Dear all,

I would like to report a typo in our algorithm 4 on page 7, and a thank you to Vadim Lyubashevshy for pointing this out to us.

Line 7 of algorithm 4 should be: $(\bfu, \bfv) \gets (\bfu_0, \bfv_0) + (p\bfa\bff + \bfa\bff)$

Line 8 of algorithm 4 should be: $\|p\bfa\bff\| \leq B_s$, ...

In both cases the term $p$ is missing for $p\bfa\bff$.

This error occurs when we created the specification for submission and trying to combining two academic paper together to fit in one description.

We note that this was indeed a typo in the specification - both the implementation and the original paper captures the algorithm correctly.

Happy holidays!

Cheers,

Zhenfei Zhang
Dear pqNTRUSign submitters,

It appears that pqNTRUSign as specified is vulnerable to a chosen message attack. If an attacker obtains two signatures \((u_1, v_1)\) and \((u_2, v_2)\) for the same message with the same public key, then I believe \(((u_1-u_2)/p, (v_1-v_2)/p)\) will be a short lattice vector. It does not appear that either of the usual countermeasures for this sort of attack (randomizing the hash, or derandomizing the lattice sampling) are employed. If we missed something, please comment.

Ray Perlner
From: Perlner, Ray (Fed)  
Sent: Monday, March 12, 2018 5:22 PM  
To: pqc-comments  
Cc: pqc-forum@list.nist.gov  
Subject: RE: OFFICIAL COMMENT: pqNTRUSign

I should probably clarify. I do not know how to use the fact that two signatures can be queried which are equal (mod q) (mod p) in a concrete attack. Nonetheless, it is unclear how a simulator along the lines of the one described in section 3.3 of pqNTRUSign.pdf could efficiently produce a transcript containing such signatures. As such, it seems at least to be a problem with your proof of transcript security.

From: Perlner, Ray (Fed)  
Sent: Monday, March 12, 2018 1:42 PM  
To: pqc-comments <pqc-comments@nist.gov>  
Cc: pqc-forum@list.nist.gov  
Subject: OFFICIAL COMMENT: pqNTRUSign

Dear pqNTRUSign submitters,

It appears that pqNTRUSign as specified is vulnerable to a chosen message attack. If an attacker obtains two signatures (u1, v1) and (u2, v2) for the same message with the same public key, then I believe ((u1-u2)/p, (v1-v2)/p) will be a short lattice vector. It does not appear that either of the usual countermeasures for this sort of attack (randomizing the hash, or derandomizing the lattice sampling) are employed. If we missed something, please comment.

Ray Perlner
Hi Ray,

Thank you very much for the analysis. I think there are two points here.
1. Is it safe for the signer to sign a same message twice, and
2. How effective is this attack - a.k.a. how small is this vector compared to Gaussian heuristics.

Here are our thoughts:

We agree that the simulation does not take into accounts that a signer signs two identical message digests. The simulator we outlined in section 3.3 of the report cannot simulate a signature transcript that happens to congruent to a zero vector.

However, we argue that the reason to show that a signature scheme is simulatable is to demonstrate that one can produce a "signature" without the knowledge of the secret key.

The attack here is beyond the purpose of the simulation, in that the output of the attack, namely, \((u_1, v_1), (v_1, v_2)\), will no longer resemble a valid signature, since they will be way bigger than a valid signature with overwhelming probability.

This indeed brings us to the second point - how big is this vector \((u_1, v_2)/p, (v_1, v_2)/p\).

By construction, \(u_i\) is either a Gaussian with sigma \(\approx \sqrt{q}\); or is uniform between \((-q/2+B_s)/p\) and \((q/2-B_s)/p\) for some small bound \(B_s\).

\(v_i\) is uniform between \((-q/2+B_t)/p\) and \((q/2-B_t)/p\) for some small bound \(B_t\).

The \(l_2\) norm of the vector \((a_i, b_i)\) will be at least around \(\sqrt{Nq + Nq^2/p^2}\);

and that of \((a_1-a_2, b_1-b_2)\) will be even larger, and perhaps counter the gain of division by \(p\), with \(p = 2\) in the current setting.

\(|a_1-a_2, b_1-b_2|\) will be much greater than the Gaussian heuristic length \(\approx \sqrt{Nq/\pi}\) of the lattice. So it will not be a short vector within this lattice.

Having said above, it is true that an attacker is able to produce some information on the lattice that isn't directly derivable from the public key.

As you have pointed out, there is an easy remedy, i.e., to include a salt in the hash when computing the message digest to randomize the digest.

We did not include this salt since this will increase the signature size by 256 bits; and we do not see how this information may aid the attacker.

We are happy to revise the scheme to include this salt if requested and allowed.

An alternative solution is to make the signing algorithm deterministic so that we always get the same signature given a same message.

This is indeed the implementation we have submitted as per the requirement of KAT.

Please let us know your comments and questions!

The pqNTRUSign Team