From:	Dustin Moody
To:	Moody, Dustin (Fed); Liu, Yi-Kai (Fed); Chen, Lily (Fed); Daniel Smith; Perlner, Ray A. (Fed); Alperin-Sheriff, Jacob (Fed); Miller, Carl A. (Fed); Peralta, Rene C. (Fed); Jordan, Stephen P (Fed); Bassham, Lawrence E. (Fed)
Subject:	PQC comments
Date:	Tuesday, September 20, 2016 8:34:13 AM
Attachments:	Comments9-1-18.zip

Everyone,
Sorry I have been out of contact. I am in Toronto, and forgot to bring my RSA token, so it's been harder to access my NIST email. Thank you Lily for sending out a file with the PQC comments. I'm attaching an updated version which has a couple of comments she missed. The most important of which is Jintai Ding's. (The other ones were basically spam or not serious ones).

Thank you to Jacob for organizing the comments into the sections of our CFP. Everyone please read the comments, and we will discuss them at our next meeting. Please also feel free to discuss them before then, so we can have a productive meeting.

Dustin

Public Comments Received on the
Draft Proposed Submission Requirements and Evaluation Criteria
for the Post-Quantum Cryptography Standardization Process
(Comment Period Closed: 9/16/2016)

Dear Ms. Chen,

Thank you for the opportunity to submit comments on the Submission Requirements and Evaluation Criteria to be used in NIST's process for standardizing quantum-resistant public-key cryptographic algorithms.

Mozilla's mission as a non-profit organization is to promote openness, innovation, and opportunity online. Protecting the security of Internet communications is a core part of that mission. Mozilla is a major user of cryptographic standards. Our products engage in billions of HTTPS transactions per day, and we maintain of one the most widely used open-source cryptographic libraries. We are also deeply involved in the standardization of cryptographic protocols in the IETF. Eric Rescorla, a Mozilla fellow, is editor of the TLS specification, Richard Barnes is a former member of the Internet Engineering Steering Group, and several other Mozilla staff are active in cryptography-related IETF working groups. It is from this perspective that we offer our comments.

Our primary concern with the proposed process is that it needs to ensure that the algorithms standardized through it work in the real world. The draft documents provided for comment present problems at both legal and technical levels.

1. Submitted algorithms must be usable without compensation to patent holders (RAND-Z, not only RAND) and implementations must bear an open-source license

The draft Call for Proposals is correct to note that "royalty-free availability of cryptosystems and implementations has facilitated adoption". It is surprising that this observation is followed by allowances for royalty-bearing cryptosystems, e.g., in the Statement by Patent Owners 2.D.2. Allowing royalty-bearing cryptosystems to be submitted will inhibit both the thorough evaluation of proposals and their eventual adoption by industry.

As the draft CFP acknowledges in several places, contributions by the broader research community will be essential in helping NIST make a thorough evaluation of the submitted algorithms. In order to make these contributions, members of the community including researchers in the commercial and academic sectors will need be able to implement the submitted algorithms. A requirement to license patents for such implementations would make it impossible for many researchers to participate in evaluation of algorithms, undermining the completeness and the legitimacy of the NIST process.

In this context, it should be noted that U.S. Courts have all but eliminated the availability of the "experimental use defense" to patent infringement: "[R]egardless of whether a particular institution or entity is engaged in an endeavor for commercial gain, so long as the act is in furtherance of the alleged infringer's legitimate business and is not solely for amusement, to satisfy idle curiosity, or for strictly philosophical inquiry the act does not qualify for the very narrow and strictly limited experimental use defense. Moreover, the profit or non-profit status of the user is not determinative." *Madey v. Duke Univ.*, 307 F.3d 1351, 1362 (Fed. Cir. 2002) (finding university's research projects with no commercial application still "unmistakably further the institution's legitimate business objectives" in education). (See also *Soitec, S.A. v. Silicon Genesis Corp.*, 81 Fed.Appx. 734, 737 (Fed.Cir. 2003), "There is no fair use or research and development exception for infringement of normal commercial processes.")

There is also a need for researchers to be able to use and modify the submitted implementations in order to evaluate the costs and benefits of the algorithm in different contexts. For example, a researcher might adapt the optimized implementation to run on a machine architecture common in mobile devices to see if the algorithm is suitable for use in that environment. In order to allow this usage, it is imperative that the submitted implementations be licensed under an open-source license, and in particular one that allows for the creation of derivative works. We encourage NIST to specify a small set of acceptable open-source licenses. There are several such licenses available: Many of the policies in the US CIO's list of licensing resources recommend the CC0 or CC-BY licenses; we would also find licenses such as the MIT, BSD, Mozilla Public License, or Apache Public License acceptable.

Standardization of a royalty-bearing algorithm would strongly inhibit industry adoption of the algorithm, especially in open standards organizations and open-source projects. The IETF has historically avoided standardization of royalty-bearing algorithms, so it would be difficult to establish the ancillary protocol standards needed to integrate a NIST-standard algorithm into Internet protocols.

Open source projects such as OpenSSL and NSS are critical to the deployment of cryptographic protocols on the Internet. These projects are often unable to use algorithms that are only available through royalty-bearing licenses. In particular, it would be extremely difficult for Mozilla to include a royalty-bearing algorithm in its products (including Firefox) even if it were standardized by NIST. The only use of royalty-bearing technologies in Firefox today (the H.264 video codec) was only possible because an existing license holder offered to cover royalty costs for Firefox users and because significant engineering effort was spent enabling the codec to be distributed within the terms of the license.

2. Algorithms need to be evaluated as they will be used

It is crucial for the success of this process that NIST not evaluate submitted algorithms in

the abstract, but as they will be deployed in modern information security systems. To that end, we are glad to see that the evaluation criteria place the impact on Internet protocols as a primary measure of an algorithm's utility.
Along these lines, it should be noted that verifying that an algorithm is IND-CCA2 secure in the abstract might not mean that it provides this level of security in practice. For example, if there are assumptions underlying the IND-CCA2 proof that a protocol cannot meet, then the algorithm might not provide an acceptable level of security for that protocol. The evaluation criteria should make clear that algorithms must not rely on assumptions in security proofs that cannot be satisfied by common security protocols.

While we agree with NIST's choice to rule hybrid algorithms out of scope for this process, it is nonetheless true that hybrid algorithms will be an important part of the deployment process for post-quantum algorithms. We encourage NIST to consider the suitability of algorithms for use in hybrid schemes as an evaluation criterion, with preference for algorithms that are more amenable to hybridization.

Looking at how public-key algorithms are used in modern Internet protocols, it is clear that key establishment and signature are much more important features to implement than public-key encryption. Forward secrecy in particular has been a feature that the community of TLS operators has worked very hard to make pervasive, in order to guard against temporary compromises. Even messaging security protocols, which have traditionally relied fairly heavily on public-key encryption, have been moving toward frameworks that provide more forward secrecy by relying more on key establishment instead of public-key encryption. We would be comfortable if NIST de-emphasized or dropped public-key encryption from evaluation, especially given that in many cases, it can be replaced by a combination of key establishment and symmetric encryption.

Finally, the selection of x64 as a reference platform is understandable, but perhaps not a complete reflection of modern computing environments. It is increasingly common for cryptography to be done on mobile devices, mostly using ARM architectures, and operators are increasingly selecting algorithms based on their performance in mobile environments (e.g., preferring ChaCha20 over AES). Emerging platforms for the Internet of Things will likely be bringing similar constraints in the near future. We would encourage NIST to include one or more mobile and/or IoT platforms in their evaluations, either directly or by working with the community to ensure that algorithms are evaluated in these contexts.

We are grateful to the NIST for the opportunity to comment on on this process. We look forward to working with NIST and the broader community to ensure that the Internet can be kept secure even if quantum cryptanalysis becomes feasible.

Respectfully submitted,
Richard Barnes, Firefox Security Lead
James Jones, Cryptographic Engineering Manager

>>
Dear Dr. Chen,

Congratulations on initiating a standardization effort for post-quantum cryptography. In general the effort sounds useful and carefully planned, and I look forward to providing whatever assistance I can.

I have comments on several topics, which I have tried to sort here into decreasing order of importance.

1. Quantitatively comparing post-quantum public-key security levels is going to be a nightmare. I see only two ways that submitters a year from now can possibly be "confident that the specified security target is met or exceeded": (1) overkill; (2) overconfidence. Many users will not be satisfied with overkill, and NIST should not encourage overconfidence.

   For comparison, let's ignore quantum computers for a moment. Imagine asking someone to choose DSA key sizes to be confident about reaching a pre-quantum 2^256 security target, the largest number in NIST's list of preselected security levels. Should be easy, right?

