From: zhenfei <zzhang@onboardsecurity.com>
Sent: Tuesday, December 26, 2017 8:25 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: pqNTRUSign
Attachments: signature.asc

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\bfg)

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

From: Perlner, Ray (Fed)
Sent: Monday, March 12, 2018 1:42 PM
To: pqc-comments
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

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

From: Zhenfei Zhang <zzhang@onboardsecurity.com>
Sent: Tuesday, March 13, 2018 9:13 AM
To: Perlner, Ray (Fed)
Cc: pqc-comments; pqc-forum@list.nist.gov; Cong Chen; William Whyte; Jeffrey Hoffstein
Subject: Re: [pqc-forum] RE: OFFICIAL COMMENT: pqNTRUSign

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, ((u1-u2), (v1-v2)), 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 ((u1-u2)/p, (v1-v2)/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 l2 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 \e} 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

From: Perlner, Ray (Fed)
Sent: Thursday, March 15, 2018 3:35 PM
To: Zhenfei Zhang
Cc: pqc-comments; pqc-forum@list.nist.gov; Cong Chen; William Whyte; Jeffrey Hoffstein
Subject: RE: [pqc-forum] RE: OFFICIAL COMMENT: pqNTRUSign

Thanks for the quick response.

I agree that it seems difficult to turn my observation into a forgery attack. Nonetheless, I have to disagree with your claim that the attack is "beyond the purpose of the simulation argument." The point of simulation arguments in the context of transcript security is to rule out the possibility that the adversary can learn anything about the lattice that couldn't be learned from the public key alone, which you seem to concede the adversary can do if the signer signs the same message digest more than once.

Regarding possible countermeasures – we are not accepting tweaks at this time. That said, if pqNTRUSign does advance to the point that we are considering tweaks, we would probably feel more comfortable with pqNTRUSign if it came with either an explicit specification of one of the standard countermeasures (randomized hashing / derandomized sampling) or a much stronger argument for transcript security in the presence of multiple signatures on the same digest.

From: Zhenfei Zhang [mailto:zzhang@onboardsecurity.com]
Sent: Tuesday, March 13, 2018 9:13 AM
To: Perlner, Ray (Fed) <ray.perlner@nist.gov>
Cc: pqc-comments <pqc-comments@nist.gov>; pqc-forum@list.nist.gov; Cong Chen <cchen@onboardsecurity.com>; William Whyte <wwhyte@onboardsecurity.com>; Jeffrey Hoffstein <jhoff@math.brown.edu>
Subject: Re: [pqc-forum] RE: OFFICIAL COMMENT: pqNTRUSign

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, ((u1-u2), (v1-v2)), 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 ((u1-u2)/p, (v1-v2)/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 l2 norm of the vector (a_i,b_i) will be at least around \sqrt{Nq + Nq^2/p^2};

From: Zhenfei Zhang <zzhang@onboardsecurity.com>
Sent: Saturday, March 17, 2018 10:11 AM
To: Perlner, Ray (Fed)
Cc: pqc-comments; pqc-forum@list.nist.gov; Cong Chen; William Whyte; Jeffrey Hoffstein
Subject: Re: [pqc-forum] RE: OFFICIAL COMMENT: pqNTRUSign

Hi Ray,

> Regarding possible countermeasures – we are not accepting tweaks at this time. That said, if pqNTRUSign does advance to the point that we are considering tweaks, we would probably feel more comfortable with pqNTRUSign if it came with either an explicit specification of one of the standard countermeasures (randomized hashing / derandomized sampling) or a much stronger argument for transcript security in the presence of multiple signatures on the same digest.

Thanks. Totally understood.
Although we would like to stress that the currently implementation is derandomized already - so the attack shall not work.

Zhenfei

On Thu, Mar 15, 2018 at 3:35 PM, Perlner, Ray (Fed) <ray.perlner@nist.gov> wrote:

> Thanks for the quick response.
>
> I agree that it seems difficult to turn my observation into a forgery attack. Nonetheless, I have to disagree with your claim that the attack is "beyond the purpose of the simulation argument." The point of simulation arguments in the context of transcript security is to rule out the possibility that the adversary can learn anything about the lattice that couldn't be learned from the public key alone, which you seem to concede the adversary can do if the signer signs the same message digest more than once.
>
> Regarding possible countermeasures – we are not accepting tweaks at this time. That said, if pqNTRUSign does advance to the point that we are considering tweaks, we would probably feel more comfortable with pqNTRUSign if it came with either an explicit specification of one of the standard countermeasures (randomized hashing / derandomized sampling) or a much stronger argument for transcript security in the presence of multiple signatures on the same digest.
>
> **From:** Zhenfei Zhang [mailto:zzhang@onboardsecurity.com]
> **Sent:** Tuesday, March 13, 2018 9:13 AM
> **To:** Perlner, Ray (Fed) <ray.perlner@nist.gov>
> **Cc:** pqc-comments <pqc-comments@nist.gov>; pqc-forum@list.nist.gov; Cong Chen <cchen@onboardsecurity.com>; William Whyte <wwhyte@onboardsecurity.com>; Jeffrey Hoffstein <jhoff@math.brown.edu>
> **Subject:** Re: [pqc-forum] RE: OFFICIAL COMMENT: pqNTRUSign
>
> Hi Ray,

From: Gregor Seiler <gseiler@inf.ethz.ch>
Sent: Saturday, March 17, 2018 6:44 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: pqNTRUSign

Dear Authors,

when looking at your reference code it seems to me that the norm check on af is missing in the rejection sampling during signing for the uniform case.

