NTRU Prime

20171130

Principal submitter

This submission is from the following team, listed in alphabetical order:

- Daniel J. Bernstein, University of Illinois at Chicago
- Chitchanok Chuengsatiansup, INRIA and ENS de Lyon
- Tanja Lange, Technische Universiteit Eindhoven
- Christine van Vredendaal, Technische Universiteit Eindhoven

E-mail address (preferred): `authorcontact-ntruprime@box.cr.yp.to`

Telephone (if absolutely necessary): +1-312-996-3422

Postal address (if absolutely necessary): Daniel J. Bernstein, Department of Computer Science, University of Illinois at Chicago, 851 S. Morgan (M/C 152), Room 1120 SEO, Chicago, IL 60607–7053.

Auxiliary submitters: There are no auxiliary submitters. The principal submitter is the team listed above.

Inventors/developers: The inventors/developers of this submission are the same as the principal submitter. Relevant prior work is credited below where appropriate.

Owner: Same as submitter.

Signature: ×. See also printed version of "Statement by Each Submitter".

Document generated with the help of `pqskeleton` version 20171123.

Contents

1 Introduction

A 2015 algorithm breaks dimension-N SVP (under plausible assumptions) in time $2^{(c+o(1))N}$ as $N \to \infty$ with $c \approx 0.292$. See [9]. For comparison, the best algorithm known just five years earlier had a much worse $c \approx 0.415$, and the best algorithm known just ten years before that took time $2^{\Theta(N \log N)}$.

Gentry's original FHE system at STOC 2009, with standard "cyclotomic" choices of rings, is now known (again under plausible assumptions) to be broken in polynomial time by a quantum algorithm. See [12]. Peikert claimed in 2015 that the weakness in Gentry's system was specific to Gentry's short generators and inapplicable to Ideal-SVP:

> Although cyclotomics have a lot of structure, nobody has yet found a way to exploit it in attacking Ideal-SVP/BDD ... For commonly used rings, principal ideals are an extremely small fraction of all ideals. ... The weakness here is not so much due to the structure of cyclotomics, but rather to the extra structure of principal ideals that have short generators.

However, the attack was then combined with further features of cyclotomics to break Ideal-SVP (again under plausible assumptions) with approximation factor $2^{N^{1/2+o(1)}}$, a terrifying advance compared to the previous $2^{N^{1+o(1)}}$. See [24].

As these attack examples illustrate, the security of lattice-based cryptography is not well understood. There are serious risks of further advances in

- SVP algorithms,
- algorithms that exploit the "approximation factors" used in cryptography,
- algorithms that exploit the structure of cryptographic problems such as LWE,
- algorithms that exploit the multiplicative structure of *efficient* cryptographic problems such as Ring-LWE,
- algorithms that exploit the structure of these problems for the *specific* rings chosen by users, and
- algorithms to break cryptosystems without breaking these problems.

The point of this submission is that the attack surface in lattice-based cryptography can be significantly reduced with only a minor loss of efficiency. In fact, despite the extra security criteria imposed below, the two cryptosystems in this submission are two of the smallest and fastest lattice-based cryptosystems.

2 General algorithm specification (part of 2.B.1)

This submission provides two key-encapsulation mechanisms: "Streamlined NTRU Prime" and "NTRU LPRime".

2.1 Streamlined NTRU Prime parameter space

Streamlined NTRU Prime has parameters (p, q, w) subject to the following restrictions: p is a prime number; q is a prime number; w is a positive integer; $2p \geq 3w$; $q \geq 16w + 1$; $x^p - x - 1$ is irreducible in the polynomial ring $(\mathbb{Z}/q)[x]$.

We abbreviate the ring $\mathbb{Z}[x]/(x^p - x - 1)$, the ring $(\mathbb{Z}/3)[x]/(x^p - x - 1)$, and the field $(\mathbb{Z}/q)[x]/(x^p - x - 1)$ as $\mathcal{R}$, $\mathcal{R}/3$, and $\mathcal{R}/q$ respectively. We refer to an element of $\mathcal{R}$ as **small** if all of its coefficients are in $\{-1, 0, 1\}$, and **weight w** if exactly w of its coefficients are nonzero.

Streamlined NTRU Prime also has the following parameters: an encoding of public keys as strings; an encoding of rounded ring elements (see below) as strings; and a hash function mapping each small polynomial to two fixed-length output strings, a "confirmation" and a "session key".

2.2 Streamlined NTRU Prime key generation

The receiver generates a public key as follows:

- Generate a uniform random small element $g \in \mathcal{R}$. Repeat this step until g is invertible in $\mathcal{R}/3$. (There are various standard ways to test invertibility: for example, one can check divisibility of g by the irreducible factors of $x^p - x - 1$ modulo 3, or one can deduce invertibility as a side effect of various algorithms to compute $1/g$ in $\mathcal{R}/3$.)
- Generate a uniform random small weight-w element $f \in \mathcal{R}$. (Note that f is nonzero and hence invertible in $\mathcal{R}/q$, since $w \geq 1$.)
- Compute $h = g/(3f)$ in $\mathcal{R}/q$. (By assumption q is a prime larger than 3, so 3 is invertible in $\mathcal{R}/q$, so $3f$ is invertible in $\mathcal{R}/q$.)
- Encode h as a string $\overline{h}$, using the aforementioned encoding of public keys as strings. The public key is $\overline{h}$.
- Save the following secrets: f in $\mathcal{R}$; and $1/g$ in $\mathcal{R}/3$.

2.3 Streamlined NTRU Prime encapsulation

The sender generates a ciphertext as follows:

- Decode the public key $\overline{h}$, obtaining $h \in \mathcal{R}/q$.
- Generate a uniform random small weight-w element $r \in \mathcal{R}$.
- Compute $hr \in \mathcal{R}/q$.
- Round each coefficient of hr, viewed as an integer between $-(q-1)/2$ and $(q-1)/2$, to the nearest multiple of 3, producing $c \in \mathcal{R}$. (If $q \in 1+3\mathbb{Z}$ then each coefficient of c is in $\{-(q-1)/2, \ldots, -6, -3, 0, 3, 6, \ldots, (q-1)/2\}$. If $q \in 2+3\mathbb{Z}$ then each coefficient of c is in $\{-(q+1)/2, \ldots, -6, -3, 0, 3, 6, \ldots, (q+1)/2\}$. Rounding adds an element from $\{-1, 0, 1\}$ to each coefficient.)
- Encode c as a string $\underline{c}$, using the aforementioned encoding of rounded ring elements as strings.
- Hash r, obtaining a confirmation C and a session key K. The ciphertext is the concatenation $C\,\underline{c}$.

2.4 Streamlined NTRU Prime decapsulation

The receiver decapsulates a ciphertext $C\,\underline{c}$ as follows:

- Decode $\underline{c}$, obtaining $c \in \mathcal{R}$.
- Multiply by $3f$ in $\mathcal{R}/q$.
- View each coefficient of $3fc$ in $\mathcal{R}/q$ as an integer between $-(q-1)/2$ and $(q-1)/2$, and then reduce modulo 3, obtaining a polynomial e in $\mathcal{R}/3$.
- Multiply by $1/g$ in $\mathcal{R}/3$.
- Lift e/g in $\mathcal{R}/3$ to a small polynomial $r' \in \mathcal{R}$.
- Compute c', C', K' from r' as in encapsulation.
- If r' is small, r' has weight w, $c' = c$, and $C' = C$, then output K'. Otherwise output False.

If $C\,\underline{c}$ is a legitimate ciphertext then c is obtained by rounding the coefficients of hr to the nearest multiples of 3; i.e., $c = m + hr$ in $\mathcal{R}/q$, where m is small. All coefficients of the polynomial $3fm + gr$ in $\mathcal{R}$ are in $[-8w, 8w]$ by Theorem 2 below, and thus in $[-(q-1)/2, (q-1)/2]$ since $q \geq 16w+1$. Viewing each coefficient of $3fc = 3fm+gr$ as an integer in $[-(q-1)/2, (q-1)/2]$ thus produces exactly $3fm+gr \in \mathcal{R}$, and reducing modulo 3 produces $gr \in \mathcal{R}/3$; i.e., $e = gr$ in $\mathcal{R}/3$, so $e/g = r$ in $\mathcal{R}/3$. Lifting now produces exactly r since r is small; i.e., $r' = r$. Hence $(c', C', K') = (c, C, K)$. Finally, $r' = r$ is small, r' has weight w, $c' = c$, and $C' = C$, so decapsulation outputs $K' = K$, the same session key produced by encapsulation.

Theorem 1 *Fix integers $p \geq 3$ and $w \geq 1$. Let $r, g \in \mathbb{Z}[x]$ be polynomials of degree at most $p-1$ with all coefficients in $\{-1, 0, 1\}$. Assume that r has at most w nonzero coefficients. Then $gr \bmod x^p - x - 1$ has each coefficient in the interval $[-2w, 2w]$.*

Theorem 2 *Fix integers $p \geq 3$ and $w \geq 1$. Let $m, r, f, g \in \mathbb{Z}[x]$ be polynomials of degree at most $p-1$ with all coefficients in $\{-1, 0, 1\}$. Assume that f and r each have at most w nonzero coefficients. Then $3fm + gr \bmod x^p - x - 1$ has each coefficient in the interval $[-8w, 8w]$.*

2.5 NTRU LPRime parameter space

NTRU LPRime has parameters (p, q, w, δ, I) subject to the following restrictions: p is a prime number; q is a prime number; w, δ, I are positive integers; $2p \geq 3w$; I is a multiple of 8; $p \geq I$; $q \geq 16w + 2\delta + 3$; $x^p - x - 1$ is irreducible in the polynomial ring $\mathcal{R}/q$.

As before, we abbreviate the ring $\mathbb{Z}[x]/(x^p - x - 1)$, the ring $(\mathbb{Z}/3)[x]/(x^p - x - 1)$, and the field $(\mathbb{Z}/q)[x]/(x^p - x - 1)$ as $\mathcal{R}$, $\mathcal{R}/3$, and $\mathcal{R}/q$ respectively. We refer to an element of $\mathcal{R}$ as **small** if all of its coefficients are in $\{-1, 0, 1\}$, and **weight w** if exactly w of its coefficients are nonzero.