   Here are some costs in the literature for computing multiplicative-group discrete logarithms by NFS index calculus, and thus breaking DSA:
   - Exponent 2.080 from 1992 Gordon. What I mean here is that the cost is asymptotically L^(2.080...+o(1)), assuming standard conjectures, with the usual definition of L in NFS.
   - Exponent 1.922 from 1993 Schirokauer.
   - Exponent 1.901 from 2003 Matyukhin.
   - Exponent 1.442 per target from 2006 Commeine--Semaev, after per-prime precomputation with exponent 1.901.
   - Exponent 1.231 per target from 2013 Barbulescu, after per-prime precomputation with exponent 1.638, after one-time precomputation with exponent 1.901. (I doubt that 1.231 is optimal.)

   All of these are exponents in an unrealistic model of computation where storage and communication are free. In a realistic model, I presume that 1.901 would go up to 1.976 by an adaptation of a factorization algorithm that I published in 2001, and 1.638 would go up to 1.704 for a batch of targets by an adaptation of my 2014 "Batch NFS" paper with Lange. I don't know what would happen to the 1.231.

   One might try to argue that the 1.901 and 1.976 have been stable for a decade, and that multi-target attacks don't matter. But NIST's call explicitly, and sensibly, asks for "resistance to multi-key attacks". Multi-target attacks _do_ matter, and the current _exponent_ for the security of DSA against these attacks is only three years old.

Furthermore, a closer look shows that there are many more improvements that reduce concrete attack costs by reducing the "o(1)" quantities. Sloppily replacing o(1) with 0, as NIST apparently did to obtain its current recommendation of 15360 bits for DSA for >=2^256 security, is unjustified. Even for the simple case of single-key attacks, figuring out the o(1) with moderate precision at such large sizes is a difficult research project, extrapolating far beyond experimental ranges. At this point I wouldn't hazard a guess as to whether NIST's 2^256 is an overestimate or an underestimate.

If someone makes enough progress on this research project to announce that the single-key attack cost is actually between 2^245 and 2^250, will NIST withdraw its DSA standard? (What if there is a 2^100 attack against RSA-2048, rather than the commonly quoted 2^112?) If not, then what exactly is the point of asking people to be "confident" that 2^256 is "met or exceeded"? More to the point, experts are _not_ confident, even when multi-target attacks are ignored.

Some people are even more worried by the recent drastic reduction of security levels for pairing-based cryptography, by other index-calculus optimizations. Fortunately, DSA conservatively chose prime fields, avoiding the subfield/automorphism structures exploited in the latest attacks (and in the most complicated previous variants of NFS---one of the warning signals that led ECRYPT to recommend prime fields a decade ago). But the bigger picture is that index calculus is complicated and constantly improving. Would it really be so surprising to see another security loss that _does_ affect DSA?

keylength.com reports some sources making recommendations around the same 15360-bit level recommended by NIST, but also reports Lenstra and Verheul recommending 26000 or 50000 bits. The big difference here is Lenstra and Verheul leaving a security margin in case there is progress in cryptanalysis.

So, with all these numbers in mind, how should we choose DSA key sizes to be "confident" about >=2^256 pre-quantum security? Should we take NIST's overconfident 15360 bits, which definitely flunks the multi-key requirement, and isn't a solid bet even for a single key? How about 26000 bits? 50000 bits? Much bigger, considering Barbulescu's 1.231? What happens if 1.231 is improved further?

What NIST is asking post-quantum submitters to figure out is far more difficult than this DSA example, for several reasons:

- As one would expect given the history of how cryptanalytic effort has been allocated, the security picture for most post-quantum public-key algorithms is even less stable than the security picture or DSA. Example: Current algorithms for the famous shortest-vector problem take (conjecturally) time 2^((0.29...+o(1))d) in dimension d, a vast improvement compared to 2^((0.40...+o(1))d), the best result known just a few years ago.
- At this point we have only crude guesses as to the ultimate costs of different quantum operations. I understand that NIST wants to define 2^b post-quantum security as 2^b quantum AES computations, but what is the relative cost of a quantum AES computation and a lookup in an N-entry table using a quantum index? How does this depend on N? How much harder is it if the table entries are themselves quantum?
- I agree with NIST's comment that a 256-bit preimage search with Grover is actually harder than a 256-bit collision search (even if qubits are magically as cheap as bits), since Grover parallelizes poorly. I agree that the optimum value of T*sqrt(S), subject to ST=2^128 and T>=sqrt(S), is 2^(2*128/3). But T*sqrt(S) is not a user-comprehensible cost metric, and not a metric for which many subroutines have been analyzed.
- Algorithm designers benefit tremendously from being able to try out their algorithms on small-scale and medium-scale problems. An experiment can show with minimal effort that an algorithm doesn't produce the desired outputs, or that it doesn't run at the desired speed. Designers of quantum algorithms don't have this tool yet.
- How is a submitter supposed to be confident of reaching, e.g., 2^128 post-quantum security? Submitters will end up making ill-informed random guesses of which parameters to assign to which security levels. Security analysis will then throw some submissions into the Scylla of being "broken", while others will have thrown themselves into the Charybdis of being "inefficient", even though those submissions might simultaneously be _more secure and more efficient_ than other submissions that simply happened to make luckier initial guesses of target security levels.

To summarize: Well-informed long-term security assessments will not simply supersede obsolete guesswork. The guesswork will continue having an influence long after it should have been thrown away. This is a serious failure mode for the evaluation process.

Does "meet or exceed each of five target security strengths" mean that each submission has to separately target all five levels, giving designers five chances to be artificially thrown into the rocks? Is it enough to target just the top level, namely 2^256 pre-quantum security and 2^128 post-quantum security?

I found it particularly striking that this choice of top target security level was based on the security achieved by a secret-key system (in this case AES-256, for some reason ignoring multi-target attacks), rather than on any attempt to assess what users actually need. I'm reminded of the ludicrous requirement of 2^512 preimage resistance for SHA-3, forcing permutation-based SHA-3 submissions such as Keccak to be much larger and slower than they would otherwise have been.

If a public-key system naturally has 2^2b pre-quantum security and more than 2^b post-quantum security (I predict that this will be a common case), then choosing parameters to successfully target 2^256 pre-quantum security will be overkill for 2^128 post-quantum security---and also overkill for what users actually need. Why is this a sensible target?

If a public-key system naturally has 2^b post-quantum security and more than 2^2b pre-quantum security (I know one example like this), then choosing parameters to successfully target 2^128 post-quantum security will be overkill for 2^256 pre-quantum security. Why should the designer have to bother evaluating the pre-quantum security level?

Let me suggest a different approach:

- Leave it up to submitters to decide exactly what post-quantum security level to aim for.
- Tell them that security levels <2^64 will be viewed as "breakable", and that security levels >2^128 are unlikely to be viewed as more valuable than security level 2^128, except possibly as a buffer against future cryptanalytic progress.
- Ask them to do the most accurate job that they can of analyzing post-quantum security. Don't ask for fake confidence.
- Scrap the requirement of a pre-quantum security analysis. Users will use cheap ECC hybrids to obtain the pre-quantum security that they want.

Of course, many submissions will do a pre-quantum security analysis and then say "We don't think Grover will reduce the exponent by a factor beyond 2". Is there any problem with this? Should the number of submissions be limited by the current availability of expertise in quantum cryptanalysis?

Followup analysis will improve our understanding of the actual post-quantum security levels of various algorithms, and then NIST will look at a two-dimensional plot of speed vs. security level and decide which options are most interesting.

2. My understanding is that NIST is asking for two specific types of encryption, which NIST labels as "public-key encryption" and "key exchange". This is too

narrow: it omits important variants of public-key encryption that people should be allowed to submit.

What I suspect will be most important in the long run is a CCA2-secure "KEM". A KEM can be viewed as a limited form of public-key encryption: the only thing a ciphertext can do is to communicate a random session key. As a simple pre-quantum example, Shoup's "RSA-KEM" chooses a random number r mod pq and transmits a session key SHA-256(r) as the ciphertext r^3 mod pq. This is easier to design and analyze and implement than, say, RSA-OAEP.

(Proponents of RSA-OAEP will respond that RSA-OAEP can encrypt a short user-specified message as a ciphertext with the same length as pq. Some applications will notice the bandwidth difference. Obviously NIST should continue to allow public-key encryption as a target.)

One can easily combine a KEM with an authenticated cipher to produce a full-fledged public-key encryption scheme. But this understates the utility of a KEM: the same session key can be reused to encrypt any number of messages in both directions, whereas wrapping the KEM in a public-key encryption scheme hides this functionality. Using this public-key encryption scheme to encrypt another level of a shared session key would be frivolous extra complexity. Why not let submitters simply submit a KEM, skipping the cipher?

