From: Dan Collins <dcollinsn@gmail.com> via pqc-forum@list.nist.gov
To: pqc-forum <pqc-forum@list.nist.gov>
Subject: [pqc-forum] Layman's question about decryption failures under Kyber
Date: Saturday, July 16, 2022 10:23:55 PM ET

Greetings,

I have read some of the documents published by NIST and done some other research, in the hopes of understanding what special considerations engineers may need to understand while implementing systems which use the algorithms that were recently announced for standardization.

One item which confused me was the concept of decryption failures, for example under Kyber. I understand that for some mathematical reasons, for certain very rare values, the generated encryption key does not survive the round-trip of Encaps(Decaps(key)). However, in the Kyber paper (https://ieeexplore.ieee.org/document/8406610/ Algorithm 5), I see that in this case, some different key is returned based on a random and otherwise unused value "z" unique to the secret key.

Is this strictly done for constant-time reasons, or does this actually return some usable key? If so, how does the other party get the key H(z,H(C)) given that "z" seems to be part of the secret key?

If not, is it simply the implementer's responsibility to determine that the decrypted data is nonsense, and to restart the key exchange process?

(I realize that it may be premature to ask for implementation guidance when standards don't even exist yet, but I have engineers asking me what they may need to plan for - larger ciphertext sizes, for instance - and I'd like to start getting a clearer understanding of these algorithms.)

Thank you,

Dan Collins

From: Markku-Juhani O. Saarinen <mjos.crypto@gmail.com> via pqc-forum@list.nist.gov
To: Dan Collins <dcollinsn@gmail.com>
CC: pqc-forum <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Layman's question about decryption failures under Kyber
Date: Saturday, July 16, 2022 11:59:46 PM ET

On Sun, Jul 17, 2022 at 4:23 AM Dan Collins <dcollinsn@gmail.com> wrote:

> One item which confused me was the concept of decryption failures, for example under Kyber. I understand that for some mathematical reasons, for certain very rare values, the generated encryption key does not survive the round-trip of Encaps(Decaps(key)). However, in the Kyber paper (https://ieeexplore.ieee.org/document/8406610/ Algorithm 5), I see that in this case, some different key is returned based on a random and otherwise unused value "z" unique to the secret key.

Hi Dan,

This is higher-level structure is the Fujisaki-Okamoto transform; abbreviated "FO" below.

A sidenote: Even though that 2018 paper may be an appropriate reference in a general academic context, it no longer works as an implementation or security analysis reference. Many key details have changed that would totally break any implementation; for example, the prime field is q=7681 in that paper, and the latest v3.02 version has q=3329. I certainly hope they don't change that for the "final" tweak pre-standard version due October 1 (the q value is fixed in hardware, we have tape-outs, etc)! 😅

> Is this strictly done for constant-time reasons, or does this actually return some usable key? If so, how does the other party get the key H(z,H(C)) given that "z" seems to be part of the secret key?

The reasons are in the NIST requirement for CCA requirement for KEMs. H(z,H(C)) is indeed not a usable key but a throw-away value that is designed to be deterministic for a given ciphertext and recipient (that secret key component z.)

As for constant-time, Implementations avoid creating a timing oracle that would reveal a decryption failure. But the implications of the CCA requirement and FO for broader side-channel securite (DPA, DEMA) implementations are much more severe. Secure masked implementation of the re-encryption step and the resulting comparison is extremely tricky, especially in the case of Kyber (with its rounding and ciphertext compression transforms.)

Overall, NIST should again consider allowing relaxation of the CCA requirement for ephemeral key establishment applications, as it creates a significant latency and implementation overhead for the private key operation, and is arguably not needed in some prominent use cases.

> If not, is it simply the implementer's responsibility to determine that the decrypted data is nonsense, and to restart the key exchange process?

It seems that way. However, if an implementor wants to have a NIST-compliant module, it would be unwise to modify the Fujisaki-Okamoto transform as that is an integral part of the algorithm specification itself. Kyber with a different output transformation would no longer be Kyber.

(Even though failure conditions are not covered by current KATs, they absolutely should be. The current test vectors don't cover the correct implementation of FO or the existence of "z" at all. So future FIPS compliance / ACVP will need this.)

If a CCA KEM is used for key establishment the decryption failure will of course eventually lead to an error condition. In the context of TLS, IPSec, and SSH, the shared secrets will get passed on to a key derivation function, and ultimately at least the message authentication (MAC or AEAD) for transmitted payload data will fail. So the decryption failure is just pushed down to other components of the protocol.

> (I realize that it may be premature to ask for implementation guidance when standards don't even exist yet, but I have engineers asking me what they may need to plan for - larger ciphertext sizes, for instance - and I'd like to start getting a clearer understanding of these algorithms.)

Even though the subject matter has been discussed in length previously, I think it's important to provide feedback to NIST on this particular subject as the universal CCA (FO) requirement has an oversized impact on implementations. Clearly, performance in the ephemeral use case must be a consideration for NIST.

Cheers,

- markku

Dr. Markku-Juhani O. Saarinen <mjos@iki.fi>