of the position of SHA-2's default padding. The position of SHA-2's built-in padding is altered, placing it after the constant parameters and before the variable parameters. Once the message length is determined, this padding remains fixed and is also categorized as a constant parameter. The constant parameters only need to be processed once across multiple function invocations.

$\text{SHA2}^{\text{SLH-DSA}}$ and SHA-2 both employ the same Merkle–Damgård iteration structure, compression function, and initial vector IV. $\text{SHA2}^{\text{SLH-DSA}}$ includes two functions: $\text{SHA-256}^{\text{SLH-DSA}}$ and $\text{SHA-512}^{\text{SLH-DSA}}$. Correspondingly, $H_{\text{SHA-256}}{}^{\text{SLH-DSA}}$ and $H_{\text{SHA-512}}{}^{\text{SLH-DSA}}$ represent the compression functions of

SHA2$^{\text{SLH-DSA}}$ and SHA-512$^{\text{SLH-DSA}}$, respectively. Here, for SHA-256$^{\text{SLH-DSA}}$, b = 512, n = 256, and for SHA-512$^{\text{SLH-DSA}}$, b = 1024, n = 512, b and n denote the block length and output length, respectively. Same as in sections 10.2 and 10.3, SHA-256$^{\text{SLH-DSA}}$ is used for instantiating the four underlying functions of SLH-DSA for Security Category 1; SHA-256$^{\text{SLH-DSA}}$ and SHA-512$^{\text{SLH-DSA}}$ is used for instantiating them for Security Categories 3 and 5.

The standard instantiation of underlying functions of the SLH-DSA scheme based on SHA-2 is shown in Figure 1.

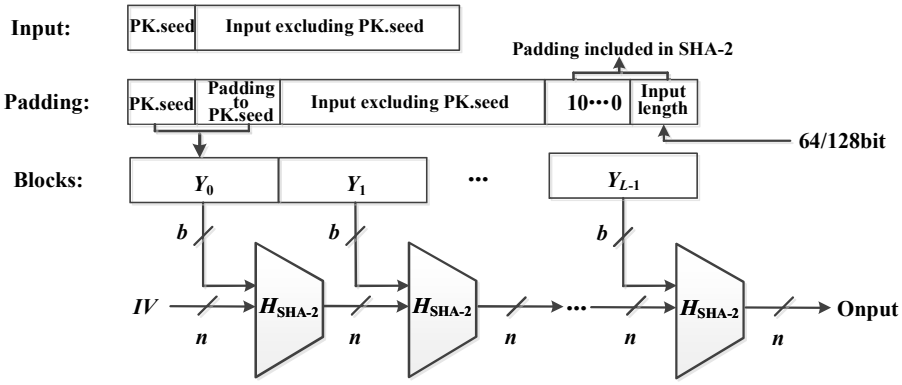


Figure 1: Instantiation of underlying functions of the SLH-DSA scheme based on SHA-2

Instantiation of underlying functions of the SLH-DSA scheme based on SHA-2$^{\text{SLH-DSA}}$ is shown in Figure 2.

- **Performance**

In the following, we compare the performance of the instantiation of the four underlying functions of the SLH-DSA based on SHA-2 and SHA2$^{\text{SLH-DSA}}$. It compares the reduction in the number of input message blocks, as well as the reduction in the number of steps required to process constants and variables of the input for the instantiation functions of PRF, F, H and T_l in the latter compared to the former. The analysis uses the internal computational steps of the compression function as the fundamental unit to measure the instantiation performance of the SLH-DSA scheme.

Here are the meanings of the various parameters:

- Original Block Count: The total number of input blocks when instantiated based on SHA-2.

- Current Block Count: The total number of input blocks when instantiated based on SHA2$^{\text{SLH-DSA}}$.

- Block Count Reduction: The reduction in the total number of input blocks, calculated as the difference between the original block count and the current block count.

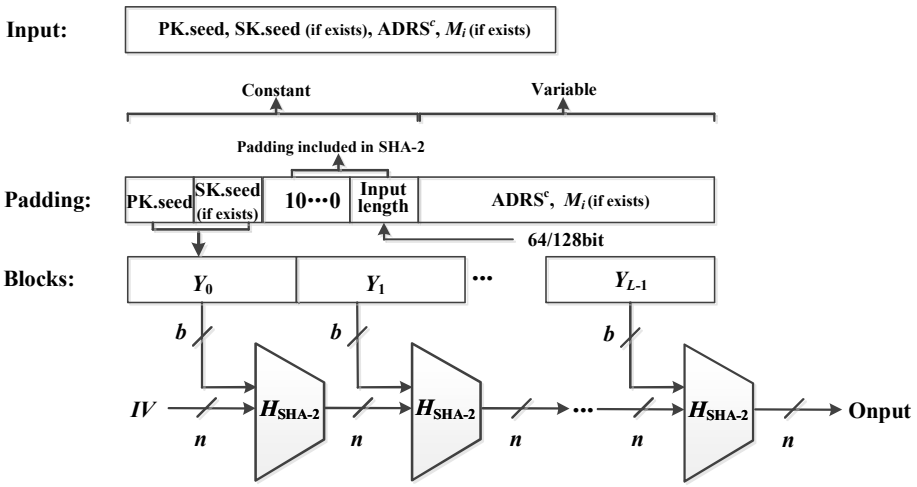


Figure 2: Instantiation of underlying functions of the SLH-DSA scheme based on SHA-2$^{\text{SLH-DSA}}$

- Block Reduction Ratio: The ratio of the reduction in the total number of input blocks for the current block count compared to the original block count.

