From: EL HASSANE LAAJI <e.laaji@ump.ac.ma>
Sent: Saturday, May 25, 2019 8:34 PM
To: pqc-forum
Subject: [pqc-forum] ROUND 2 OFFICIAL COMMENT: NTRUEncrypt & NTRU

Hello NTRU team
Can you say me, why you didn't keep the NTRUencrypt-1024 release, is it because of speed performance or security performance.
Best regards.
--

From: John Schanck <jschanck@uwaterloo.ca>
Sent: Tuesday, May 28, 2019 4:31 PM
To: EL HASSANE LAAJI
Cc: pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: NTRUEncrypt & NTRU

Dear El Hassane Laaji,

* EL HASSANE LAAJI <e.laaji@ump.ac.ma> [2019-05-26 00:34:09 +0000]:
> Can you say me, why you didn't keep the NTRUencrypt-1024 release, is
> it because of speed performance or security performance.

Thanks for your question. To clarify for others, the "NTRUencrypt-1024"
parameter set was proposed in the first round NTRUEncrypt submission for use with the ss-ntru-pke and ss-ntru-kem schemes. I'll split your question into two parts:
 - Why didn't we recommend ss-ntru?
 - Why didn't we recommend an NTRU variant that uses Z[x]/(x^1024 + 1)?

Regarding ss-ntru:

At a fixed security level, NTRU and LWE schemes have a trade-off triangle between
 1. the correctness of the decryption procedure,
 2. the width of the coefficient distributions,
 3. the compactness of public keys and ciphertexts.

The second round NTRU team wanted a compact scheme with a correct decryption procedure. The coefficient distribution used in ss-ntru is not compatible with that goal.

Regarding Z[x]/(x^1024 + 1):

It's not clear to us that there's a real need for an NTRU parameter set with such a large n. The largest n that we recommend is 821.

Best,
John (on behalf of the NTRU team)

From: EL HASSANE LAAJI <e.laaji@ump.ac.ma>
Sent: Tuesday, October 1, 2019 7:27 AM
To: pqc-forum; pqc-comments
Subject: NTRU comment

Hello NTRU team;
I have a remark:
In "NTRU Algorithm specification..." document ,page 17 DPE_Public_Key algorithm, in operation subsection you compute:

V0=Sq(G.f); // you compute V0 in Sq but in implementation you compute G.f in Rq.
the same for V1=Sq_inverse(V0);// you compute V1 in Sq but in implementation you compute it in Rq.
also hq=Rq(V1.f.f); you compute hq in Rq but in implementation you compute it Sq.
 see function:

```
void owcpa_keypair(unsigned char *pk,
               unsigned char *sk,
               const unsigned char seed[NTRU_SAMPLE_FG_BYTES])
{
   ....

poly_Rq_mul(Gf, G, f);//v0=f.G should be computed in Sq ?

  poly_Rq_inv(invGf, Gf);//v1=invGf should be computed in Sq ?

  poly_Rq_mul(tmp, invGf, f);

  poly_Sq_mul(invh, tmp, f);//hq=invh should be  also computed in Rq ?
....

}
```

Can you say me which are the right operations?

Best regards
H.LAAJI
Ph.D student
Mohammed First University Morocco.

Garanti sans virus. www.avast.com

From: John Schanck <jschanck@uwaterloo.ca>
Sent: Tuesday, October 1, 2019 5:10 PM
To: EL HASSANE LAAJI
Cc: pqc-forum; pqc-comments
Subject: Re: [pqc-forum] NTRU comment

* EL HASSANE LAAJI <e.laaji@ump.ac.ma> [2019-10-01 12:26:42 +0100]:
> Hello NTRU team;

Hello El Hassane,

>   poly_Rq_inv(invGf, Gf);//v1=invGf should be computed in Sq ?

> Can you say me which are the right operations?

The "poly_Rq" in the name just refers to the fact that Gf and invGf are encoded as elements of Rq. As long as Gf is not zero, our implementation of poly_Rq_inv will return invGf that satisfies invGf * Gf = 1 mod (q, Phi_n).

If Gf happens to be invertible in Rq, then our implementation of poly_Rq_inv will return invGf that satisfies invGf * Gf = 1 mod (q, Phi_1Phi_n).
This condition is not necessary, which is why the specification only asks for an inverse in Sq.

