From: Danny Niu <dannyniu@hotmail.com>
Sent: Sunday, October 4, 2020 1:12 AM
To: pqc-comments
Cc: pqc-forum
Subject: ROUND 3 OFFICIAL COMMENT: CRYSTALS-DILITHIUM

Hello Crystals team.

I was wondering if it's possible to use smaller parameter sets
to decrease the signature size, instead of relying on aggressive
compression techniques used in the current form of Dilithium.

Specifically, if we were to preserve the Z_q[X]/(X^256+1) module
and matrix row and column counts in the current specification, and
decrease 'q' and acceptance threshold of signature variable 'z', and
avoid public-key compression, would we be able to achieve comparable
bandwidth efficiency and security? Had the team done some calculation
on this?

Thanks.

From: Vadim Lyubashevsky <vadim.lyubash@gmail.com>
Sent: Monday, February 8, 2021 4:35 PM
To: pqc-comments
Cc: pqc-forum
Subject: ROUND 3 OFFICIAL COMMENT: CRYSTALS-DILITHIUM

Dear all,

Dustin and the NIST team pointed out to us on Friday that one of our internal hash functions, which converts the message to a digest (line 10 in Figure 4), is hashing to a 384-bit value. This was large enough for NIST level 3 security (which was our maximum security level in round 2), but would need to be 512 for NIST level 5 security. We forgot to make this change and we thank the NIST team for pointing it out. We have since updated our code and spec at

https://github.com/pq-crystals/dilithium
and
https://pq-crystals.org/dilithium/resources.shtml

Since this is an internal variable, it has no impact on the parameter sizes. The runtime differences are also negligible. Since this change does affect the KATs, we took it as an opportunity to make a few additional small changes that we were holding off on. To harmonize the randomness expansion function in the key generation and signing, we are now using SHAKE-256 with 512-bit secret seeds for both. We also reduced the output size of the public key hash (tr on line 7) to 256 bits. And we simplified the pseudo-code throughout the spec by merging the names of H and CRH to just H (since they both use SHAKE-256 with varying output sizes).

Best,

Vadim
(On behalf of the Dilithium team).

From: pqc-forum@list.nist.gov on behalf of Christopher J Peikert <cpeikert@alum.mit.edu>
Sent: Saturday, October 9, 2021 12:55 PM
To: pqc-forum
Subject: [pqc-forum] ROUND 3 OFFICIAL COMMENT: CRYSTALS-Dilithium

This comment conveys the simple observation that the finalist signature scheme Dilithium has the "strong binding" and "(message) binding" security properties, based (tightly) on the assumed collision resistance of the hash function H=SHAKE-256.

Such binding properties can be important for non-repudiation (see, e.g., [1,2,3]), and I suggest that NIST should consider them as part of its evaluation.

(It's possible that this has been noticed before; I searched through the Dilithium documents and other sources, and didn't find anything about it.)

Informally, "strong binding" says that a signature sigma uniquely determines the public key and message (if any) for which it is valid. More precisely, it should be infeasible for an adversary to generate

(pk, M, pk', M', sigma)

where Verify(pk, M, sigma) and Verify(pk', M', sigma) both accept, and (pk, M) != (pk', M'). Importantly, all of the components may be "maliciously generated," i.e., they need not be valid outputs of the scheme's algorithms.

Strong binding trivially implies "(message) binding," which is defined in the same way, but with the requirement that pk=pk' (and hence M != M'). This means that a valid signature uniquely determines the message for which it is valid, but not necessarily the public key.

The formal proof of strong binding for Dilithium (which will appear in a future work) is a straightforward exercise based on this observation: a valid signature contains the hash value c~ = H(mu, w'_1), where mu = H(H(pk), M) and w'_1 is computed efficiently from the public key and signature. Therefore, one can easily extract an H-collision from any successful break of strong binding.

More generally, it appears that any signature scheme where a valid signature contains a collision-resistant hash of the full public key and message will have strong binding. We leave the formalization of this to other work.

(In practice, it is important to hash the given *representations* of the public key and message, as opposed to any "deserialized" versions that may have non-unique but valid representations.)

Regarding the other signature finalists and alternates: I have not carefully checked whether they plausibly have (strong) binding or not, but it would be worthwhile to investigate. The same kind of proof does not work for Falcon, both due to the structure of the scheme's signatures, and because HashToPoint's input does not include the public key. Perhaps there is a different kind of argument, or an explicit counterexample.

