From: Bassham, Lawrence E (Fed)
To: Moody, Dustin (Fed)
Subject: Re: [Pqc-forum] API for PQC algorithms
Date: Monday, October 31, 2016 2:22:21 PM

Ok. Thanks

On 10/31/16, 2:18 PM, "Moody, Dustin (Fed)" <dustin.moody@nist.gov> wrote:

Larry,
I'll subscribe you so you see all the future messages. I don't think we need to respond right away (nor do I think we even have to respond). Just wanted you to be aware.

Dustin

-----Original Message-----
From: Bassham, Lawrence E (Fed)
Sent: Monday, October 31, 2016 2:15 PM
To: Moody, Dustin (Fed) <dustin.moody@nist.gov>
Subject: Re: [Pqc-forum] API for PQC algorithms

Thanks. Looks like I need to get subscribed. I don't think I have a problem with the first batch of comments about the order of the arguments and the naming of the KEM functions. We can discuss the random input portion. You aren't looking for an update really soon are you? We can wait the couple of weeks or whatever you were thinking for this brief comment period. See if anything else comes in.

Larry

On 10/31/16, 2:06 PM, "Moody, Dustin (Fed)" <dustin.moody@nist.gov> wrote:

In case you aren't subscribed to the pqc-forum, here are some comments by Dan about the API

-----Original Message-----
From: pqc-forum-bounces@nist.gov [mailto:pqc-forum-bounces@nist.gov] On Behalf Of D. J. Bernstein
Sent: Monday, October 31, 2016 4:57 AM
To: pqc-forum <pqc-forum@nist.gov>
Subject: Re: [Pqc-forum] API for PQC algorithms

Some suggestions for the syntax of the crypto_kem API:

* Key generation should be crypto_kem_keypair(pk,sk) by analogy to
the other functions in SUPERCOP generating public and secret keys:
crypto_encrypt_keypair, crypto_sign_keypair, and crypto_dh_keypair.

* Encryption should put ct,ss before pk: outputs are always before
inputs in SUPERCOP (and NaCl). Having ct before ss is good: it fits
the keypair pattern of public output before secret output.

* Decryption should similarly put ss before ct,sk. Again having ct
before sk is good, as in (e.g.) crypto_dh; more broadly, secret
keys as inputs are always at the end.

* It's common in the literature to abbreviate encapsulation (or
encapsulate) as encaps or encap or enc. The shortest version, enc,
has the virtue of reducing distraction for people who are thinking

"encrypt" and who aren't already steeped in the relatively obscure
"encapsulate" terminology. Similarly I'd suggest dec.

Regarding KATs (not just for crypto_kem), I'd suggest scrapping the extra API calls (and the RANDOMBYTES definition) and instead telling implementors to use randombytes() to generate randomness. This has at least four advantages:

* It's already supported and used by SUPERCOP for generating KATs
  (and also works in NaCl to read /dev/urandom).

* It avoids a profusion of extra function calls.

* It avoids having to pass an RNG argument around through all the
  subroutines that need (or might need) the RNG.

* The implementor doesn't have to bother calculating a maximum number
  of random bytes needed (or a maximum reasonable number for the
  common case that there's no limit).

I'm not aware of any advantages of having an extra RNG argument. It's sometimes useful to analyze a randomized algorithm _semantically_ as a deterministic algorithm applied to a separate string-of-random-bits input, and there is a fringe argument that any other _syntax_ for algorithms should be disallowed, but this argument is ignored by approximately 100% of algorithm designers and programmers.

---Dan