From: Kelsey, John M. (Fed)
To: Bassham, Lawrence E. (Fed)
Subject: Re: CTR_DRBG for PQC
Date: Monday, April 17, 2017 1:34:46 PM

If we implement the Instantiate function, then we need to provide Update with some provided_data from randombytes(). The other place we use Update is within the Generate function. Since we're not going to accept any additional input, we would in principle call it only once, at the end of a Generate request. There, the provided_data is set to a block of zero bits.

The Generate function is only allowed to generate up to 4096 bytes per call. So if you're implementing a smart buffered form of Generate, you need to take this into account.

Do you know if all that PQC algorithms at least know how many RNG calls they're going to need? If so, it would be really simple to just request enough bytes to meet them all (perhaps with two or three Generate requests).

--John

From: "Bassham, Lawrence E (Fed)" <lawrence.bassham@nist.gov>
Date: Monday, April 17, 2017 at 1:26 PM
To: "Kelsey, John M. (Fed)" <john.kelsey@nist.gov>
Subject: Re: CTR_DRBG for PQC

I get all that. My question is "what's the external/provided data?" For what we're doing, implementing a randombytes() style function, there is no "provided_data". Is it required?

From: "Kelsey, John M. (Fed)" <john.kelsey@nist.gov>
Date: Monday, April 17, 2017 at 1:20 PM
To: "Bassham, Lawrence E (Fed)" <lawrence.bassham@nist.gov>
Subject: Re: CTR_DRBG for PQC

Larry,

The idea with the Update function is that it takes your existing K and V, and updates it in a way that guarantees that:

a. If you start out secure, then even if the attacker gets to choose your provided_data, you end up secure.

b. If the provided data is secure (it has at least 128 bits of entropy), then even if the attacker completely knows K and V going in, you end up secure.

All the descriptions of the DRBGs in 90A suffer from a sort of programming problem: a DRBG is properly an ***object***, with attributes and methods.

CTR_DRBG has the following state:
    K
    V
    Reseed_counter

Update() is really an internal method that updates K and V from the input it's given.

Update() is defined in two different scenarios—the one we care about is one where we have a full-entropy input source. (We'll assume randombytes() is a full-entropy source for our purposes.) So we ask for 256 bits of randomness from randombytes(), and then we end up just generating 256 bits from K and V in counter mode, XORing in the external data, and then using the resulting 256 bits as the new K and V.

Does that answer your question?

--John

From: "Bassham, Lawrence E (Fed)" <lawrence.bassham@nist.gov>
Date: Monday, April 17, 2017 at 1:12 PM
To: "Kelsey, John M. (Fed)" <john.kelsey@nist.gov>
Subject: CTR_DRBG for PQC

John,

I'm implementing the CTR_DRBG we discussed. I have a question. From 800-90A it looks like I need to get bits from an entropy source to fill K and V (let's say 256-bits total (128-bit key and 128-bit V). Those are in "defines" so I can adjust those. Can I ignore the "provided_data" below? You don't have any notion of it in your pseudocode.

Larry

CTR_DRBG_Update:
  Input: bitstring (*provided_data*, *Key*, *V*).