From: Moody, Dustin (Fed)
To: Perlner, Ray A. (Fed)
Subject: FW: API changes
Date: Tuesday, August 8, 2017 11:51:09 AM

I'm forwarding these notes from our last meeting to you. I think we've covered most of it (at least 1-4 already), but if you see anything that we already have decided and haven't let the forum know, we probably should. The KAT calls are from 6. Below. I think we're waiting on 5 still until Larry is ready....

From: Moody, Dustin (Fed)
Sent: Tuesday, August 01, 2017 10:09 AM
To: Bassham, Lawrence E (Fed) <lawrence.bassham@nist.gov>
Cc: internal-pqc <internal-pqc@nist.gov>
Subject: API changes

Larry,

I'm summarizing what we discussed at our last meeting regarding Dan's recommendations (Everyone - please check I got it right. See the text in red below where I wasn't sure). Can you change the API in accordance to this? Then we can let people know when we're ready.
Thanks,

Dustin

1. **Recommendation to NIST:** Adjust the API to clearly allow submissions to call crypto_hash_sha256.

Response: OK - we can do this.

2. **Recommendation to NIST:** Adjust the API to clearly allow submissions to call GMP.

Response: We require the reference implementation to be self-contained. The optimized version can call libraries. We will have some standard libraries (GMP, NTL?). We may even have a VM interface for people to check their implementations with us.

3. **Recommendation to NIST:** Make an explicit list of automatic conversions that NIST is guaranteed to apply.

Response: We've already dealt with this, by pointing to where the conversions are on our FAQ.

4. **Can the optimized software use vector instructions? Assembly?** Yes, of course. Most cryptographic speed records use vector instructions and/or assembly. **Recommendation to NIST:** Clearly state that this is allowed.

Response: Additional implementations can have this, but the optimized implementation which is required in the submission should not, because it needs to run on a wide variety of platforms.

5. Sometimes a simple reference implementation will have a loop of processing random numbers, say 1024 times calling randombytes(...,4) and processing the result. An optimized implementation might instead call randombytes(...,4096) and use vector instructions to process the results. Will the resulting bytes from randombytes be identical for known-answer tests? SUPERCOP now guarantees that the answer is yes. **Recommendation to NIST:** Make the same guarantee in the NIST API.

Response: Using the AES CTR DRBG and John's seed-expander will provide the same guarantee.

6.**Recommendation to NIST:** Change the NIST API to allow randombytes to be called as often as desired. Explicitly say that randombytes should be used whenever the specification calls for randomness. Rename keygenerate and encapsulate and decapsulate as keypair and enc and dec (but don't do this if you're still insisting on incompatibilities in the randomness semantics!). Scrap the extra KAT calls: they're a big step backwards from SUPERCOP's automatic known-answer tests.

Response: We can do away with the extra KAT calls, and change the names of the variables (in our API document, and our FAQ). As I understand, we allow randombytes to be called as often as desired, correct? And we are OK with saying that randombytes should be used whenever the specification calls for randomness, right?