Sincerely yours in cryptography,
Chris

[1] https://eprint.iacr.org/2020/1244
[2] https://eprint.iacr.org/2020/823

[3] https://link.springer.com/chapter/10.1007/BFb0034842

--

From: pqc-forum@list.nist.gov on behalf of Markku-Juhani O. Saarinen <mjos.crypto@gmail.com>
Sent: Saturday, October 9, 2021 4:58 PM
To: pqc-forum
Cc: cpei...@alum.mit.edu
Subject: [pqc-forum] Re: ROUND 3 OFFICIAL COMMENT: CRYSTALS-Dilithium

Hello!

I think Chris's comment is a good illustration of the reasons why system integrators should be very careful when they interpret or modify PQC algorithms. There are usually excellent -- but sometimes non-obvious -- reasons for why the proposals do certain things the way they do them. Detail is important.

For example, the security properties that Chris describes do not seem to hold with the simplified interpretation of Dilithium described in a recent ETSI document [1]. This is because (as far as I can see) the variant of Dilithium in [1] removes the public-key prefix binding from the hash function.

It is tempting to use Dilithium, Falcon, and SP 800-208 HBS (LMS/XMSS) without their respective hash prefixes in PQC/PKI integration -- as essentially traditional hash-and-sign algorithms. This is because the internal APIs and certificate handling logic of PKI implementations is often built around that paradigm. In addition to breaking the "strong binding" property, the simplification introduces additional security risks stemming from hash function collision attacks.

As Chris notes, strong binding (and the related collision attack countermeasure) is dependant on unambiguous public key serialization. There are proposals such as [2] that diverge from the public key serialization methods specified by the Dilithium design team, and which have been evaluated in the NIST process. The [2] proposal adds a sequence tag, and individual string length separators for the (rho,t1) components. The "internal" separator bytes break verification unless both signer and verifier just implement a weakened hash-and-sign mode (or if the signer always adds the same non-standard padding bytes, or if a re-serialization step is implemented for hash computation.) Note that the additional encoding is unnecessary for X.509 interoperability; in addition to OID, the entire public key can be encoded as a single OCTET STRING (as the standard encoding is unambiguous); this also results in a 10-byte saving in certificate length.

Cheers,
- Markku

[1] ETSI, "CYBER: Quantum-Safe Signatures" ETSI TR 103 616 V1.1.1 (September 2021.)
https://www.etsi.org/deliver/etsi_tr/103600_103699/103616/01.01.01_60/tr_103616v010101p.pdf
[2] van Vredendaal et al. "RFC Quantum Safe Key Identification and Serialisation V1." (Aug 2021.) *(Note: Despite the title, this is not an IETF RFC or even an Internet-Draft.)* https://docs.google.com/document/d/1MbSf7e9NIZ0XCEpJ9Kpdxe04Z5HlvvgOBTUX4uvM1i0/edit?usp=sharing

From: pqc-forum@list.nist.gov on behalf of Cas Cremers <cas.cremers@gmail.com>
Sent: Monday, October 11, 2021 10:58 AM
To: pqc-forum
Cc: mjos....@gmail.com; cpei...@alum.mit.edu
Subject: [pqc-forum] Re: ROUND 3 OFFICIAL COMMENT: CRYSTALS-Dilithium

Dear all,

we agree with Chris Peikert that the next generation of signature schemes should fulfill security notions beyond unforgeability.
Beyond UnForgeability Features (BUFF) of signature schemes (including those mentioned by Chris Peikert) were treated in our S&P paper this year [1], which I mentioned on this mailing list [2] (2020-12-08 1418 UTC+1), and we presented at the 3rd NIST PQC workshop [3, Session VIII, talk 2].

We used the names of some of my prior work [4] for the notions:
- *Exclusive Ownership* denotes that a signature is bound to a single public key. (This notion comes in a few flavors depending on how many keys are maliciously chosen.)
- *Message-Binding Signatures* denotes that a signature is bound to a single message.
- *Non Re-Signability* denotes that one cannot create a signature without knowledge of the message, even if given a signature of the message under a different key.

We have shown that
1) Regarding the NIST round 3 submissions, Dilithium and Picnic fulfill all notions as is (we are currently working on proofs for SPHINCS+). FALCON, Rainbow, and GeMSS do not fulfill all notions. See [1, Table 1] for an overview of our analysis.
2) Our generic transformation: Signing H(pk, m) and additionally adding H(pk,m) to a signature results in a scheme that achieves all three Beyond UnForgeability Features (BUFF) (when using a collision resistant hash function). Notably, the Falcon team is considering this. This mechanism, of course, can be used to secure all finalists.

