From: vadim1980@gmail.com on behalf of Vadim Lyubashevsky <vadim.lyubash@gmail.com>
Sent: Tuesday, January 02, 2018 11:34 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: CRYSTALS-DILITHIUM

Dear all,

We are very grateful to Peter Pessl for notifying us of an implementation error in a randomness generator of our NIST submission. The bug was in the function rej_gamma1m1 in poly.c and consisted of accidentally overwriting a variable prior to using it. This function is used for sampling the masking vector y (line 13 of Figure 4 in the supporting documentation), and the result of the bug was that the same randomness ended up being used for pairs of consecutive coefficients, whereas the specification demands that all the coefficients be independent.

This reuse of randomness can easily be exploited to recover the secret key and we thus emphasize that the software, in the state submitted to NIST, should not be used in any real application.

We fixed the bug in an updated version of the software, which is available from the CRYSTALS website at https://pq-crystals.org/dilithium/resources.shtml. On the site, we also re-packaged the NIST submission package to include the updated KAT vectors.

Sincerely,

The CRYSTALS Team

From: 4akolzinaolga@gmail.com
Sent: Monday, March 26, 2018 11:03 AM
To: pqc-forum
Cc: pqc-comments
Subject: [pqc-forum] Re: OFFICIAL COMMENT: CRYSTALS-DILITHIUM

Dear all!

Typically, the standards (such as X9.98, SHA - 3) use Converting Between Bit Strings and Right-Padded Octet Strings.
Why do you use Converting Between Bit Strings and Left-Padded Octet Strings?

24 bits instead of 23 bits usage for A matrix generation needs to generate extra 256 pseudo random bits for each matrix element. Is this for security reasons or for code simplicity?

Digital signature verification
It isn't clear, why send the public key in the field before hashing, and don't hash it directly?

I.e. operators:

```
for(i = 0; i < CRYPTO_PUBLICKEYBYTES; ++i)
    m[CRYPTO_BYTES - CRYPTO_PUBLICKEYBYTES + i] = pk[i];
```

and

```
shake256(m + CRYPTO_BYTES - CRHBYTES, CRHBYTES,
           m + CRYPTO_BYTES - CRYPTO_PUBLICKEYBYTES,
CRYPTO_PUBLICKEYBYTES);
```

replace with operators:

```
shake256(m + CRYPTO_BYTES - CRHBYTES, CRHBYTES,
           pk, CRYPTO_PUBLICKEYBYTES);
```

Thank you! Best regards,

I. Gorbenko, E. Kachko, M. Yesina, O. Akolzina

From: Gregor Seiler <gseiler@inf.ethz.ch>
Sent: Thursday, April 05, 2018 9:38 AM
To: 4akolzinaolga@gmail.com; pqc-forum
Cc: pqc-comments
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: CRYSTALS-DILITHIUM

Dear all,

On 26.03.2018 17:03, 4akolzinaolga@gmail.com wrote:
> Typically, the standards (such as X9.98, SHA - 3) use Converting
> Between Bit Strings and Right-Padded Octet Strings.
>
> Why do you use Converting Between Bit Strings and Left-Padded Octet Strings?

We don't entirely understand where you are referring to when you say we are converting between bit strings and "left-padded octet strings". We repeatedly pack vectors of unsigned b-bit integers into byte strings for various different b. But since the integer vectors always have 256 coefficients there is no padding needed. We store the bytes of the integers in little-endian order as most modern CPUs use this byte ordering. So if we would for example convert a single integer to a byte string then the most significant bits of the last byte would be zero.
See Section 5.2 in the specification for an explanation of our data formats.

> 24 bits instead of 23 bits usage for A matrix generation needs to
> generate extra 256 pseudo random bits for each matrix element. Is this
> for security reasons or for code simplicity?

Using 23 random bits per coefficient in the sampling of A would be much more difficult to implement and would also not save any randomness because one would still need 5 blocks of SHAKE-128 output per polynomial. Notice that we sample each polynomial in the matrix A from a separate output stream of SHAKE-128 corresponding to a distinct input. Although this means we throw away more randomness compared to sampling all polynomials from the same stream, it is necessary in order to take advantage of a vectorized SHAKE implementation, which we do in our optimized implementation which uses AVX2 instructions.

> Digital signature verification
>
> It isn’t clear, why send the public key in the field before hashing,
> and don’t hash it directly?
>
> I.e. operators:
>
> for(i = 0; i < CRYPTO_PUBLICKEYBYTES; ++i)
>
> m[CRYPTO_BYTES- CRYPTO_PUBLICKEYBYTES+ i] = pk[i];
>
> and
>
> shake256(m+ CRYPTO_BYTES- CRHBYTES, CRHBYTES,
>
> m+ CRYPTO_BYTES- CRYPTO_PUBLICKEYBYTES, CRYPTO_PUBLICKEYBYTES);
>
> replace with operators:
>

> shake256(m+ CRYPTO_BYTES- CRHBYTES, CRHBYTES,
>
> pk, CRYPTO_PUBLICKEYBYTES);

We could directly hash the public key as you suggest without first copying it to the message buffer, but it wouldn't make verification any faster. We might change this in a more optimized version of our AVX2 code together with omissions of other copying that make the reference code more readable.

Regards,
Gregor