Sometimes people reduce the security goals and design KEMs to encrypt just one message, _without_ chosen-ciphertext security. Here is the application to keep in mind:

- a client generates a KEM public key;
- a server uses this to transmit a random session key;
- messages are signed by long-term keys for authentication;
- the KEM private key and session key are erased after the session.

This is how New Hope works inside TLS. The signatures (if handled properly) prevent attackers from choosing any ciphertexts. So why not let people submit single-message non-CCA2-secure KEMs?

(I don't like the TLS/SIGMA approach to secure sessions: it is error-prone and excessively complex. This is not a broadcast scenario; authentication does not require signatures. I prefer the simplicity of using pure encryption: the long-term key is an encryption key, and the soon-to-be-erased short-term key is another encryption key. This requires multiple-message support and CCA2 security, but my current impression is that this robustness has only minor costs, and I wouldn't be surprised if the New Hope team decides to move in this direction. However, if they instead decide that CCA2 security is too expensive, they shouldn't be rejected for targeting TLS!)

What NIST calls "key exchange" in the draft sounds to me like a poorly labeled KEM with intermediate security requirements: chosen-ciphertext security seems to be required, but the interface sounds like it allows only one message before the key is thrown away. NIST should make clear if it instead meant a full-fledged KEM allowing any number of ciphertexts. Either way, NIST should explicitly allow non-CCA2-secure single-message KEMs such as New Hope.

Calling any of these systems "key exchange" is deceptive for people who expect "key exchange" to be a drop-in replacement for DH key exchange. In DH, Alice and Bob both know a shared secret as soon as they see public keys from Bob and Alice respectively, with no additional communication. As a concrete example, consider the very small number of network round trips needed to efficiently authenticate data from hidden client identities in the "CurveCP" and "Noise_XK" protocols. Here's Noise_XK using ECC:

- Alice sends her ephemeral public key eG to Bob. New session key: hash of ebG, where b is Bob's long-term key.
- Bob responds with his ephemeral public key fG, encrypted and authenticated. New session key: hash of ebG and efG.
- Alice sends her long-term public key aG to Bob, encrypted and authenticated. New session key: hash of ebG, efG, and afG.

This third packet can already include data authenticated under the last session key, and Bob immediately knows that the data is from Alice. Pure public-key encryption (without signatures) needs another round trip for authentication: Bob has to send data to Alice's long-term public key and see the reply before Bob knows it's Alice talking.

There is one notable post-quantum example of the DH data flow, namely isogeny-based crypto. Security analysis of isogeny-based crypto is clearly in its infancy, but if isogeny-based crypto does survive then the data flow will be an interesting feature. People who submit isogeny-based crypto should be allowed to submit it in a way that makes this data flow clear, rather than having to wrap it in public-key encryption.

I understand that for signatures NIST explicitly decided to disallow one data flow of clear interest, namely stateful signatures, since there is already separate ongoing standardization of stateful hash-based signatures, which are the main reason for interest in this data flow. (The security of hash-based signatures is much better understood than the security of most other public-key systems.) But for encryption I don't see how a similar limitation could be justified.

To summarize, there are at least three clearly different types of data flow of interest: public-key encryption, KEMs, and DH. Within KEMs, there are at least two security targets of interest: passive security for one message, and chosen-ciphertext security for many messages. I suggest that NIST explicitly allow

- all four of these targets;
- also the intermediate type of KEM labeled as "key exchange" in the current draft, if NIST has an application in mind; and
- any further encryption targets that NIST identifies this year as being useful.

I also suggest defining some standard conversions that NIST will apply automatically: e.g., converting a CCA2-secure KEM into CCA2-secure PKE by composition with AES-256-GCM, and converting the other way by encrypting a random 256-bit key. NIST won't want to listen to pointless arguments such as "yes we know we're worse than this PKE but it wasn't submitted to the KEM category" from KEM submitters, and won't want to have to wade through artificially bloated PKE+KEM submissions that are really just one design but want to compete in every category.

3. I have three suggestions regarding terminology.

First, the draft refers frequently to "key exchange", which as noted above ends up deceiving people. I suggest scrapping this terminology in favor of more precise terminology such as KEM and DH. (There's already a NIST standard introducing relevant names such as "C(0,2)", but I don't know how many people are familiar with these names.)

Second, the draft uses "forward secrecy" (even worse, "perfect forward secrecy") to refer to the obvious security benefits of erasing a private key. This terminology also ends up deceiving people. Last week I was speaking with a banker who thought that TLS's "perfect forward secrecy" would protect his communications against future quantum computers. I suggest avoiding this terminology and instead saying something like "Fast key generation is useful for high-frequency generation of new key pairs, which in turn allows each private key to be promptly erased."

Third, the draft says that post-quantum cryptography is "also called quantum-resistant or quantum-safe cryptography", and makes occasional use of the "quantum-resistant" terminology after that. It's true that Google finds some hits for "quantum-resistant cryptography" and "quantum-safe cryptography" (1630 and 4340, compared to 47100 for "post-quantum cryptography"), but I'm not at all sure that the people using these terms are using them with the same meaning as post-quantum cryptography, and I predict that users seeing algorithms labeled as "resistant" and "safe" will be deceived into thinking that we are more confident than can be scientifically justified.

As a concrete example, research by Makarov et al. has convincingly shown that ID Quantique's QKD products are breakable at low cost, but one of the top hits for "quantum-safe cryptography" appears to refer to those products as "provably secure quantum-safe" cryptography. I presume that snake-oil peddlers choose this terminology precisely because it is deceptive; for the same reason, I suggest that NIST avoid the terminology. As an analogy, FIPS 186-4 has the sensible title "Digital signature standard", not "Safe digital signature standard" or "Attack-resistant digital signature standard".

4. Requiring submissions to be sent by postal mail will penalize some submitters for reasons that are not connected to the quality of their submissions. For example, as far as I know, the lowest-cost way to guarantee two-day delivery of a 1kg package from Bangalore to NIST is a Fedex International Priority Pak, which costs half a week's salary for a typical Indian professor.

I understand that NIST needs a signed printed statement regarding patents etc., but this statement is not urgent: it can be sent by mail later, or hand-delivered to NIST at the first workshop.

On a related note, requiring fax numbers and telephone numbers is silly.

5. The draft needs a general round of proofreading. For example, Wiener is not "Weiner", the JoC97 link does not work, and 4.B.4 is incomplete.

---D. J. Bernstein
University of Illinois, Chicago

\>>
I think my main comment about the draft NIST call is that they need to be careful with their discussion following the target security strengths listed in section 4.A.4. It's important that they make it absolutely clear that this discussion is about parallelisation of *quantum* attacks and is not applicable to classical attacks. This is particularly true of the clause "... NIST recognizes that extremely serial or extremely parallel attacks (e.g., those that have a time depth or space complexity exceeding 2^100) may be of minimal practical importance". The danger is that someone reading this might incorrectly interpret it as a giving a memory bound for classical attacks. Given recent heated arguments around the NTRUPrime security analysis, this should be avoided if at all

possible. For example, it might be better to remove the specific figure 2^100 from the clause above. (On the other hand, if NIST do intend to give a memory bound then it should be applied consistently to *all* large-memory attacks.)

Peter Campbell
CESG

>>

To whom it may concern:

I have read through the proposed document and supporting materials and have two comments:

1 --
In section 2.C.1 (Implementations) Submitting an implementation solely on the Intel x64 processor ignores the vast and ever-growing population of smaller processors that make up the Internet of Things. Quantum-resistant solutions optimized for such a capable machine may not scale down to 8 or 16 bit microcontrollers. To that end we propose that you include such smaller devices (e.g. 16-bit MSP430, and 8-bit 8051 and/or AVR8) in your testing and evaluation.

2 --
The proposed testing API is extremely problematic. Specifically, it assumes that Keys and Signatures are a constant size. There are definitely real algorithms where this is not the case, and each keypair (and signature) generated requires dynamic memory. In order to apply these variable-length algorithms to the process would require a change to the testing API that allows for dynamic sizes.

We see two possibilities to handle this extremely important use case:
1 Set the sizes so high as to be sure to include even the largest possible keys/signatures. The problem is that this would necessarily increase the amount of memory/storage required, and it's still potentially possible to hit a sample that goes beyond the boundaries, in which case the system either has to try again or give up.
2 Fix the APIs themselves to handle dynamic-size responses. This would allow an algorithm to return data objects of varying lengths.

We would encourage taking approach #2.

