Lattice KEMs, the round-3 candidates: NTRU, NTRU Prime, SABER, Kyber, Frodo

Daniel J. Bernstein

University of Illinois at Chicago;
Ruhr University Bochum;
Academia Sinica

Widely used software already adding lattice options

2019.04: OpenSSH 8.0 adds `sntrup761` option.
Used if client and server configure it.

Widely used software already adding lattice options

2019.04: OpenSSH 8.0 adds `sntrup761` option.
Used if client and server configure it.

2019.07: Chrome adds `ntruhrss701` option.
Used for an experiment with Cloudflare servers.

Widely used software already adding lattice options

2019.04: OpenSSH 8.0 adds `sntrup761` option.
Used if client and server configure it.

2019.07: Chrome adds `ntruhrss701` option.
Used for an experiment with Cloudflare servers.

2021.05: OpenBSD adds `sntrup761` option for IPsec.
Used if client and server configure it.

Widely used software already adding lattice options

2019.04: OpenSSH 8.0 adds `sntrup761` option.
Used if client and server configure it.

2019.07: Chrome adds `ntruhrss701` option.
Used for an experiment with Cloudflare servers.

2021.05: OpenBSD adds `sntrup761` option for IPsec.
Used if client and server configure it.

2021.11: OpenSSH pre-8.9 enables `sntrup761` on server by default.
Used if client configures it.

Widely used software already adding lattice options

2019.04: OpenSSH 8.0 adds `sntrup761` option.
Used if client and server configure it.

2019.07: Chrome adds `ntruhrss701` option.
Used for an experiment with Cloudflare servers.

2021.05: OpenBSD adds `sntrup761` option for IPsec.
Used if client and server configure it.

2021.11: OpenSSH pre-8.9 enables `sntrup761` on server by default.
Used if client configures it.

These encryption layers are *added* to X25519 encryption (ECC).
If lattices are completely broken, still have pre-quantum security.

Why lattices attract attention, part 1: not too big

`sntrup761`: 2197 bytes total for key + ciphertext.
If application can cache key: 1039 bytes for ciphertext.

Why lattices attract attention, part 1: not too big

`sntrup761`: 2197 bytes total for key + ciphertext.
If application can cache key: 1039 bytes for ciphertext.

`ntruhrss701`: 2276 bytes total for key + ciphertext.
If application can cache key: 1138 bytes for ciphertext.

Why lattices attract attention, part 1: not too big

`sntrup761`: 2197 bytes total for key + ciphertext.
If application can cache key: 1039 bytes for ciphertext.

`ntruhrss701`: 2276 bytes total for key + ciphertext.
If application can cache key: 1138 bytes for ciphertext.

Compare to 64 bytes total for X25519 key + ciphertext.

Why lattices attract attention, part 1: not too big

`sntrup761`: 2197 bytes total for key + ciphertext.
If application can cache key: 1039 bytes for ciphertext.

`ntruhrss701`: 2276 bytes total for key + ciphertext.
If application can cache key: 1138 bytes for ciphertext.

Compare to 64 bytes total for X25519 key + ciphertext.
Also compare to 2 megabytes for typical web page.
How many applications will notice 2276 extra bytes?

