NTRU

Cong Chen, Oussama Danba, Jeffrey Hoffstein,
Andreas Hülsing, Joost Rijneveld, Tsunekazu Saito,
John M. Schanck, Peter Schwabe, William Whyte,
Keita Xagawa, Takashi Yamakawa, Zhenfei Zhang

Third round update
June 9, 2021

No changes

- No changes to the scheme.
- No new parameter sets.

What might change?

- Revised security analysis.
- Maybe a small tweak to the KDF.

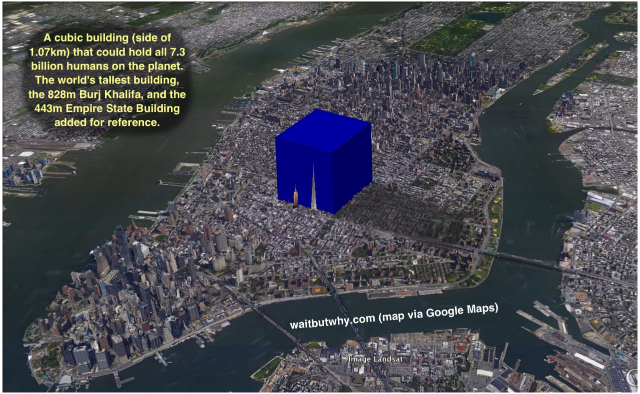
Security updates

- **Ducas** (EUROCRYPT 2018):
  "Dimensions for free".
- **Dachman-Soled–Ducas–Gong–Rossi** (CRYPTO 2020):
  More refined analysis of the blocksize needed in primal attack.
  New "leaky-LWE-estimator" scripts.
- **Albrecht–Gheorghiu–Postlethwaite–Schanck** (ASIACRYPT 2020):
  Heuristic non-asymptotic costs for one subroutine of sieving-based attacks.
- **May**, "How to meet ternary LWE keys" (CRYPTO 2021):
  Improved combinatorial attacks.
- **Nguyen**, "Boosting the hybrid attack on NTRU" (later today!).

"Beyond Core-SVP" estimate

Defined in Section 5.2 of Round 3 Kyber document.
Accounts for **dimensions-for-free** and **progressive sieving**.
Gives a circuit model **gate count**. Ignores data locality.

	ntruhrss701	ntruhps2048677
lattice attack dim. d	1328	1228
BKZ-blocksize	466	496
d_{4f}	41	43
sieving dimension	425	453
$\log_2$(gates)	**168**	**176**
$\log_2$(memory in bits)	**105**	**111**



Volume of $\approx 2^{52}$ microSD cards.

Assume **1 petabyte** per card, then 2^{52} cards hold roughly the amount of memory used in the 2^{168} operation attack on ntruhrss701.

Reaction attacks / Key reuse attacks

Background:

- **Hall–Goldberg–Schneier** (ICICS, 1999): Reaction attacks on McEliece, Hwang–Rao, Ajtai–Dwork, Goldreich–Goldwasser–Halevi. Suggestion that all "closest-point cryptosystems" systems are vulnerable.
- **Hoffstein–Silverman** (NTRU Tech Report 15, 1999): Reaction attacks on NTRU.
- **Jaulmes–Joux** (CRYPTO 2000): Reaction attacks on NTRU with a certain message padding mechanism.

Very well understood. Eliminated by CCA protections, but see fault injection literature.

Reaction attacks / Key reuse attacks

Recent papers, e.g. ePrint 2019/1022, ePrint 2021/168, ...

- Target a "CPA version" of NTRU that is claimed to be interesting for efficiency reasons.
- No data to support the claim that "CPA versions" are significantly more efficient.

Are “CPA versions” efficient?

- ntruhrss701 (CCA-secure)
  Decaps: **58219** cycles
- ntruhrss701 + {improper key} (CCA-secure)
  Decaps: **45580** cycles (-21.7%)
- ntruhrss701 + {improper key, explicit rejection} (CCA-secure)
  Decaps: **34101** cycles (-41.4%)
- ntruhrss701 + {improper key, explicit rejection, no message check} (CPA-secure)
  Decaps: **33891** cycles (-41.8%)

Dropping CCA protections does not significantly improve performance.

Symmetric primitive negotiation

It's annoying that "ntruhrss701" implies the use of SHA3-256 for key derivation.

- Not everyone likes this choice! Google/Cloudflare experiment used SHA256.
- Risk of incompatibility / proliferation of parameter sets. (Magnified because the other KEMs have the same problem.)
- The implicit rejection step is fragile and hard to test.

Symmetric primitive negotiation

Suggestion:

- Unify KDFs for the remaining KEMs.
- Avoid proliferation of parameter sets by defining

  $$\mathrm{PQKem}(\textsf{ntruhrss701}, \textsf{HKDF-SHA256})$$

  like DHKem from HPKE[1] (+ implicit rejection).
- Do this **now**, before picking a winner.

[1] Section 4.1 of https://www.ietf.org/archive/id/draft-irtf-cfrg-hpke-09.html

Alternative key derivation

Current construction:

1. (ok, msg) = Decaps(c, sk1)
2. k1 = H(msg)
3. k2 = H(sk2 ++ c)
4. output = k1 if ok else k2

Alternative:

1. (ok, msg) = Decaps(c, sk1)
2. ikm = msg `if` ok `else` sk2
3. k1 = KDF(ok ++ ikm, kem_context)

Alternative key derivation

Current construction:

1. (ok, msg) = Decaps(c, sk1)
2. k1 = H(msg)
3. k2 = H(sk2 ++ c)
4. output = k1 if ok else k2

Alternative:

1. (ok, msg) = Decaps(c, sk1)
2. ikm = msg `if` ok `else` sk2
3. k1 = KDF(ok ++ ikm, kem_context)

- Requires length('sk2') = length('msg').
- kem_context includes c.
- Implementation should follow Sec. 15 of Bernstein–Persichetti 2018.

Which parameter set should I use?

Use ntruhrss701 or ntruhps2048677.

- Default to ntruhrss701.
- Consider ntruhps2048677 if
  1. you understand the cost of constant-time sorting, and
  2. you need the 208 byte savings in key and ciphertext sizes $(1138 \to 930)$.

Why NTRU?

- The performance is excellent.
- There's no performance benefit to skipping CCA protections.
- The choice of symmetric primitives is limited to the choice of KDF.
- The size/security tradeoffs are very good.
- You don't have to think about decryption failures.
- You don't have to think about patents.

Supplemental material

“Beyond Core-SVP” estimate

	ntruhps2048509	ntruhps4096821
lattice attack dim. d	955	1526
BKZ-blocksize	364	615
d_{4f}	35	50
sieving dimension	329	565
$\log_2$(gates)	**139**	**209**
$\log_2$(memory in bits)	**84**	**134**