To this end we would propose a change to the API that enables dynamic responses, perhaps something like the following (with similar changes for the KAT versions):

```
typedef struct {unsigned long long len; unsigned char*
```

```
buf;}buffer_t;
typedef buffer_t PublicKey;
typedef buffer_t PrivateKey;
typedef buffer_t Signature;

int crypto_sign_keypair_dyn(
PublicKey* pk,
PrivateKey* sk
);

int crypto_sign_dyn(
Signature *sig,
const unsigned char *m, unsigned long long mlen,
const PrivateKey sk
);

int crypto_sign_open_dyn(
const unsigned char *m, unsigned long long mlen,
const Signature sig,
const PublicKey pk
);

void free_buffer(buffer_t buf);
```

Thanks for your consideration,

Derek Atkins
Chief Technology Officer
SecureRF Corporation

>>

Comments and questions on the NIST call for PQC standards.

Proposed Minimum Acceptability Requirements

For Part 2,

what if the submission infringes on others' patent or patent application and does not disclose it?

In Part 4, it says:

"The submission package shall provide concrete values for any parameters and settings required to meet or exceed (to the best of the submitter's knowledge) the relevant security targets in Section 4.A.4, for the appropriate security models in Sections 4.A.2 and 4.A.3."

Does this mean for each security targets, a submission can have more than 1 set of parameters?

Must each submission submit at least one set of parameters for each security target?
Proposed Evaluation Criteria

4.A.4 Target Security Strengths,

Should the memory complexity be taken into account for classical attacks? If an attack on a scheme requires an tremendous memory, can it be considered secure?

For the quantum attack, should the number of quantum bits need be considered?

4.C Algorithm and Implementation Characteristics

How about the versification of the implementations? Should the implementations be easily versified that it indeed implements what is theoretical requires? (Like sampling etc?)

4.B.4 Decryption Failures

“Some public-key encryption algorithms, even when correctly implemented, will occasionally produce ciphertexts that cannot be decrypted. For most”

What is the threshold for decryption failure?

Jintai Ding
University of Cincinatti

>>

Dear NIST,

I have the following two suggestions.

1. For the part "Algorithm Specifications And Supporting Documentation".

In Section 2.B.1. paragraph 3 the current text is:
"To facilitate the analysis of these algorithms by the cryptographic community, submitters are encouraged to also specify parameter sets that provide lower security levels, and to provide concrete examples that demonstrate how certain parameter settings affect the feasibility of known cryptanalytic attacks."

I suggest this sentence to be moved as a separate section that states the following:

"To facilitate the analysis of the submitted algorithms by the cryptographic community, submitters are required to specify parameter sets that provide lower security levels, and to provide concrete examples that demonstrate how certain parameter settings affect the feasibility of known cryptanalytic attacks."

2. Then in connection with this change, in the part "Proposed Evaluation Process" in Section 5.A the paragraph

"When evaluating algorithms, NIST will make every effort to obtain public input and will encourage the review of the submitted algorithms by outside organizations; however, the final decision as to which (if any) algorithm(s) will be selected for standardization is the responsibility of NIST."

to be changed to the following paragraph

"When evaluating algorithms, NIST will make every effort to obtain public input and will encourage the review of the submitted algorithms by outside organizations; NIST encourages the reviewers to demonstrate their findings and attacks both on the versions with parameters that achieve full security levels, as well as on the provided parameter sets with lower security levels; however, the final decision as to which (if any) algorithm(s) will be selected for standardization is the responsibility of NIST."

Rationale for these suggestions:
NIST crypto competitions are highly respected events in the cryptographic and information security community. It is a prestige to participate in the competition and to publish attacks on the proposed algorithms. In the heat of the debates and the competition, there will be a lot of overrated attacks that actually are not so efficient as the attackers would claim. I am proposing the above changes in order to protect the dignity of both the submitters and the attackers and to save a precious time and efforts by the NIST employees and the whole crypto community. If in the submission documentation there are obligatory test parameters that have very low security margin, any published attack on the schemes is encouraged to be demonstrated *practically* on those low level parameters. That will be seen as a correct and honest attempt to analyze the scheme, not just as a malicious attempt to discredit the attacked algorithm.

Best regards,
Danilo Gligoroski

Norwegian University of Science and Technology (NTNU)

>>

Hello,

I have one comment on the document « Proposed Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process ».

In Section 4.A.4, five target security strengths are listed. On security strength 2 and 4, it is explained that the reference is brute-force collision attacks against SHA-256/SHA3-256 and SHA-384/SHA3-384.

However, in the paper « Cost analysis of hash collisions : Will quantum computers make SHARCS obsolete ? » by Daniel J. Bernstein (https://cr.yp.to/hash/collisioncost-20090517.pdf), it is explained that :
« There is a popular myth that the Brassard-Hoyer-Tapp algorithm reduces the cost of b-bit hash collisions from 2^(b/2) to 2^(b/3); this myth rests on a nonsensical notion of cost and is debunked in this paper. »

And later in the same paper :
« The best time claimed by Brassard, Hoyer, and Tapp in [6], and by Grover and Rudolph in [10]|is 2(b/2) /M^(1/2) on a size-M quantum computer. »

Based on this paper, it would mean that:
- For level 2 : 128 bits classical security /80 bits quantum security with the reference to a quantum brute-force collision attack on SHA-256/SHA3-256 would require a quantum computer of size 2^96 to find a collision on SHA-256/SHA3-256.
- For level 4 : 192 bits classical security / 128 bits quantum security with the reference to a quantum brute-force collision attack on SHA-384/SHA3-384 would require a quantum computer of size 2^128 to find a collision on SHA-384/SHA3-384

My comment is that a clarification seems needed on the meaning of the target security strengths 2 and 4, assuming that they are kept in the final version.

Regards

Aline Gouget
Gemalto

>>

Maybe NIST could consider another set of evaluation critera, resistance against traditional physical attacks. Something, along the line described below.
--- ---
Implementation issues – Traditional attacks

The algorithm has to have a reasonably complex implementation, which resists known (published) types of physical attacks, with documented exceptions. The document describing exceptions tells, in what type of environment the implementation works safely, and what kind of physical protection it may need. E.g. FIB probing or photo voltaic charge detection can be prevented by physical means, like chip covers, but preventing leaks of secrets by timing- or simple power analysis needs careful implementations.
A submission my state that their algorithm is intended only in physically protected environments, where side channel attacks are prevented by the physical protection.

Side channel attacks

Some of the following side channel attacks (maybe even more?) need to be mitigated by proper implementations:
☐ Timing, EM radiation, SPA, DPA (High-order, multivariate…) attacks
A potential family of protection measures may include
☐ Random masking schemes on keys or on the secret input – such that the masking and de-masking procedure is simple enough, such that they can be made of low leakage

Fault injection attacks

The algorithm should have implementations of reasonable complexity, which leak no secrets at a small number (e.g. < 4) of targeted faults.
--- ---
Laszlo Hars, PhD
Chief Crypto Architect
Boeing Secure Computing Solutions

>>

To whom it may concern:

My name is David Jao. I am the designer of post-quantum cryptosystems based on the isogeny problem over supersingular elliptic curves.

I would like to draw your attention to the following areas in your post-quantum cryptography draft requirements and evaluation criteria which I believe would benefit from further clarification:

(Section 2.B.1) In prior NIST standardization processes, there was only one functionality being evaluated (e.g. block ciphers for AES, and hash functions for SHA3). In this draft, we have potentially up to three distinct pieces of functionality (encryption, signatures, and key exchange) being evaluated at the same time. Will NIST be evaluating all the algorithms in a single submission package together, or will the three types of schemes be evaluated separately? If the latter, why not just accept separate submissions in each category rather than combining schemes of each type into one submission? It would be helpful if NIST could clarify the rationale behind accepting multiple items in one submission. For example: "If a submission includes more than one type of scheme, NIST will evaluate the schemes of each type separately. However, submitters may choose to combine different types of schemes into a single submission in order to share software code among multiple schemes within the submission."

(Section 2.C.1) Does the requirement for ANSI C source code preclude the use of assembly language optimizations? Your draft proposal does not specifically address this question. I would like to see assembly optimizations (at least inline ASM) allowed for the optimized implementation, because otherwise the implementation would not be representative of real-world conditions, especially for number-theoretic cryptography which relatively speaking benefits more from assembly optimization than other families of cryptosystems. It seems to me to be a little inconsistent to specify a target platform (Intel x64) and not allow platform-specific optimizations.

(Section 4.A.2) IND-CCA2 makes perfect sense for public-key encryption, as well as key transport, but does not apply to key establishment in isolation. It is not clear what security model NIST is proposing for key establishment. All existing security models for key establishment that I'm aware of are rather heavyweight, and the vast majority are tailored to authenticated key exchange, which you mention only in Section 4.C.1.
As I am not an expert in security models for key establishment, I defer to others on the question of what model to use. If NIST requires external assistance in this regard then a public request for input would be appropriate.