- Block Reduction Rate of Occurrence: The probability of the aforementioned block reduction occurring.

- Original Constant Calculation Steps: The number of steps required to process input constants when instantiated based on SHA-2.

- Current Constant Calculation Steps: The number of steps required to process input constants when instantiated based on SHA2$^{\text{SLH-DSA}}$.

- Reduction Ratio of Constant calculation: The ratio of the reduction in calculation steps for the current constants compared to the original constant.

- Original Variable Calculation Steps: The number of steps required to process input variables when instantiated based on SHA-2.

- Current Variable Calculation Steps: The number of steps required to process input variables when instantiated based on SHA2$^{\text{SLH-DSA}}$.

- Reduction Ratio of Variable calculation: The ratio of the reduction in calculation steps for the current variables compared to the original variables.

The performance for the instantiation of the underlying functions of the SLH-DSA scheme based on SHA-2 and SHA2$^{\text{SLH-DSA}}$, considering security categories and specific functions, are presented in Table 1 and Table 2, respectively. It is worth noting that several parameters related to the function T_l in the table are represented as "-", as a result of the need to categorize discussions based on different values of l. The analysis result for the function T_l is averaged across all possible values of l after a comprehensive discussion.

To begin with, when considering the number of blocks in the input, classified based on security categories and underlying functions respectively, the average of the overall block reduction ratio is 21.53%; the average block re-

Table 1: Performance Analysis of SLH-DSA Scheme Instantiated with Different Security Categories based on SHA-2 and SHA-2$^{\text{SLH-DSA}}$

Security Category	Function	Original Block Count	Current Block Count	Block Count Reduction	Block Reduction Ratio	Block Reduction Rate of Occurrence	Original Constant Calculation Steps	Current Constant Calculation Steps	Reduction Ratio of Constant Calculation	Original Variable Calculation Steps	Current Variable Calculation Steps	Reduction Ratio of Variable Calculation
1	PRF_{sk}	2	1	1	50.00%	100%	64	10	84.38%	64	54	15.63%
1	T_l	-	-	-	29.17%	75%	64	-	62.50%	-	-	7.81%
1	F	2	1	1	50.00%	100%	64	6	90.63%	64	58	9.38%
1	H	2	2	0	0.00%	0%	64	66	-3.13%	64	62	3.13%
				Average	32.29%	68.75%	64	-	58.59%			8.98%
3	PRF_{sk}	2	2	0	0.00%	0%	64	74	-15.63%	64	54	15.63%
3	T_l	-	-	-	50.00%	100.00%	80	-	83.75%	-	-	16.25%
3	F	2	2	0	0.00%	0%	64	68	-6.25%	64	60	6.25%
3	H	2	1	1	50.00%	100%	80	7	91.25%	80	73	8.75%
				Average	25.00%	50.00%	72	49.67	38.28%			11.72%
5	PRF_{sk}	2	2	0	0.00%	0%	64	74	-15.63%	64	54	15.63%
5	T_l	-	-	-	29.17%	75%	80	-	41.25%	-	-	19.06%
5	F	2	2	0	0.00%	0%	64	66	-3.13%	64	62	3.13%
5	H	2	2	0	0.00%	0%	80	85	-6.25%	80	75	6.25%
				Average	7.29%	18.75%	72	75	4.06%			11.02%
				Overall Average	21.53%	45.83%			33.65%			10.57%

Table 2: Performance Analysis of SLH-DSA Scheme Instantiated with Specific Functions based on SHA-2 and SHA-2$^{\text{SLH-DSA}}$

Security Category	Function	Block Reduction Ratio	Block Reduction Rate of Occurrence	Reduction Ratio of Constant Calculation	Reduction Ratio of Variable Calculation
1	PRF_{sk}	50.00%	100.00%	84.38%	15.63%
3	PRF_{sk}	0.00%	0.00%	-15.63%	15.63%
5	PRF_{sk}	0.00%	0.00%	-15.63%	15.63%
	Average	16.67%	33.33%	17.71%	15.63%
1	T_l	29.17%	75.00%	62.50%	7.81%
3	T_l	50.00%	100.00%	83.75%	16.25%
5	T_l	29.17%	75.00%	41.25%	19.06%
	Average	36.11%	83.33%	62.50%	14.38%
1	F	50.00%	100.00%	90.63%	9.38%
3	F	0.00%	0.00%	-6.25%	6.25%
5	F	0.00%	0.00%	-3.13%	3.13%
	Average	16.67%	33.33%	27.08%	6.25%
1	H	0.00%	0.00%	-3.13%	3.13%
3	H	50.00%	100.00%	91.25%	8.75%
5	H	0.00%	0.00%	-6.25%	6.25%
	Average	16.67%	33.33%	27.29%	6.04%
	Overall Average	21.53%	45.83%	33.65%	10.57%

duction rate of occurrence stands at 45.83%. These findings reveal a marked reduction in the number of blocks for the underlying functions.

