NTRU-HRSS-KEM

Andreas Hülsing¹, Joost Rijneveld², John Schanck³, Peter Schwabe²

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 $R = (\mathbb{Z}^n, +, \ast)$, where $\ast$ 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 $[-t/2, t/2)$. 

Parameters: $n, p, q \in \mathbb{Z}$ with $\gcd(p, q) = 1$ and $p \ll q$.

Sample spaces $L_f, L_g, L_r, L_m$ 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 $\{-1, 0, 1\}^n$. 

2 / 14
NTRU (Hoffstein–Pipher–Silverman 1998)

Arithmetic is in $R = (\mathbb{Z}^n, +, \ast)$, where $\ast$ 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 $\{-1, 0, 1\}^n$. 


Arithmetic is in \( R = (\mathbb{Z}^n, +, \star) \), where \( \star \) 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 $(f \odot F_q) \mod q = 1$.
3: (Try to) compute $F_p$ such that $(f \odot F_p) \mod p = 1$.
4: If step 2 or step 3 fails, go to 1.
5: $h = (p \odot g \odot F_q) \mod q$.

**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 $(f \star F_q) \mod q = 1$.
3. (Try to) compute $F_p$ such that $(f \star F_p) \mod p = 1$.
4. If step 2 or step 3 fails, go to 1.
5. $h = (p \star g \star F_q) \mod q$.

**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. $c = (r \star h + m) \mod q$.

**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 $(f \odot F_q) \mod q = 1$.
3: (Try to) compute $F_p$ such that $(f \odot F_p) \mod p = 1$.
4: If step 2 or step 3 fails, go to 1.
5: $h = (p \odot g \odot F_q) \mod q$.

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: $c = (r \odot h + m) \mod q$.
Output: Ciphertext $c$.

Decryption
Input: Ciphertext $c$.
1: $v = (c \odot f) \mod q$.
2: $m' = (v \odot F_p) \mod p$.
Output: $m'$. 
Why HPS98 decryption works

**Decryption**

**Input:** Ciphertext $c$.
1: $v = (c \otimes f) \mod q$.
2: $m' = (v \otimes F_p) \mod p$.

**Output:** $m'$.

Crucial step is:

$$v = (c \otimes f) \mod q$$
Why HPS98 decryption works

Decryption

Input: Ciphertext $c$.
1: $v = (c \otimes f) \mod q$.
2: $m' = (v \otimes F_p) \mod p$.
Output: $m'$.

Recall:

- $c = (r \otimes h + m) \mod q$.

Crucial step is:

$$v = (c \otimes f) \mod q \equiv (r \otimes h + m) \otimes f \pmod{q}$$
Why HPS98 decryption works

Decryption

Input: Ciphertext $c$.
1. $v = (c \otimes f) \mod q$.
2. $m' = (v \otimes F_p) \mod p$.
Output: $m'$.

Recall:
- $c = (r \otimes h + m) \mod q$.
- $h = (p \otimes g \otimes F_q) \mod q$.

Crucial step is:

\[
v = (c \otimes f) \mod q \equiv (r \otimes h + m) \otimes f \pmod{q} \\
\equiv (r \otimes p \otimes g \otimes F_q + m) \otimes f \pmod{q}
\]
Why HPS98 decryption works

**Decryption**

**Input:** Ciphertext \( c \).

1. \( v = (c \circ f) \mod q \).
2. \( m' = (v \circ F_p) \mod p \).

**Output:** \( m' \).

Recall:

- \( c = (r \circ h + m) \mod q \).
- \( h = (p \circ g \circ F_q) \mod q \).
- \( (F_q \circ f) \mod q = 1 \).

Crucial step is:

\[
\begin{align*}
v &= (c \circ f) \mod q \
   &= (r \circ h + m) \circ f \pmod{q} \
   &
   \equiv (r \circ p \circ g \circ F_q + m) \circ f \pmod{q} \
   &
   \equiv r \circ p \circ g + m \circ f \pmod{q}.
\end{align*}
\]
Why HPS98 decryption works

**Decryption**

**Input:** Ciphertext \( c \).
1. \( v = (c \star f) \mod q \).
2. \( m' = (v \star F_p) \mod p \).

**Output:** \( m' \).

Recall:
- \( c = (r \star h + m) \mod q \).
- \( h = (p \star g \star F_q) \mod q \).
- \( (F_q \star f) \mod q = 1 \).

