NIST PQ Submission: NTRUEncrypt A lattice based encryption algorithm

Cong Chen[2], Jeffrey Hoffstein[1], William Whyte[2], and Zhenfei Zhang[2]

[1] Brown University, Providence RI, USA, jhoff@math.brown.edu
[2] OnBoard Security, Wilmington MA, USA,
{cchen,wwhyte,zzhang}@onboardsecurity.com

1 Cover Sheet

This is an overview document of the NTRU lattice-based cryptosystem for submission to the NIST post-quantum cryptography call for standardization. The submitted cryptosystem consists of :

- ntru-pke: a public key encryption (PKE) scheme based on the "*original NTRU*" encryption algorithm by Hoffstein, Pipher and Silverman [26] with parameter sets derived from a recent revision [24], that achieves CCA-2 security via NAEP transformation [30];
- ntru-kem: a key encapsulation mechanism (KEM) using the above public key encryption algorithm;
- ss-ntru-pke: a public key encryption scheme based on the provable secure NTRU encryption algorithm [36] that achieves CCA-2 security via NAEP transformation [30];
- ss-ntru-kem: a key encapsulation mechanism (KEM) using the above public key encryption algorithm.

This documents addresses the following requirements:

- **Specifications**
- **Performance analysis**
- **A statement of the advantages and limitations**
- **Cover sheet**
- **Reference implementation**
- **Security analysis**
- **Statement of IPR**

Submission information:

- **Principal submitter:** Zhenfei Zhang, zzhang@onboardsecurity.com, Onboard Security, 187 Ballardvale St. Suite A202, Wilmington, MA, 01887, U.S.
- **Auxiliary submitters:** Chen Cong, Jeffrey Hoffstein and William Whyte.
- **Inventors of the cryptosystem:** Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte and Zhenfei Zhang.

– **Name of the owner of the cryptosystem:** Onboard Security Inc.
– **Backup point of contact:** William Whyte, wwhyte@onboardsecurity.com, Onboard Security, 187 Ballardvale St. Suite A202, Wilmington, MA, 01887, U.S.

The following academic papers contain cryptographic designs, hardness results, security analysis and parameter derivations related to the submitted cryptosystem.

– *NTRU, A ring-based public key cryptosystem*, 1998.
– *NAEP: provable security in the presence of decryption failures*, 2003.
– *A hybrid lattice-reduction and meet-in-the-middle attack against NTRU*, 2007.
– *Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches*, 2008.
– *Making NTRU as secure as worst-case problems over ideal lattices*, 2011.
– *Choosing parameters for NTRUEncrypt*, 2017.

Additional information related to implementations, such as public key/private key encodings, conversions, etc. can be found in the additional supporting document:

– *Efficient Embedded Security Standard (EESS) #1.* Version 3.3, 2017.

2 Algorithm Specifications

2.1 Notation

We use lower case bold letters for vectors, upper case bold letters for matrices. For a polynomial $f(x) = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1}$, we denote its vector form by $\mathbf{f} := \langle f_0, f_1, \ldots, f_{n-1}\rangle$. We sometimes abuse the notation of vector and polynomial when there is no ambiguity. For a polynomial/vector $\mathbf{f}$, the norms are $\|\mathbf{f}\| := \sqrt{\sum_{i=0}^{n-1} f_i^2}$ and $\|\mathbf{f}\|_\infty := \max(|f_i|)$.

We often use the polynomial rings $\mathcal{R}_q := \mathbb{Z}[x]/F(x)$ with $F(x) = x^n \pm 1$. A cyclic rotated matrix of a polynomial $f(x)$ over the ring $\mathcal{R}_q$ is a matrix $\mathbf{M} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$ with $\mathbf{f}_i = f(x)x^{i-1} \bmod F(x)$. If $F(x) = x^n - 1$ it is literally cyclic, and it is close to cyclic, up to signs, if $F(x) = x^n + 1$.

