From: D. J. Bernstein <djb@cr.yp.to>
To: pqc-comments@nist.gov
CC: pqc-forum@list.nist.gov
Subject: ROUND 3 OFFICIAL COMMENT: NTRU
Date: Tuesday, August 30, 2022 06:09:49 AM ET
Attachments: smime.p7m

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