NTRU LPRime also has the following parameters: an encoding of rounded ring elements (see below) as strings; a hash function mapping each I-bit string to three fixed-length output strings, a "cipher key" and a "confirmation" and a "session key"; a function Small from the set of cipher keys to the set of small weight-w elements in $\mathcal{R}$; a function Generator from a set of "seed" strings to $\mathcal{R}/q$; a function Top from $(\mathbb{Z}/q)^I$ to a fixed-length set of strings; and a function Right from the same set of strings to $(\mathbb{Z}/q)^I$ such that each coordinate of the difference $\text{Right}(\text{Top}(C)) - C$ is in $\{0, 1, \ldots, \delta\}$ for each $C \in (\mathbb{Z}/q)^I$.

2.6 NTRU LPRime key generation

The receiver generates a public key as follows:

- Generate a uniform random seed S.
- Compute $G = \text{Generator}(S) \in \mathcal{R}/q$.
- Generate a uniform random small weight-w element $a \in \mathcal{R}$.
- Compute $aG \in \mathcal{R}/q$.
- Round each coefficient of aG, viewed as an integer between $-(q-1)/2$ and $(q-1)/2$, to the nearest multiple of 3, producing $A \in \mathcal{R}$.
- Encode A as a string $\underline{A}$. The public key is the concatenation $S\underline{A}$.
- Save the secret a.

2.7 NTRU LPRime encapsulation

The sender generates a ciphertext as follows:

- Decode the public key $S\underline{A}$, obtaining a seed S and a polynomial $A \in \mathcal{R}$.
- Compute $G = \text{Generator}(S) \in \mathcal{R}/q$.
- Generate a uniform random I-bit string $r = (r_0, r_1, \ldots, r_{I-1})$.
- Hash r, obtaining a cipher key k, a confirmation H, and a session key K.
- Compute $b = \text{Small}(k) \in \mathcal{R}$.
- Compute bG in $\mathcal{R}/q$.
- Compute bA in $\mathcal{R}/q$. (Only the bottom I coefficients of bA, the coefficients $(bA)_0, (bA)_1, \ldots, (bA)_{I-1}$ of $x^0, x^1, \ldots, x^{I-1}$ respectively, will be used; other coefficients do not need to be computed.)
- Round each coefficient of bG, viewed as an integer between $-(q-1)/2$ and $(q-1)/2$, to the nearest multiple of 3, producing $B \in \mathcal{R}$.
- Encode B as a string $\underline{B}$.
- Compute $C = (C_0, C_1, \ldots, C_{I-1}) \in (\mathbb{Z}/q)^I$ as follows: $C_j = (bA)_j + r_j(q-1)/2$.
- Compute $\tilde{C} = \text{Top}(C)$.
- The ciphertext is the concatenation $H\underline{B}\tilde{C}$. The session key is K.

2.8 NTRU LPRime decapsulation

The receiver decapsulates a ciphertext $H\underline{B}\tilde{C}$ as follows:

- Decode $\underline{B}$, obtaining $B \in \mathcal{R}$.
- Compute $T = \text{Right}(\tilde{C}) \in (\mathbb{Z}/q)^I$.
- Compute aB in $\mathcal{R}/q$. (Only the bottom I coefficients of aB will be used.)
- Compute $r'_0, r'_1, \ldots, r'_{I-1} \in \{0, 1\}$ as follows. View $T_j - (aB)_j + 4w + 1 \in \mathbb{Z}/q$ as an integer between $-(q-1)/2$ and $(q-1)/2$. Then r'_j is the sign bit of this integer: 1 if the integer is negative, otherwise 0.
- Compute a ciphertext c' and session key K' from $r' = (r'_0, r'_1, \ldots, r'_{I-1})$ as in encapsulation.
- If the ciphertext c' is $H\underline{B}\tilde{C}$, then output K'. Otherwise output False.

The public key A is obtained by rounding the coefficients of aG to the nearest multiples of 3; i.e., $A = aG + d$ in $\mathcal{R}/q$, where d is small.

If $H\underline{B}\tilde{C}$ is a legitimate ciphertext then $\underline{B}$ is an encoding of B which is obtained by rounding the coefficients of bG to the nearest multiples of 3; i.e., $B = bG + e$ in $\mathcal{R}/q$, where e is small, and $\tilde{C} = \mathrm{Top}(C)$ with $C_j = (bA)_j + r_j(q-1)/2$.

By construction the functions Top and Right are such that each coordinate of $\mathrm{Right}(\mathrm{Top}(C)) - C$ is in $\{0, 1, \ldots, \delta\}$ for each $C \in (\mathbb{Z}/q)^I$, i.e,. $\mathrm{Right}(\mathrm{Top}(C))_j - C_j \in \{0, 1, \ldots, \delta\}$.

Then

$$\begin{aligned}
T_j - (aB)_j + 4w + 1 &= \mathrm{Right}(\mathrm{Top}(C))_j - (a(bG+e))_j + 4w + 1 \\
&= \mathrm{Right}(\mathrm{Top}(C))_j - C_j + C_j - ((abG)_j + (ae)_j) + 4w + 1 \\
&= \mathrm{Right}(\mathrm{Top}(C))_j - C_j + (bA)_j + r_j(q-1)/2 - ((abG)_j + (ae)_j) + 4w + 1 \\
&= \mathrm{Right}(\mathrm{Top}(C))_j - C_j + (baG)_j + (bd)_j + r_j(q-1)/2 - ((abG)_j + (ae)_j) + 4w + 1 \\
&= \mathrm{Right}(\mathrm{Top}(C))_j - C_j + (bd)_j - (ae)_j + 4w + 1 + r_j(q-1)/2 \in \mathbb{Z}/q.
\end{aligned}$$

All coefficients of the polynomials bd and ae are in $[-2w, 2w]$ by Theorem 1, thus

$$1 \leq \mathrm{Right}(\mathrm{Top}(C))_j - C_j + (bd)_j - (ae)_j + 4w + 1 \leq 8w + \delta + 1.$$

Viewing each coefficient of $T_j - (aB)_j + 4w + 1$ as an integer in $[-(q-1)/2, (q-1)/2]$ thus produces an integer in $[1, 8w+\delta+1]$ if and only if $r_j = 0$ and an integer in $[-(q-1)/2, -(q-1)/2 + 8w + \delta]$ if and only if $r_j = 1$ because $8w + \delta + 1 \leq (q-1)/2$ by construction.

This means that $r'_j = r_j$, thus $r' = r$ and $c' = c$, so decapsulation outputs $K' = K$, the same session key produced by encapsulation.

3 List of parameter sets (part of 2.B.1)

3.1 Parameter set `kem/sntrup4591761`

Streamlined NTRU Prime with $p = 761$, $q = 4591$, $w = 286$, and the following functions.

Encoding of public keys as strings: View the input polynomial in little-endian form as a sequence of coefficients of $x^0, x^1, \ldots, x^{764}$. The coefficients of $x^{761}, \ldots, x^{764}$ are always 0.

View each coefficient in $\mathbb{Z}/4591$ as an element of $\{-2295, \ldots, 2295\}$. Add 2295 to obtain an element of $\{0, \ldots, 4590\}$.

Write each batch of 5 elements c_0, c_1, c_2, c_3, c_4 in radix $6144 = 3 \cdot 2^{11}$ as the integer $c_0 + 6144c_1 + 6144^2c_2 + 6144^3c_3 + 6144^4c_4$. This integer is below $6144^5 < 2^{63}$. Write this integer as 8 bytes in little-endian form.

This produces $8(765/5) = 1224$ bytes. The last 6 bytes are always 0 and are suppressed, so a public key is encoded as 1218 bytes.

Encoding of rounded ring elements as strings: View the input polynomial in little-endian form as a sequence of coefficients of $x^0, x^1, \ldots, x^{761}$. The coefficient of x^{761} is always 0.

View each coefficient in $\mathbb{Z}/4591$ as an element of $\{-2295, -2292, \ldots, 2292, 2295\}$; recall that ciphertext coefficients are always multiples of 3. Add 2295 to obtain an element of $\{0, 3, \ldots, 4587, 4590\}$. Divide by 3 to obtain an element of $\{0, 1, \ldots, 1530\}$.

Write each batch of 3 elements c_0, c_1, c_2 in radix $1536 = 3 \cdot 2^9$ as the integer $c_0 + 1536c_1 + 1536^2c_2$. This integer is below $1536^3 < 2^{32}$. Write this integer as 4 bytes in little-endian form.

This produces $4(762/3) = 1016$ bytes. The last byte is always 0 and is suppressed, so a rounded ring element is encoded as 1015 bytes.

Hash function: View the input polynomial r in little-endian form as a sequence of coefficients of $x^0, x^1, \ldots, x^{763}$. The coefficients of $x^{761}, x^{762}, x^{763}$ are always 0.

Add 1 to each coefficient, obtaining an element of $\{0, 1, 2\}$. Write each batch of 4 elements in radix 4, obtaining a byte. Overall this produces $764/4 = 191$ bytes.

Hash the resulting byte string with SHA-512, obtaining a 256-bit confirmation followed by a 256-bit session key.

3.2 Parameter set `kem/ntrulpr4591761`

NTRU LPRime with $p = 761$, $q = 4591$, $w = 250$, $\delta = 292$, $I = 256$, and the following functions.

Encoding of rounded ring elements as strings: Same as in `sntrup4591761`.

Hash function: View the 256-bit string r in little-endian form as a 32-byte string, i.e. the first byte of r is $r_0 + 2r_1 + \cdots + 128r_7$, the next byte is $r_8 + 2r_9 + \cdots + 128r_{15}$, etc.

Hash r with SHA-512, obtaining a 32-byte cipher key k followed by a 32-byte intermediate key k'. Hash k' with SHA-512, obtaining a 32-byte confirmation followed by a 32-byte session key.

Mapping to $\mathcal{R}$: For each 32-byte string k, $\text{Small}(K) \in \mathcal{R}$ is defined as follows:

- Use AES-256-CTR with key k, starting from counter 0, to generate $4p$ bytes of output.
- View each 4 bytes of output in little-endian form, obtaining p elements of $\{0, 1, \ldots, 2^{32} - 1\}$.
- Clear the bottom bit of each of the first w integers; now each of those integers is 0 modulo 2.

- Set the bottom bit, and clear the next bit, of each of the remaining $p - w$ integers; now each of those integers is 1 modulo 4.
- Sort the integers.
- Reduce each integer modulo 4, and subtract 1, obtaining p elements of $\{-1, 0, 1\}$, of which exactly w are nonzero.
- View these elements as a polynomial in little-endian form, namely Small(K).