When examining the reduction in computational steps for the constant portion, categorized by underlying functions and security categories, the overall reduction ratio of constant calculation is 33.65%. The underlying functions exhibit a significant reduction in computational steps for the constants. The reduction ratio of computational steps for the variables is the most critical parameter. categorized by underlying functions and security categories, the overall reduction ratio of variable calculation is 10.57%. Consequently, there is an overall trend of reduction in computational steps for the variables.

- **Security**

In the instantiation of the SLH-DSA scheme, the underlying SHA-2 function is regarded as a computational collision-resistant Hash function. Here, $\mathrm{SHA2}^{\text{SLH-DSA}}$ function also possesses computational collision resistance security.

Theorem 1 *$SHA2^{SLH\text{-}DSA}$: $\{0,1\}^* \rightarrow \{0,1\}^n$ is a collision-resistant Hash function if its compression function $H_{SHA\text{-}2}{}^{SLH\text{-}DSA}$: $\{0,1\}^b \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a collision-resistant compression function.*

Proof: Omitted.

In summary, an alternative design approach for a Hash function based on SHA-2 is presented, which can enhance the performance of the SLH-DSA scheme when instantiating its underlying functions. Its drawback is that it is not a widely adopted standard Hash function, we measure the performance of SLH-DSA scheme instantiation based on the number of computational steps within the compression function and require targeted recording of the results of constant part calculations for different underlying functions.

8. Comments from Siwei Sun, October 23, 2023

Dear Dustin,

I have some questions on the draft of FIPS 205.

In Algorithm 18 slh_sign(M, SK), md is the first \ceil{ka/8} bytes of the computed digest. Subsequently, in Line 21, md is signed with a FORS instance.
However, a FORS instance can only sign ka-bit messages. In the original SPHINCS+ submission, the first ka bits is extracted and signed by a FORS instance.

So, in this draft, it is a mistake or there are some other changes to make the FORS instance sign a \ceil{ka/8}-byte message?

Thank you very much.

Best regards,
Siwei

9. Comments from Danny Niu, October 25, 2023

In NIST.FIPS-205.ipd, section 2.3 Mathematical Symbols, line 434, it says:

> ... outputs the most significant l bytes of ...

The use of "most significant" is problematic because it means that, a byte is multiplied with the highest power of log_2(BYTE_BITS) in a machine integer word.

When talking about a byte string, such as the one Trunc operates on, it's better to say "left-most" (since majority of the world will interpret it as logically foremost) or equivalent wording if found appropriate.

In page 15, the pseudocodes for base_2^b, the variable 'baseb' is used - first in the outer for loop, and then returned. However, this variable is not declared or initialized before (although it can be deduced to be initialized as array of bit strings / integers.

Step 1-3 of the xmss_node and fors_node currently return NULL as an error indication. I propose the following change for consideration as improvement.

These subroutines are not externally visible, so mis-using them by providing erroneous parameters is unlikely. Also, returning NULL seem to suggest that it's a valid return value, instead of an indication of error, what's more, that error is not handled by any of their callers.

I suggest changing the behavior of each of these 3 lines in xmss_node and fors_node to asserts/invariants/report-error-if's.

10.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

11.Comments from Niklas Börbacke, November 3, 2023

Erratum:

FYI; On line 26 in Algorithm 16 on page 32 the line ends with a semicolon (it is "return pk;". No other return statement in the standard ends with a semicolon, so I assume that this is a typo.

Regards,
Niklas Börbacke

12.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:

Lines 265, 522 and 540 - As SP800-89 is now a dated publication (published in 2006 and largely referencing FIPS PUB 186-3 exclusively using SHA1) we'd suggest references are updated to the specific section from that that that is expected to be applied. In particular as written that standard explicitly covers key validation for DSA, ECDSA and RSA - as such it's not immediately obvious what parts of that standard are expected to apply in the case of validating public keys from SLH-DSA.

Proposed fix - Add an explicit reference in the test to the sections from SP 800-89 that FIPS 203 expects to apply in the context of SLH-DSA. Independent of this, SP 800-89 should be updated given its recent reference from FIPS PUB 186-5 and proposed reference from FIPS PUB 203, 204 and 205.

Comment 2:

Lines 526-528 - To avoid potential confusion with available options to an 'Approved RBG' we'd propose explicitly stating here that "values shall be taken from the output of 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.

A consistent update should be made to lines 898-899 that separately reference the suite of three publications associated with the requirement to use an approved random bit generator.

There is potential for confusion by referencing all three standards. In particular, whether outputs from a 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 use of a DRBG and separately makes it clear that DRBG may be seeded using external noise sources.