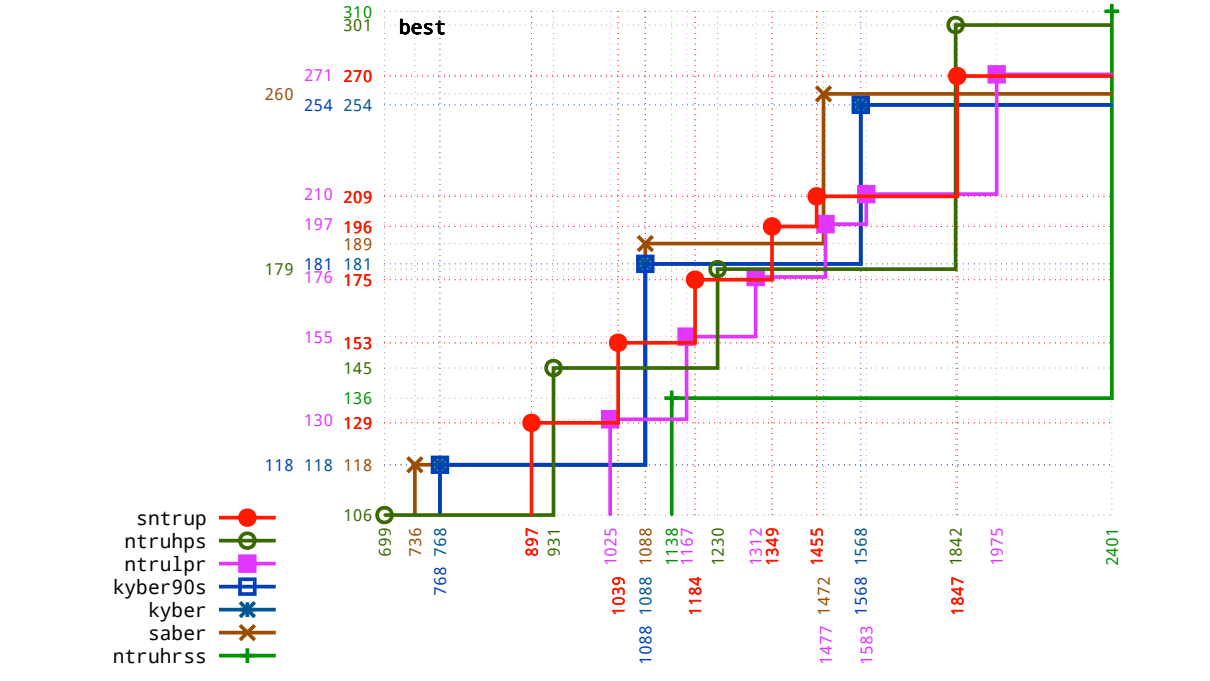
Why lattices attract attention, part 1: not too big

`sntrup761`: 2197 bytes total for key + ciphertext.
If application can cache key: 1039 bytes for ciphertext.

`ntruhrss701`: 2276 bytes total for key + ciphertext.
If application can cache key: 1138 bytes for ciphertext.

Compare to 64 bytes total for X25519 key + ciphertext.
Also compare to 2 megabytes for typical web page.
How many applications will notice 2276 extra bytes?

If larger sizes are acceptable, can increase security level.
Next slide: "Core-SVP" security estimate vs. ciphertext size.


best
sntrup
ntruhps
ntrulpr
kyber90s
kyber
saber
ntruhrss

Frodo is different from the other lattice KEMs

`frodo640`: 19336 bytes total for key + ciphertext.
If application can cache key: 9720 bytes for ciphertext.

Frodo is different from the other lattice KEMs

`frodo640`: 19336 bytes total for key + ciphertext.
If application can cache key: 9720 bytes for ciphertext.

Frodo avoids the security risks of structured lattices:

> *Given the unpredictable long-term outlook for algebraically structured lattices, and because any post-quantum standard should remain secure for decades into the future—including against new quantum attacks—we have based our proposal on the algebraically unstructured, plain LWE problem with conservative parameterizations . . .*

NTRU, NTRU Prime, SABER, Kyber use structured lattices.

Can we afford unstructured lattices?

2018 Google decided not to experiment with 10KB ciphertext:

. . . probably not preferable for real-time TLS connections.

Can we afford unstructured lattices?

2018 Google decided not to experiment with 10KB ciphertext:

> *. . . probably not preferable for real-time TLS connections.*

2020 NIST claimed that 10KB ciphertext is not "acceptable":

> *NIST's first priority for standardization is a KEM that would have acceptable performance in widely used applications overall. As such, possible standardization for FrodoKEM can likely wait until after the third round.*

Can we afford unstructured lattices?

2018 Google decided not to experiment with 10KB ciphertext:

> *. . . probably not preferable for real-time TLS connections.*

2020 NIST claimed that 10KB ciphertext is not "acceptable":

> *NIST's first priority for standardization is a KEM that would have acceptable performance in widely used applications overall. As such, possible standardization for FrodoKEM can likely wait until after the third round.*

If application *can* actually afford 10KB for ciphertext, then what's safer: `frodo640`, or scaled-up `ntruhrss4723`? This is a difficult question about managing attack risks.

Why lattices attract attention, part 2: fast

If size is the top priority, isogenies are better than lattices.
`sikep434`: 676 bytes total for key + ciphertext.
(If keys are cached: 128-byte `mceliece348864` ciphertext.)

But people complain about `sikep434` speed: 10265631 cycles dec
on 3GHz Skylake core ($3 \cdot 10^9$ cycles per second).
For comparison, X25519: 95437 cycles dec.

Why lattices attract attention, part 2: fast

If size is the top priority, isogenies are better than lattices.
`sikep434`: 676 bytes total for key + ciphertext.
(If keys are cached: 128-byte `mceliece348864` ciphertext.)