Mapping to $\mathcal{R}/q$: The set of seeds is the set of 32-byte strings. For each 32-byte string K, Generator(K) $\in \mathcal{R}/q$ is defined as follows:

- Use AES-256-CTR with key K, starting from counter 0, to generate $4p$ bytes of output.
- View each 4 bytes of output in little-endian form, obtaining p elements of $\{0, 1, \ldots, 2^{32} - 1\}$.
- Reduce each of these elements modulo q, obtaining p elements of $\{0, 1, \ldots, q - 1\}$.
- Obtain p elements of $\{-(q-1)/2, \ldots, (q-1)/2\}$ by subtractng $(q-1)/2$ from each integer.
- View these elements as a polynomial in little-endian form, namely Generator(K).

Top bits: For each $C \in (\mathbb{Z}/q)^{256}$, Top($C$) is a 128-byte string defined as follows:

- View each C_j as an integer between -2295 and 2295.
- Compute $T_j = \lfloor (114(C_j + 2156) + 16384)/32768 \rfloor \in \{0, 1, \ldots, 15\}$ for each j.
- Define Top(C) $= (T_0 + 16T_1, T_2 + 16T_3, \ldots, T_{254} + 16T_{255})$.

For each 128-byte string T, Right(T) $\in (\mathbb{Z}/q)^{256}$ is defined as follows:

- Extract $T_0, T_1, \ldots, T_{255} \in \{0, 1, \ldots, 15\}$ from T in little-endian form.
- Compute $R_j = 287T_j - 2007$ for each j.
- Define Right(T) $= (R_0, R_1, \ldots, R_{255})$.

One can check each integer $c \in \{-2295, \ldots, 2295\}$ to see that $(287t - 2007) - c \in \{0, 1, \ldots, 292\}$ where $t = \lfloor (114(c + 2156) + 16384)/32768 \rfloor$.

4 Design rationale (part of 2.B.1)

There are many different ideal-lattice-based public-key encryption schemes in the literature, including many versions of NTRU; many Ring-LWE-based cryptosystems; and now Streamlined NTRU Prime and NTRU LPRime. These are actually many different points in a high-dimensional space of possible cryptosystems. We give a unified description of the advantages and disadvantages of what we see as the most important options in each dimension, in particular explaining the choices that we made in Streamlined NTRU Prime and NTRU LPRime. Beware that there are many interactions between options. For example, using Gaussian errors is incompatible with eliminating decryption failures, because there is always a small probability of large samples combining with large values. Using *truncated* Gaussian errors is compatible with eliminating decryption failures, but requires a much larger modulus q. Neither of these options is compatible with the simple tight KEM that we use.

4.1 The ring

The choice of cryptosystem includes a choice of a monic degree-p polynomial $P \in \mathbb{Z}[x]$ and a choice of a positive integer q. As in Section 2, we abbreviate the ring $\mathbb{Z}[x]/P$ as $\mathcal{R}$, and the ring $(\mathbb{Z}/q)[x]/P$ as $\mathcal{R}/q$.

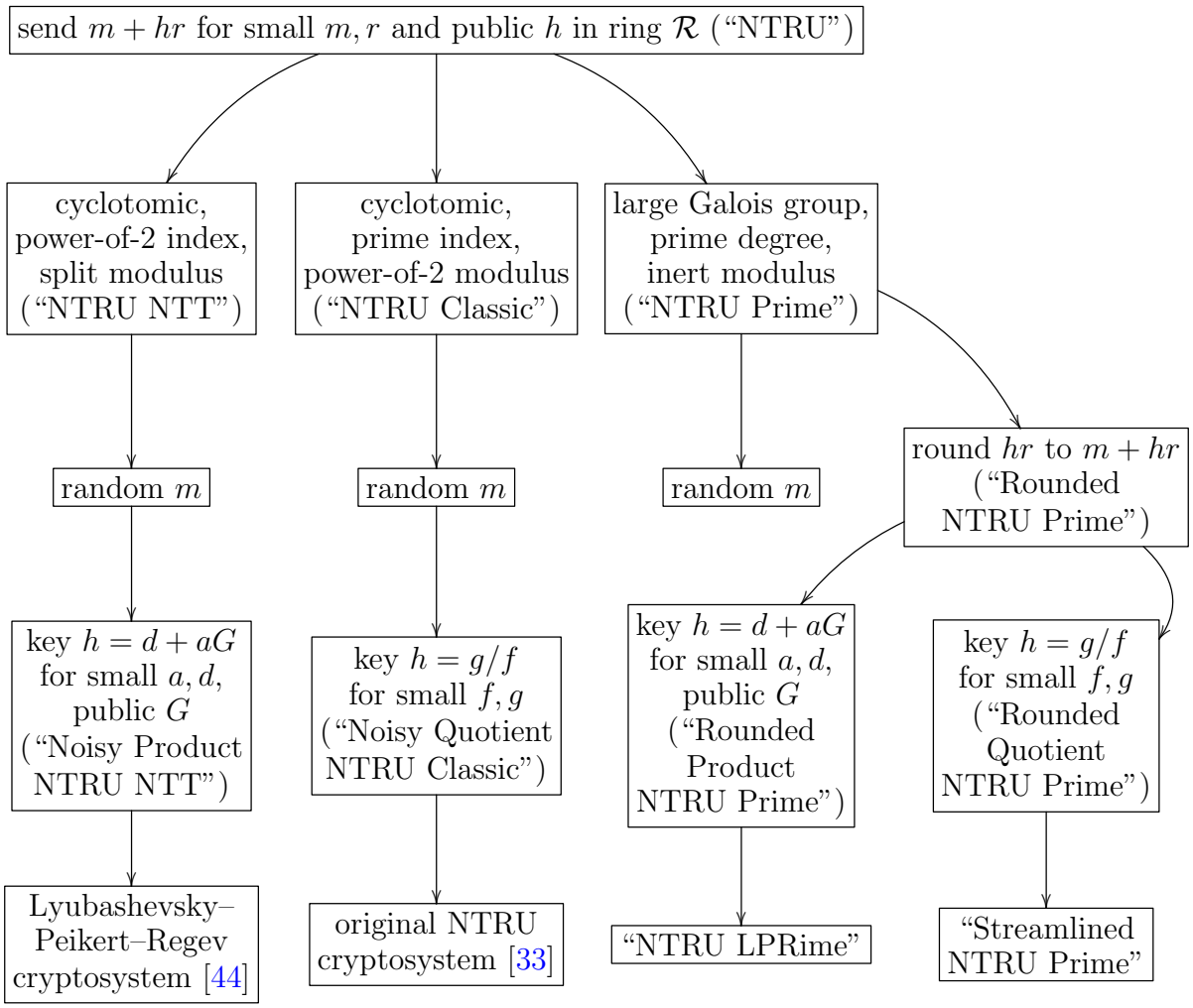
Common choices of $\mathcal{R}/q$ are as follows:

- "NTRU Classic": Rings of the form $(\mathbb{Z}/q)[x]/(x^p - 1)$, where p is a prime and q is a power of 2, are used in the original NTRU cryptosystem [33].
- "NTRU NTT": Rings of the form $(\mathbb{Z}/q)[x]/(x^p + 1)$, where p is a power of 2 and $q \in 1 + 2p\mathbb{Z}$ is a prime, are used in typical "Ring-LWE-based" cryptosystems such as [3].
- "NTRU Prime": Fields of the form $(\mathbb{Z}/q)[x]/(x^p - x - 1)$, where p is prime, are used in this submission.

NTRU Prime uses a prime-degree number field with a large Galois group and an inert modulus, minimizing the number of ring homomorphisms available to the attacker. As an analogy, conservative prime-field discrete-logarithm systems also minimize the number of ring homomorphisms available to the attacker.

We expect the future situation, like the current situation, to be a mix of the following three scenarios:

- Some lattice-based systems are broken whether or not they have unnecessary homomorphisms. As an analogy, some discrete-logarithm systems are broken whether or not they have unnecessary homomorphisms.



- Some lattice-based systems are unbroken whether or not they have unnecessary homomorphisms. As an analogy, some discrete-logarithm systems are unbroken whether or not they have unnecessary homomorphisms.
- Some lattice-based systems are broken *only if* they have unnecessary homomorphisms. As an analogy, some discrete-logarithm systems are broken only if they have unnecessary homomorphisms. Eliminating unnecessary homomorphisms rescues these systems, and removes the need to worry about what attackers can do with these homomorphisms.

The current situation is that homomorphisms eliminated by NTRU Prime are used in the following attack papers: [18], [28], [23], [20], [24], and [8]. See our "NTRU Prime" paper for further details.

4.2 The public key

The receiver's public key, which we call h, is an element of $\mathcal{R}/q$.

4.3 Inputs and ciphertexts

In the original NTRU system, ciphertexts are elements of the form $m + hr \in \mathcal{R}/q$. Here $h \in \mathcal{R}/q$ is the public key as above, and m, r are small elements of $\mathcal{R}$ chosen by the sender.

Subsequent systems labeled as "NTRU" have generally extended ciphertexts to include additional information, for various reasons explained below; but these cryptosystems all share the same core design element, sending $m + hr \in \mathcal{R}/q$ where m, r are small secrets and h is public. We suggest systematically using the name "NTRU" to refer to this design element, and more specific names (e.g., "NTRU Classic" vs. "NTRU Prime") to refer to other design elements.

We refer to (m, r) as "input" rather than "plaintext" because in any modern public-key cryptosystem the input is randomized and is separated from the sender's plaintext by symmetric primitives such as hash functions. See Section 4.5.

In the original NTRU specification [33], m was allowed to be any element of $\mathcal{R}$ having all coefficients in a standard range. The range was $\{-1, 0, 1\}$ for all of the suggested parameters, with q not a multiple of 3, and we focus on this case for simplicity (although we note that some other lattice-based cryptosystems have taken the smaller range $\{0, 1\}$, or sometimes larger ranges).

Current NTRU Classic specifications such as [32] prohibit m that have an unusually small number of 0's or 1's or -1's. For random m, this prohibition applies with probability $<2^{-10}$, and in case of failure the sender can try encoding the plaintext as a new m, but this is problematic for applications with hard real-time requirements. The reason for this prohibition is that NTRU Classic gives the attacker an "evaluate at 1" homomorphism from $\mathcal{R}/q$ to $\mathbb{Z}/q$, leaking $m(1)$. The attacker scans many ciphertexts to find an occasional ciphertext where the value $m(1)$ is particularly far from 0; this value constrains the search space for the corresponding m by enough bits to raise security concerns. In NTRU Prime, $\mathcal{R}/q$ is a field, so this type of leak cannot occur.