Comment 3:

Line 526 - Linked to the comment above, we propose removing the word 'freshly' from the sentence 'For each invocation of key generation each of these values shall be **freshly** generated using an approved random bit generator'. Provided entropy has been protected, there is no requirement or benefit to requiring entropy be generated at the point of generating a signature and where this has the potential to impact the performance of many modules where entropy is generated during periods of inactivity and stored in an entropy pool awaiting use. This standard shouldn't introduce any additional requirements on entropy (intentional or not) that aren't present in the NIST CTG standards for Approved RBG.

If the worry is that entropy in storage could be compromised or corrupted, the same would apply to keys used by this algorithm.

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

13.Comments from John Gray, November 17, 2023

In section 9.4 of SLH-DSA and 7.1 there is mention of accommodating pre-hashing. This is great to see! I am requesting that you define separate Object Identifiers for the pre-hashed versions verses the non pre-hashed versions.

Since you accommodate for prehash in FIPS 204 and 205, having a separate OID definition or defined domain separation would prevent application incompatibility, otherwise applications would need to have pre-shared information amongst themselves (or try multiple times in the case of failures, which isn't helpful). So please define either different OIDs or domain separated versions in those drafts. I think using different OIDs would be the easiest for implementors otherwise the format of the message would have to first be parsed to determine which method to use (and anytime we implementors have to parse stuff opens up the possibility of errors)… 😊

Thanks for receiving my comments.

In regards to SLH-DSA, another comment I have is whether it is possible to specify less conservative parameters which would reduce the size of the signatures. I think in practice people will not be performing 2^60 signatures on anything. For example, a PKI that serves a large organization of 10,000 people are likely not going to perform even 2^24 signatures in the lifetime of its root key. Shorter lived intermediate CA's would likely perform even less. Having a greater choice of parameter sizes to help mitigate the size of SLH-DSA signature size with guidelines on security and expected number signatures would be helpful.

Cheers,

John Gray
Senior Principal Software Applications Developer

14. Comments from the National Security Agency, November 20, 2022

Main comment: Having 12 different parameter sets is not conducive to inter-operability and may cause practical usage issues if different groups implement different versions. We suggest NIST chooses a smaller number of options. For instance, NIST should select either the small signature parameters or the fast signature parameters, but not both.

Typos and minor comments:

- Line 221 – The example of floor should be floor(-2.1)=-3.
- Line 633 – 'String X' should likely be 'array X'.
- Line 639 – x_{n-2}, x_{n-1} rather than the opposite order.
- Top of section 4.4 - To help developers, suggest clarifying that the n-byte strings will always have n<= 8.
- Footnote 6 – typo at versus as. Also, suggest "in order to simplify notation" at the end of the footnote.
- Algorithm 4: Suggest to indicate that a 0 start index indicates the "bottom" of a chain.

15.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.

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

16.Comments from Markku-Juhani Saarinen, November 21, 2023

Dear NIST,

There has been some discussion in the IETF LAMPS WG around SLH-DSA signatures. Credit is due to { Falko Strenzke, David A. Cooper, Scott Fluhrer, Mike Ounsworth, John Gray, Tim Hollebeek, Darren Johnson, .. }, even though I'm to blame for errors in the following and some members of this group may prefer other solutions to the one I am proposing below.

From an engineering perspective, it is non-ideal that SLH-DSA, as specified in FIPS 205 IPD, potentially ("sometimes") requires two passes over the message M to sign it. The specification leaves it to the application whether M is the message itself or a hash M = H(M'). There is no built-in domain separation between M and H(M').

Quote from Section 9.4 ("Prehash SLH-DSA"): *"For some use cases, these issues may be addressed by signing a digest of the message rather than signing the message directly. In order to maintain the same level of security strength, the digest that is signed needs to be generated using an approved hash function or extendable-output function (XOF) (e.g., from FIPS 180-4 [12] or FIPS 202 [10]) that provides at least 8n bits of classical security strength against both collision and second preimage attacks [10, Table 4]."*

Hence, a user (adversary) can seemingly state to the verification function whether a (say) 512-bit blob M is a message itself, or hash M=SHAKE256(M1), M=SHA2-512(M2), or M=SHA3-512(M3) -- the same signature authenticates all. In this scenario, EUF-CMA -- the stated security goal of FIPS signature standards -- is potentially violated since the same signature that was queried for M1/M2/M3 is also a valid signature for M itself (different message).

This is a very general issue, but here's a little example scenario: Imagine a remote firmware update that is authenticated with a signature of M = SHA3-512(firmware). An adversary modifies the metadata to state that, "oh, here actually, M=firmware itself -- the update just happens to be only 512 bits long." Since the metadata is not authenticated by the signature, the device replaces its firmware with that garbage. As a result, the device is permanently bricked (perhaps by remote control.)

Proposed change:

Eliminate the option to sign M directly -- always require a hash input as M. Consequently, the signing function never requires the M to be hashed twice.

There are several options to do this. I'd prefer to align the interface and signature logic with ML-DSA / Dilithium (and to implement the same interface later for FN-DSA / Falcon as well). The module would always be passed "mu" to represent the message to be signed/verified: mu = H(PK || M)

Where PK = (PK.seed, PK.root) in the case of SPHINCS+. This maps to "tr" in Dilithium, albeit without the hash (in Dilithium tr=H(PK)). In both cases, mu has at least "double" (collision-resistant) length -- perhaps always 512 bits as in Dilithium.