Also I have trouble understanding why you use q/4 - 25 as the bound on the signature r + af.

Following your comment from December 26 that af in the pseudocode for signing should be replaced by paf = 2af, I interpret the bound Bs = 98 in your parameter table as the bound on paf so that the bound on af is equal to 49. Then should the bound on u = u_p + pr + paf not be q/2 - 98 and hence the one on r + af not be equal to (q/2 - 98)/2 - 1 = q/4 - 50?

When adding a condition that rejects if ||af|| > 49 and changing the bound on the signature to q/4 - 50, I find experimentally that the average number of rejections for the 100 KATs goes up from 23 to 113.
Accordingly the signing time goes up to 344 ms on my laptop.

Best,
Gregor

From:	Perlner, Ray (Fed)
Sent:	Monday, March 19, 2018 3:16 PM
To:	Zhenfei Zhang
Cc:	pqc-comments; pqc-forum@list.nist.gov; Cong Chen; William Whyte; Jeffrey Hoffstein
Subject:	RE: [pqc-forum] RE: OFFICIAL COMMENT: pqNTRUSign

On examining your submitted reference implementation, it appears you are calling randombytes inside both your your Gaussian (DCG) and uniform (pol_unidrnd) sampling routines. As a point of clarification, for the purpose of security evaluation, we do not consider code that calls randombytes to be derandomized.

From our FAQ https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/faqs question 015:

"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().**"

From: Zhenfei Zhang [mailto:zzhang@onboardsecurity.com]
Sent: Saturday, March 17, 2018 10:11 AM
To: Perlner, Ray (Fed) <ray.perlner@nist.gov>
Cc: pqc-comments <pqc-comments@nist.gov>; pqc-forum@list.nist.gov; Cong Chen <cchen@onboardsecurity.com>; William Whyte <wwhyte@onboardsecurity.com>; Jeffrey Hoffstein <jhoff@math.brown.edu>
Subject: Re: [pqc-forum] RE: OFFICIAL COMMENT: pqNTRUSign

Hi Ray,

> Regarding possible countermeasures – we are not accepting tweaks at this time. That said, if pqNTRUSign does advance to the point that we are considering tweaks, we would probably feel more comfortable with pqNTRUSign if it came with either an explicit specification of one of the standard countermeasures (randomized hashing / derandomized sampling) or a much stronger argument for transcript security in the presence of multiple signatures on the same digest.

Thanks. Totally understood.
Although we would like to stress that the currently implementation is derandomized already - so the attack shall not work.

Zhenfei

On Thu, Mar 15, 2018 at 3:35 PM, Perlner, Ray (Fed) <ray.perlner@nist.gov> wrote:

> Thanks for the quick response.
>
> I agree that it seems difficult to turn my observation into a forgery attack. Nonetheless, I have to disagree with your claim that the attack is "beyond the purpose of the simulation argument." The point of simulation arguments in the context of transcript security is to rule out the possibility that the adversary can learn anything about the lattice that couldn't be learned from the public key alone, which you seem to concede the adversary can do if the signer signs the same message digest more than once.

From: Zhenfei Zhang <zzhang@onboardsecurity.com>
Sent: Monday, March 26, 2018 1:01 PM
To: Gregor Seiler
Cc: pqc-comments; pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: pqNTRUSign

Hi all,

Thanks Gregor for spotting the errors in our code and parameters.

> when looking at your reference code it seems to me that the norm check on af is missing in the rejection sampling during signing for the uniform case.

This is correct.

> Also I have trouble understanding why you use q/4 - 25 as the bound on the signature r + af.

> Following your comment from December 26 that af in the pseudocode for signing should be replaced by paf = 2af, I interpret the bound Bs = 98 in your parameter table as the bound on paf so that the bound on af is equal to 49. Then should the bound on u = u_p + pr + paf not be q/2 - 98 and hence the one on r + af not be equal to (q/2 - 98)/2 - 1 = q/4 - 50?

Yes the norm bound should be q/4 - 50 rather than q/4 - 25 as in the current param.c.

We will revise the code and the performance section in the report; and post it on our website:
https://www.onboardsecurity.com/nist-post-quantum-crypto-submission

Regards,
Zhenfei

On Sat, Mar 17, 2018 at 6:43 PM, Gregor Seiler <gseiler@inf.ethz.ch> wrote:

> Dear Authors,
>
> when looking at your reference code it seems to me that the norm check on af is missing in the rejection sampling during signing for the uniform case.
>
> Also I have trouble understanding why you use q/4 - 25 as the bound on the signature r + af.
>
> Following your comment from December 26 that af in the pseudocode for signing should be replaced by paf = 2af, I interpret the bound Bs = 98 in your parameter table as the bound on paf so that the bound on af is equal to 49. Then should the bound on u = u_p + pr + paf not be q/2 - 98 and hence the one on r + af not be equal to (q/2 - 98)/2 - 1 = q/4 - 50?
>
> When adding a condition that rejects if ||af|| > 49 and changing the bound on the signature to q/4 - 50, I find experimentally that the average number of rejections for the 100 KATs goes up from 23 to 113. Accordingly the signing time goes up to 344 ms on my laptop.
>
> Best,
> Gregor
>
>