Streamlined NTRU Prime actually uses a different type of ciphertext, which we call a "rounded ciphertext". The sender chooses a small r as input and computes $hr \in \mathcal{R}/q$. The sender obtains the ciphertext by rounding each coefficient of hr, viewed as an integer between $-(q-1)/2$ and $(q-1)/2$, to the nearest multiple of 3. This ciphertext can be viewed as an example of the original ciphertext $m + hr$, but with m chosen so that each coefficient of $m + hr$ is in a restricted subset of $\mathbb{Z}/q$.

With the original ciphertexts, each coefficient of $m + hr$ leaves 3 possibilities for the corresponding coefficients of hr and m. With rounded ciphertexts, each coefficient of $m + hr$ also leaves 3 possibilities for the corresponding coefficients of hr and m, except that the

boundary cases $-(q-1)/2$ and $(q-1)/2$ (assuming $q \in 1+3\mathbb{Z}$) leave only 2 possibilities. In a pool of 2^{64} rounded ciphertexts, the attacker might find one ciphertext that has 15 of these boundary cases out of 761 coefficients; these occasional exceptions have very little impact on known attacks. It would be possible to randomize the choice of multiples of 3 near the boundaries, but we prefer the simplicity of having the ciphertext determined entirely by r. It would also be possible to prohibit ciphertexts at the boundaries, but as above we prefer to avoid restarting the encryption process.

More generally, we say "Rounded NTRU" for any NTRU system in which m is chosen deterministically by rounding hr to a standard subset of $\mathbb{Z}/q$, and "Noisy NTRU" for the original version in which m is chosen randomly. Rounded NTRU has two advantages over Noisy NTRU. First, it reduces the space required to transmit $m+hr$. Second, the fact that m is determined by r simplifies protection against chosen-ciphertext attacks; see Section 4.5.

[49, Section 4] used an intermediate non-deterministic possibility to provide some space reduction for a public-key cryptosystem: first choose m randomly, and then round $m+hr$, obtaining $m'+hr$. The idea of rounded hr as a *deterministic* substitute for noisy $m+hr$ was introduced in [7] in the context of a symmetric-key construction, was used in [5] to construct another public-key encryption system, and was further studied in [13] and [4]. All of the public-key cryptosystems in these papers have ciphertexts longer than Noisy NTRU, but applying the same idea to Noisy NTRU produces Rounded NTRU, which has shorter ciphertexts.

4.4 Key generation and decryption

In the original NTRU cryptosystem, the public key h has the form $3g/f$ in $\mathcal{R}/q$, where f and g are secret. Decryption computes $fc = fm+3gr$, reduces modulo 3 to obtain fm, and multiplies by $1/f$ to obtain m.

Streamlined NTRU Prime changes the position of the 3, taking h as $g/(3f)$ rather than $3g/f$. Decryption computes $3fc = 3fm+gr$, reduces modulo 3 to obtain gr, and multiplies by $1/g$ to obtain r. This change lets us compute (m,r) by first computing r and then multiplying by h, whereas otherwise we would first compute m and then multiply by $1/h$. One advantage is that we skip computing $1/h$; another advantage is that we need less space for storing a key pair. This $1/h$ issue does not arise for NTRU variants that compute r as a hash of m, but those variants are incompatible with rounded ciphertexts, as discussed in Section 4.5.

More generally, we say "Quotient NTRU" for NTRU with h computed as a ratio of two secret small polynomials. An alternative is what we call "Product NTRU", namely NTRU with h of the form $d+aG$, where a and d are secret small polynomials. Here $G \in \mathcal{R}/q$ is public, like h, but unlike h it does not need a hidden multiplicative structure: it can be, for example, a standard chosen randomly by a trusted authority, or output of a long hash function applied to a standard randomly chosen seed, or (as proposed in [3]) output of a long hash function applied to a per-receiver seed supplied along with h as part of the public key.

Product NTRU does not allow the same decryption procedure as Quotient NTRU. The first

Product NTRU system, introduced by Lyubashevsky, Peikert, and Regev in [44] (originally in talk slides in 2010), sends $e + rG$ as additional ciphertext along with $m + hr + M$, where, as before, m and r are small polynomials, e is another small polynomial, and M is a polynomial consisting of solely 0 or $\lfloor q/2 \rfloor$ in each position. The receiver computes $(m + hr + M) - a(e + rG) = M + m + dr - ae$, and rounds to 0 or $\lfloor q/2 \rfloor$ in each position, obtaining M. Note that $m + dr - ae$ is small, since all of m, d, r, a, e are small.

The ciphertext size here, two elements of $\mathcal{R}/q$, can be improved in various ways. One can replace hr with fewer coefficients, for example by summing batches of two or three coefficients [53], before adding M and m. Rounded Product NTRU rounds $hr+M$ to obtain $m+hr+M$, rounds rG to obtain $e+rG$, and (to similarly reduce key size) rounds aG to obtain $d+aG$. Decryption continues to work even if $m + hr + M$ is compressed to two bits per coefficient.

A disadvantage of Product NTRU is that r is used twice, exposing approximations to both rG and hr. This complicates security analysis compared to simply exposing an approximation to hr. State-of-the-art attacks against Ring-LWE, which reveals approximations to any number of random public multiples of r, are significantly faster for many multiples than for one multiple. Perhaps this indicates a broader weakness, in which each extra multiple hurts security.

Quotient NTRU has an analogous disadvantage: if one moves far enough in the parameter space [39] then state-of-the-art attacks distinguish g/f from random more efficiently than they distinguish $m + hr$ from random. Perhaps this indicates a broader weakness. On the other hand, if one moves far enough in another direction in the parameter space [61], then g/f has a security proof.

We find both of these issues worrisome: it is not at all clear which of Product NTRU and Quotient NTRU is a safer option.[1] We see no way to simultaneously avoid both types of complications. We have opted to present details of Streamlined NTRU Prime, an example of Quotient NTRU Prime; and of NTRU LPRime, an example of Product NTRU Prime.

If exposing approximations to two multiples of r damages the security of Product NTRU, perhaps exposing fewer bits does less damage. The compression techniques mentioned above, such as replacing $m + hr + M$ with fewer coefficients and releasing only a few top bits of each coefficient, naturally expose fewer bits than uncompressed ciphertexts. NTRU LPRime releases a few top bits of each of the bottom coefficients of $m + hr + M$, enough coefficients to communicate a hard-to-guess input M.

The Quotient NTRU literature, except for the earliest papers, takes f of the form $1 + 3F$, where F is small. This eliminates the multiplication by the inverse of f modulo 3. In Streamlined NTRU Prime we have chosen to skip this speedup for two reasons. First, in the long run we expect cryptography to be implemented in hardware, where a multiplication

[1] Peikert claimed in [50], modulo terminology, that Product NTRU is "at least as hard" to break as Quotient NTRU (and "likely strictly harder"). This claim ignores the possibility of attacks against the reuse of r in Product NTRU. There are no theorems justifying Peikert's claim, and we are not aware of an argument that eliminating this reuse is less important than eliminating the g/f structure. For comparison, switching from NTRU NTT and NTRU Classic to NTRU Prime eliminates structure used in some state-of-the-art attacks without providing new structure used in other attacks.

in $\mathcal{R}/3$ is far less expensive than a multiplication in $\mathcal{R}/q$. Second, this speedup requires noticeably larger keys and ciphertexts for the same security level, and this is important for many applications, while very few applications will notice the CPU time for Streamlined NTRU Prime.

4.5 Padding, KEMs, and the choice of q

In Streamlined NTRU Prime and NTRU LPRime we use the modern "KEM+DEM" approach introduced by Shoup; see [58]. This approach is much nicer for implementors than previous approaches to public-key encryption. For readers unfamiliar with this approach, we briefly review the analogous options for RSA encryption.

RSA maps an input m to a ciphertext $m^e \bmod n$, where (n, e) is the receiver's public key. When RSA was first introduced, its input m was described as the sender's plaintext. This was broken in reasonable attack models, leading to the development of various schemes to build m as some combination of fixed padding, random padding, and a short plaintext; typically this short plaintext is used as a shared secret key. This turned out to be quite difficult to get right, both in theory (see, e.g., [59]) and in practice (see, e.g., [46]), although it does seem possible to protect against arbitrary chosen-ciphertext attacks by building m in a sufficiently convoluted way.

The "KEM+DEM" approach, specifically Shoup's "RSA-KEM" in [58] (also called "Simple RSA"), is much easier:

- Choose a uniform random integer m modulo n. This step does not even look at the plaintext.
- To obtain a shared secret key, simply apply a cryptographic hash function to m.
- Encrypt and authenticate the sender's plaintext using this shared key.

Any attempt to modify m, or the plaintext, will be caught by the authenticator.

"KEM" means "key encapsulation mechanism": $m^e \bmod n$ is an "encapsulation" of the shared secret key $H(m)$. "DEM" means "data encapsulation mechanism", referring to the encryption and authentication using this shared secret key. Authenticated ciphers are normally designed to be secure for many messages, so $H(m)$ can be reused to protect further messages from the sender to the receiver, or from the receiver back to the sender. It is also easy to combine KEMs, for example combining a pre-quantum KEM with a post-quantum KEM, by simply hashing the shared secrets together.

When NTRU was introduced, its input (m, r) was described as a sender plaintext m combined with a random r. This is obviously not secure against chosen-ciphertext attacks. Subsequent NTRU papers introduced various mechanisms to build (m, r) as increasingly convoluted combinations of fixed padding, random padding, and a short plaintext.

It is easy to guess that KEMs simplify NTRU, the same way that KEMs simplify RSA; we are certainly not the first to suggest this. However, all the NTRU-based KEMs we have found in the literature (e.g., [60] and [55]) construct the NTRU input (m, r) by hashing a shorter input and verifying this hash during decapsulation; typically r is produced as a hash of m. These KEMs implicitly assume that m and r can be chosen independently, whereas rounded ciphertexts (see Section 4.3) have r as the sole input. It is also not clear that generic-hash chosen-ciphertext attacks against these KEMs are as difficult as inverting the NTRU map from input to ciphertext: the security theorems are quite loose.

We instead follow a simple generic KEM construction introduced in the earlier paper [25, Section 6] by Dent, backed by a tight security reduction [25, Theorem 8] saying that generic-hash chosen-ciphertext attacks are as difficult as inverting the underlying function:

- Like RSA-KEM, this construction hashes the input, in our case r, to obtain the session key.
- Decapsulation verifies that the ciphertext is the correct ciphertext for this input, preventing per-input ciphertext malleability.
- The KEM uses additional hash output for key confirmation, making clear that a ciphertext cannot be generated except by someone who knows the corresponding input.

Key confirmation might be overkill from a security perspective, since a random session key will also produce an authentication failure; but key confirmation allows the KEM to be audited without regard to the authentication mechanism, and adds only 3% to our ciphertext size.