Crucial step is:

\[
v = (c \star f) \mod q \equiv (r \star h + m) \star f \pmod{q} \\
\equiv (r \star p \star g \star F_q + m) \star f \pmod{q} \\
\equiv r \star p \star g + m \star f \pmod{q}.
\]

Correctness depends on equality in

\[
(c \star f) \mod q \overset{?}{=} r \star p \star g + m \star f.
\]
Why HPS98 decryption works

Decryption

Input: Ciphertext \( c \).
1: \( v = (c \otimes f) \mod q \).
2: \( m' = (v \otimes F_p) \mod p \).
Output: \( m' \).

Recall:
- \( c = (r \otimes h + m) \mod q \).
- \( h = (p \otimes g \otimes F_q) \mod q \).
- \((F_q \otimes f) \mod q = 1\).
Why HPS98 decryption works

**Decryption**

**Input:** Ciphertext $c$.
1. $v = (c \odot f) \mod q$.
2. $m' = (v \odot F_p) \mod p$.

**Output:** $m'$.

**Recall:**
- $c = (r \odot h + m) \mod q$.
- $h = (p \odot g \odot F_q) \mod q$.
- $(F_q \odot f) \mod q = 1$.

Equality in

$$(c \odot f) \mod q \overset{?}{=} r \odot p \odot g + m \odot f$$

holds when

$$|r \odot p \odot g + m \odot f|_{\infty} < q/2.$$
Why HPS98 decryption works

Decryption

**Input:** Ciphertext \( c \).
1. \( v = (c \odot f) \mod q \).
2. \( m' = (v \odot F_p) \mod p \).
**Output:** \( m' \).

Recall:
- \( c = (r \odot h + m) \mod q \).
- \( h = (p \odot g \odot F_q) \mod q \).
- \( (F_q \odot f) \mod q = 1 \).

Equality in

\[(c \odot f) \mod q \overset{?}{=} r \odot p \odot g + m \odot f\]

holds when

\[|r \odot p \odot g + m \odot f|_\infty < q/2.\]

Parameters, incl. \( L_f, L_g, L_r, L_m \), are chosen to ensure this usually holds. It is possible to choose parameters for which this always holds.
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 $q = 2^{3.5+\log n}$. 
NTRU-HRSS

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

Define

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

and

$$\mathcal{T}_+ = \{ v \in \mathcal{T} : \langle x \ordan v, v \rangle \geq 0 \}.$$
NTRU-HRSS

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

Define

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

and

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

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 \( q = 2^{[3.5+\log n]} \).

Define
\[
\mathcal{T} = \{ v \in \{-1, 0, 1\}^n : v_{n-1} = 0 \}
\]
and
\[
\mathcal{T}_+ = \{ v \in \mathcal{T} : \langle x \otimes v, v \rangle \geq 0 \}.
\]

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.
Key Generation

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

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 $(f \odot F_q) \mod q \equiv 1$ in $S$.
3: *(Try to)* compute $F_p$ such that $(f \odot F_p) \mod p \equiv 1$ in $S$.
4: If step 2 or step 3 fails, go to 1.
5: $h = (p \otimes (x - 1) \otimes g \otimes F_q) \mod q$.

Output: Private key $(f, F_p)$ and public key $h$. 
**Key Generation**

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

**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 $(f \odot F_q) \mod q \equiv 1$ in $S$.
3: Compute $F_p$ such that $(f \odot F_p) \mod p \equiv 1$ in $S$.
4: $h = (p \odot (x - 1) \odot g \odot F_q) \mod q$.
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: $c = (r \odot h + \text{LiftP}(m)) \mod q$.
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 $(f \odot F_q) \mod q \equiv 1$ in $S$.
3: Compute $F_p$ such that $(f \odot F_p) \mod p \equiv 1$ in $S$.
4: $h = (p \odot (x - 1) \odot g \odot F_q) \mod q$.
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: $c = (r \odot h + \text{LiftP}(m)) \mod q$.