Hence, in the first instance of M in the signing, just change M to "mu":
Alg. 18, slh_sign(). Line 7: R <- PRF_msg(SK.prf, opt_rand, mu)

Since the PK is already in "mu", we may remove it from "digest":
Alg. 18, slh_sign(). Line 10: digest <- H_msg(R, mu)
Alg. 19, slh_verify(). Line 9: digest <- H_msg(R, mu)

Note the mapping; R = "rho'" in Dilithium: SK.prf = "K" in Dilithium, opt_rand = "rnd" in Dilithium. This corresponds to the BUFF transform (See Fig 6 https://ia.cr/2020/1525), which Dilithium has.

Regards,

Dr. Markku-Juhani O. Saarinen

Professor of Practice, Tampere University

17.Comments from the Canadian Centre for Cyber Security, November 21, 2023

Hello,

Please find attached our comments for the FIPS 205, Stateless Hash-Based Digital Signature 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.

Regards,

Comment attached.

FIPS 205: SLH-DSA Comments

Small edits:

- Line 304: "the match" should be changed to "to match".
- Line 585, 592, 596, etc.: Writing "WOTS_HASH (defined to be 0)", instead of "WOTS_HASH (0)", makes the meaning of the value in parenthesis easier to understand. We recommend that NIST adopts this formulation throughout section 4.2.

Changes to presentation:

- The name of section 5, which is the type of scheme considered, is markedly inconsistent with the name of sections 6, 8, and 9, which are the specific scheme considered. We recommend renaming section 5 to "The Winternitz One-Time Signature+ Scheme", or otherwise make these section titles more consistent.
- On line 620, we recommend that NIST moves the definition of the "c" superscript for an address string from line 620 to somewhere in section 10, or at the very least recall this definition in section 10.
- Page 11, Footnote 4: We recommend that NIST moves the text of footnote 4 into the main body of section 4.2. The text of footnote 4 explains the relation between the constants "WOTS_HASH" et. al. and their numerical values much more clearly than "WOTS_HASH (0)" (line 585), "WOTS_PK (1)" (line 592), etc.

Suggested clarifications:

- The full name of the algorithm is "Stateless Hash-Based Signature Standard" and the scheme is described as "stateless" several times in the document. We recommend that NIST explains what is meant by "stateless" in the introduction and in section 9 (line 879), perhaps by directly contrasting with the existing stateful hash-based signature schemes.
- Lines 296-305: In the summary of changes from versions 3 and 3.1 we recommend also mentioning that the xmss_node and fors_node algorithms have been modified, in particular, in how they expect to receive their inputs.
- We recommend that NIST explicitly states the meaning of the abbreviations "sec level", "pk bytes" and "sig bytes" in Table 1 on page 38. In particular, the table is titled "SLH-DSA parameter sets", but the security level is not a "parameter" of the scheme.
- While the current draft does specify that the individual components should not be used individually, this is relegated to the last page (page 47). We recommend that NIST adds more prominent warnings throughout the draft that these components are insecure if used in a standalone manner. Such warnings could and should appear at the start of sections 5, 6, 7, and 8 as each of these sections describe signing algorithms (and, implicitly, verification algorithms) which are not guaranteed to offer any meaningful notion of security.

- Lines 553, 557: Pseudorandom functions are standard objects of study in cryptography and are typically defined to take two arguments (for example in section 3 of SP 800-108r1 they are described in this way). Moreover, these arguments have distinct roles in the standard definition of security for these objects. The PRFs defined in this standard take three arguments, and it is unclear how to understand the role of these three arguments in the context of the usual security definition. We recommend that NIST clarifies what they mean when they write "is a PRF" on lines 553 and 557.
- We recommend that NIST provides clearer guidance concerning randomized and deterministic signing in SLH-DSA.
    - In section 9.2, algorithm 18: We recommend that NIST adds an explanation, or a reference to an explanation, as to why the value opt_rand used, does not need to be generated by an approved RBG. Such an explanation is given in the SPHINCS 3.1 draft (Section 8.1.5) but has been omitted in the SLH-DSA draft.
    - In the draft ML-DSA standard, NIST makes the use of randomized signing a "**should**" clause of the standard. The SLH-DSA standard currently does not offer the same level of clarity as to when the randomized variant and the deterministic variants are to be used. We recommend that NIST make a clearer recommendation on the usage of these two variants.
- The use of the "RANDOMIZE" Boolean in Algorithm 18 is not discussed anywhere in the text, making it unclear if this value is considered a parameter of the scheme or an input to the algorithm. We recommend that NIST clarifies this question further. Related to this point, we recommend that NIST rewrites lines 4 to 6 of algorithm 18 in the SLH-DSA draft in the same style as line 7 of algorithm 7 in the ML-DSA draft.
- We recommend that the standard include additional clarifications on whether it is acceptable or not to use the same key pair for both the randomized and deterministic variants. Note that in the FIPS 186-5 Digital Signature Standard, ECDSA also has deterministic and randomized versions, and section 6.2 explicitly states that “(Deterministic) ECDSA keys **shall** only be used for the generation and verification of (deterministic) ECDSA digital signatures.” We recommend providing a similar clarification in this standard.
- Algorithms 4, 8, 14: In algorithm 15, it is specified that NULL is the zero-length byte string. We recommend that NIST make it clearer if this is also true for the NULL values appearing in algorithms 4, 8, and 14. If this is the case, then this definition should be added to section 2.3.