For a real a, we let $\lfloor a \rceil$ denote the closet integer to a. For an integer a, we use $[a]_q$ to denote $a \bmod q$; $\lfloor a \rfloor_p := (a - [a]_p)/p$ for the operation of rounding a to the closest multiple of p. Modular operations are center lifted, for example $a \bmod q$ returns an integer within $-q/2$ and $q/2$. These notations are also extended to vectors and matrices.

We set the notation:

$$\begin{aligned}
\mathcal{B}_N &= \{\text{binary polynomials}\} \\
\mathcal{T}_N &= \{\text{trinary polynomials}\} \\
\mathcal{T}_N(d,e) &= \left\{ \begin{array}{l} \text{trinary polynomials with exactly} \\ d \text{ ones and } e \text{ minus ones} \end{array} \right\}
\end{aligned}$$

If N is fixed we will write $\mathcal{B}$, $\mathcal{T}$ and $\mathcal{T}(d,e)$ instead.

2.2 NTRU and lattices

A lattice $\mathcal{L}$ is a discrete sub-group of $\mathbb{R}^n$, or equivalently, the set of all the integral combinations of $d \leq n$ linearly independent vectors over $\mathbb{R}$:

$$\mathcal{L} \coloneqq \mathbb{Z}\mathbf{b}_1 + \mathbb{Z}\mathbf{b}_2 + \cdots + \mathbb{Z}\mathbf{b}_d, \mathbf{b}_i \in \mathbb{R}^n.$$

$\mathbf{B} \coloneqq (\mathbf{b}_1, \ldots, \mathbf{b}_d)^T$ is called a basis of $\mathcal{L}$.

Definition 1 (γ-SVP and uSVP). *Given a lattice $\mathcal{L}$, finding a vector that is no longer than $\gamma \cdot \lambda_1(\mathcal{L})$ is called the* approximate shortest vector problem (γ-SVP)*, where λ_1 is the first minima, i.e, the length of the shortest vector, of the lattice.*

Given a particular lattice $\mathcal{L}$, where there exists a unique shortest non-zero vector, finding this vector is called the unique shortest vector problem.

Let $f(x)$, $g(x)$ and $h(x)$ be 3 polynomials in $\mathcal{R}_q$, where $f(x)$ and $g(x)$ have very small coefficients; $h(x) = p^{-1}g(x)f^{-1}(x)$. We express by $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ the vector form of the polynomials. Also let $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ be the matrix obtained from nega-cyclic rotations. The NTRU lattice with regard to h is defined as

$$\mathcal{L}_h = \{(u, v) \in \mathcal{R}_q^2 : uh = v\}$$

or rather, the vector/matrix form:

$$\mathcal{L}_h = \{(\mathbf{u}, \mathbf{v}) : \mathbf{uH} = \mathbf{v} \bmod q\}$$

where there exists a public basis $\mathbf{P} = \begin{bmatrix} 0 & q\mathbf{I}_N \\ \mathbf{I}_N & \mathbf{H} \end{bmatrix}$ and a secret generator $[p\mathbf{F}|\mathbf{G}]$.

Definition 2 (NTRU assumption). *Given $\mathbf{h}$, it is hard to find $\mathbf{f}$ and $\mathbf{g}$.*

The NTRU assumption can be reduced to the uSVP for the NTRU lattice.

2.3 Auxiliary functions

Let us first define some auxiliary functions. Those functions are the building blocks for our algorithm. We give a generic description for those functions. Implementers may choose to use better (more secure or more efficient) instantiations when and if they are available. Also note that Gaussian samplers are only used by ss-ntru-pke and ss-ntru-kem.

Hash function. Through out the paper we will use HASH to denote a cryptographic secure hash function that takes arbitrary input length and outputs a binary string with arbitrary length. In our submission, we use SHA-512 for such an instantiation.

Seed expansion function. We will use a Salsa20 [10] based pseudo-random number generator as the seed expansion function. Salsa20 is a fast and well accepted stream cipher. It is up to 5 times faster than the AES based solutions through our benchmark on computers that do not have AES-NI instructions. We remark that although we did not use the AES-based seed expansion function provided by NIST for efficiency reasons, such a modification can be made quite easily.