Dent's security analysis assumes that decryption works for all inputs. We achieve this in Streamlined NTRU Prime by requiring $q \geq 16w + 1$. Recall that decryption sees $3fm + gr$ in $\mathcal{R}/q$ and tries to deduce $3fm + gr$ in $\mathcal{R}$; the condition $q \geq 16w + 1$ guarantees that this works, since each coefficient of $3fm + gr$ in $\mathcal{R}$ is between $-(q-1)/2$ and $(q-1)/2$ by Theorem 2. Taking different shapes of m, r, f, g, or changing the polynomial $P = x^p - x - 1$, would change the bound $16w + 1$; for example, replacing g by $1 + 3G$ would change $16w + 1$ into $24w + 3$.

Similarly, NTRU LPRime takes $q \geq 16w + 2\delta + 3$ to avoid decryption failures. Sending along merely top bits of $m + hr + M$ means that there is an additional error, producing a slightly worse bound than in the Streamlined NTRU Prime case. Another difference in details is that decryption reconstructs only M, not m; NTRU LPRime chooses r deterministically[2] as a hash of M.

In lattice-based cryptography it is standard to take somewhat smaller values of q. The idea is that coefficients in $3fm + gr$ are produced as sums of many $+1$ and -1 terms, and these

[2]This requires another layer of security analysis beyond Dent's security analysis. The core question is whether it is hard to recover a random M from ciphertext and public key, when r is chosen randomly. The next question, the extra layer, is whether it is hard to recover a random M from ciphertext and public key, when r is chosen as a hash of M. The third question, addressed by Dent's security analysis, is whether the KEM is hard to break.

terms *usually* cancel, rather than conspiring to produce the maximum conceivable coefficient. However, this idea led to attacks that exploited occasional decryption failures; see [35] and, for an analogous attack on code-based cryptography using QC-MDPC codes, [30]. It is common today to choose q so that decryption failures will occur with, e.g., probability 2^{-80}; but this does not meet Dent's assumption that decryption always works. This nonzero failure rate appears to account for most of the complications in the literature on NTRU-based KEMs. We prefer to guarantee that decryption works, making the security analysis simpler and more robust.

4.6 The shape of small polynomials

As noted in Section 4.3, the coefficients of m are chosen from the limited range $\{-1, 0, 1\}$. The NTRU literature [33, 37, 31, 32] generally puts the same limit on the coefficients of r, g, and f, except that if f is chosen with the shape $1 + 3F$ (see Section 4.4) then the literature puts this limit on the coefficients of F. Sometimes these "ternary polynomials" are further restricted to "binary polynomials", excluding coefficient -1.

The NTRU literature further restricts the Hamming weight of r, g, and f. Specifically, a cryptosystem parameter is introduced to specify the number of 1's and -1's. For example, there is a parameter t (typically called "d" in NTRU papers) so that r has exactly t coefficients equal to 1, exactly t coefficients equal to -1, and the remaining $p - 2t$ coefficients equal to 0. These restrictions allow decryption for smaller values of q (see Section 4.5), saving space and time. Beware, however, that if t is *too* small then there are attacks; see our security analysis in Section 6.

In Streamlined NTRU Prime we keep the requirement that r have Hamming weight $w = 2t$, and keep the requirement that these w nonzero coefficients are all in $\{-1, 1\}$, but we drop the requirement of an equal split between -1 and 1. This allows somewhat more choices of r. The same comments apply to f. Similarly, we require g to have all coefficients in $\{-1, 0, 1\}$ but the distribution is otherwise unconstrained. We also require that f and g be invertible in $\mathcal{R}/q$, which simply means nonzero given that $P(x)$ is irreducible for NTRU Prime, and that g be invertible in $\mathcal{R}/3$.

These changes would affect the conventional NTRU decryption procedure: they expand the *typical* size of coefficients of fm and gr, forcing larger choices of q to avoid *noticeable* decryption failures. But we instead choose q to avoid *all* decryption failures (see Section 4.5), and these changes do not expand our *bound* on the size of the coefficients of fm and gr.

In NTRU LPRime we similarly choose small weight-w polynomials with coefficients in $\{-1, 0, 1\}$ without restricting the distribution of -1 and 1 beyond the weight.

Elsewhere in the literature on lattice-based cryptography one can find larger coefficients: consider, e.g., the quinary polynomials in [27], and the even wider range in [3]. In [61], the coefficients of f and g are sampled from a very wide discrete Gaussian distribution, allowing a proof regarding the distribution of g/f. However, this appears to produce *worse* security for any given key size. Specifically, there are no known attack strategies blocked

by a Gaussian distribution, while the very wide distribution forces q to be very large to enable decryption (see Section 4.5), producing a much larger key size (and ciphertext size) for the same security level. Furthermore, wide Gaussian distributions are practically always implemented with variable-time algorithms, creating security problems, as illustrated by the successful cache-timing attacks in [17] and [51].

5 Detailed performance analysis (2.B.2)

5.1 Description of platform

The following measurements were collected using `supercop-20170904` running on a computer named `titan0`. The CPU on `titan0` is an Intel Xeon E3-1275 v3 (Haswell) running at 3.5 GHz. Turbo Boost is disabled. `titan0` has 32GB of RAM and runs Ubuntu 14.04. Benchmarks used `./do-part`, which ran on one core of the CPU. The compiler list was reduced to just `gcc -march=native -mtune=native -O3 -fomit-frame-pointer -fwrapv`.

NIST says that the "NIST PQC Reference Platform" is "an Intel x64 running Windows or Linux and supporting the GCC compiler." `titan0` is an Intel x64 running Linux and supporting the GCC compiler. Beware, however, that different Intel CPUs have different cycle counts.

5.2 Time

In the first measurement run (many timings), the median encapsulation time for `sntrup4591761` was 59456 cycles, and the median decapsulation time was 97684 cycles. Timings were practically identical in the second measurement run (59476, 97624) and the third measurement run (59508, 97692).

Key-generation time was slower, over 6 million cycles. With more effort one can eliminate most of these cycles,[3] but our current key-generation cost is already negligible. Specifically:

- The standard design goal of IND-CCA2 security means that it is safe to generate a key once and use the key any number of times. The situation in several recent lattice-based KEMs (for example, BCNS [15], New Hope [3], and Frodo [14]) is completely different: they are not designed to resist, and do not resist, chosen-ciphertext attacks, so they generate a new key for every ciphertext, so their key-generation time is important.
- Forward secrecy does *not* require constant generation of new keys. A typical quad-core 3GHz server generating a new short-term key every minute is using under 1/100000 of its CPU time on key generation with our current software.

[3] For example, "fast gcd" techniques incorporate subquadratic-time multiplication methods such as Karatsuba's method, and are compatible with constant-time computations.

- A user who (for some reason) wants to generate many keys more quickly than this can use Montgomery's trick to batch the inversions. Montgomery's trick replaces (e.g.) 1000 inversions with 2997 multiplications and just 1 inversion. This reduces the cost of generating each key below 300000 cycles.

Our software is analogous to the original Curve25519 software [10], which emphasized encryption/decryption speed and did not bother speeding up occasional key-generation computations.

`ntrulpr4591761` is estimated to be somewhat slower than `sntrup4591761`, although it is faster than `sntrup4591761` for key generation.

5.3 Space

Public keys for `sntrup4591761` occupy 1218 bytes. Ciphertexts occupy only 1047 bytes. Secret keys occupy 1600 bytes.

Public keys for `ntrulpr4591761` occupy 1047 bytes. Ciphertexts occupy 1175 bytes. Secret keys occupy 1238 bytes.

5.4 How parameters affect performance

Encapsulation and decapsulation involve a few multiplications in the ring $\mathcal{R}/q$. The asymptotic cost of multiplication, as p and q grow, is essentially linear in $p \log_2 q$, the number of bits in a ring element. Other operations scale at least as well as this.

6 Analysis of known attacks (2.B.5)

We start with existing *pre-quantum* NTRU attack strategies, adapt those strategies to the context of Streamlined NTRU Prime, and quantify their effectiveness. In particular, we account for the impact of changing $x^p - 1$ to $x^p - x - 1$, and using small f rather than $f = 1+3F$ with small F. For comparability we assume here that the weight w in Streamlined NTRU Prime is taken as $2t$, where t is the number of 1's and the number of -1's in the original NTRU cryptosystem.

We consider NTRU LPRime in Section 6.8. We consider post-quantum security in Section 6.6.

6.1 Warning: underestimates are dangerous

Underestimating attack cost can *damage* security, for reasons explained in [11, full version, Appendix B.1.2], so we prefer to use accurate cost estimates. However, accurately evaluating

the cost of lattice attacks is generally quite difficult. The literature very often explicitly resorts to underestimates. Comprehensively fixing this problem is beyond the scope of this submission, but we have started work in this direction, as illustrated by Section 6.7. At the same time it is clear that the best attack algorithms known today are much better than the best attack algorithms known a few years ago, so it is unreasonable to expect that the algorithms have stabilized. We plan to periodically issue updated security estimates to reflect ongoing work.

6.2 Meet-in-the-middle attack

Odlyzko's meet-in-the-middle attack [36, 34] on NTRU works by splitting the space of possible keys $\mathcal{F}$ into two parts such that $\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2$. Then in each loop of the algorithm partial keys are drawn from $\mathcal{F}_1$ and $\mathcal{F}_2$ until a collision function (defined in terms of the public key h) indicates that $f_1 \in \mathcal{F}_1$ and $f_2 \in \mathcal{F}_2$ have been found such that $f = f_1 + f_2$ is the private key.

The number of choices for f is $\binom{p}{t}\binom{p-t}{t}$ in original NTRU and $\binom{p}{2t}2^{2t}$ in Streamlined NTRU Prime. A first estimate is that the number of loops in the algorithm is the square root of the number of choices of f. However, this estimate does not account for equivalent keys. In NTRU Classic, a key (f, g) is equivalent to all of the rotated keys $(x^i f, x^i g)$ and to the negations $(-x^i f, -x^i g)$, and the algorithm succeeds if it finds any of these rotated keys. The $2p$ rotations and negations are almost always distinct, producing a speedup factor very close to $\sqrt{2p}$.