But people complain about `sikep434` speed: 10265631 cycles dec on 3GHz Skylake core ($3 \cdot 10^9$ cycles per second).

For comparison, X25519:	95437 cycles dec.
`saber2`:	86246 cycles dec.
`ntruhrss701`:	61530 cycles dec.
`sntrup761`:	48799 cycles dec.
`kyber768`:	43744 cycles dec.
`kyber90s768`:	24933 cycles dec.

Two different optimization goals

If goal is to minimize enc + dec time, best option is Quotient NTRU: original 1998 Hoffstein–Pipher–Silverman NTRU. Keygen: $G = e/a$. Enc: $B = Gb + d$. Dec: ...

If goal is to minimize keygen + enc + dec time, best option is Product NTRU: 2010 Lyubashevsky–Peikert–Regev (LPR). Keygen: $A = aG + e$. Enc: $B = Gb + d$, $C = M + Ab + c$. Dec: ...

Two different optimization goals

If goal is to minimize enc + dec time, best option is Quotient NTRU: original 1998 Hoffstein–Pipher–Silverman NTRU. Keygen: $G = e/a$. Enc: $B = Gb + d$. Dec: ...

If goal is to minimize keygen + enc + dec time, best option is Product NTRU: 2010 Lyubashevsky–Peikert–Regev (LPR). Keygen: $A = aG + e$. Enc: $B = Gb + d$, $C = M + Ab + c$. Dec: ...

NTRU's `ntruhrss` and `ntruhps` options:	Quotient NTRU.
NTRU Prime's `sntrup` option:	Quotient NTRU.
NTRU Prime's `ntrulpr` option:	Product NTRU.
SABER:	Product NTRU.
Kyber:	Product NTRU.

Kyber is different from the other lattice KEMs

Kyber is designed for speed on big Intel CPUs. In particular:

- Kyber uses "NTT-friendly rings".
- Kyber sends data "in the NTT domain".

Kyber is different from the other lattice KEMs

Kyber is designed for speed on big Intel CPUs. In particular:

- Kyber uses "NTT-friendly rings".
- Kyber sends data "in the NTT domain".

The same decisions penalize Kyber on smaller platforms.
Example: SABER hardware is more efficient than Kyber hardware.
Example: Kyber has trouble using existing RSA/ECC coprocessors.

Kyber is different from the other lattice KEMs

Kyber is designed for speed on big Intel CPUs. In particular:

- Kyber uses "NTT-friendly rings".
- Kyber sends data "in the NTT domain".

The same decisions penalize Kyber on smaller platforms.
Example: SABER hardware is more efficient than Kyber hardware.
Example: Kyber has trouble using existing RSA/ECC coprocessors.

Is speed a problem? If yes, are Intel CPUs the biggest problem?

Kyber is different from the other lattice KEMs

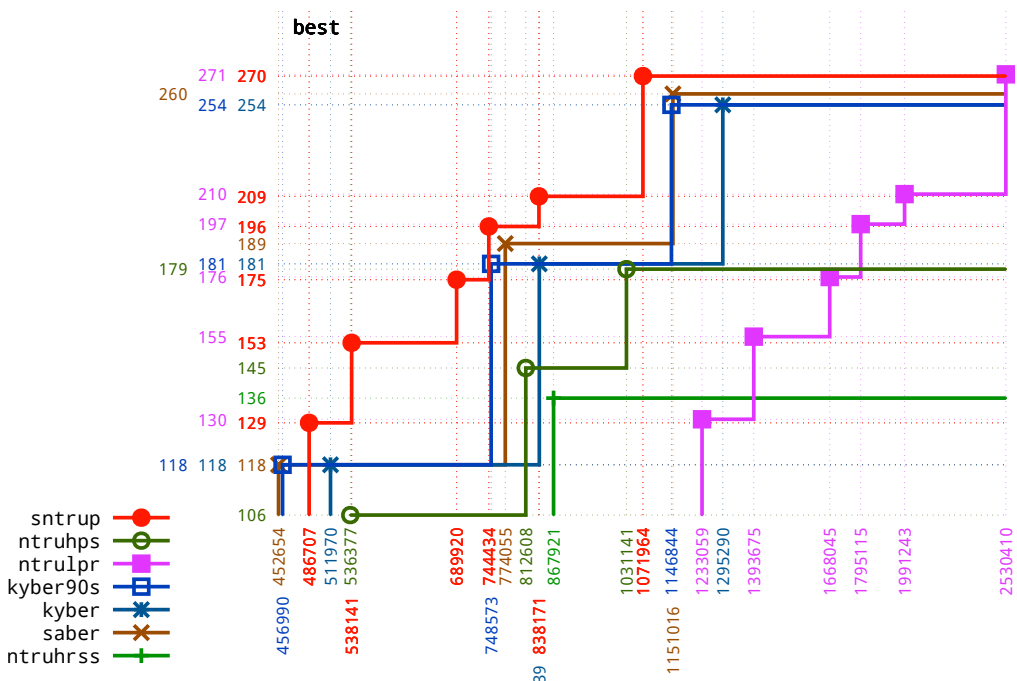
Kyber is designed for speed on big Intel CPUs. In particular:

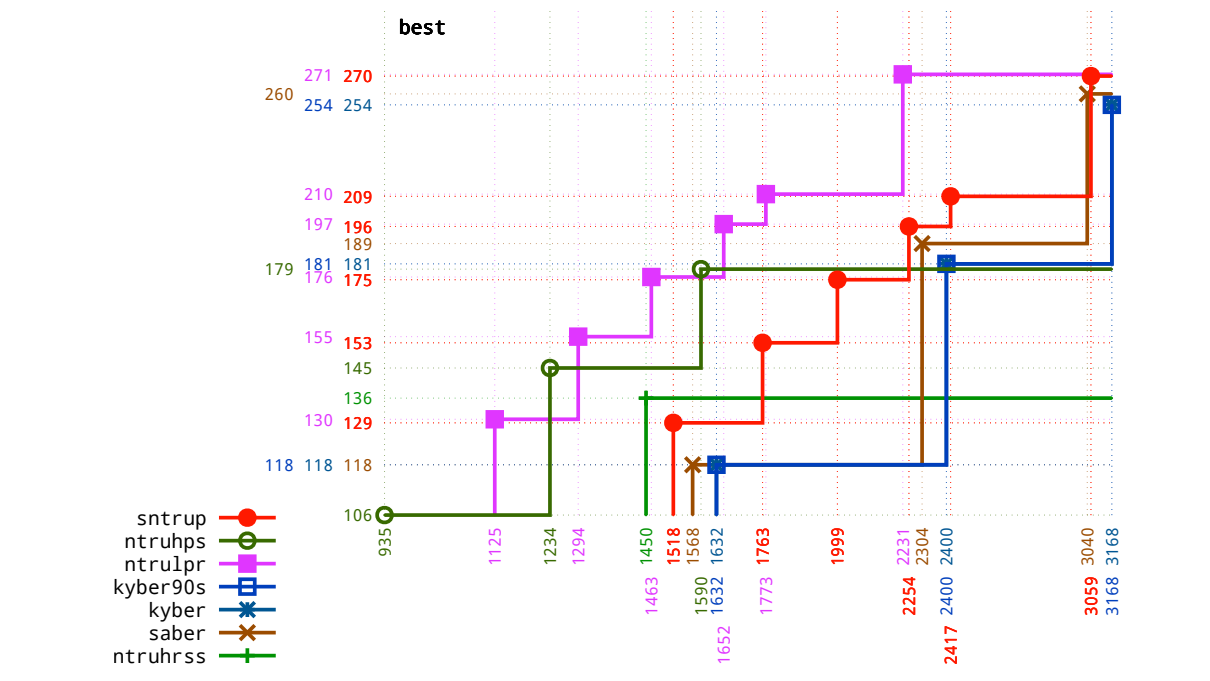
- Kyber uses "NTT-friendly rings".
- Kyber sends data "in the NTT domain".

The same decisions penalize Kyber on smaller platforms.
Example: SABER hardware is more efficient than Kyber hardware.
Example: Kyber has trouble using existing RSA/ECC coprocessors.

Is speed a problem? If yes, are Intel CPUs the biggest problem?

