From:	Bassham, Lawrence E (Fed)
To:	Bassham, Lawrence E. (Fed)
Subject:	FW: API
Date:	Monday, April 3, 2017 2:02:41 PM

From: "Bassham, Lawrence E (Fed)" <lawrence.bassham@nist.gov>
Date: Monday, April 3, 2017 at 12:39 PM
To: "Peralta, Rene (Fed)" <rene.peralta@nist.gov>, "Moody, Dustin (Fed)" <dustin.moody@nist.gov>, "Perlner, Ray (Fed)" <ray.perlner@nist.gov>, "Alperin-Sheriff, Jacob (Fed)" <jacob.alperin-sheriff@nist.gov>
Subject: Re: API

Rene,

I just question your "at this point in the process... [ballpark will do]." When will you want more concrete numbers? We won't be changing the API later.

As Ray points out, the one call to randombytes() can do everything. The problem/conflict is that *requires* one call to do both things (harvest entropy and call to DRBG to expand). For the timing testing we want and need distinct calls to happen. The test code has a built-in function that implements randombytes(). We (or at least I) want people to call Dan's randombytes() to get pseudoentropy (my word for it) and then call DRBG to expand as necessary. Apparently, there may be a wide variety of needs/requirements for that expansion so let the submitter determine what's necessary for their implementation. Dan's statement of "Do we want designers and implementers of post-quantum algorithms to be forced to worry about the details of how this is accomplished?" (referring to a call to randombytes() to obtain all randomness needed). Well, yes I do - at least to some extent. They must define the properties that randomness needs in their algorithm. I heard people on the forum saying that a full blown NIST-approved DRBG was overkill for what their algorithm needed and a much more rudimentary whitening type of function would suffice. If that's the case they can define it in their spec and submit that. I don't know how Dan's Salsa and ChaChaCha fits into this. Are these full blown DRBG's, but not NIST approved? Would we allow them?

So, we could skip the whole requirement regarding a DRBG in the proposals. Use a generic randombytes() to provide entropy/randomness for the algorithms. SUPERCOP has a built-in function for this. Just keep calling it every time you need randomness. In a production environment, you would replace it with a construct like Ray demonstrated. That all works fine. The problem with it is that you don't get accurate performance that reflects the time spent in a DRBG. From Dan's post:

> The lattisigns512 authors needed enough random bytes that they noticed
> the slowness of the /dev/urandom-based randombytes() implementation, so
> they replaced randombytes() with a faster implementation using Salsa20.
> But we certainly want this done centrally, providing fast randombytes()
> for all algorithms to use, rather than having each post-quantum designer
> forced to worry about how to do this safely.

Two problems with that observation:

1) The "providing fast randombytes() for all algorithms to use". Nice in theory, but as I pointed out some algorithms indicated that there may be drastically different requirements on the "randomness" (Laszlo Hars post). Whether Laszlo's post is theoretical or he had something more concrete in mind isn't clear.
2) Calling /dev/urandom is in essence getting entropy every time you ask for randomness. You do that once and then call a DRBG for additional bits as necessary. Should be much faster than their initial implementation without going to Salsa20 (again, don't know how we feel about that).

I understand that it is hard/impossible to define/select a DRBG that performs well across all platforms (built-in AES instructions, etc). So, if the PQ computation time will heavily outweigh the entropy expansion (DRBG) than I'm ok with the simpler construct. Let's discuss.

(Apologies for the stream of conscience. Just trying to get something before we discuss.)
Larry

From: "Peralta, Rene (Fed)" <rene.peralta@nist.gov>
Date: Monday, April 3, 2017 at 11:16 AM
To: "Moody, Dustin (Fed)" <dustin.moody@nist.gov>, "Perlner, Ray (Fed)" <ray.perlner@nist.gov>, "Bassham, Lawrence E (Fed)" <lawrence.bassham@nist.gov>
Cc: "Peralta, Rene (Fed)" <rene.peralta@nist.gov>
Subject: Re: API

Thanks Dustin,

I was worried that doing what Dan wants would not allow the production code
to access two separate functionalities:

1) get true entropy
2) get pseudo-randomness

But Ray just pointed out that one can eventually code randombytes(..)

like

```
randombytes(..)
{
  static boolean harvested_entropy;
  if (not harvested_entropy) harvest_entropy(..);
  harvested_entropy = true;
  call a DRBG;
}
```

So my main issue goes away.

There still is the issue that doing what Dan says will mask the true cost of calling a DRBG. But at this point in the process I am not too concerned with performance other than ballpark.

Rene.

****
Dr. Rene Peralta
Computer Security Division
NIST
(301) 975-8702
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20899-8930
****

From: Moody, Dustin (Fed)
Sent: Monday, April 3, 2017 10:37 AM
To: Peralta, Rene (Fed)
Subject: API

Rene,
Dan posted a response to our API over the weekend. We will meet to discuss at 2pm, if you are interested. If not – no worries.

Dustin