Cheers,
John

From:	EL HASSANE LAAJI <e.laaji@ump.ac.ma>
Sent:	Wednesday, October 9, 2019 1:11 PM
To:	pqc-forum; pqc-comments
Subject:	[pqc-forum] NTRU comments
Attachments:	NTRU4096821_owcpa.rar

Dear NTRU team;
I have some remarks:
1- In NTRU4096821 release, in file owcpa.c in owcpa_keypair(..) function .
I remark that the poly_S3_inv(invf_mod3, f); is very costly,
the speed of key generation function is about 215 ms(test performed in PC-I7 2Ghz, 8go).
I successfully reduce the cost to only 106 ms, by using another inverse function ntru_ring_inv(...);.
(you find attached to this message the files modified: owcpa.c ; poly.c; poly.h)

2- Another idea, why you do not choose F=3*f+1 to void computing inverse of f mod 3.
3- In my opinion, the use of S domain is unnecessary; why you do not use only R domain.
Best regards
--

From: John Schanck <jschanck@uwaterloo.ca>
Sent: Thursday, October 10, 2019 2:32 PM
To: EL HASSANE LAAJI
Cc: pqc-forum
Subject: Re: [pqc-forum] NTRU comments

* EL HASSANE LAAJI <e.laaji@ump.ac.ma> [2019-10-09 18:10:46 +0100]:
> Dear NTRU team;

Dear El Hassane,

Thank you for these comments.

> 1- In NTRU4096821 release, in file owcpa.c in owcpa_keypair(..) function .
> I remark that the poly_S3_inv(invf_mod3, f); is very costly,

This is true. But note that the reference implementation is not currently an optimization target for us, and we are unlikely to release an optimized plain c implementation in the near future. Our main focus is on platform specific optimizations.

That said, we plan on restructuring our supercop submission to make it easier for you and others to contribute optimized subroutines for benchmarking. The "factored" implementations of Streamlined NTRU Prime should give you a sense of what this will look like.

> 2- Another idea, why you do not choose F=3*f+1 to void computing
> inverse of f mod 3.

We discuss this in the submission document. See Sections 2.4.1 and 5.2.

Best,
John

PS. It looks like you are working on an old version of our source code.
I keep a public github repository up to date with our private development repository: https://github.com/jschanck/ntru Please feel free to get in touch off-list if you'd like to learn more about where your efforts may be most useful.

From: EL HASSANE LAAJI <e.laaji@ump.ac.ma>
Sent: Friday, October 11, 2019 2:49 PM
To: John Schanck
Cc: pqc-forum
Subject: [pqc-forum] Re: NTRU comments

I work on NTRU scheme that was submitted to the second round of the NIST's Post-Quantum Cryptography project in March 2019.
Is this really old release? What are the principal changes on NTRU4096821?

Le jeudi 10 octobre 2019, John Schanck <jschanck@uwaterloo.ca> a écrit :

> * EL HASSANE LAAJI <e.laaji@ump.ac.ma> [2019-10-09 18:10:46 +0100]:
> > Dear NTRU team;
>
> Dear El Hassane,
>
> Thank you for these comments.
>
> > 1- In NTRU4096821 release, in file owcpa.c in owcpa_keypair(..) function .
> > I remark that the poly_S3_inv(invf_mod3, f); is very costly,
>
> This is true. But note that the reference implementation is not currently
> an optimization target for us, and we are unlikely to release an optimized
> plain c implementation in the near future. Our main focus is on platform
> specific optimizations.
>
> That said, we plan on restructuring our supercop submission to make it
> easier for you and others to contribute optimized subroutines for
> benchmarking. The "factored" implementations of Streamlined NTRU Prime
> should give you a sense of what this will look like.
>
> > 2- Another idea, why you do not choose F=3*f+1 to void computing inverse
> > of f mod 3.
>
> We discuss this in the submission document. See Sections 2.4.1 and 5.2.
>
> Best,
> John
>
> PS. It looks like you are working on an old version of our source code.
> I keep a public github repository up to date with our private
> development repository: https://github.com/jschanck/ntru
> Please feel free to get in touch off-list if you'd like to learn more
> about where your efforts may be most useful.

