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
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:

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

and

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

replace with operators:

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

Thank you! Best regards,

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