18.Comments from Cloudflare, November 21, 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/

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.

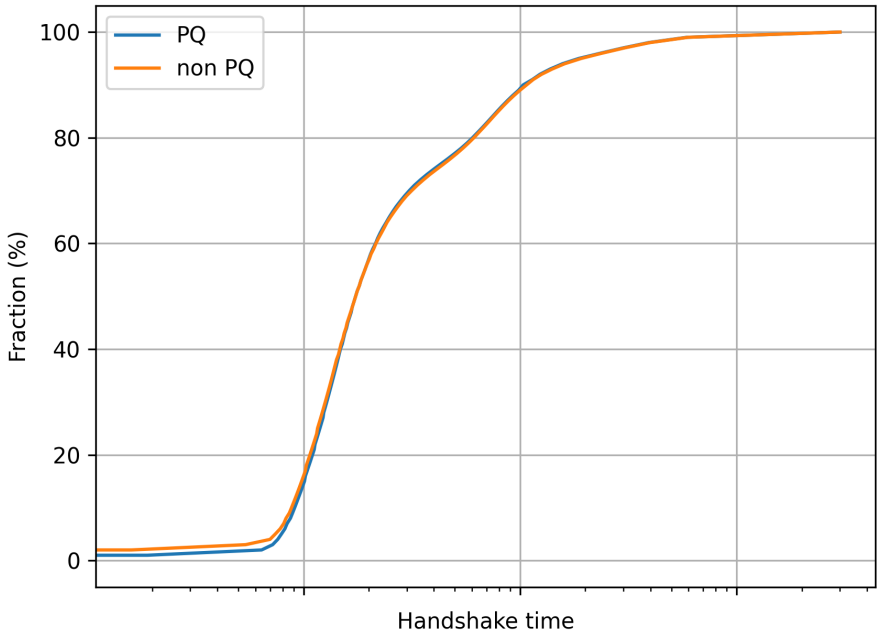
The designers of SHAKE / SHA3 have reiterated their 2016 proposal to use a variant of 12 rounds as was originally proposed, which has now been dubbed “TurboSHAKE”. We are comfortable with the wide security margin of the 12-round variant, and would welcome NIST *replacing* SHAKE / SHA3 in these drafts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



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.)

19.Comments from Dev Null, November 21, 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.

20.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

21.Comments from Falko Strenzke, November 22, 2023

The following proposal is a result of the recent discussion on the LAMPS list about safe signature-separability defences and safe optional pre-hashing for the PQC signature schemes with contributions from David Cooper, Markku Saarinen, Darren Johnson and others. However, with this specific form of proposal I still can only claim to speak for myself.
I propose to build the following two features into ML-DSA (FIPS 204) and SLH-DSA (FIPS 205):

1. Introduction of a binary flag in the algorithm's interface that gives domain separation between the two uses of directly signing the message and signing the hash of the message (pre-hashing).
2. Introduction of a context string into the algorithm's interface of a length between 0 and 255 octets that allows the protocol or application to define a custom domain separation string.

This proposal is meant to be equivalent to the construction of the respective features of Ed25519ctx/Ed25519ph or Ed448/Ed448ph in RFC 8032.
The reason for the first point is the need to avoid any ambiguity with respect to what is the signed message given the alternate options of direct-signing and sign-pre-hashed-message. When lacking this domain separation feature in ML-DSA, protocols that allow both variants and cannot provide authentic information about the variant during signature verification will be subjected to signature forgery attacks.
The reason for the second point is that currently there are plans to design composite signature schemes that fulfil non-separability notions, i.e. make it impossible that a signature is stripped from the set of signatures in a composite signature and the remaining signature(s) appear as (a) valid signature(s) of the message. Existing protocols that cannot provide for authentic information about the nature (composite/standalone) of the signature algorithm during verification can achieve non-separability with a means of domain separation in the signature algorithm. The protocol can thus use the context string to ensure domain separation between composite and standalone use and possibly realize further security features such as domain separation between applications.
- Falko

22. Comments from J. Harvey, B. Kaliski, A. Fregly, S. Sheth, Verisign, November 22, 2023

Dear NIST,

We appreciate NIST's leadership in post-quantum cryptography and the three draft standards that have resulted from that effort. Attached please find a brief set of comments provided in response to NIST's call for public comments on the proposed standard "FIPS 205: Stateless Hash-Based Signature Standard."

Best Regards,
Joe Harvey

Comment attached.

Comments on FIPS 205 (Draft): Stateless Hash-Based Signature Standard

Joe Harvey, Burt Kaliski, Andrew Fregly and Swapneel Sheth
November 21, 2023

(These comments are provided in response to NIST's call for public comments on the proposed standard "FIPS 205: Stateless Hash-Based Signature Standard."[1])

We appreciate NIST's leadership in post-quantum cryptography and the three draft standards that have resulted from that effort.

