From:	Dang, Quynh H. (Fed)
To:	Perlner, Ray A. (Fed); Kelsey, John M. (Fed); Moody, Dustin (Fed); internal-pqc; (b) (6)
Subject:	Re: The path to standardization
Date:	Saturday, June 13, 2020 7:12:32 AM

so we should keep investigating that possibility/option.

Quynh.

From: Perlner, Ray A. (Fed) <ray.perlner@nist.gov>
Sent: Friday, June 12, 2020 3:03 PM
To: Kelsey, John M. (Fed) <john.kelsey@nist.gov>; Moody, Dustin (Fed) <dustin.moody@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: RE: The path to standardization

That seems correct to me.

Ray

From: Kelsey, John M. (Fed) <john.kelsey@nist.gov>
Sent: Friday, June 12, 2020 3:01 PM
To: Perlner, Ray A. (Fed) <ray.perlner@nist.gov>; Moody, Dustin (Fed) <dustin.moody@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: Re: The path to standardization

So, it seems to me that this wouldn't change its standing compared to Rainbow. That is, if Rainbow still looks solid at the end of round three and its IP situation doesn't look too toxic, we will go with Rainbow. If either the security picture or the IP picture looks bad for Rainbow, then we will consider GeMSS, perhaps with some tweaks for improved efficiency.

Does that seem right?

--John

From: "Perlner, Ray A. (Fed)" <ray.perlner@nist.gov>
Date: Friday, June 12, 2020 at 14:52
To: "Kelsey, John M. (Fed)" <john.kelsey@nist.gov>, "Moody, Dustin (Fed)" <dustin.moody@nist.gov>, internal-pqc <internal-pqc@nist.gov>
Subject: RE: The path to standardization

Signing and verifying should be 25% faster and signatures should be 24-30 bits shorter depending on GeMSS, RedGeMSS, or BlueGeMSS. Keygen and public key size would not change. (This is for category 1. I don't think we can reduce the number of iterations for category 3 and 5.)

From: Kelsey, John M. (Fed) <john.kelsey@nist.gov>

Sent: Friday, June 12, 2020 2:23 PM
To: Perlner, Ray A. (Fed) <ray.perlner@nist.gov>; Moody, Dustin (Fed) <dustin.moody@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: Re: The path to standardization

Daniel and Ray,

What would change if GeMSS made that tweak? Would it just be faster, or would the key size change?

Thanks,

--John

From: "Perlner, Ray A. (Fed)" <ray.perlner@nist.gov>
Date: Friday, June 12, 2020 at 13:31
To: "Moody, Dustin (Fed)" <dustin.moody@nist.gov>, "Kelsey, John M. (Fed)" <john.kelsey@nist.gov>, internal-pqc <internal-pqc@nist.gov>
Subject: RE: The path to standardization

I also mostly agree with the summaries with a couple of quibbles:

I second Daniel A's point that NTRU's main path to standardization involves IPR concerns for the newer lattice schemes.

Regarding tweaks. Daniel and I have been considering a specific tweak to GeMSS (3 instead of 4 rounds for the Feistel Patarin construction, leading to somewhat better performance.) You currently have that listed as tweaks discouraged.

From: Moody, Dustin (Fed) <dustin.moody@nist.gov>
Sent: Friday, June 12, 2020 1:26 PM
To: Kelsey, John M. (Fed) <john.kelsey@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: Re: The path to standardization

John,
I mostly agree with those informal explanations. I think for SIKE though, the description might not be completely right. I don't think SIKE needs big tweaks. The actual algorithm is very stable. We'd like more confidence in the security, but mostly we want improved performance.

I think we can include some of this type of reasoning in the report, and people should add what they think is needed. Some of it is already covered, or is probably clear enough (I think

the lattice finalists know they need to beat each other, as we state that we'll only choose one).

Dustin

From: Kelsey, John M. (Fed) <john.kelsey@nist.gov>
Sent: Friday, June 12, 2020 12:48 PM
To: internal-pqc <internal-pqc@nist.gov>
Subject: The path to standardization

Everyone,

This is my attempt to very briefly and informally state the path to standardization. In this list, I consider three categories: finalists, fallback alternates, and round 4 alternates.

My question is, does this more-or-less succinctly capture the path to standardization for each algorithm? Obviously we'll state this in a more polished and formal way in the report, but am I missing anything or wrong about anything?

Thinking about the algorithms this way makes me think we should be more clear about the distinction between round 3 finalists, round 3 fallbacks, and round 4 alternates, because these are quite different. We want minimal tweaks for finalists and fallbacks, but we encourage tweaks for our alternates—they won't be standardized until after somethink like a fourth round.

Finalists:

a. Classic McEliece

- Don't get broken

b. Kyber
c. Saber
d. NTRU

- Beat the other two, don't get broken
- NTRU: Be the last one standing when K and S both look shaky due to excessive optimization or insufficient grey hairs.

e. Falcon
f. Dilithium

- Beat the other one, don't get broken
- Falcon: Show your floating point stuff doesn't drag you down

g. Rainbow

- Don't get broken AND
- Don't be too terrible on IP issues

Alternates (Fallbacks): (Tweaks discouraged)

h. HQC

- BIKE doesn't go forward AND
- We see need for another code-based KEM.

i. GeMSS

- Rainbow doesn't go forward AND
- We see need for another multivariate signature (aka everything in signature finalists dies)

j. SPHINCS+

- Dilithium and Falcon get broken

OR

- We see demand for a paranoid signature option

k. NTRU Prime

- Advances in structured lattice analysis undermine Saber, Kyber, and NTRU AND
- Those advances do not undermine NTRU Prime AND
- We are comfortable with NTRU Prime's parameter selection based on their costing of attacks

l. Frodo

- Advances in structured lattice analysis undermine Saber, Kyber, and NTRU AND
- Those advances do not undermine Frodo

OR

- We see a need for a paranoid lattice KEM option

Alternates (Round Four): (Tweaks encouraged)

m. PICNIC

- Continued progress gives much better performance that SPHINCS+ AND
- We see need for a symmetric-only signature AND
- Scheme ripens enough that we're sure it's nailed down including LowMC security

n. BIKE

- Nail down decryption failures, add level 5 parameters AND
- We see need for another code-based KEM

o. SIKE

- Don’t get broken AND
- Scheme and problem ripen enough we’re comfortable standardizing it