(Section 4.A.4) Typically, it is not possible to tune the classical and quantum security levels of a scheme separately; a given choice of parameters will imply a fixed classical security level and a fixed (possibly different, but not independently tunable) quantum security level. For example, any isogeny-based scheme with 128-bit classical security automatically has 80 bits quantum security; therefore security level number 1 in this section is superfluous for isogenies, as any such parameter choice automatically satisfies security level number 2. It would be helpful to have explicit guidance on what to do in such situations. I suggest adding an explicit guideline to ignore such inapplicable security levels.

(Section 4.A.4) In your FAQ (http://csrc.nist.gov/groups/ST/post-quantum-crypto/faq.html) you state that "quantum security should be defined as the minimum possible value of log(depth*(squareroot (space))) PLUS A CONSTANT" (emphasis added). The phrase "plus a constant" in my interpretation allows for some fudge factor (note the sign of the constant is not restricted to being positive!), so that something which (for example) strictly speaking might provide only 125 bits of security could be considered to provide 128 bits. Unfortunately, this phrasing does not appear in the PDF of your actual draft proposal. Instead, the draft proposal uses the phrasing "meet or exceed" which is less flexible. For number-theoretic cryptography, crossing a machine-level word size boundary incurs a huge performance penalty, and for this reason it is extremely common to use parameters which meet a given security level only up to the addition of a small constant (e.g. Curve25519 provides only 125-bit security). Therefore I would like to ask that the draft proposal be amended to include the "plus a constant" phrasing.

(Section 4.B) This section lists cost considerations which apply specifically to public-key cryptosystems and signature size, but does not list any cost considerations which apply specifically to key exchange. I would suggest that some attempt be made to specify some cost considerations for key exchange protocols, or else explicitly request comments on this topic from the public. Examples of cost considerations specific to key exchange include the number of rounds of communication, the number of static keys and ephemeral keys required, and whether or not the protocol supports (or alternatively requires) synchronous and/or asynchronous communication.

David Jao
University of Waterloo

>>

Comments on "Proposed Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process", draft from August 2016

In general I'm positively impressed with the document and how it reflects discussions e.g. during the PQCrypto workshop. In the following I will raise one major and one minor issue and then give some detailed comments and suggestions. Please feel free to contact me if any of this is unclear.

The current document is still inconsistent in what categories NIST is asking for submissions. This matches the discussions in February when it was left open whether NIST would ask for a key exchange mechanism. The current document first speaks of 'key exchange' and later of 'key establishment'. The API documentation uses both words interchangeably.

It should be made clear what precisely is asked for. Most people understand key exchange to match the functionality that Diffie-Hellman key exchange is offering: two keypairs determine a shared secret without communication; keys can be reused, e.g. A's published key can lead to a shared secret with Bob, using Bob's public key, and one with Charlie, using Charlie's public key. This is the functionality matching eBACS's DH function API: given one public key and one secret key, compute the shared key.

The functionality described early in the draft call for proposals changes this to distinguish between an initiator message and a responder message. This does not match common understanding of key exchange, which is also why the eBACS API does not fit.

Later on the call document -- now speaking about key establishment -- highlights the desired result: key transport and forward secrecy. The latter implies that new public keys must be generated frequently, requiring efficient key generation and small key sizes for transmission. I think it makes much more sense to ask for submissions for this scenario and a scenario with long-term public keys instead of asking for submissions for key exchange and encryption.

Realistically, public-key encryption is used only to transmit a key which is then used in a symmetric cipher; this is also recognized in the call document. The formal treatment of this is most advanced in the KEM/DEM framework: the public-key system is used as a key-encapsulation mechanism, which ensures that sender and receiver obtain the same key, and that key is then used in the data-encapsulation mechanism to encrypt the message. This avoids issues of padding.

To summarize, I recommend asking for submissions for two types of KEM:

- KEMs in which the receiver has a long-term public key; obtaining the key is outside the scope of specifying the KEM and
- KEMs in which at least one side generates and transmits a fresh public key.

instead of submissions for encryption and key exchange/establishment.

The second type of KEM scheme should be efficient enough that keys can be generated for each key transport, but ideally not break down completely if keys are reused. It would be good for the submitters to specify how key reuse would affect the security of their system. I understand that the latter might be captured under the property of 'robustness'.

The minor issue is that I would recommend to request a constant-time implementation of each retained algorithm in the second round. Timing attacks are one of the most basic and thus most powerful attacks and each implementation (in software or hardware) needs to be protected against it. The call currently says that you'll take ease of SCA protection into account; that will be much easier and more meaningful if the submitter has to send in a protected implementation. This might be seen as a burden by some, so I wouldn't require it as part of the submission package, but each proposer group can get help by the time the second round comes along.

Editorial comments:

1. p.2 "in the event that large-scale quantum computers" should be replaced by "to prepare for the event that large-scale quantum computers" (or similar). It is too late to change once a QC is built. Even if long-term security is not a concern, roll out would take too long. (This is captured well a few paragraphs further down but confusing here).
2. p.4 It is unusual for key-exchange schemes to distinguish between initiator and responder messages. It is normal to request that the scheme defines a shared secret for each pair of public keys. If the definition is different this should be stated early on. It might be that you're instead asking for public-key encryption schemes for one-time use public keys with fast key generation (which is different from the typical DH message flow). See above.
3. p.5 In case a submitter has submitted his implementation to eBACS there will be benchmarks on a multitude of processors. Describing all the platforms and all the results would unnecessarily blow up the submission document. I recommend to allow inclusion by reference to the page for the primitive on bench.cr.yp.to. Of course, the submitter should still be free to highlight some processors and implementations if he chooses to and then be required to describe the platform, so this is a suggestion to permit a reference in addition.
4. p.7 Do you really want to receive all pdf files of papers or are links to public versions of the paper sufficient? Can people set up a webpage with supplemental material including links? I foresee a problem with copyrights: authors usually have the right to put their author copy online; if their work is relevant to my submission, I can put a link to their work on my homepage without violating any copyright, but I cannot submit the paper to NIST and make a statement about the copyright. This basically means that I cannot use papers published by others.
5. What do you mean by "unusual vulnerabilities"? Would this be e.g. key reuse in a scheme where decryption failures can be used to determine the secret key? or the fragility in ECDSA with nonce reuse? It would be good if you could be more

specific. For the avoidance of doubt, please specify whether assembly subroutines or intrinsics are acceptable.

6. p.10 "the quantum-resistant algorithm evaluation process": elsewhere you've changed to 'post-quantum' so I suggest to adjust the phrasing here to match.
7. p.13 Same comment regarding key exchange being asymmetric in initiator and responder as above.
8. p.16 You mean 2^k executions of AES on the given architecture? See the detailed analysis of the costs of using Grover on AES (PQCrypto 2016); are you considering the estimated cost of 2^32 to equal 1? I've seen the FAQ on this topic, but that didn't help. Some algorithms suffer much more from requiring the steps to be reversible than others, so it will be necessary for cryptographers to understand quantum algorithms in any case. In principle this is not a new problem -- 1 ECC operation is not the same an AES operation and we don't even know how to define the exact security level of elliptic curves. Counting operations in quantum algorithms is at least as hard. While I think that we cannot reach a way of comparing security between AES and post-quantum systems, I strongly suggest that systems using similar primitives count their efficiency the same way, e.g. code-based systems against which information-set decoding is the most efficient, should have a consistent way of using the cost of one loop; same for lattice-based systems using sieving. These ways might not be accurate in the end, but at least they allow comparisons within one class of algorithms. Eventually it is necessary to compare systems across different primitives, but by then more detailed research on current and quantum attacks will have happened.
9. p.17 I often encounter practitioners who take "perfect forward secrecy" to mean that a later attack cannot do harm and misunderstand it to mean that they can continue to use ECC till quantum computers arrive. They are surprised when they understand that having the public messages + keys is enough to later on break the scheme with a quantum computer. Due to this confusion I have started to use "key erasure" for this concept; given that this is not yet a common term it is necessary to add a parenthetical comment "(also known as perfect forward secrecy)" for now. Please be more specific when referring to this concept. Do you accept schemes that become insecure if the same key is reused or do you mean to ask for schemes which have very fast key generation time and do not require much space for the key transmission?
10. p.17 While it is grammatical to say 'resistance to side-channel attack' I would suggest to use the plural 'attacks' here, because there are many different attacks and a system might not be equally defendable against all of them. It might be useful to include a ranking of what types of attacks must be covered, e.g. timing attacks are applicable in significantly more situations than power analysis.
11. Regarding multi-key attacks: a brute force attack is always sped up when many targets can be attacked; you might want to specify that this would be with respect to the best attack or be more precise in the 'an advantage'. Also it

depends on the number of available keys -- after very many keys, brute force search might be the best possibility, so limiting to 2^64 or 2^96 keys seems reasonable.