The structure of the NTRU Prime ring is less friendly to this attack. Say f has degree $p-c$; typically c is around $p/2t$, since there are $2t$ terms in f. Multiplying f by $x, x^2, \ldots, x^{c-1}$ produces elements of $\mathcal{F}$, but multiplying f by x^c replaces x^{p-c} with x^p mod $x^p - x - 1 = x + 1$, changing its weight and thus leaving $\mathcal{F}$. It is possible but rare for subsequent multiplications by x to reenter $\mathcal{F}$. Similarly, one expects only about $p/2t$ divisions by x to stay within $\mathcal{F}$, for a total of only about p/t equivalent keys, or $2p/t$ when negations are taken into account. We have confirmed these estimates with experiments.

One could modify the attack to use a larger set $\mathcal{F}$, but this seems to lose more than it gains. Furthermore, similar wraparounds for g compromise the effectiveness of the collision function. To summarize, the extra term in $x^p - x - 1$ seems to increase the attack cost by a factor around $\sqrt{t}$, compared to NTRU Classic; i.e., the rotation speedup is only around $\sqrt{2p/t}$ rather than $\sqrt{2p}$.

On the other hand, some keys f allow considerably more rotations. We have decided to assume a speedup factor of $\sqrt{2(p-t)}$, since we designed some pathological polynomials f with that many (not consecutive) rotations in the set. For random r the speedup is much smaller. This means that the number of loops before this attack is expected to find f is bounded by

$$L = \sqrt{\binom{p}{2t}2^{2t}} \Big/ \sqrt{2(p-t)}. \tag{1}$$

In each loop, t vectors of size p are added and their coefficients are reduced modulo q. We thus estimate the attack cost as Lpt. The storage requirement of the attack is approximately $L \log_2 L$. We can reduce this storage by applying collision search to the meet-in-the-middle attack (see [48, 62]). In this case we can reduce the storage capacity by a factor s at the expense of increasing the running time by a factor $\sqrt{s}$.

6.3 Streamlined NTRU Prime lattice

As with NTRU Classic, we can embed the problem of recovering the private keys f, g into a lattice problem. Saying $3h = g/f$ in $\mathcal{R}/q$ is the same as saying $3hf + qk = g$ in $\mathcal{R}$ for some polynomial k; in other words, there is a vector (k, f) of length $2p$ such that

$$\begin{pmatrix} k & f \end{pmatrix} \begin{pmatrix} qI & 0 \\ H & I \end{pmatrix} = \begin{pmatrix} k & f \end{pmatrix} B = \begin{pmatrix} g & f \end{pmatrix},$$

where H is a matrix with the i'th vector corresponding to $x^i \cdot 3h \bmod x^p - x - 1$ and I is the $p \times p$ identity matrix. We will call B the *Streamlined NTRU Prime public lattice basis*. This lattice has determinant q^p. The vector (g, f) has norm at most $\sqrt{2p}$. The Gaussian heuristic states that the length of the shortest vector in a random lattice is approximately $\det(B)^{1/(2p)}\sqrt{\pi e p} = \sqrt{\pi e p q}$, which is much larger than $\sqrt{2p}$, so we expect (g, f) to be the shortest nonzero vector in the lattice.

Finding the secret keys is thus equivalent to solving the Shortest Vector Problem (SVP) for the Streamlined NTRU Prime public lattice basis. The fastest currently known method to solve SVP in the NTRU public lattice is the hybrid attack, which we discuss below.

A similar lattice can be constructed to instead try to find the input pair (m, r). However, there is no reason to expect the attack against (m, r) to be easier than the attack against (g, f): r has the same range as f, and m has essentially the same range as g. Recall that Streamlined NTRU Prime does not have the NTRU Classic problem of leaking $m(1)$. There are occasional boundary constraints on m (see Section 4.3), and there is also an $\mathcal{R}/3$ invertibility constraint on g, but these effects are minor.

6.4 Hybrid security

The best known attack against the NTRU lattice is the hybrid lattice-basis-reduction-and-meet-in-the-middle attack described in [34]. The attack works in two phases: the reduction phase and the meet-in-the-middle phase.

Applying lattice-basis-reduction techniques will mostly reduce the middle vectors of the basis [56]. Therefore the strategy of the reduction phase is to apply lattice-basis reduction, for example BKZ 2.0 [21], to a submatrix B' of the public basis B. We then get a reduced basis $T = UBY$:

$$\left(\begin{array}{c|c|c} I_u & 0 & 0 \\ \hline 0 & U' & 0 \\ \hline 0 & 0 & I_{u'} \end{array}\right) \cdot \left(\begin{array}{c|c|c} qI_u & 0 & 0 \\ \hline * & B' & 0 \\ \hline * & * & I_{u'} \end{array}\right) \cdot \left(\begin{array}{c|c|c} I_u & 0 & 0 \\ \hline 0 & Y' & 0 \\ \hline 0 & 0 & I_{u'} \end{array}\right) = \left(\begin{array}{c|c|c} qI_u & 0 & 0 \\ \hline * & T' & 0 \\ \hline * & * & I_{u'} \end{array}\right)$$

Here Y is orthonormal and T' is again in lower triangular form.

In the meet-in-the-middle phase we can use a meet-in-the-middle algorithm to guess options for the last u' coordinates of the key by guessing halves of the key and looking for collisions. If the lattice basis was reduced sufficiently in the first phase, a collision resulting in the private key will be found by applying a rounding algorithm to the half-key guesses. More details on how to do this can be found in [34].

To estimate the security against this attack we adapt the analysis of [32] to the set of keys that we use in Streamlined NTRU Prime. Let u be the dimension of I_u and u' be the dimension of $I_{u'}$. For a sufficiently reduced basis the meet-in-the-middle phase will require on average

$$-\frac{1}{2}\left(\log_2(2(p-t)) + \sum_{0 \le a \le \min\{2t,u'\}} 2^a \binom{u'}{a} v(a) \log_2(v(a))\right) \tag{2}$$

work, where the $\log_2(2(p-t))$ term accounts for equivalent keys and

$$v(a) = \frac{2^{2t-a}\binom{p-u'}{2t-a}}{2^{2t}\binom{p}{2t}} = \frac{2^{-a}\binom{p-u'}{2t-a}}{\binom{p}{2t}}. \tag{3}$$

The quality of a basis after lattice reduction can be measured by the Hermite factor $\delta = ||\mathbf{b_1}||/\det(B)^{1/p}$. Here $||\mathbf{b_1}||$ is the length of the shortest vector among the rows of B. To be able to recover the key in the meet-in-the-middle phase, the $(2p-u-u') \times (2p-u-u')$ matrix T' has to be sufficiently reduced. For given u and u' this is the case if the lattice reduction reaches the required value of δ. This Hermite factor has to satisfy

$$\log_2(\delta) \le \frac{(p-u)\log_2(q)}{(2p-(u+u'))^2} - \frac{1}{2p-(u'+u)}. \tag{4}$$

We use the BKZ 2.0 simulator of [21] to determine the best BKZ 2.0 parameters, specifically the "block size" β and the number of "rounds" n, needed to reach a root Hermite factor δ. To get a concrete security estimate of the work required to perform BKZ-2.0 with parameters β and n we use the conservative formula determined by [32] from the experiments of [22]:

$$\text{Estimate}(\beta, p, n) = 0.000784314\beta^2 + 0.366078\beta - 6.125 + \log_2(p \cdot n) + 7. \tag{5}$$

This estimate and the underlying experiments rely on "enumeration"; see Section 6.7 for a comparison to "sieving". This analysis also assumes that the probabily of two halves of the key colliding is 1. We will also conservatively assume this, but a more realistic estimate can be found in [63]. Using these estimates we can determine the optimal u and u' to attack a parameter set and thereby estimate its security.

Lastly we note that this analysis is easily adaptable to generalizing the coefficients to be in the set $\{-d, -(d-1), \ldots, d-1, d\}$ by replacing base 2 in the exponentiations in Equations 1, 2 and 3 with $2d$. In this case however the range of t, by a generalization of Theorem 2, decreases to $q > 16(d^3 + d^2)t$.

6.5 Algebraic attacks

The attack strategy of Ding [26], Arora–Ge [6], and Albrecht–Cid–Faugère–Fitzpatrick–Perret [2] takes subexponential time to break dimension-n LWE with noise width $o(\sqrt{n})$, and polynomial time to break LWE with constant noise width. However, these attacks require many LWE samples, whereas typical cryptosystems in the NTRU family provide far less data to the attacker. When these attacks are adapted to cryptosystems that provide only (say) $2n$ samples, they end up taking more than $2^{0.5n}$ time, even when the noise is limited to $\{0, 1\}$. See generally [2, Theorem 7] and [43, Case Study 1].

6.6 Quantum attacks

Grover's algorithm, amplitude amplification, and quantum walks produce better exponents for some of the subroutines used above. Preliminary estimates indicate that the overall impact on Streamlined NTRU Prime security levels is much less than the impact upon AES-256 security levels. Further analysis is required.

6.7 Memory, parallelization, and sieving algorithms

The security estimates above rely on enumeration algorithms [52, 29, 38, 32]. For very large dimensions, the performance of enumeration algorithms is slightly super-exponential and is known to be suboptimal. The provable sieving algorithms of Pujol and Stehlé [54] solve dimension-β SVP in time $2^{2.465\ldots\beta+o(\beta)}$ and space $2^{1.233\ldots\beta+o(\beta)}$, and more recent SVP algorithms [1] take time $2^{\beta+o(\beta)}$. More importantly, under heuristic assumptions, sieving is much faster. The most recent work on lattice sieving (see [9, 42]) has pushed the heuristic complexity down to $2^{0.292\ldots\beta+o(\beta)}$.

Simply comparing 0.292β to enumeration exponents suggests that sieving could be faster than enumeration for sizes of β of relevance to cryptography. However, this comparison ignores two critical caveats regarding the performance of sieving. First, a closer look at polynomial factors indicates that the $o(\beta)$ here is positive. Consider, e.g., [9, Figure 3], which reports a best fit of $2^{0.387\beta-15}$ for its fastest sieving experiments. The comparison in [47] takes this caveat into account and concludes that the sieving cutoff is "far out of reach".

Second, sieving uses much more storage as β grows: at least $2^{0.208\ldots\beta+o(\beta)}$ bits of storage, again with positive $o(\beta)$. It is not known how to reduce the storage without large increases in the number of operations. Furthermore, sieving is bottlenecked by random access to

storage, and this random access also becomes slower as the amount of storage increases. The slowdown is approximately the square root of the storage in realistic cost models; see, e.g., [16].

Enumeration fits into very little memory even for large β. Kuo, Schneider, Dagdelen, Reichelt, Buchmann, Cheng, and Yang [41] showed that enumeration parallelizes effectively within and across GPUs. An attacker who can afford enough hardware for sieving for large β can instead use the same amount of hardware for enumeration, obtaining an almost linear parallelization speedup.