Most of our evaluation has focused on the hash-based signature schemes selected by NIST, including the stateful schemes in SP 800-208[2] and SPHINCS+, the stateless scheme that is the basis for FIPS 205. While we do not have any significant technical comments on SLH-DSA, we would like to offer three general recommendations on uses of the algorithm and its components.

1 Allowing Randomized Prehashing

The draft allows implementations that want to reduce the size of the message inputs to the SLH-DSA operations to use a prehash mode that computes a digest of the message using an approved hash function or extensible-output function (XOF) and then applies the SLH-DSA operations to the digest. The draft requires that the hash function or XOF be secure against collision and second preimage attacks.

We recommend that FIPS 205 also allow implementations to compute the digest using randomized hashing. As Section 9.4 states, the security of SLH-DSA's operations do not depend on collision resistance. Allowing randomization would extend this property to the prehash mode.

In its recent withdrawal[3] of SP 800-106,[4] its previously approved randomized hashing technique, NIST confirmed that randomized hashing was still allowed in FIPS-approved systems. If this position holds for FIPS 205, then we suggest adding a statement like "implementations using prehash mode may use randomization to strengthen the collision resistance provided by the approved hash function or XOF."

2 Specifying a Randomized Hashing Technique

The draft FIPS 205 provides three candidate techniques for randomized hashing: the $\mathbf{H}_{msg}$ instantiations in Sections 10. We recommend that FIPS 205 specify a way to use these techniques for prehashing. Recall that $\mathbf{H}_{msg}$ has four inputs: a randomizer *R*, a public seed **PK**.seed, a public root **PK**.root, and a message *M*. To separate the use of $\mathbf{H}_{msg}$ for prehashing the full message from its use in processing the

[1] NIST. *FIPS 205 (Initial Public Draft): Stateless Hash-Based Digital Signature Standard.* August 24, 2023. https://csrc.nist.gov/pubs/fips/205/ipd
[2] *NIST SP 800-208: Recommendation for Stateful Hash-Based Signature Schemes.* October 2020. https://csrc.nist.gov/pubs/sp/800/208/final
[3] NIST. *Withdrawal of NIST Special Publication 800-106, Randomized Hashing for Digital Signatures.* December 15, 2022. https://csrc.nist.gov/News/2022/withdrawal-of-nist-sp-800-106
[4] *NIST SP 800-106: Randomized Hashing for Digital Signatures.* February 2009. https://csrc.nist.gov/pubs/sp/800/106/final

digest, the inputs should be distinct for the different purposes. (Similar remarks apply to $\mathbf{PRF}_{msg}$ if it is also used during prehashing.)

One approach for separating the inputs would be for an implementation to pass different **PK**.seed and **PK**.root values to $\mathbf{H}_{msg}$ during prehashing from the values passed during the SLH-DSA operations, which are drawn from the SLH-DSA public key. The implementation could generate the separate values at random and maintain them with the SLH-DSA public key, or derive them from the SLH-DSA public key.

Another approach would be for the implementation to prepend a designated **ADRS** value to the message input to $\mathbf{H}_{msg}$ during prehashing and a different **ADRS** value to the digest during the SLH-DSA operations. It's not necessary to modify SLH-DSA to achieve this effect, as an implementation can simply prepend the second **ADRS** value to the digest that it passes to SLH-DSA, and SLH-DSA will then pass the prepended value to $\mathbf{H}_{msg}$. (This is the approach we took in the specification of the Merkle Tree Ladder (MTL) mode of operation[5] — see Section 4.6 of that specification for discussion.)

While there may not be any problem using other randomized hashing techniques with SLH-DSA, it would be helpful both for security and interoperability to give implementers a recommended approach.

3 Extensibility and Component Reuse

The prehash mode of SLH-DSA is one example of how SLH-DSA and its internal hash and pseudorandom functions can be reused as components in other cryptographic techniques. The randomized prehash mode suggested above and MTL mode are two others.

Emerging proposals for hybrid signatures offer additional examples of techniques that could reuse signature schemes such as SLH-DSA and their components when combining a classical algorithm and a post-quantum algorithm, e.g., as contemplated in Bindel and Hale's recent note.[6] Some techniques may require slight changes to the signature schemes, for instance making an external value an input to an internal hash function. It would be helpful to have flexibility for these kinds of future extensions. (As a related example, NIST's key establishment specifications, e.g., SP 800-56A[7], allow such "supplementary information" to be input to its internal key derivation functions.)

There may also be benefit in using common internal hash and pseudorandom functions across both SLH-DSA and ML-DSA. While the draft standards for these algorithms both use approved hash functions, they use them in very different ways. A common approach would reduce the number of components that need to be analyzed, built and tested.

We recommend further design efforts on these uses, extensions and "modes of operation" as SLH-DSA and NIST's other standards are finalized and deployed. Much like prehashing, the goal of these of efforts would be to address additional operational and security objectives, and thereby increase the adoption, use, and impact of the new post-quantum algorithms that NIST is standardizing.

