From: ducas <L.Ducas@cwi.nl>
Sent: Thursday, July 30, 2020 9:49 AM
To: pqc-comments
Cc: pqc-forum
Subject: ROUND 3 OFFICIAL COMMENT: NTRU

Dear all,

following valuable comments and references from John Schanck, we have updated our report:

LWE with Side Information: Attacks and Concrete Security Estimation
Dana Dachman-Soled and Léo Ducas and Huijing Gong and Mélissa Rossi
https://eprint.iacr.org/2020/292

This updates now also considers the symmetries in the NTRU problem in Section 6.3, and discuss the (known) ways of exploiting it in a primal attack. In particular, we found that the technique of May and Silverman is in fact slightly counter-productive, if one accounts the accumulated probabilities of finding each rotation of the secret key.

The quantitative gain from this analysis remains low (e.g. from 379 bikz to 368 bikz for ntruhps2048509, improving the attack by about 3-4 bits).

This does *__not__* contradict the claims of the NTRU Specifications document, which only claimed 359 bikz because of conservative simplifications. We hope this clarify certain details of NTRU's cryptanalysis.

Best regards,

Dana, Léo, Huijing and Mélissa

From: Simone Dutto <simone.dutto@polito.it>
Sent: Friday, July 31, 2020 10:49 AM
To: pqc-comments
Cc: pqc-forum
Subject: ROUND 3 OFFICIAL COMMENT: NTRU
Attachments: fixed-sample.patch

Dear NTRU team,

while working with the C implementations found in the ZIP file attached to your submission (2nd round), we found a minor error in the code.
Specifically, in all files sample.c, line 86 should be

```
s[4*i+3] =  (u[15*i+11] & 0xfc)      + (u[15*i+12] << 8) + (u[15*i+13] << 16) + (u[15*i+14] << 24);
```

instead of

```
s[4*i+3] =  (u[15*i+11] & 0xfc)      + (u[15*i+12] << 8) + (u[15*i+13] << 15) + (u[15*i+14] << 24);
```

This is not a mandatory correction but, without it, the implemented sampling is not the one described in the documentation.

We saw that this correction is also necessary in the current version of the implementation at
https://github.com/jschanck/ntru/blob/master/ref-common/sample.c
Attached to this email there is the related github patch.

Hoping this comment will help, we thank you for your great work.

Best regards,
Simone Dutto

From: D. J. Bernstein <djb@cr.yp.to>
Sent: Friday, September 18, 2020 4:21 AM
To: pqc-comments
Cc: pqc-forum
Subject: ROUND 2 OFFICIAL COMMENT: NTRU
Attachments: signature.asc

I've posted a new paper "A discretization attack" identifying an NSA-exploitable weakness in some standardization processes:

https://cr.yp.to/papers.html#categories

The NISTPQC process has exactly this weakness. The paper uses NTRU vs.
Kyber as a case study, showing the results of hypothetical pro-NTRU and pro-Kyber discretization attacks. The paper also identifies claims in NIST IR 8309 regarding NTRU that match the results of a hypothetical pro-Kyber discretization attack and that do not match the facts. I am therefore filing this OFFICIAL COMMENT to

(1) dispute what NIST IR 8309 says regarding NTRU and

(2) request transparency regarding the NISTPQC process so that the public can see whether a discretization attack was carried out.

Full details appear in the paper.

---Dan

P.S. My question "What exactly has NSA told NIST regarding NISTPQC, regarding security levels or otherwise?" (email dated 2 Aug 2020
11:50:26 +0200) remains unanswered.

From: 'Moody, Dustin (Fed)' via pqc-forum <pqc-forum@list.nist.gov>
Sent: Monday, September 28, 2020 5:17 PM
To: D. J. Bernstein
Cc: pqc-forum
Subject: [pqc-forum] Re: ROUND 2 OFFICIAL COMMENT: NTRU

Dan,

In response to your two points in your official comment on NTRU.