We do not mean to suggest that the operation-count ratio should be multiplied by the sieving storage (accounting for this enumeration speedup) *and* further by the square root of the storage (accounting for the cost of random access inside sieving): this would ignore the possibility of a speedup from parallelizing sieving. "Mesh" sorting algorithms such as the Schnorr–Shamir algorithm [57] sort n small items in time just $O(\sqrt{n})$, which is optimal in realistic models of parallel computation; these algorithms can be used as subroutines inside sieving, reducing the asymptotic cost penalty to just $2^{0.104\ldots\beta+o(\beta)}$. However, this is still much less effective parallelization than [41].

This cost penalty for sieving is ignored in measurements such as [45] and [9, Figure 3], and in the resulting comparisons such as [47]. These measurements are limited to sieving sizes that fit into DRAM on a single computer, and do not account for the important increase in memory cost as β increases. Another way to see the same issue would be to scale sieving *down* to a small enough size to fit into GPU multiprocessors; this would demonstrate a sieving speedup for smaller β, for fundamentally the same reason that there will be a sieving slowdown for larger β.

In the absence of any realistic analyses of sieving cost for large β, we have decided to omit sieving from our security estimates. There is very little reason to believe that sieving can beat enumeration inside any attack that fits within our security target.

6.8 Attacks against NTRU LPRime

NTRU LPRime is similar to Streamlined NTRU Prime from an attack perspective. In particular, a lattice attack that finds small (m, r), given a random h and given $c = m + hr$, breaks both Streamlined NTRU Prime and NTRU LPRime.

Above we focused on the similar problem of finding small (g, f) given h and given $0 = g - 3hf$. Having to consider this second problem is a complication avoided by NTRU LPRime, and if this problem is easier then NTRU LPRime could be more secure than Streamlined NTRU Prime.

On the other hand, NTRU LPRime has its own complication, namely that it also releases an approximation to a second multiple of r. If this makes the r-recovery problem easier then NTRU LPRime could be less secure than Streamlined NTRU Prime. As noted above, we find both of these complications worrisome; further security analysis is required.

A simpler issue is that, for the same sizes p and q, NTRU LPRime places slightly smaller limits on the weight w than Streamlined NTRU Prime does. A smaller weight reduces the quantitative security level against various attack strategies discussed above.

7 Expected strength (2.B.4) in general

7.1 Security definitions

Our security goal is IND-CCA2. See Section 8 for quantitative estimates of the security of specific parameter sets.

Our general strategy for handling multi-target attacks is to aim for a very high single-target security level, and then rely on the fact that T-target attacks gain at most a factor T. We have not introduced complications aimed at making multi-target attacks even more difficult.

Our current software allows multiple encodings of ciphertexts and keys, for example allowing 4591 as a synonym for 0. NIST has stated a preference for implementations that enforce unique encodings, and we plan to adjust our software accordingly.

Lattice-based encryption also has various symmetries, analogous to well-known ECC symmetries. For example, if c (plus confirmation) is a Streamlined NTRU Prime ciphertext under public key h, then $-c$ (plus the same confirmation) is a Streamlined NTRU Prime ciphertext for the same session key under public key $-h$.

7.2 Rationale

See Section 6 for an analysis of known attacks.

Algorithm 1 searches for (p, q, t, λ), where λ is Section 6's estimate of the *pre-quantum* security level for parameters (p, q, t) with $w = 2t$. The subroutine nextprime(i) returns the first prime number $>i$. The subroutine viableqs(p, q_b) returns all primes q larger than p and smaller than q_b for which it holds that $x^p - x - 1$ is irreducible in $(\mathbb{Z}/q)[x]$. The subroutine mitmcosts uses the estimates from Equation (1) to determine the bitsecurity level of the parameters against a straightforward meet-in-the-middle attack. To find u, u', β, n we set u to the hybridbkzcost of the previous iteration (initially 0) and do a binary search for u' such that the two phases of the hybrid attack are of equal cost. For each u' we determine the Hermite factor required with Equation (4), use the BKZ-2.0 simulator to determine the optimal β and n to reach the required Hermite factor and use Equations (5) and (2) to determine the hybridbkzcost and hybridmitmcost.

Note that this algorithm outputs the largest value of t such that there are no decryption failures according to Theorem 2 and that no more than 2/3 of the coefficients of f are set. Experiments show that decreasing t to t_1 linearly decreases the security level by approximately $t - t_1$.

Algorithm 1: Determine parameter sets for security level above ℓ.

Input: Upper bound q_b for q, range $[p_1, p_2]$ for p, lower bound ℓ for security level
Result: Viable parameters p, q and t with security level λ.
$p \leftarrow p_1 - 1$ (the prime we are currently investigating)
while $p \leq p_2$ **do**
  $p \leftarrow \text{nextprime}(p)$
  $Q \leftarrow \text{viableqs}(p, q_b)$
  for $q \in Q$ **do**
    $t \leftarrow \min\{\lfloor (q-1)/32 \rfloor, \lfloor p/3 \rfloor\}$
    $\lambda_1 \leftarrow \text{mitmcosts}(p, t)$
    if $\lambda_1 \geq \ell$ **then**
      Find u, u', β, n such that BKZ-2.0 costs are approximately equal to meet-in-the-middle costs in the hybrid attack.
      $\lambda_2 \leftarrow \max\{\text{hybridbkzcost}, \text{hybridmitmcost}\}$
      return $p, q, t, \min\{\lambda_1, \lambda_2\}$

See the NTRU Prime paper for a table of Streamlined NTRU Prime parameter sets with $465 < p < 970$ and $q < 20000$. Our recommended parameters $(p, q, w) = (761, 4591, 286)$ with estimated pre-quantum security 2^{248} provide an excellent tradeoff between size and security level.

The analysis of NTRU LPRime parameter sets works the same way and gives 2^{225} for our recommended parameters $(p, q, w) = (761, 4591, 250)$.

8 Expected strength (2.B.4) for each parameter set

8.1 Parameter set `kem/sntrup4591761`

Category 5. 2^{248} is marginally smaller than 2^{256}, but we expect that further analysis along the lines of [63], together with analysis of memory and communication costs, will show that this parameter set is much more expensive to break than AES-256.

8.2 Parameter set `kem/ntrulpr4591761`

Category 5. 2^{225} is noticeably smaller than 2^{256}, due to the smaller polynomial weight compared to `sntrup4591761`. However, note that the analysis by Wunderer [63] showed an underestimate of security by a factor of at least 2^{30} for a previous set of parameters analyzed by the methodology above; memory and communication costs are likely to be on an even larger scale. Hence, we expect that further analysis will show that this parameter set is harder to break than AES-256. See also Section 6.8.

9 Advantages and limitations (2.B.6)

There are several proposals of lattice-based cryptosystems that appear to provide high security with keys and ciphertexts fitting into just a few kilobytes. This proposal is designed to have the smallest attack surface, minimizing the number of avenues available to cryptanalysts. Some recent attacks against lattice-based cryptosystems rely on homomorphisms eliminated by this proposal.

At the same time this proposal provides unusually small sizes and excellent speed. One of the reasons for this performance is that this proposal provides the flexibility to target any desired lattice dimension rather precisely, without the "jumps" that appear in most proposals. Future advances in understanding the exact security level of lattice-based cryptography will allow this proposal to be tuned accordingly.

Beware, however, that there are other recent attacks against lattice-based cryptography, including impressive advances against SVP. As noted before, the security of lattice-based cryptography is not well understood. This is a general limitation of lattice-based cryptography. The same limitation is shared by many—but not all—post-quantum proposals.

References

[1] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: Extended abstract. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 733–742. ACM, 2015. http://arxiv.org/abs/1412.7994.

[2] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Algebraic algorithms for LWE problems. *ACM Comm. Computer Algebra*, 49(2):62, 2015. https://eprint.iacr.org/2014/1018.

[3] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 327–343. USENIX Association, 2016. https://eprint.iacr.org/2015/1092.

[4] Jacob Alperin-Sheriff and Daniel Apon. Dimension-preserving reductions from LWE to LWR. *IACR Cryptology ePrint Archive*, 2016:589, 2016. https://eprint.iacr.org/2016/589.

[5] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited – new reduction, properties and applications. In Canetti and Garay [19], pages 57–74. https://eprint.iacr.org/2013/098.

[6] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011. https://users.cs.duke.edu/~rongge/LPSN.pdf.

[7] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012. https://eprint.iacr.org/2011/401.

[8] Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, and Christine van Vredendaal. Short generators without quantum computers: The case of multiquadratics. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 27–59, 2017. https://multiquad.cr.yp.to.

[9] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Krauthgamer [40], pages 10–24. https://eprint.iacr.org/2015/1128.

[10] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006. https://cr.yp.to/papers.html#curve25519.

[11] Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2013. https://cr.yp.to/papers.html#nonuniform.

[12] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Krauthgamer [40], pages 893–902. http://fangsong.info/files/pubs/BS_SODA16.pdf.

[13] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In Eyal Kushilevitz and Tal

Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2016. https://eprint.iacr.org/2015/769.

[14] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In *ACM Conference on Computer and Communications Security*, pages 1006–1018. ACM, 2016. https://eprint.iacr.org/2016/659.

[15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society, 2015. https://eprint.iacr.org/2014/599.

[16] Richard P. Brent and H. T. Kung. The area-time complexity of binary multiplication. *J. ACM*, 28(3):521–534, 1981. http://maths-people.anu.edu.au/~brent/pd/rpb055.pdf.

[17] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, Gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 323–345. Springer, 2016. https://eprint.iacr.org/2016/300.

[18] Peter Campbell, Michael Groves, and Dan Shepherd. Soliloquy: a cautionary tale, 2014. http://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07_Groves_Annex.pdf.

[19] Ran Canetti and Juan A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*. Springer, 2013.

[20] Hao Chen, Kristin Lauter, and Katherine E. Stange. Vulnerable Galois RLWE families and improved attacks. *IACR Cryptology ePrint Archive*, 2016. https://eprint.iacr.org/2016/193.

[21] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.

[22] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates (full version), 2011. http://www.di.ens.fr/~ychen/research/Full_BKZ.pdf.

[23] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 559–585. Springer, 2016. https://eprint.iacr.org/2015/313.

[24] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short Stickelberger class relations and application to Ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 324–348, 2017. https://eprint.iacr.org/2016/885.

[25] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2003. https://eprint.iacr.org/2002/174.

[26] Jintai Ding. Solving LWE problem with bounded errors in polynomial time. *IACR Cryptology ePrint Archive*, 2010:558, 2010. https://eprint.iacr.org/2010/558.