From: John Schanck <jschanck@uwaterloo.ca>
Sent: Friday, October 11, 2019 3:16 PM
To: EL HASSANE LAAJI
Cc: pqc-forum
Subject: [pqc-forum] Re: NTRU comments

* EL HASSANE LAAJI <e.laaji@ump.ac.ma> [2019-10-11 19:49:02 +0100]:
> I work on NTRU scheme that was submitted to the second round of the
> NIST's Post-Quantum Cryptography project
> <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Subm
> issions>
> in
> March 2019.
> Is this really old release? What are the principal changes on NTRU4096821?

I announced new software on this list on August 4th, and I announced further improvements to the software at the Second PQC Standardization Conference.

You can see a detailed list of changes here:
https://github.com/jschanck/ntru/commits/master

Some commits that affect ntruhps4096821:
* Aug 26: "Factor inversion routines out of ref-common/poly.c"
* Aug 27: "Factor optimization targets out of poly.c"
* Aug 29: "Update avx2-hps4096821"
* Aug 30: "avx2-hps4096821: better poly_r2_mul"

Best,
John

From: D. J. Bernstein <djb@cr.yp.to>
Sent: Tuesday, December 8, 2020 11:12 PM
To: pqc-comments
Cc: pqc-forum
Subject: Re: [pqc-forum] Re: ROUND 2 OFFICIAL COMMENT: NTRU
Attachments: signature.asc

This message is specifically concerning the following incorrect claims from NIST IR 8309 regarding the round-2 submissions: NTRU "is not quite at the level of the highest-performing lattice schemes" and "has a small performance gap in comparison to KYBER and SABER".

In fact, NTRU is sometimes better in performance than Kyber and SABER, and sometimes worse. The claims do not say this; they are blanket claims that NTRU is "not quite at the level" of the others.

I request that NIST clearly admit for the record that these claims from NIST IR 8309 are false.

Regarding content, the claims are disproven by, e.g., the following numbers (quoted from https://cr.yp.to/papers.html#categories), which are also regarding the round-2 submissions:

* "Consider an application that requires Core-SVP to be at least 2^128. NTRU's ntruhps2048677 meets this requirement with 931-byte ciphertexts, while Kyber's kyber768 needs 1088-byte ciphertexts." (Also, in this situation SABER needs 1088-byte ciphertexts.)

* "An application limited to 1024-byte ciphertexts obtains a comfortable-sounding 2^145 Core-SVP from NTRU and only 2^111 Core-SVP from Kyber." (Also, in this situation SABER claimed only 2^125, never mind the subsequent correction saying only 2^118.)

Again, if NIST had instead written that NTRU outperforms Kyber and SABER in _some_ ways and not in _other_ ways, then the contradiction would have disappeared, but that isn't what NIST wrote. If NIST had expanded its comparison to include NTRU Prime (the first NIST IR 8309 quote above is ambiguous on this point but the second quote isn't), then the 2^128 would favor NTRU Prime but the 1024 would still favor NTRU, so the claims would still be wrong.

Regarding procedures, I already filed an OFFICIAL COMMENT in September to (inter alia) "dispute what NIST IR 8309 says regarding NTRU". In response, NIST took the following steps to dodge admitting error:

(1) Saying that it "did not do a detailed dive into all possible application scenarios".

    Is this supposed to be suggesting that it's hard to imagine applications "limited to 1024-byte ciphertexts"? That it's hard to imagine applications asking for "Core-SVP to be at least 2^128"? That it's okay for NIST to make false blanket claims if it hasn't taken enough time to see any of the (many) exceptions?

(2) Pointing to NTRU's "slower key generation".

Under a security requirement of (e.g.) Core-SVP >= 2^128, Kyber
and SABER outperform NTRU in key-generation time. However, under
the same security requirement, NTRU outperforms Kyber and SABER
in (e.g.) ciphertext size.

The dispute isn't about the existence of performance metrics
where Kyber and SABER outperform NTRU. The dispute is about
NIST's exaggerations.

(3) Saying that scenarios exist where NTRU "could" outperform Kyber
"or" SABER.

This is not the same as clearly admitting that NTRU _does_
outperform Kyber and SABER for, e.g., ciphertext size in
applications asking for Core-SVP to be at least 2^128. This is
also not the same as clearly admitting error in NIST IR 8309.

Readers who take the time to go through the details can see that NIST is wrong, but this isn't a substitute for a clear statement from NIST admitting that it's wrong. It's bad enough that these and other errors appeared in NIST IR 8309 in the first place; having the errors corrected for the record shouldn't require this much effort.

---Dan