12. The 'compromise a single key pair' case could be made more precise: I assume you mean attack any single key, so 1 out of n vs. n out of n; using a bit of notation should help here.
13. "established body of cryptographic research" is too narrow and excludes work done in number theory or complexity theory which studies the same problems but at different venues (compare to RSA drawing on the body of factorization research, which gets published in crypto venues, but also at ANTS, Math Comp,, and other journals).
14. p.18 same comment regarding "perfect forward secrecy" vs "key erasure"(twice)
15. 4.B.4: I assume that this text deals with accidental decryption failures which are a nuisance and should thus be avoided. I suggest adding that you consider attacks using decryption failures as attacks, if failures are sufficiently common or can be triggered by an attacker.
16. Nitpick: what do you mean by 'encrypting the same _cipher_text'? Btw. I'm not sure that one can always reach _acceptable_ rates, this really depends on how bad the scheme is.
17. p.20 "All proposed changes must be proposed by the submitter;" I would add that the submitter can submit implementations prepared by third party with the permission of the third party. At least my understanding is that you want to ensure that the proposer endorses the implementation, but it is not necessary for the implementer to become part of the proposers team.
18. Other files: kat.pdf still includes instructions to Sara. (twice). api-notes.pdf:
19. Skipping most comments because the specifications of what is wanted are still not fixed.
20. Please note that the supercop benchmarking framework generates KATs from submissions; submitters can also specify these in a separate file. This means that you don't need to change the API to include those.

Tanja Lange
TU Eindhoven
>>>

Dear NIST,
I have two comments about the draft document.

The first concerns the target security requirements in section 4.A.4. I do not understand the relationship that is drawn between the security of public key primitives and brute-force attacks on SHA/AES. Unlike SHA/AES, the best attacks against public key primitives are not brute force, so there is no reason to assume that the effect of Grover's algorithm on the quantum security of such primitives is analogous to its effect on symmetric ones such as SHA/AES.

Of course, when public key primitives use SHA as a sub-routine, the parameters of SHA should be set appropriately to resist quantum attacks (for example, in Fiat-Shamir constructions, one can use a hash function with 128-bit outputs to have 128 bits of classical security in the random oracle model, but would most likely need to use SHA-256 for 128-bits of quantum security.) But just because one needs to increase the security of the hash function does not imply that anything needs to be increased in the rest of the construction. For example, there are no known quantum algorithms for lattice reduction that outperform classical ones by any significant margin. Thus other than adjusting for a larger output from SHA, there would be no reason to increase the hardness of the lattice problem in the aforementioned Fiat-Shamir example.

Perhaps something reasonable that could be mandated is that if one uses hash functions or block ciphers within the primitive, then they must at a minimum have all the classical/quantum security features of SHA-256 and AES-256 (or one can just use SHA-256 or AES-256). But I believe that it would be very wasteful to set parameters so that the whole public key scheme is 256-bit secure classically when what we really want is that the scheme cannot be broken in 2^128 time on a quantum computer.

My second comment/question is about 4.B.4. Would it be possible for NIST to specify precisely what are the acceptable rates of decryption or key-agreement failures? If these failures lead to attacks, this is of course unacceptable. But if, for example, with probability 2^-30 the key-transport protocol fails and thus needs to be redone, is this something that's acceptable?

Thank you very much and best regards,

Vadim Lyubashevsky
IBM Research - Zurich
>>>

Dear NIST Team,

we have a comment to your DRAFT Call for Proposals:

For proposals of digital signatures, could you make a more clear separation between a mode vs. a underlying primitive that instantiates this mode?

We believe such separation would be less applicable to other categories like encryption or key-exchange, but a similar distinction came up, e.g., in the CAESAR competition.

Best regards,

Christian Rechberger, Martin Lauridsen
Technical University of Denmark, DTU
Willi Meier
FHNW

>>>

Please find enclosed comments from us on the proposed submission and evaluation criteria for the Post-Quantum Cryptography Standardization Process.

Page 3, first paragraph of Section 2.

The current submission process has a single deadline. Given the current state of post-quantum cryptography, it may be preferable to separate submissions into several generations to allow for new findings to be accommodated.

Page 7, first paragraph of Section 2.C.1

It is unclear the reason to include optimized source code within the submission package. Typically, optimizations are a way for industry to differentiate product offerings from each other and as such should be considered out of scope for the standardization process.

In addition, optimized code will often contain assembly which goes against the specification requirement of “written in ANSI C”.

Page 7, second paragraph of Section 2.C.1

Perhaps add encryption, decryption and shared secret generation for completeness.

Page 8, first paragraph of Section 2.C.3

Typically source code is also considered to be “written material”. To avoid ambiguity, perhaps reword as “supported documents.”

Page 10, last paragraph of Section 2.D.1

The first sentence says “the quantum-resistant algorithm evaluation process.” For consistency we think you mean “the post-quantum algorithm evaluation process.”

Page 12, first paragraph of Section 2.E

The list includes “source code”. The reference to English is ambiguous then since source code would be written in ANSI C. Instead do you mean that comments in source code should be written in English?

Page 14, Section 4.A.1

Instead of saying “IPSec” perhaps instead use “IPSec/IKE” is IKE is where the public key cryptography is. You may also consider adding S/MIME.

Thanks,

Mike Brown and Atsushi Yamada
Isara
>>>

Dear Sir or Madam,

the Bundesdruckerei is producing ID documents and runs a trust center for various use cases. We are currently involved in the EU project PQCRYPTO, the ISO SC27 WG2 efforts for Post-Quantum-secure algorithms and we are working on post quantum secure implementations with our subsideary company Genua. We are very excited that NIST is moving forward in standardization of PQC.

Unfortunately, standardization committees in general have suffered from a decline in credebility in the past years. Many people think that the standardization process can be manipulated by powerful industry lobbying and governmental intrests. We think, that a modern standardization should include the maximum amount of transparency possible. NIST has done a great job with the AES and the SHA-3 competition. We hope that this success can also be achieved with the standardization of Post-Quantum-Cryptography.

PQC will most likely be used for applications with long term security. Those applications are already in danger today, because encrypted communication can be stored forever and could be analyzed later with a quantum computing. We see a big need for PQC as of today. But of course, new algorithms should be evaluated well from many aspects. The mandatory criteria you have outlined are certainly necessary. Though we are looking at many of your optional features as mandatory.

One of the main problems is that PQC should receive a good amount of cryptanalytic attention before standardization. Therefore, we need to measure the confidence in an algorithm somehow. Means to create this confidence may, for example, be a proof of security, the number of scientific citations/reviews or simply the time an algorithm has been out there for public review. With the urge today, this effectively means that we should concentrate the standardization efforts on algorithms that are known for a longer time. There are some promising algorithms that have been developed in the past years, but evaluation still needs some time.

Another essential part in the process of establishing this confidence is exposing a detailed and well-supported design rationale. This rationale can be used by experts to verify that the design indeed follows the design strategy (recall the curves that were not generated according to the public procedures), and verify whether the security margins offered by the design are consistent with the design strategy and target.

We hope this helps you refining your evaluation critarias. We are looking forward hearing your feedback.

Best regards,

Frank Morgner
BSI
>>>

16 September 2016
National Institute of Standards and Technology
Computer Security Division

Subject: Comments on *Proposed Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*

Microsoft Corporation appreciates the opportunity to submit comments on the subject draft. We believe that this standardization process is both timely and critically important. NIST's past standardization activities for AES and SHA-3 were outstanding examples of evaluations of candidate cryptographic algorithms, and the algorithms selected through these processes are now being deployed throughout industry worldwide. We look forward to a similar outcome from this process, and an open and transparent process with clear technical guidelines and evaluation criteria will help ensure that the results of this process are trusted and credible.
Executive summary and high-level recommendations

We have suggestions that we believe will improve the proposed standardization process and the outcome. Our comments focus on the following areas:

- Intellectual Property Rights
- Performance Measurement
- Evaluation Under Real-World Scenarios
- Security Levels
- International Standardization

A. Intellectual Property Rights:

The current draft includes the following statement on Intellectual Property in Section 2.D:

> *NIST has observed that royalty-free availability of cryptosystems and implementations has facilitated adoption of cryptographic standards in the past. As part of its evaluation of a PQC cryptosystem for standardization, NIST will consider the assurances made in the statements by the submitter(s) and any patent owner(s), with a strong preference for submissions as to which there are commitments to license, without compensation, under reasonable terms and conditions that are demonstrably free of unfair discrimination.*

Further, the proposed required Statement by Patent Owner(s) in Section 2.D.2 explicitly allows for a patent holder to select an option of RAND (reasonable and on discriminatory) licensing with compensation.

This is a change from the SHA-3 competition in that royalty-free licensing is not required by the proposal but is merely a factor to be considered. We have seen in the past how ambiguity and licensing have hampered the adoption of new cryptographic technologies. It is critical that NIST maintain the same intellectual property rights disclosure and release requirements that were set out for the SHA-3 competition, namely that all submitters be required to release any and all IP claims as a condition of entry, and that each submitter agree to unrestricted, royalty-free use of their work.

Additionally, we note that the proposed approach to Intellectual Property Rights for this competition conflicts with NIST's stated commitment in NISTIR 7977 on this specific issue. See NISTIR 7977, Section 7 ("Policies and Procedures for the Life Cycle Management of Cryptographic Standards and Guidelines), Subsection 4 ("Define a Specific Plan and Process"), bullet point "Hold a Competition" (bottom of page 18 of NISTIR 7977), where NIST writes [emphasis added]:

> If NIST decides to pursue the development of a standard or guideline, it may use an open competition. When a competition is used, interested parties will have an opportunity to participate in the competition by reviewing core requirements and evaluation criteria, publishing research papers, submitting comments, and attending public workshops. Researchers worldwide may contribute candidate

> designs and papers on the theory, cryptanalysis and performance of the candidates. The winning submitters are recognized, *but agree to relinquish claim to intellectual property rights for their design so that the winning candidate can be available for royalty-free use.*

This process is clearly a competition as defined in NISTIR 7977, so NIST must adhere to the IPR commitments it made for competitions in that document. To that end, Microsoft strongly suggests that the "reasonable terms and conditions" IPR language be struck from the proposal in favor of the exact language used in the SHA-3 competition, guaranteeing that the selected algorithms be "available on a worldwide, non-exclusive, royalty-free basis."

B. Performance Measurement

i. Constant Time Implementations

Section 2.C.1 ("Implementations") references "optimized implementations" that will be used for performance benchmarking. Real-world applications of cryptographic schemes require constant-time implementations as a minimum to protect against timing and cache-timing attacks.

To ensure that "optimized implementations" reflect what would be deployed, and to enable apples-to-apples comparisons, all "optimized implementations" submitted for this effort should be designed to be constant-time. Second-round updates to submissions may make updates to fix constant-time-related bugs in first-round submissions.

ii. Performance Tooling

The performance evaluation of "optimized implementations" must be done by NIST directly or by an independent and neutral third party not affiliated with any party involved in any submission. The tools used in this evaluation must be open, independent, auditable and neutral, their code must be freely published for inspection, and must not be owned by or affiliated with any party involved in any submission. No submitter can be involved in performance evaluation in any capacity.

iii. Performance Testing Scenarios

The performance evaluation should cover the following platforms at a minimum: a 64-bit processor "server class" and a 32-bit processor "mobile class". In addition, testing should be conducted on 8-bit and 32-bit microcontrollers, and be evaluated on at least one alternative hardware platform (e.g., FPGA).

C. Evaluation Under Real-World Scenarios

i. Hybrid Modes:

In Section 1, NIST writes that "hybrid modes" which combine quantum-resistant cryptographic algorithms with existing (not necessarily quantum-resistant) cryptographic algorithms are out of scope for the competition. We believe this limitation is overly restrictive for two reasons. First, some proposed quantum-resistant schemes may have benefits when combined with certain classical schemes, and NIST's evaluation process should be able to consider such benefits[1]. Second, ease of integration and engineering compatibility with classical cryptography must be a consideration in the evaluation of submitted algorithms as a desirable property. It is most likely that post-quantum cryptographic schemes will be deployed in such hybrid modes first and be used alongside classical cryptography for a significant amount of time. Candidate quantumresistance schemes must be evaluated in the wider context in which they will be applied, which will include integration with classical cryptography.

ii. Protocol Scenarios

NIST should identify several high-priority protocol scenarios, such as TLS, for evaluating and testing submitted schemes. Ease of integration with the most commonly used security protocols and performance in such scenarios must be an important evaluation criteria.

In the second round, those candidates selected to continue on should be asked to apply their submissions to selected real-world protocols, such as TLS, to further the evaluation.

D. Security Levels

In Section 4.A.4 ("Target Security Strengths"), NIST identifies five target security strengths for which submitters will be asked to provide parameter sets. We are concerned that the lowest security strengths identified are too low: the requirements should encourage strong and conservative security levels. There are also too many security strengths specified. Reducing the number of parameter sets required of submissions will simplify the evaluation. We suggest that NIST remove target levels (1), (2) and (3) and replace them with a target level of 128 bits classical security / 128 bits

[1] For a practical example of such ancillary benefits see C. Costello, P. Longa and M. Naehrig, *Efficient Algorithms for Supersingular Isogeny Diffie-Hellman,* recently presented at Crypto 2016 and available online at http://eprint.iacr.org/2016/413. In this paper the authors present a post-quantum key agreement scheme based on supersingular isogenies, and in Section 8 they present a strong ECDH+SIDH hybrid ("BigMont") that leverages the underlying field arithmetic of the post-quantum scheme to provide a parallel ECDH key exchange for very little overhead. NIST's current proposed language would prohibit NIST from considering hybrid benefits from such schemes.

quantum security, and that this new level be the minimum target level. Target levels 4) 192 bits classical security / 128 bits quantum security and 5) 256 bits classical security / 128 bits quantum security should be consolidated to one level, and then a third higher level should be added to provide more breathing room in the face of continuing cryptanalytic advances. We suggest that this new higher level be 256 bits classical security / 192 bits quantum security.

Any scheme that has an efficiency or technical obstacle, must clearly justify the limitations that prevent it from achieving the desired security level.

E. International Standardization

In our letter of 25 March 2015 and the accompanying formal comments on the then draft NISTIR 7977, Microsoft stressed the importance of submitting key NIST standards to standards development organizations (SDOs) with international scope, in particular standards that result from competitions. For this competition we strongly encourage NIST to plan to submit the selected algorithms to one or more international SDOs after the resulting FIPS or SPs are completed. There is an opportunity when establishing new post-quantum cryptography standards to have fewer national variations worldwide.

Conclusion
We believe these modifications will strengthen the proposed process, will ensure the strongest technical outcome from the evaluation, and will provide the best transparency and assurance to the community.

Thank you again for the opportunity to submit these comments. We would be happy to discuss these recommended modifications, or any other aspect of the draft, with you.

Sincerely,
Brian A. LaMacchia, Ph.D.
Director, Security and Cryptography, Technology and Research Group
Microsoft Corporation
>>

General Comments:

* The terminology in the document should be more consistently used; for example, quantum-resistant vs. post-quantum.

* The document should make the use of the term "parameters" clearer; whether "parameters" refer to a general parameter set for the primitive, a specific choice of public parameters or even a full suite of test parameters. For example, ECDH with a 384-bit prime curve, ECDH with the P-384 curve or ECH with curve P-384 and test key pairs.

* There should be more care with the use of pseudo-normative language (shall/should and must/may) as this could lead to problems later in the competition. As an example, if the README file for a submission doesn't list all of the files on the CD, then someone could claim that it is not a "complete and proper" submission as it fails to meet the mandatory requirements in section 2.C.4:

o The "README" file shall list all files that are included on this disc with a brief description of each.

Specific Comments:

Section 3.3.c - The scheme shall be capable of supporting a message size up to 2^63 bits.". For hash-based signatures, the number of messages that can be signed is limited by the parameter set used. If the submission is supposed to provide concrete values for the parameter sets, should there also be a requirement on the number of messages that it must be capable of signing? A bound of 2^64 messages is given in Section 4.A.3 but then it states that "attacks involving more messages may also be considered".

Section 4.A.2 - "NIST intends to standardize at least one scheme that enables "semantically secure encryption" with respect to adaptive chosen ciphertext attacks". This statement is slightly confusing. Does this imply that NIST might also standardize other encryption schemes that are not secure against active attacks? It is not clear as well what security model is expected for key agreement.

Section 4.A.4, paragraphs 4-7 - While understanding the desire to find a way to fairly address the issues with parallelization, this discussion somewhat undermines the clear definition of security targets given in the previous paragraphs. One can imagine arguments similar to those about what does and doesn't constitute an attack during the SHA-3 competition.

Section 4.A.4, paragraph 7 (starting "Since NIST's goal") - implies that they are equating s bits of quantum security with the time t it takes to break a 2s-bit key by Grover's algorithm given w much space. It is commonly held that for Grover's algorithm, we have t^2 * w = 2^(2s). If that is NIST's definition of quantum security, it should be explicitly stated. Does this mean that an attack that uses 85 bits of quantum memory and 85 bits of quantum depth is not feasible because one could only attack a 255-bit key with Grover's algorithm using such a circuit?