[27] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Canetti and Garay [19], pages 40–56. https://eprint.iacr.org/2013/383.

[28] Kirsten Eisenträger, Sean Hallgren, and Kristin E. Lauter. Weak instances of PLWE. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2014. https://eprint.iacr.org/2014/784.

[29] Ulrich Fincke and Michael Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–471, 1985. http://www.ams.org/journals/mcom/1985-44-170/S0025-5718-1985-0777278-8/S0025-5718-1985-0777278-8.pdf.

[30] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on mdpc with cca security using decoding errors. Cryptology ePrint Archive, Report 2016/858, 2016. https://eprint.iacr.org/2016/858.

[31] Philip S. Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 437–455, 2009.

[32] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. *IACR Cryptology ePrint Archive*, 2015. https://eprint.iacr.org/2015/708.

[33] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[34] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, 2007. https://www.iacr.org/archive/crypto2007/46220150/46220150.pdf.

[35] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246. Springer, 2003. http://www.di.ens.fr/~pointche/Documents/Papers/2003_crypto.pdf.

[36] Nick Howgrave-Graham, Joseph H Silverman, and William Whyte. A meet-in-the-middle attack on an NTRU private key. Technical report, NTRU Cryptosystems, June 2003. Report, 2003. https://www.securityinnovation.com/uploads/Crypto/NTRUTech004v2.pdf.

[37] Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3, 2005. https://eprint.iacr.org/2005/045.

[38] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 193–206, New York, NY, USA, 1983. ACM.

[39] Paul Kirchner and Pierre-Alain Fouque. Comparison between subfield and straightforward attacks on NTRU. Cryptology ePrint Archive, Report 2016/717, 2016. https://eprint.iacr.org/2016/717.

[40] Robert Krauthgamer, editor. *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. SIAM, 2016.

[41] Po-Chun Kuo, Michael Schneider, Özgür Dagdelen, Jan Reichelt, Johannes A. Buchmann, Chen-Mou Cheng, and Bo-Yin Yang. Extreme enumeration on GPU and in clouds: How many dollars you need to break SVP challenges. In Bart Preneel and

Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 176–191. Springer, 2011. http://www.iis.sinica.edu.tw/papers/byyang/12158-F.pdf.

[42] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2015. https://eprint.iacr.org/2014/744.pdf.

[43] Vadim Lyubashevsky. Future directions in lattice cryptography (talk slides), 2016. http://troll.iis.sinica.edu.tw/pkc16/slides/Invited_Talk_II--Directions_in_Practical_Lattice_Cryptography.pptx.

[44] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013. https://eprint.iacr.org/2012/230.

[45] Artur Mariano, Christian H. Bischof, and Thijs Laarhoven. Parallel (probable) lock-free hash sieve: A practical sieving algorithm for the SVP. In *44th International Conference on Parallel Processing, ICPP 2015, Beijing, China, September 1-4, 2015*, pages 590–599. IEEE Computer Society, 2015. https://eprint.iacr.org/2015/041.

[46] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. Revisiting SSL/TLS implementations: New Bleichenbacher side channels and attacks. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 733–748. USENIX Association, 2014. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/meyer.

[47] Daniele Micciancio and Michael Walter. Fast lattice point enumeration with minimal overhead. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 276–294. SIAM, 2015. https://eprint.iacr.org/2014/569.

[48] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12(1):1–28, 1999. http://people.scs.carleton.ca/~paulv/papers/JoC97.pdf.

[49] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009. https://eprint.iacr.org/2008/481.

[50] Chris Peikert. "A useful fact about Ring-LWE that should be known better: it is *at least as hard* to break as NTRU, and likely strictly harder. 1/" (tweet), 2017. http://archive.is/B9KEW.

[51] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. In *CCS*, pages 1843–1855. ACM, 2017. https://eprint.iacr.org/2017/490.

[52] Michael Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *SIGSAM Bull.*, 15(1):37–44, February 1981.

[53] Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 2013. https://www.ei.rub.de/media/sh/veroeffentlichungen/2013/08/14/lwe_encrypt.pdf.

[54] Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009. https://eprint.iacr.org/2009/605.

[55] Halvor Sakshaug. Security analysis of the NTRUEncrypt public key encryption scheme, 2007. brage.bibsys.no/xmlui/bitstream/handle/11250/258846/426901_FULLTEXT01.pdf.

[56] Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *STACS*, volume 2607 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2003. http://www.math.uni-frankfurt.de/~dmst/research/papers/schnorr.random_sampling.2003.ps.

[57] Claus-Peter Schnorr and Adi Shamir. An optimal sorting algorithm for mesh connected computers. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 255–263. ACM, 1986.

[58] Victor Shoup. A proposal for an ISO standard for public key encryption. *IACR Cryptology ePrint Archive*, 2001. https://eprint.iacr.org/2001/112.

[59] Victor Shoup. OAEP reconsidered. *J. Cryptology*, 15(4):223–249, 2002. https://eprint.iacr.org/2000/060.

[60] Martijn Stam. A key encapsulation mechanism for NTRU. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 410–427. Springer, 2005.

[61] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 27–47. Springer, 2011. https://www.iacr.org/archive/eurocrypt2011/66320027/66320027.pdf.

[62] Christine van Vredendaal. Reduced memory meet-in-the-middle attack against the NTRU private key. *LMS Journal of Computation and Mathematics*, 19(A):43–57, 001 2016. https://eprint.iacr.org/2016/177.

[63] Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates, 2016. https://eprint.iacr.org/2016/733.

A Statements

These statements "must be mailed to Dustin Moody, Information Technology Laboratory, Attention: Post-Quantum Cryptographic Algorithm Submissions, 100 Bureau Drive – Stop 8930, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, or can be given to NIST at the first PQC Standardization Conference (see Section 5.C)."

First blank in submitter statement: full name. Second blank: full postal address. Third, fourth, and fifth blanks: name of cryptosystem. Sixth and seventh blanks: describe and enumerate or state "none" if applicable.

First blank in patent statement: full name. Second blank: full postal address. Third blank: enumerate. Fourth blank: name of cryptosystem.

First blank in implementor statement: full name. Second blank: full postal address. Third blank: full name of the owner.

A.1 Statement by Each Submitter

I, _____, of _____, do hereby declare that the cryptosystem, reference implementation, or optimized implementations that I have submitted, known as _____, is my own original work, or if submitted jointly with others, is the original work of the joint submitters. I further declare that (check one):

- *I do not hold and do not intend to hold any patent or patent application with a claim which may cover the cryptosystem, reference implementation, or optimized implementations that I have submitted, known as _____ OR (check one or both of the following):*
  - *to the best of my knowledge, the practice of the cryptosystem, reference implementation, or optimized implementations that I have submitted, known as _____ may be covered by the following U.S. and/or foreign patents:*
    _____
  - *I do hereby declare that, to the best of my knowledge, the following pending U.S. and/or foreign patent applications may cover the practice of my submitted cryptosystem, reference implementation or optimized implementations:*
    _____

I do hereby acknowledge and agree that my submitted cryptosystem will be provided to the public for review and will be evaluated by NIST, and that it might not be selected for standardization by NIST. I further acknowledge that I will not receive financial or other compensation from the U.S. Government for my submission. I certify that, to the best of my knowledge, I have fully disclosed all patents and patent applications which may cover my cryptosystem, reference implementation or optimized implementations. I also acknowledge and agree that the U.S. Government may, during the public review and the evaluation process, and, if my submitted cryptosystem is selected for standardization, during the lifetime of the standard, modify my submitted cryptosystem's specifications (e.g., to protect against a newly discovered vulnerability).

I acknowledge that NIST will announce any selected cryptosystem(s) and proceed to publish the draft standards for public comment.

I do hereby agree to provide the statements required by Sections 2.D.2 and 2.D.3, below, for any patent or patent application identified to cover the practice of my cryptosystem, reference implementation or optimized implementations and the right to use such implementations for the purposes of the public review and evaluation process.

I acknowledge that, during the post-quantum algorithm evaluation process, NIST may remove my cryptosystem from consideration for standardization. If my cryptosystem (or the derived cryptosystem) is removed from consideration for standardization or withdrawn from consideration by all submitter(s) and owner(s), I understand that rights granted and assurances made under Sections 2.D.1, 2.D.2 and 2.D.3, including use rights of the reference and optimized implementations, may be withdrawn by the submitter(s) and owner(s), as appropriate.

Signed:

Title:

Date:

Place:

A.2 Statement by Patent (and Patent Application) Owner(s)

If there are any patents (or patent applications) identified by the submitter, including those held by the submitter, the following statement must be signed by each and every owner, or each owner's authorized representative, of each patent and patent application identified.

I, _____, of _____, am the owner or authorized representative of the owner (print full name, if different than the signer) of the following patent(s) and/or patent application(s): _____

and do hereby commit and agree to grant to any interested party on a worldwide basis, if the cryptosystem known as _____ is selected for standardization, in consideration of its evaluation and selection by NIST, a non-exclusive license for the purpose of implementing the standard (check one):

- *without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination, OR*
- *under reasonable terms and conditions that are demonstrably free of any unfair discrimination.*

I further do hereby commit and agree to license such party on the same basis with respect to any other patent application or patent hereafter granted to me, or owned or controlled by me, that is or may be necessary for the purpose of implementing the standard.

I further do hereby commit and agree that I will include, in any documents transferring ownership of each patent and patent application, provisions to ensure that the commitments and assurances made by me are binding on the transferee and any future transferee.

I further do hereby commit and agree that these commitments and assurances are intended by me to be binding on successors-in-interest of each patent and patent application, regardless of whether such provisions are included in the relevant transfer documents.

I further do hereby grant to the U.S. Government, during the public review and the evaluation process, and during the lifetime of the standard, a nonexclusive, nontransferrable, irrevocable, paid-up worldwide license solely for the purpose of modifying my submitted cryptosystem's specifications (e.g., to protect against a newly discovered vulnerability) for incorporation into the standard.

Signed:

Title:

Date:

Place:

A.3 Statement by Reference/Optimized Implementations' Owner(s)

The following must also be included:

I, _____, _____, am the owner or authorized representative of the owner _____ of the submitted reference implementation and optimized implementations and hereby grant the U.S. Government and any interested party the right to reproduce, prepare derivative works based upon, distribute copies of, and display such implementations for the purposes of the post-quantum algorithm public review and evaluation process, and implementation if the corresponding cryptosystem is selected for standardization and as a standard, notwithstanding that the implementations may be copyrighted or copyrightable.

Signed:

Title:

Date:

Place: