NTRU-HRSS-KEM

Andreas Hülsing[1], Joost Rijneveld[2], John Schanck[3], Peter Schwabe[2]

[1] Eindhoven University of Technology, The Netherlands
[2] Radboud University, Nijmegen, The Netherlands
[3] Institute for Quantum Computing, University of Waterloo, Canada

2018-04-13

NTRU (Hoffstein–Pipher–Silverman 1998)

Arithmetic is in $R \cong \mathbb{Z}[x]/(x^n - 1)$

NTRU (Hoffstein–Pipher–Silverman 1998)

Arithmetic is in $R = (\mathbb{Z}^n, +, \circledast)$, where $\circledast$ is cyclic convolution.

NTRU (Hoffstein–Pipher–Silverman 1998)

Arithmetic is in $R = (\mathbb{Z}^n, +, \circledast)$, where $\circledast$ is cyclic convolution.
Reduction modulo an integer t is into the interval $[-t/2, t/2)$.

NTRU (Hoffstein–Pipher–Silverman 1998)

Arithmetic is in $R = (\mathbb{Z}^n, +, \circledast)$, where $\circledast$ is cyclic convolution.
Reduction modulo an integer t is into the interval $[-t/2, t/2)$.

Parameters: $n, p, q \in \mathbb{Z}$ with $\gcd(p, q) = 1$ and $p \ll q$.
Sample spaces $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_r$, and $\mathcal{L}_m$ are sets of "short" elements of R.

NTRU (Hoffstein–Pipher–Silverman 1998)

Arithmetic is in $R = (\mathbb{Z}^n, +, \circledast)$, where $\circledast$ is cyclic convolution.
Reduction modulo an integer t is into the interval $[-t/2, t/2)$.

Parameters: $n, p, q \in \mathbb{Z}$ with $\gcd(p, q) = 1$ and $p \ll q$.
Sample spaces $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_r$, and $\mathcal{L}_m$ are sets of "short" elements of R.

For concreteness, think: n prime, $q = 2^{\lfloor \log n \rfloor + O(1)}$, and $p = 3$.
Sample spaces are subsets of $\{-1, 0, 1\}^n$.

NTRU (Hoffstein–Pipher–Silverman 1998)

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: (Try to) compute F_q such that $(f \circledast F_q) \bmod q = 1$.
3: (Try to) compute F_p such that $(f \circledast F_p) \bmod p = 1$.
4: If step 2 or step 3 fails, go to 1.
5: $h = (p \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

NTRU (Hoffstein–Pipher–Silverman 1998)

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: (Try to) compute F_q such that $(f \circledast F_q) \bmod q = 1$.
3: (Try to) compute F_p such that $(f \circledast F_p) \bmod p = 1$.
4: If step 2 or step 3 fails, go to 1.
5: $h = (p \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

Encryption

Input: Message $m \in \mathcal{L}_m$.
1: Sample r from $\mathcal{L}_r$.
2: $c = (r \circledast h + m) \bmod q$.

Output: Ciphertext c.

NTRU (Hoffstein–Pipher–Silverman 1998)

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: (Try to) compute F_q such that $(f \circledast F_q) \bmod q = 1$.
3: (Try to) compute F_p such that $(f \circledast F_p) \bmod p = 1$.
4: If step 2 or step 3 fails, go to 1.
5: $h = (p \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

Encryption

Input: Message $m \in \mathcal{L}_m$.
1: Sample r from $\mathcal{L}_r$.
2: $c = (r \circledast h + m) \bmod q$.

Output: Ciphertext c.

Decryption

Input: Ciphertext c.
1: $v = (c \circledast f) \bmod q$.
2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Why HPS98 decryption works

Decryption

Input: Ciphertext c.

1: $v = (c \circledast f) \bmod q$.

2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Crucial step is:

$$v = (c \circledast f) \bmod q$$

Why HPS98 decryption works

Decryption

Input: Ciphertext c.

1: $v = (c \circledast f) \bmod q$.

2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Recall:

- $c = (r \circledast h + m) \bmod q$.

Crucial step is:

$$v = (c \circledast f) \bmod q \equiv (r \circledast h + m) \circledast f \pmod q$$

Why HPS98 decryption works

Decryption

Input: Ciphertext c.

1: $v = (c \circledast f) \bmod q$.

2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Recall:

- $c = (r \circledast h + m) \bmod q$.
- $h = (p \circledast g \circledast F_q) \bmod q$.

Crucial step is:

$$\begin{aligned} v = (c \circledast f) \bmod q &\equiv (r \circledast h + m) \circledast f \pmod q \\ &\equiv (r \circledast p \circledast g \circledast F_q + m) \circledast f \pmod q \end{aligned}$$

Why HPS98 decryption works

Decryption

Input: Ciphertext c.

1: $v = (c \circledast f) \bmod q$.

2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Recall:

- $c = (r \circledast h + m) \bmod q$.
- $h = (p \circledast g \circledast F_q) \bmod q$.
- $(F_q \circledast f) \bmod q = 1$.

Crucial step is:

$$\begin{aligned} v = (c \circledast f) \bmod q &\equiv (r \circledast h + m) \circledast f \pmod q \\ &\equiv (r \circledast p \circledast g \circledast F_q + m) \circledast f \pmod q \\ &\equiv r \circledast p \circledast g + m \circledast f \pmod q. \end{aligned}$$

Why HPS98 decryption works

Decryption

Input: Ciphertext c.

1: $v = (c \circledast f) \bmod q$.

2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Recall:

- $c = (r \circledast h + m) \bmod q$.
- $h = (p \circledast g \circledast F_q) \bmod q$.
- $(F_q \circledast f) \bmod q = 1$.

Crucial step is:

$$\begin{aligned} v = (c \circledast f) \bmod q &\equiv (r \circledast h + m) \circledast f \pmod q \\ &\equiv (r \circledast p \circledast g \circledast F_q + m) \circledast f \pmod q \\ &\equiv r \circledast p \circledast g + m \circledast f \pmod q. \end{aligned}$$

Correctness depends on equality in

$$(c \circledast f) \bmod q \stackrel{?}{=} r \circledast p \circledast g + m \circledast f.$$

Why HPS98 decryption works

Decryption

Input: Ciphertext c.

1: $v = (c \circledast f) \bmod q$.

2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Recall:

- $c = (r \circledast h + m) \bmod q$.
- $h = (p \circledast g \circledast F_q) \bmod q$.
- $(F_q \circledast f) \bmod q = 1$.

Why HPS98 decryption works

Decryption

Input: Ciphertext c.

1: $v = (c \circledast f) \bmod q$.

2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Recall:

- $c = (r \circledast h + m) \bmod q$.
- $h = (p \circledast g \circledast F_q) \bmod q$.
- $(F_q \circledast f) \bmod q = 1$.

Equality in

$$(c \circledast f) \bmod q \stackrel{?}{=} r \circledast p \circledast g + m \circledast f$$

holds when

$$|r \circledast p \circledast g + m \circledast f|_\infty < q/2.$$

Why HPS98 decryption works

Decryption

Input: Ciphertext c.

1: $v = (c \circledast f) \bmod q$.

2: $m' = (v \circledast F_p) \bmod p$.

Output: m'.

Recall:

- $c = (r \circledast h + m) \bmod q$.
- $h = (p \circledast g \circledast F_q) \bmod q$.
- $(F_q \circledast f) \bmod q = 1$.

Equality in

$$(c \circledast f) \bmod q \stackrel{?}{=} r \circledast p \circledast g + m \circledast f$$

holds when

$$|r \circledast p \circledast g + m \circledast f|_\infty < q/2.$$

Parameters, incl. $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$, are chosen to ensure this usually holds. It is possible to choose parameters for which this always holds.

NTRU-HRSS

NTRU-HRSS

Arithmetic is still in $R \cong \mathbb{Z}[x]/(x^n - 1)$,

NTRU-HRSS

Arithmetic is still in $R \cong \mathbb{Z}[x]/(x^n - 1)$,
but now we will pay attention to the fact that

$$x^n - 1 = (x - 1)(x^{n-1} + x^{n-2} + \cdots + x + 1).$$

NTRU-HRSS

Arithmetic is still in $R \cong \mathbb{Z}[x]/(x^n - 1)$,
but now we will pay attention to the fact that

$$x^n - 1 = (x - 1)\underbrace{(x^{n-1} + x^{n-2} + \cdots + x + 1)}_{\Phi_n}.$$

It will be helpful to define $S \cong \mathbb{Z}[x]/(\Phi_n)$.

NTRU-HRSS

Parameters: Prime n for which both 2 and 3 generate $(\mathbb{Z}/n)^\times$, $p = 3$, and $q = 2^{\lceil 3.5+\log n \rceil}$.

NTRU-HRSS

Parameters: Prime n for which both 2 and 3 generate $(\mathbb{Z}/n)^\times$, $p = 3$, and $q = 2^{\lceil 3.5 + \log n \rceil}$.

Define

$$\mathcal{T} = \{v \in \{-1, 0, 1\}^n \ : \ v_{n-1} = 0\}$$

and

$$\mathcal{T}_+ = \{v \in \mathcal{T} : \langle x \circledast v, v \rangle \geq 0\}.$$

NTRU-HRSS

Parameters: Prime n for which both 2 and 3 generate $(\mathbb{Z}/n)^\times$, $p = 3$, and $q = 2^{\lceil 3.5 + \log n \rceil}$.

Define

$$\mathcal{T} = \{v \in \{-1, 0, 1\}^n \; : \; v_{n-1} = 0\}$$

and

$$\mathcal{T}_+ = \{v \in \mathcal{T} : \langle x \circledast v, v \rangle \geq 0\}.$$

Sample spaces: $\mathcal{L}_f = \mathcal{L}_g = \mathcal{T}_+$ and $\mathcal{L}_r = \mathcal{L}_m = \mathcal{T}$.

NTRU-HRSS

Parameters: Prime n for which both 2 and 3 generate $(\mathbb{Z}/n)^\times$, $p = 3$, and $q = 2^{\lceil 3.5+\log n \rceil}$.

Define

$$\mathcal{T} = \{v \in \{-1, 0, 1\}^n \ : \ v_{n-1} = 0\}$$

and

$$\mathcal{T}_+ = \{v \in \mathcal{T} : \langle x \circledast v, v \rangle \geq 0\}.$$

Sample spaces: $\mathcal{L}_f = \mathcal{L}_g = \mathcal{T}_+$ and $\mathcal{L}_r = \mathcal{L}_m = \mathcal{T}$.

For the experts: We want to do NTRU in $S = \mathbb{Z}[x]/(\Phi_n)$, but we want perfect correctness and small q. The usual decryption algorithm in S costs us a factor of 2 in q. Better decryption algorithms require analysis of "gap failures" (see: Silverman, NTRU Tech Report #11, 2001). Using $\mathcal{T}_+$ saves us a factor of $\sqrt{2}$, with little effort.

NTRU-HRSS

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: (Try to) compute F_q such that $(f \circledast F_q) \bmod q \equiv 1$ in S.
3: (Try to) compute F_p such that $(f \circledast F_p) \bmod p \equiv 1$ in S.
4: If step 2 or step 3 fails, go to 1.
5: $h = (p \circledast (x-1) \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

NTRU-HRSS

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: ~~(Try to)~~ compute F_q such that $(f \circledast F_q) \bmod q \equiv 1$ in S.
3: ~~(Try to)~~ compute F_p such that $(f \circledast F_p) \bmod p \equiv 1$ in S.
4: ~~If step 2 or step 3 fails, go to 1.~~
5: $h = (p \circledast (x-1) \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

NTRU-HRSS

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.

2: Compute F_q such that $(f \circledast F_q) \bmod q \equiv 1$ in S.

3: Compute F_p such that $(f \circledast F_p) \bmod p \equiv 1$ in S.

4: $h = (p \circledast (x-1) \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

NTRU-HRSS

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: Compute F_q such that $(f \circledast F_q) \bmod q \equiv 1$ in S.
3: Compute F_p such that $(f \circledast F_p) \bmod p \equiv 1$ in S.
4: $h = (p \circledast (x-1) \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

Encryption

Input: Message $m \in \mathcal{L}_m$.

1: Sample r from $\mathcal{L}_r$.
2: $c = (r \circledast h + LiftP(m)) \bmod q$.

Output: Ciphertext c.

NTRU-HRSS

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: Compute F_q such that $(f \circledast F_q) \bmod q \equiv 1$ in S.
3: Compute F_p such that $(f \circledast F_p) \bmod p \equiv 1$ in S.
4: $h = (p \circledast (x-1) \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

Encryption

Input: Message $m \in \mathcal{L}_m$.
1: Sample r from $\mathcal{L}_r$.
2: $c = (r \circledast h + \mathit{LiftP}(m)) \bmod q$.

Output: Ciphertext c.

Where

$$\mathit{LiftP}(m) = (x-1) \circledast m_0$$

with $m_0 \in \mathcal{T}$ and

$$\mathit{LiftP}(m) \equiv m \text{ in } S/p.$$

NTRU-HRSS

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: Compute F_q such that $(f \circledast F_q) \bmod q \equiv 1$ in S.
3: Compute F_p such that $(f \circledast F_p) \bmod p \equiv 1$ in S.
4: $h = (p \circledast (x-1) \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

Encryption

Input: Message $m \in \mathcal{L}_m$.

1: Sample r from $\mathcal{L}_r$.
2: $c = (r \circledast h + \mathit{LiftP}(m)) \bmod q$.

Output: Ciphertext c.

NTRU-HRSS

Key Generation

1: Sample f and g from $\mathcal{L}_f$ and $\mathcal{L}_g$.
2: Compute F_q such that $(f \circledast F_q) \bmod q \equiv 1$ in S.
3: Compute F_p such that $(f \circledast F_p) \bmod p \equiv 1$ in S.
4: $h = (p \circledast (x-1) \circledast g \circledast F_q) \bmod q$.

Output: Private key (f, F_p) and public key h.

Encryption

Input: Message $m \in \mathcal{L}_m$.
1: Sample r from $\mathcal{L}_r$.
2: $c = (r \circledast h + \mathit{LiftP}(m)) \bmod q$.

Output: Ciphertext c.

Decryption

Input: Ciphertext c.
1: $v = (c \circledast f) \bmod q$.
2: $u = (u \circledast F_p) \bmod p$.
3: $m' = (u - u_{n-1} \cdot \Phi_n) \bmod p$.

Output: m'

Correctness condition

NTRU-HRSS decryption will succeed if

$$|r \circledast p \circledast (x-1) \circledast g + \mathit{LiftP}(m) \circledast f|_\infty < q/2.$$

Correctness condition

NTRU-HRSS decryption will succeed if

$$|r \circledast p \circledast (x-1) \circledast g + \mathit{LiftP}(m) \circledast f|_\infty < q/2.$$

The triangle inequality gives:

$$\begin{aligned} |r \circledast p \circledast (x-1) \circledast g|_\infty &< 2pn. \\ |\mathit{LiftP}(m) \circledast f|_\infty &< 2n. \end{aligned}$$

Correctness condition

NTRU-HRSS decryption will succeed if

$$|r \circledast p \circledast (x-1) \circledast g + LiftP(m) \circledast f|_\infty < q/2.$$

The triangle inequality gives:

$$|r \circledast p \circledast (x-1) \circledast g|_\infty < 2pn.$$
$$|LiftP(m) \circledast f|_\infty < 2n.$$

But we prove that for $f, g \in \mathcal{T}_+$

$$|r \circledast p \circledast (x-1) \circledast g|_\infty < \sqrt{2}pn.$$
$$|LiftP(m) \circledast f|_\infty < \sqrt{2}n.$$

Why not just do NTRU in S?

"NTRU in S" decryption will succeed if

$$|r \circledast p \circledast g + m \circledast f - b\Phi_n|_\infty < q/2,$$

where b is the coefficient of x^{n-1} in $r \circledast p \circledast g + m \circledast f$.

Why not just do NTRU in S?

"NTRU in S" decryption will succeed if

$$|r \circledast p \circledast g + m \circledast f - b\Phi_n|_\infty < q/2,$$

where b is the coefficient of x^{n-1} in $r \circledast p \circledast g + m \circledast f$.

Without knowing more about b, success is only guaranteed when

$$|r \circledast p \circledast g + m \circledast f|_\infty < q/4.$$

Why not just do NTRU in S?

"NTRU in S" decryption will succeed if

$$|r \circledast p \circledast g + m \circledast f - b\Phi_n|_\infty < q/2,$$

where b is the coefficient of x^{n-1} in $r \circledast p \circledast g + m \circledast f$.

Without knowing more about b, success is only guaranteed when

$$|r \circledast p \circledast g + m \circledast f|_\infty < q/4.$$

Known (1996?) workaround: translate by $\delta\Phi_n$ before "mod p".

Why not just do NTRU in S?

"NTRU in S" decryption will succeed if

$$|r \circledast p \circledast g + m \circledast f - b\Phi_n|_\infty < q/2,$$

where b is the coefficient of x^{n-1} in $r \circledast p \circledast g + m \circledast f$.

Without knowing more about b, success is only guaranteed when

$$|r \circledast p \circledast g + m \circledast f|_\infty < q/4.$$

Known (1996?) workaround: translate by $\delta\Phi_n$ before "mod p".
Open problems:

- Choose δ in constant time.
- Save a factor $\geq \sqrt{2}$ using this approach.

How the NTRU submissions avoid decryption failures

How the NTRU submissions avoid decryption failures

- NTRU-PKE $n = 743, p = 3, q = 2048$:
  - fixed weight 494 for f and g,
  - uniform trinary for r and m,
  - expected failure rate 2^{-112} (w.r.t. honest r and m).
- SS-NTRU-PKE $n = 1024$, $p = 2$, $q = 2^{30} + 2^{13} + 1$:
  - wide gaussian for f, g, r, and m,
  - expected failure rate 2^{-80} (w.r.t. honest r and m).
- Streamlined NTRU Prime $n = 761, p = 3, q = 4591$:
  - fixed weight 286 for f and r,
  - uniform trinary for g and m.
- NTRU-HRSS $n = 701$, $p = 3$, $q = 8192$:
  - uniform $\mathcal{T}_+$ for f and g,
  - uniform trinary for r and m.

Note: these are the distributions assumed in correctness proofs, not necessarily the distributions that are used in implementations.

How the NTRU submissions avoid decryption failures

- NTRU-PKE $n = 743, p = 3, q = 2048$:
  - fixed weight 494 for f and g,
  - uniform trinary for r and m,
  - expected failure rate 2^{-112} (w.r.t. honest r and m).
- SS-NTRU-PKE $n = 1024$, $p = 2$, $q = 2^{30} + 2^{13} + 1$:
  - wide gaussian for f, g, r, and m,
  - expected failure rate 2^{-80} (w.r.t. honest r and m).
- Streamlined NTRU Prime $n = 761, p = 3, q = 4591$:
  - fixed weight 286 for f and r,
  - uniform trinary for g and m.
- NTRU-HRSS $n = 701$, $p = 3$, $q = 8192$:
  - uniform $\mathcal{T}_+$ for f and g,
  - uniform trinary for r and m.

Note: these are the distributions assumed in correctness proofs, not necessarily the distributions that are used in implementations.

How the NTRU submissions avoid decryption failures

- NTRU-PKE $n = 743, p = 3, q = 2048$:
  - fixed weight 494 for f and g,
  - uniform trinary for r and m,
  - expected failure rate 2^{-112} (w.r.t. honest r and m).
- SS-NTRU-PKE $n = 1024$, $p = 2$, $q = 2^{30} + 2^{13} + 1$:
  - wide gaussian for f, g, r, and m,
  - expected failure rate 2^{-80} (w.r.t. honest r and m).
- Streamlined NTRU Prime $n = 761, p = 3, q = 4591$:
  - fixed weight 286 for f and r,
  - uniform trinary for g and m.
- NTRU-HRSS $n = 701$, $p = 3$, $q = 8192$:
  - uniform $\mathcal{T}_+$ for f and g,
  - uniform trinary for r and m.

Note: these are the distributions assumed in correctness proofs, not necessarily the distributions that are used in implementations.

How the NTRU submissions avoid decryption failures

- NTRU-PKE $n = 743, p = 3, q = 2048$:
  - fixed weight 494 for f and g,
  - uniform trinary for r and m,
  - expected failure rate 2^{-112} (w.r.t. honest r and m).
- SS-NTRU-PKE $n = 1024$, $p = 2$, $q = 2^{30} + 2^{13} + 1$:
  - wide gaussian for f, g, r, and m,
  - expected failure rate 2^{-80} (w.r.t. honest r and m).
- Streamlined NTRU Prime $n = 761, p = 3, q = 4591$:
  - fixed weight 286 for f and r,
  - uniform trinary for g and m.
- NTRU-HRSS $n = 701$, $p = 3$, $q = 8192$:
  - uniform $\mathcal{T}_+$ for f and g,
  - uniform trinary for r and m.

Note: these are the distributions assumed in correctness proofs, not necessarily the distributions that are used in implementations.

How the NTRU submissions avoid decryption failures

- NTRU-PKE $n = 743, p = 3, q = 2048$:
  - fixed weight 494 for f and g,
  - uniform trinary for r and m,
  - expected failure rate 2^{-112} (w.r.t. honest r and m).
- SS-NTRU-PKE $n = 1024$, $p = 2$, $q = 2^{30} + 2^{13} + 1$:
  - wide gaussian for f, g, r, and m,
  - expected failure rate 2^{-80} (w.r.t. honest r and m).
- Streamlined NTRU Prime $n = 761, p = 3, q = 4591$:
  - fixed weight 286 for f and r,
  - uniform trinary for g and m.
- NTRU-HRSS $n = 701$, $p = 3$, $q = 8192$:
  - uniform $\mathcal{T}_+$ for f and g,
  - uniform trinary for r and m.

Note: these are the distributions assumed in correctness proofs, not necessarily the distributions that are used in implementations.

The "evaluate at 1" map

The "evaluate at 1" map

Recall: $R \cong \mathbb{Z}[x]/(x^n - 1)$ and

$$x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \cdots + x + 1).$$

The "evaluate at 1" map

Recall: $R \cong \mathbb{Z}[x]/(x^n - 1)$ and

$$x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \cdots + x + 1).$$

So $x \mapsto 1$ is a ring homomorphism $R \to \mathbb{Z}$.

The "evaluate at 1" map

Recall: $R \cong \mathbb{Z}[x]/(x^n - 1)$ and

$$x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \cdots + x + 1).$$

So $x \mapsto 1$ is a ring homomorphism $R \to \mathbb{Z}$.
This implies, e.g.,

$$c(1) = pr(1)h(1) + m(1) \bmod q.$$

The “evaluate at 1” map

Three solutions:

Control sample spaces.

- NTRU-PKE.

Multiply the HPS98 values of h and m by $(x - 1)$.

- NTRU-HRSS.

Use a different ring.

- SS-NTRU-PKE.
- NTRU Prime.

CCA transform

We use a OWCPA-PKE to CCA-KEM transform due to Dent.

CCA transform

We use a OWCPA-PKE to CCA-KEM transform due to Dent.

CCA-Encaps:

- Sample $m \in \mathcal{T}$.
- Hash m to get coins for encryption and a session key.
- Encrypt m, using the coins to sample $r \in \mathcal{T}$.
- Output ciphertext and session key.

CCA transform

We use a OWCPA-PKE to CCA-KEM transform due to Dent.

CCA-Encaps:

- Sample $m \in \mathcal{T}$.
- Hash m to get coins for encryption and a session key.
- Encrypt m, using the coins to sample $r \in \mathcal{T}$.
- Output ciphertext and session key.

CCA-Decaps: Decrypt, re-encrypt, and compare.

CCA transform

We use a OWCPA-PKE to CCA-KEM transform due to Dent.

CCA-Encaps:

- Sample $m \in \mathcal{T}$.
- Hash m to get coins for encryption and a session key.
- Encrypt m, using the coins to sample $r \in \mathcal{T}$.
- Output ciphertext and session key.

CCA-Decaps: Decrypt, re-encrypt, and compare.

Note: Our submission includes an additional hash for a QROM proof.
Accounts for 141 bytes of the ciphertext.

Parameters, security, and performance

We claim $n = 701$ ($q = 8192$) meets requirements of security category 1.

	Cycles*
Keygen:	294 847
Encaps:	38 456
Decaps:	68 458

	Bytes
sk:	1422
pk:	1140
c:	$1140 + 141$

* Optimized AVX2 impl. on 3.5 GHz Intel Core i7-4770K CPU.

Recap

Pros:

- No decryption failures.
- Simple CCA transform (no padding mechanism).
- No fixed weight distributions.
- Public keys and ciphertexts map to 0 under $x \mapsto 1$.
- No invertibility checks in key gen.
- New routines (*LiftP*, sampling from $\mathcal{T}_+$) are cheap.

Cons:

- q is a factor of $\sqrt{2}$ larger than in HPS98 (for same correctness).
- Need to compute F_p.