There are definitely quantum algorithms which may be of interest that parallelize better than that. For example, on arxiv.org there is a 2012 paper titled Efficient Distributed Quantum Computing which shows that for a certain quantum problem, you have t^2*w^2 = 2^(2s) (i.e., Grover square root the circuit area, not circuit depth). As a result, there may be some hash-related properties that you can solve a 360-bit hard problem classically.

NIST should be aware that using the given definition for quantum resistance may inadvertently call some things weak that they don't necessarily want to, given that this costing method is very generous towards higher-memory methods.

Section 5.A - Wouldn't NIST have to have compelling reasons to select an algorithm for standardization if it didn't receive additional analysis during the later phases? Even with compelling reasons, wouldn't there be a strong push back from the community? It might be worthwhile to provide clarification for this scenario.

Some minor editorial comments on spelling/grammar have been omitted.

National Security Agency
>>

Dear NIST post-quantum team,

Thank you very much for running this competition. As you are asking for comments on the draft, here are a few things that you might want to consider:

- In Section 4.A.2 ("Security Model for Encryption/Key-Establishment") you are treating Encryption and key exchange together and are asking for IND-CCA2 security. It may be interesting to distinguish the two cases of public-key encryption (to a long-term key, requiring CCA security) and ephemeral key exchange, which needs only passive security. For example, the "NewHope" key exchange by Alkim, Ducas, Pöppelmann and myself, which is currently used in Google's post-quantum experiment, is explicitly *not* offering CCA security. We could, for the sake of the competition, modify it to achieve this goal, but this would sacrifice security and performance for the ephemeral case, where CCA security is not required. On August 26, Jacob Alperin-Sheriff sent us (the NewHope authors) an e-mail suggesting that NIST might be intersted in receiving NewHope (or maybe by next year an improved version) as a submission to the competition, but with the current call I don't see how it would fit in.

- Section 4.A.4 asks for submissions at different security strenghts. What I find interesting is that there is no level offering the same security against classical and quantum attacks. For example, imagine an algorithm that offers N bits of security both against classical and quantum attackers. Personally, I want crypto that offers 128 bits of long-term security, so the only way to fit this algorithm into the proposed security levels is to scale N to 192, although most users would be happy with 128-bits of pre-quantum and post-quantum security.

Best regards,

Peter Schwabe
Digital Security Group
Radboud University
>>>

Dear NIST,

I have one suggestion for the Proposed Requirements and Evalutation Cruteria (DRAFT).

Section 2.B.1, paragraph 3:
"For algorithms that have tunable parameters (such as the dimension of some underlying vector space, or the number of equations and variables), the submission document shall specify concrete values for these parameters. If possible, the submission should specify several parameter sets that allow the selection of a range of possible security/performance tradeoffs. In addition, the submitter should provide an analysis of how the security and performance of the algorithms depend on these parameters."

Suggestion:
" For algorithms that have tunable parameters (such as the dimension of some underlying vector space, or the number of equations and variables), the submission document shall specify concrete values for these parameters and the submitter is required to provide an analysis of how the security and performance of the algorithms depend on these parameters. If possible, the submission should specify several parameter sets that allow the selection of a range of possible security/performance tradeoffs.

Thank you.

Best regards,

Seidl Jan
Deloitte
>>>

Comment on Post-Quantum Cryptography Requirements and Evaluation Criteria

September 16, 2016

Thank you for the opportunity to provide feedback on the upcoming NIST Post-Quantum Cryptography project.

The draft submission requirements set out a clear plan for a transparent process that will lead to the identification of one or more post-quantum technologies with confidence in the result. Please find below comments on six aspects of the submission requirements which I believe will further improve the process.

1) Security levels and refinement

The draft submission requirements ask for parameter sets at five target security levels, the lowest 3 of which are at 64, 80, and 96 bits of quantum security. Reducing the number of target security levels will simplify submissions and allow cryptanalytic research to focus on fewer parameters. I suggest omitting security levels below 128 bits of quantum security.

Given that some submissions will be based on mathematical problems for which cryptanalysis continues to advance, it seems plausible that security levels of parameter sets may evolve, either favourably or unfavourably, during the evaluation process. It would be unfortunate if promising submissions were disqualified because of cryptanalytic advances shaved e.g. 10 bits of security off of a 128-bit-level submission. I suggest that NIST aim to include some room for refinement of parameters in these scenarios, possibly (a) between the first and the second round, or (b) during a round with a minor update, or (c) by incorporating an even higher security level (say, 192-bit quantum security) to provide breathing room.

2) Royalty-free

NIST should require submitters to meet the same intellectual property requirements in the previous SHA-3 competition, namely that if their submission is selected then the submitters agree to place no restrictions on use of the algorithm, making it available on a worldwide, non-exclusive, royalty-free basis.

3) Key establishment / KEM security

Early text from NIST asked which model to use for key establishment; the current submission requirements say IND-CCA security. I support this security requirement. Among other reasons, it is compatible with the notion used in security proofs of TLS [Jager et al. CRYPTO 2012, Krawczyk et al. CRYPTO 2013].

4) Application-level performance

NIST should identify a variety of application scenarios and evaluate submissions in these scenarios. Given that many post-quantum schemes will involve trade-offs between runtime, memory, and communication, evaluating these schemes in applications may provide surprising results compared to standalone evaluation. TLS should certainly be one of these application scenarios. Given the complexity of adding new algorithms to TLS implementations and fairly benchmarking such a system under realistic loads, NIST may want to apply this evaluation only to second round candidates, and NIST may also want to provide a standard interface and code for integration, rather than requiring submitters to each implement this themselves. For example, it should be possible to modify OpenSSL or another open source TLS implementation to include a ciphersuite that calls in to the NIST-specified PQC API.

5) Transition from traditional to post-quantum algorithms

In the years following this NIST process, standards and implementers will likely gradually transition to post-quantum cryptography, running traditional and post-quantum algorithms side-by-side. While the NIST process rightly focuses on evaluating the security and practicality of post-quantum algorithms, one aspect of practicality is enabling a smooth transition. NIST may want to include as a positive evaluation criteria any characteristics of the scheme which enable a smoother transition, for example a post-quantum scheme that can be easily run in a hybrid mode alongside a tradition algorithm and (safely) share some parameters.

6) Key exchange API

The current C API for key exchange has four functions:

    crypto_keyestablishment_initiator_generate
    crypto_keyestablishment_responder_generate
    crypto_keyestablishment_initiator_recover
    crypto_keyestablishment_responder_recover

The responder_generate function outputs a responder public key (ker) and a responder private key (skr), and then the responder_recover function outputs the shared secret (ss). This makes sense for plain Diffie-Hellman protocols, but may not make as much sense for some other protocols. KEMs generically are separated into 3 algorithms:

KeyGen, Encaps, Decaps; the Encaps algorithm might be easily separable into responder_generate and responder_recover, but not necessarily.

(For example, in the Peikert [PQCrypto 14], BCNS [IEEE S&P 15], and NewHope [USENIX 2016] ring-LWE key exchange protocols, the responder's secret key is ostensibly s'; in responder_generate, the responder would compute the outgoing message and reconciliation data (u and r in NewHope); in responder_recover, the responder would need to recompute all of these values in order to then derive the shared secret (mu). Of course an implementation could use the skr output of responder_generate to store the shared secret and then just use responder_recover to output that, but this seems to go against the spirit of the API.)

I recommend revising the API for key exchange to have three algorithms: initiator_generate, responder, initiator_recover.

Thank for your consideration.

Sincerely,

Dr Douglas Stebila
Assistant Professor
Department of Computing and Software, Faculty of Engineering
McMaster University
Hamilton, Ontario, Canada

>>

Dear Madam/Sir,

I have a few of comments on the document "Proposed Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process".

In Sections 4.A.2 and 4.A.3, you set the number of decryption (resp. signature) queries, that an attacker against a proposed encryption (resp. signature) scheme can make, to at most 2^64.
I find this very low compared to the targets of security mentioned in Section 4.A.4. What is the rationale for not letting the adversary make essentially as many queries as the target security?

I am a bit confused by Section 4.A.4. Clearly, the classical and quantum bit security of a given scheme can differ. But why are the ratios 1/2 and 2/3 put forward as targets? This seems driven by search and collision-search, but these algorithms may not be so relevant for the schemes that will be proposed.
We could very well imagine that for some proposed schemes, the ratio will be 1, and for

others it will be 1/10. As the focus is on quantum security, it may be tempting to focus on quantum bit-security targets, possibly with an additional requirement of not getting below a certain (and higher) classical bit security.

Best regards,
Damien Stehlé
Ecole Normale Superieure de Lyon