Where
\[
\text{LiftP}(m) = (x - 1) \odot m_0
\]
with $m_0 \in \mathcal{T}$ and $\text{LiftP}(m) \equiv m$ in $S/p$.

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 $(f \ast F_q) \mod q \equiv 1$ in $S$.
3: Compute $F_p$ such that $(f \ast F_p) \mod p \equiv 1$ in $S$.
4: $h = (p \ast (x - 1) \ast g \ast F_q) \mod q$.

**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: $c = (r \ast h + \text{LiftP}(m)) \mod q$.

**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 \((f \odot F_q) \mod q \equiv 1\) in \( S \).
3: Compute \( F_p \) such that \((f \odot F_p) \mod p \equiv 1\) in \( S \).
4: \( h = (p \odot (x - 1) \odot g \odot F_q) \mod q \).

**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: \( c = (r \odot h + \text{LiftP}(m)) \mod q \).

**Output:** Ciphertext \( c \).

**Decryption**

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

**Output:** \( m' \)
Correctness condition

NTRU-HRSS decryption will succeed if

\[ \left| r \otimes p \otimes (x - 1) \otimes g + \text{LiftP}(m) \otimes f \right|_\infty < q/2. \]
Correctness condition

NTRU-HRSS decryption will succeed if

$$|r \otimes p \otimes (x - 1) \otimes g + \text{LiftP}(m) \otimes f|_\infty < \frac{q}{2}.\)$$

The triangle inequality gives:

$$|r \otimes p \otimes (x - 1) \otimes g|_\infty < 2pn.$$

$$|\text{LiftP}(m) \otimes f|_\infty < 2n.$$
Correctness condition

NTRU-HRSS decryption will succeed if

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

The triangle inequality gives:

$$\left| r \star p \star (x - 1) \star g \right|_{\infty} < 2pn.$$  
$$\left| \text{LiftP}(m) \star f \right|_{\infty} < 2n.$$  

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

$$\left| r \star p \star (x - 1) \star g \right|_{\infty} < \sqrt{2pn}.$$  
$$\left| \text{LiftP}(m) \star f \right|_{\infty} < \sqrt{2n}.$$
Why not just do NTRU in $S$?

“NTRU in $S$” decryption will succeed if

$$\left| r \circ p \circ g + m \circ f - b\Phi_n \right|_\infty < \frac{q}{2},$$

where $b$ is the coefficient of $x^{n-1}$ in $r \circ p \circ g + m \circ f$. 
Why not just do NTRU in $S$?

“NTRU in $S$” decryption will succeed if

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

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

Without knowing more about $b$, success is only guaranteed when

$$|r \circ p \circ g + m \circ f|_\infty < q/4.$$
Why not just do NTRU in $S$?

“NTRU in $S$” decryption will succeed if

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

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

Without knowing more about $b$, success is only guaranteed when

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

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

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

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

Without knowing more about $b$, success is only guaranteed when

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

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

Open problems:
- Choose $\delta$ in constant time.
- Save a factor $\geq \sqrt{2}$ using this approach.
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 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.*
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

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

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

This implies, e.g.,

\[
c(1) = \text{pr}(1)h(1) + m(1) \mod q.
\]

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.
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.,

$$c(1) = pr(1)h(1) + m(1) \mod q.$$
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 $m \in T$.
- Hash $m$ to get coins for encryption and a session key.
- Encrypt $m$, using the coins to sample $r \in T$.
- 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.
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.
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.

<table>
<thead>
<tr>
<th></th>
<th>Cycles*</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Keygen:</td>
<td>294,847</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encaps:</td>
<td>38,456</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decaps:</td>
<td>68,458</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|          |          |          |          |
| Bytes    |          |          |          |
| sk:      | 1,422    |          |          |
| pk:      | 1,140    |          |          |
| c:       | 1,140 + 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 ($\text{LiftP}$, sampling from $\mathcal{T}_+$) are cheap.

Cons:

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