Best regards,

Cas, Samed, Rune, Marc, and Christian

[1] https://eprint.iacr.org/2020/1525
[2] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/NJdNwrSC4hU/m/0KQ5v1N6AwAJ
[3] https://www.nist.gov/video/third-pqc-standardization-conference-session-viii-security-part-ii-and-candidate-updates
[4] https://eprint.iacr.org/2020/823

From: pqc-forum@list.nist.gov on behalf of Christopher J Peikert <cpeikert@alum.mit.edu>
Sent: Monday, October 11, 2021 11:13 AM
To: Cas Cremers
Cc: pqc-forum; mjos....@gmail.com
Subject: [pqc-forum] Re: ROUND 3 OFFICIAL COMMENT: CRYSTALS-Dilithium

Dear Cas and all, thank you for these references (which we will cite in our related work). I am happy to know about this systematic treatment of these issues.

One question: am I correct in concluding that "Exclusive Ownership" + "Message Binding" implies the "Strong Binding" property mentioned in my original message? (That is: breaking strong binding implies breaking one of the two former properties.)

Sincerely yours in cryptography,
Chris

From: pqc-forum@list.nist.gov on behalf of Cas Cremers <cas.cremers@gmail.com>
Sent: Monday, October 11, 2021 11:34 AM
To: Christopher J Peikert
Cc: pqc-forum; mjos....@gmail.com
Subject: [pqc-forum] Re: ROUND 3 OFFICIAL COMMENT: CRYSTALS-Dilithium

Dear Chris,

As you can check syntactically, SBS is exactly the conjunction of M-S-UEO (Malicious Strong Exclusive Ownership) and MBS as both defined in [1], i.e. a conjunction of previous properties and not strictly a new one.

[1] https://eprint.iacr.org/2020/823

Best,

Cas

From: pqc-forum@list.nist.gov on behalf of Christine Cloostermans <cvvrede@gmail.com>
Sent: Tuesday, October 12, 2021 2:51 AM
To: Markku-Juhani O. Saarinen
Cc: pqc-forum; cpei...@alum.mit.edu
Subject: Re: [pqc-forum] Re: ROUND 3 OFFICIAL COMMENT: CRYSTALS-Dilithium

Hi Markku,

Thank you for your feedback - this discussion is a dimension that is important, namely the interface between NIST's schemes and managing keys that those schemes require.

> There are proposals such as [2] that diverge from the public key serialization methods specified by the Dilithium design team, and which have been evaluated in the NIST process.

We would like to point out that this work is focused entirely on managing keys and not in any way on the schemes themselves. It has already found several issues with the specifications in a number of submitted schemes. This work is in preparation for a submission to IETF where a broader community can debate the content and has already received feedback from authors of most schemes.

There was no guidance during the NIST process on how keys should be managed - only an interface that required a serialized blob. Over the rounds serialization formats have changed and it is likely that this will be the case in the future. Interoperability between early implementations has not been easy for this reason. Add to this the fact that many legacy environments will not be able to manage even the smaller of the PQC key sets (Isogenies aside) unless they are stored/transported in a compressed format. For these reasons we would argue that binary blobs are not an appropriate way to manage keys and we need a discussion on how to do that in an agile but safe way. We hope that having that discussion early and at the right venue will prevent many of the mistakes that were made with ECC and help with the divergence between standards.

Your feedback is useful in the context of security considerations for key management.

Regards, on behalf of the team,

Christine

Op za 9 okt. 2021 om 22:57 schreef Markku-Juhani O. Saarinen <mjos.crypto@gmail.com>:

> Hello!
>
> I think Chris's comment is a good illustration of the reasons why system integrators should be very careful when they interpret or modify PQC algorithms. There are usually excellent -- but sometimes non-obvious -- reasons for why the proposals do certain things the way they do them. Detail is important.
>
> For example, the security properties that Chris describes do not seem to hold with the simplified interpretation of Dilithium described in a recent ETSI document [1]. This is because (as far as I can see) the variant of Dilithium in [1] removes the public-key prefix binding from the hash function.
>
> It is tempting to use Dilithium, Falcon, and SP 800-208 HBS (LMS/XMSS) without their respective hash prefixes in PQC/PKI integration -- as essentially traditional hash-and-sign algorithms. This is because the internal APIs and certificate handling logic of PKI implementations is often built around that paradigm. In addition to breaking the "strong binding" property, the simplification introduces additional security risks stemming from hash function collision attacks.