---

[5] J. Harvey *et al. Merkle Tree Ladder Mode (MTL) Signatures.* Revised October 23, 2023. https://datatracker.ietf.org/doc/draft-harvey-cfrg-mtl-mode/

[6] N. Bindel and B. Hale. *A Note on Hybrid Signature Schemes*. IACR ePrint 2023/423, revised July 22, 2023. https://eprint.iacr.org/2023/423

[7] *NIST SP 800-56A: Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography.* Rev. 3, April 2018. https://csrc.nist.gov/pubs/sp/800/56/a/r3/final

23.Comments from Panos Kampanakis, AWS, November 22, 2023

Dear Dustin, NIST PQ Project team,

We would like to provide feedback on the SLH-DSA FIPS-205 draft as well:

We want to suggest for NIST consider SHA-256 only for Level 1 and exclude SHA-512 for Levels 3, 5. Levels 3 and 5 should only leverage SHAKE256. SHAKE256 should still be available for Level 1. The advantage of only using SHA-256 for Level 1 allows for more resource constrained or performance demanding uses to be able to use the SHA-2 family especially for the faster and leaner Level 1 SLH-DSA parameters. Using SHAKE256 for all levels beyond that allows for SHAKEs to be used as the hash function of the future. SHAKEs should be the hashes of choice due to their superiority in all signature schemes and such an approach enables that future.

On line 971, footnote 17 mentions that 1^10 messages per second would take 58 years to run out of signatures which is a great point. You could also have an example from the real-world for readers to appreciate the scale of this number. https://ct.cloudflare.com/ shows today we have about 260K signatures / certs issued per hour. That would give us about 633,000 signatures per second. That amounts to about 10^6 signatures per second. So, even if all certs of the world were issued by one SPHINCS+ key / CA which would never be the case of course, it would take 58*10000 years to run out of signatures.

After line 983, we also suggest to add a mention of the security level not degrading fast even if you sign more than 2^64 messages. Fig 4 and 5 in https://eprint.iacr.org/2022/1725.pdf plots how slow the security level drops when the scheme is for 2^10 or 2^20 signatures and someone mistakenly signs 2^30 or more. The SPHINCS+ team could generate similar tables for the default SPHINCS+ schemes to show that the security level will be pretty high even after signing 2^70 or more.

The paragraph starting on line 935 discusses digest-then-sign. We want to suggest to be prescriptive about the digest function and not leave it open to interpretation. For example, it should suggest the use of SHA2-512 or SHAKE256 with 64byte output as a conservative choice for all uses. SHA2-512 could be used for the SHA2 parameter sets. SHAKE256 could be used for the SHAKE parameter sets. Given that digesting would make sense for large messages, there is no performance benefit in digesting with SHA2-256 over SHA2-512 because SHA2-512 will be faster [1], [2]. So, SHA2-512 and SHAKE256 are sufficient. Even if NIST does not want to make recommendations about the digest function in the SP, we think there is benefit in calling out the conservative options SHA2-512 and SHAKE256 with 64-byte output which will not materially affect performance for large messages.

Also on the predigesting topic. There was a recent discussion in the NIST list about updating ML-DSA and SLH-DSA to include some domain separation for when signing M and H(M) so that an application which supports both cannot suffer existential forgeries where two different messages M and H(M) can be validated by the same signature (depending on the verifier logic). Although this would be a problem for use-cases that need to be able to sign M and H(M), these are not many. X.509 does not need to sign M and H(M), it can only do M and that is what draft-ietf-lamps-dilithium-certificates specifies. Other applications are similar, they don't need to support both M and H(M). Similarly, CMS allows for both, but in the case of H(M) you can add any arbitrary attributes for domain separation in the optional SignedAttributes. For context, in the past EdDSA took a similar path which ended up being futile. It specified pureEdDSA and prehashEdDSA. That ended in unnecessary complications in the signature although in the end pureEdDSA was mostly adopted in various standards. Thus, we are against tweaking SLH-DSA which would complicate it and alter their security proof by introducing targeted collisions. Applications that need to support both signing M and H(M) should specify any domain separation they need. We suggest for NIST to update section 9.4 for SLH-DSA to spell out the application existential forgery concern and suggest domain separation for H(M) to prevent it in the application that uses SLH-DSA.

line 945: Prescribe SHAKE256 or SHA-512 depending on the parameter set. And not leave it to interpretation and letting adopters use SHA3, or SHA256

Additionally, a couple of minor suggestions

- Line 220: We suggest to add brief description of faults attacks how they work and their protections. Something to explain that they interfere with the signing step and could introduce malicious or inadvertent fault that could allow for grafting a SPHINCS+ subtree and forging a signature. There are various protections which include caching or calculating the signature multiple times before releasing it. The details of course will be referenced in the literature.
- Line 923: The paragraph starting on line 980, explains the security levels for each parameter and points to the literature "([4, 23]") for a more thorough analysis. We suggest to give a pointer to section 7.1.1 in [4] which has the formulas of the security levels in bits.

Regards,
Panos Kampanakis
Amazon Web Services (AWS)