Lattice performance winner depends on the environment.
Next slide: "Core-SVP" vs. ARM Cortex-M4 dec cycles.
Slide after: "Core-SVP" vs. size of secret key.


best
271
270
260
254
254
210
209
197
196
189
179
181
181
176
175
155
153
145
136
130
129
118
118
118
106
sntrup
ntruhps
ntrulpr
kyber90s
kyber
saber
ntruhrss
452654
486707
511970
536377
689920
744434
774055
812608
867921
1031141
1071964
1146844
1233059
1295290
1393675
1668045
1795115
1991243
2530410
456990
538141
748573
838171
1151016


best
sntrup
ntruhps
ntrulpr
kyber90s
kyber
saber
ntruhrss
271
270
260
254
254
210
209
197
196
189
179
181
181
176
175
155
153
145
136
130
129
118
118
118
106
935
1125
1234
1294
1450
1463
1518
1568
1590
1632
1632
1652
1763
1773
1999
2231
2304
2254
2400
2400
2417
3040
3059
3168
3168

Patents

NIST has observed that royalty-free availability of cryptosystems and implementations has facilitated adoption of cryptographic standards in the past. For that reason, NIST believes it is critical that this process leads to cryptographic standards that can be freely implemented in security technologies and products.

Patents

> *NIST has observed that royalty-free availability of cryptosystems and implementations has facilitated adoption of cryptographic standards in the past. For that reason, NIST believes it is critical that this process leads to cryptographic standards that can be freely implemented in security technologies and products.*

NIST required submitters to disclose their patents.
We've also heard about further patents from non-submitters, and there are probably many more that have been kept quiet.
NIST discouraged public analysis of known patents. Why???
NIST tried to quietly buy out some patents—and failed.

Some interesting lattice patents

Original NTRU was patented. Patent expired in 2017.

Some interesting lattice patents

Original NTRU was patented. Patent expired in 2017.

U.S. patent 9094189 until 2032 threatens Product NTRU (LPR). Was filed before LPR was published. Kept quiet for many years. Litigation against this patent was filed in 2017 and gave up in 2021.

Some interesting lattice patents

Original NTRU was patented. Patent expired in 2017.

U.S. patent 9094189 until 2032 threatens Product NTRU (LPR). Was filed before LPR was published. Kept quiet for many years. Litigation against this patent was filed in 2017 and gave up in 2021.

U.S. patent 9246675 until 2033 threatens Product NTRU with compressed ciphertexts. Was filed before 2014 Peikert paper claimed LPR ciphertext compression as an "innovation". Apparently stopped Google's first post-quantum experiment, 2016.

Some interesting lattice patents

Original NTRU was patented. Patent expired in 2017.

U.S. patent 9094189 until 2032 threatens Product NTRU (LPR). Was filed before LPR was published. Kept quiet for many years. Litigation against this patent was filed in 2017 and gave up in 2021.

U.S. patent 9246675 until 2033 threatens Product NTRU with compressed ciphertexts. Was filed before 2014 Peikert paper claimed LPR ciphertext compression as an "innovation". Apparently stopped Google's first post-quantum experiment, 2016.

Ongoing arguments: "Non-applicability ... to Kyber and Saber"; but "doctrine of equivalents"; NIST's secret patent analysis; ...

Do we know how much lattice attacks cost?

Original view of the "Core-SVP" security estimate:
"conservative" lower bound on actual security level. However:

- For large key sizes, my 2020.05 analysis suggests that known attacks cost less than Core-SVP.
- For Kyber-512, Core-SVP was only 111 bits.

Do we know how much lattice attacks cost?

Original view of the "Core-SVP" security estimate: "conservative" lower bound on actual security level. However:

- For large key sizes, my 2020.05 analysis suggests that known attacks cost less than Core-SVP.
- For Kyber-512, Core-SVP was only 111 bits.

2020.10: Kyber changes parameters, changes security analysis. Estimates that Kyber-512 security is 8 bits above AES-128, but says we "need more research into this" and there are "foreseeable improvements"; says security could be 8 bits below AES-128.

NTRU, Kyber, SABER all include bleeding-edge parameters.

Do we know the best lattice attacks?

2018 Laarhoven–Mariano saved "between a factor 20 to 40 in the time complexity for SVP".

2018 Bai–Stehlé–Wen introduced a new variant of BKZ producing "bases of better quality" for the "same cost" of SVP.

2018 Aono–Nguyen–Shen adapted "recent quantum tree algorithms" to enumeration.

2018 D'Anvers–Vercauteren–Verbauwhede showed that "an attacker can significantly reduce the security of (Ring/Module)-LWE/LWR based schemes that have a relatively high failure rate" and that for LAC-128 "the failure rate is 2^{48} times bigger than estimated".

Do we know the best lattice attacks? part 2

2018 Hamburg pointed out that the first published "provably secure" Round5 design had disastrously high decryption-failure rate, 2^{-55}.

2019 Pellet-Mary–Hanrot–Stehlé broke through the previously claimed $\exp(n^{1/2+o(1)})$ approximation-factor "barrier" for number-theoretic attacks against Ideal-SVP.

