

From: [Moody, Dustin \(Fed\)](#)
To: [Kelsey, John M. \(Fed\)](#); [Perlner, Ray A. \(Fed\)](#); [Liu, Yi-Kai \(Fed\)](#); [Alperin-Sheriff, Jacob \(Fed\)](#); [Bassham, Lawrence E. \(Fed\)](#)
Cc: [Chen, Lily \(Fed\)](#); [Liu, Yi-Kai \(Fed\)](#)
Subject: Update PQC forum
Date: Wednesday, April 19, 2017 10:05:43 AM

John,

Now that we've heard back from Dan, it would be good to provide an update on the pqc-forum about the randomness issues (unless there is still something that needs to get ironed out with Dan). The last we said on the forum is shown below:

Do you think you could write something we could post to let people know what we're planning on?

Thanks,

Dustin

The function `randombytes()` will be available to the submitters. This is a function from the SUPERCOP test environment and should be used to generate seed values for an algorithm. `Randombytes` should only be used to seed a NIST-approved DRBG.

For functional and timing tests a deterministic generator is used inside `randombytes()` to produce the seed values. If security testing is being done simply substitute calls to a true hardware RBG inside `randombytes()`.

Function prototype for `randombytes()` is:

```
// The xlen parameter is in bytes
void randombytes(unsigned char *x,unsigned long long xlen)
```

The following demonstrate the use of the KAT and non-KAT versions of the functions to generate a key pair for encryption:

```
int crypto_encrypt_keypair_KAT(
    unsigned char *pk,
    unsigned char *sk,
    const unsigned char *randomness
)

int crypto_encrypt_keypair(unsigned char *pk, unsigned char *sk)
{
    unsigned char pk[CRYPTO_PUBLICKEYBYTES];
    unsigned char sk[CRYPTO_SECRETKEYBYTES];
    unsigned char seed[CRYPTO_RANDOMBYTES];

    randombytes(seed, CRYPTO_RANDOMBYTES);
    crypto_encrypt_keypair_KAT(pk, sk, seed);
}
```