1) You dispute what NISTIR 8309 says about NTRU

Reading your paper for further information, this seems to be disputing the sentences in our NTRU write-up that said "While NTRU is very efficient, it is not quite at the level of the highest-performing lattice schemes" and "NTRU has a small performance gap in comparison to KYBER and SABER". NTRU, Kyber, and SABER are all based on structured lattices and have very good performance. In our report, we wanted to highlight some of the differences between them. Thus the report noted: "In particular, NTRU has slower key generation than the schemes based on RLWE and MLWE." We did not do a detailed dive into all possible application scenarios. We agree that there are scenarios where NTRU could outperform Kyber or SABER. And note - we did select NTRU as a finalist along with Kyber and SABER.

2) You requested more transparency to know if NIST was subjected to a "discretization attack."

NIST certainly strives to run our PQC standardization process in an open and transparent way. We welcome suggestions to improve. However, we do not believe that a discretization attack took place, and don't feel we need to respond to claims of one. We looked at a variety of performance numbers when assessing the 2nd round candidates, and considered them from different viewpoints. When comparing similar candidates, the selections we make are always going to be subjective to some degree, and we understand not everybody will agree with them. These minor disagreements should not be interpreted as a failure of the NIST PQC process.

Dustin

From: 赵运磊 <ylzhao@fudan.edu.cn>
Sent: Thursday, May 12, 2022 5:34 AM
To: pqc-comments
Cc: pqc-forum
Subject: ROUND 3 OFFICIAL COMMENT: NTRU
Attachments: CTRU-comparisons.jpg

Dear NTRU team and dear all in PQC community:

Recently, we proposed compact NTRU-based KEM with scalable ciphertext compression, referred to CTRU. The paper is available from: https://arxiv.org/abs/2205.05413

CTRU has very concise ciphertext size, and to our knowledge it is the first NTRU-based KEM with a single polynomial ciphertext compression. Another interesting point is that it combines NTRU and RLWE, and in turn unifies RLWE and RLWR within the same scheme structure. Specifically, we also proposed compact NTRU based on RLWR, referred to CNTR that is just a simplified version of CTRU. The technique used in this work can also be applied to other NTRU variants. CTRU has remarkable performance, which is summarized in the attached table with details referred to the paper.

Any feedbacks and suggestions are appreciated from you.

All the best
Yunlei

From: D. J. Bernstein <djb@cr.yp.to>
Sent: Tuesday, August 30, 2022 6:09 AM
To: pqc-comments
Cc: pqc-forum
Subject: ROUND 3 OFFICIAL COMMENT: NTRU
Attachments: signature.asc

The NISTPQC evaluation criteria state the following:

> Schemes should ideally not fail catastrophically due to isolated
> coding errors, random number generator malfunctions, nonce reuse,
> keypair reuse (for ephemeral-only encryption/key establishment) etc.

I've posted a paper showing that the IND-CCA2 security claim for NTRU-HRSS fails catastrophically if a single bit happens to flip anywhere in the last 256 bits of NTRU-HRSS's stored secret key:

https://cr.yp.to/papers.html#ntrw

Most fault attacks require the attacker to induce faults. This attack does not. DRAM reliability figures from a study of Google's monitored, air-conditioned servers indicate that, if a billion 256-bit keys are stored in DRAM without SECDED, between 50000 and 140000 will have a bit flipped each year. Presumably typical user devices are less reliable.

The original version of NTRU-HRSS included plaintext confirmation, which blocks this attack. However, one of the changes that NTRU-HRSS made in its round-2 submission in 2019 was removing plaintext confirmation.

It is interesting to see _why_ NTRU-HRSS removed plaintext confirmation:
namely, the latest proofs did not need plaintext confirmation. Those proofs use an attack model too narrow to capture this attack. The attack does not contradict the proofs; it shows that the proofs are fragile in the presence of naturally occurring hardware failures.

This NTRU-HRSS attack appears to be within scope for NISTPQC: NIST's latest report

* says "NIST may consider selecting NTRU instead of Kyber" and
* mentions more obscure failure scenarios than DRAM bit flips.

The attack also raises questions regarding design techniques used in various other KEMs.

It's not plausible that _any_ scheme can be immune to "isolated coding errors" etc. It is, however, possible to reduce the impact of natural DRAM bit flips: software can encode secret keys and other data with SECDED.
https://pqsrc.cr.yp.to/downloads.html has a "libsecded" software library that takes roughly 1 Haswell cycle/byte for encoding and roughly
1 Haswell cycle/byte for decoding (with portable C software), so applications shouldn't notice any performance issues.

---D. J. Bernstein