2019 Guo–Johansson–Yang presented faster attacks against some systems that use error correction to (try to) reduce decryption failures. This paper violated the security claims of LAC.

2020 Bellare–Davis–Günther presented a fast break of Round2.

Do we know the best lattice attacks? part 3

2020 Dachman-Soled–Ducas–Gong–Rossi presented slightly faster attacks against the constant-sum secrets in LAC, NTRU, Round5.

2020 Doulgerakis–Laarhoven–de Weger presented "faster [sieving] methods" for SVP.

2020 Albrecht–Bai–Fouque–Kirchner–Stehlé–Wen reduced the exponent of enumeration from $\approx 0.187\beta \log_2 \beta$ to $\approx 0.125\beta \log_2 \beta$.

2020 Albrecht–Bai–Li–Rowell introduced a "practical and faster" enumeration algorithm "for reaching the same RHF in practical and cryptographic parameter ranges".

Do we know the best lattice attacks? part 4

2020 Bernard–Roux-Langlois improved the algorithm from 2019 Pellet-Mary–Hanrot–Stehlé and showed experimentally that in small dimensions the improved algorithm reaches much better approximation factors.

2021 Bi–Lu–Luo–Wang–Zhang introduced a hybrid dual attack that improves "the state-of-the-art cryptanalysis results by 2–14 bits, under the BKZ-core-SVP model".

2021 D'Anvers–Batsleer improved the "state-of-the-art multitarget failure boosting attacks", showing that "the quantum security of Saber can theoretically be reduced from 172 bits to 145 bits in specific circumstances".

Do we know the best lattice attacks? part 5

2021 May improved combinatorial attacks from exponent $0.5 + o(1)$ to exponent $0.25 + o(1)$ in the case of ternary keys; 2021 van Hoof–Kirshanova–May improved the exponents of quantum combinatorial attacks; 2021 Kirshanova–May improved the $o(1)$.

2021 Chailloux–Loyer improved quantum sieving exponents below the exponents previously claimed to be "optimal".

2021 Heiser introduced another quantum-sieving speedup that "affects the security of lattice-based encryption schemes, including NIST PQC Round 3 finalists".

2021 Guo–Johansson reduced Kyber security by several bits.

Highly unstable attack picture! What do we do?

For each KEM family: Use biggest keys you can afford.
Can also choose a KEM family to eliminate *some* attack avenues:

submission	NTRU		NTRU Prime		SABER	Kyber	Frodo
KEM family	`ntruhrss`	`ntruhps`	`sntrup`	`ntrulpr`	`saber`	`kyber`	`frodo`
lattices	risk	risk	risk	risk	risk	risk	risk
derandomization				risk	risk	risk	risk
decryption failures					risk	risk	risk
structured lattices	risk	risk	risk	risk	risk	risk	
cyclotomics	risk	risk			risk	risk	
reducibility	risk	risk			risk	risk	
quotients	risk	risk	risk				
extra samples				risk	risk	risk	risk
non-QROM FO	risk	risk	risk	risk	risk	risk	risk
non-QROM 2				risk	risk	risk	risk

NTRU Prime is different from the other lattice KEMs

Subject to the requirement of being a small lattice KEM,
NTRU Prime is the only submission systematically designed to
eliminate unnecessary complications in security review:
eliminate decryption failures, eliminate cyclotomics, etc.

NTRU Prime is different from the other lattice KEMs

Subject to the requirement of being a small lattice KEM, NTRU Prime is the only submission systematically designed to eliminate unnecessary complications in security review: eliminate decryption failures, eliminate cyclotomics, etc.

FOIA request revealed that NIST treats this as a negative feature:

- *Among the remaining, structured lattice schemes, our assessment was that cyclotomics (esp power-of-2 cyclotomics) are the clear "community standard"*
- *So, we moved Kyber, Saber, NTRU on as Finalists, but kept NTRUprime too*

NTRU Prime is different from the other lattice KEMs

Subject to the requirement of being a small lattice KEM, NTRU Prime is the only submission systematically designed to eliminate unnecessary complications in security review: eliminate decryption failures, eliminate cyclotomics, etc.

FOIA request revealed that NIST treats this as a negative feature:

- *Among the remaining, structured lattice schemes, our assessment was that cyclotomics (esp power-of-2 cyclotomics) are the clear "community standard"*
- *So, we moved Kyber, Saber, NTRU on as Finalists, but kept NTRUprime too*

Further risk-management analysis: See "Risks of lattice KEMs".