Trinary Polynomial Generation function. This implementation require a function that samples uniformly from $\mathcal{B}$, $\mathcal{T}$ and $\mathcal{T}(d, e)$. This function can be build deterministically via a *seed* and a hash function.

Discrete Gaussian sampler (DGS). Input a dimension N and a standard deviation σ it outputs a discrete Gaussian distributed vector. In this implementation we use the Box-Muller approach. We remark that there are better samplers in terms of efficiency or security. We leave the investigation of those samplers to future work.

Deterministic Discrete Gaussian sampler (DDGS). Input a dimension N, a standard deviation σ and a seed s, it deterministically outputs a discrete Gaussian distributed vector.

2.4 The schemes

We sketch the algorithms related to our proposed schemes. For simplicity and clearness of the presentation, we omit minor details in this high level description. Those include, for example, checking that the length of the message is valid, encoding/packing ring elements into binary strings and vice versa, etc. For those details we refer the readers to the implantation specification document submitted along with this document.

The ntru-pke scheme . The ntru-pke schemes use Algorithms 1, 2 and 3.

In an NTRU cryptosystem, $\mathbf{f}$ (and $\mathbf{g}$, if required) are the private keys, while $\mathbf{h}$ is the public key. Those keys can be generated via algorithm 1. Note that we use the classical NTRU flat form (non-product form, cf. [24]) keys with a pre-fixed number of $+1$s and -1s.

Algorithm 1 ntru-pke.KeyGen

Input: A parameter set Param $= \{N, p, q, d\}$ and a *seed*.

1: Instantiate Sampler with $\mathcal{T}(d+1, d)$ and *seed*;
2: $\mathbf{f} \leftarrow$ Sampler
3: **if** $\mathbf{f}$ is not invertible mod q **then** go to step 2 **end if**
4: $\mathbf{g} \leftarrow$ Sampler
5: $\mathbf{h} = \mathbf{g}/(p\mathbf{f} + 1) \bmod q$

Output: Public key $\mathbf{h}$ and secret key $(p\mathbf{f}, \mathbf{g})$

Algorithm 2 ntru-pke.ENCRYPT

Input: Public key $\mathbf{h}$, message msg of length $mlen$, a parameter set PARAM and a *seed*.

$\mathbf{m} = \text{Pad}(msg, seed)$
$rseed = \text{HASH}(\mathbf{m}|\mathbf{h})$
Instantiate Sampler with $\mathcal{T}$ and $rseed$;
$\mathbf{r} \leftarrow$ Sampler
$\mathbf{t} = \mathbf{r} * \mathbf{h}$
$tseed = \text{HASH}(\mathbf{t})$
Instantiate Sampler with $\mathcal{T}$ and $tseed$;
$\mathbf{m}_{mask} \leftarrow$ Sampler
$\mathbf{m}' = \mathbf{m} - \mathbf{m}_{mask} \pmod p$
$\mathbf{c} = \mathbf{t} + \mathbf{m}'$

Output: Ciphertext $\mathbf{c}$

The encryption algorithm in Algorithm 2 uses a padding method to deal with potential insufficient entropy in a message. Assuming the message length is valid and less than $(N - 173)$ bits , the padding algorithm works as follows:

1. Convert msg into a bit string. Each bit forms a binary coefficient for the lower part of the polynomial m, starting from coefficient 0.
2. The last 167 coefficients of $m(x)$ are randomly chosen from $\{-1, 0, 1\}$ (with an input seed). This gives over 256 bits entropy.
3. The length of msg is converted into an 8 bit binary string, and forms the last 173 to 168 coefficients of $m(x)$.

Algorithm 3 ntru-pke.DECRYPT

Input: Secret key $\mathbf{f}$, public key $\mathbf{h}$, ciphertext $\mathbf{c}$, and a parameter set PARAM.

$\mathbf{m}' = \mathbf{f} * \mathbf{c} \pmod p$
$\mathbf{t} = \mathbf{c} - \mathbf{m}$
$tseed = \text{HASH}(\mathbf{t})$
Instantiate Sampler with $\mathcal{T}$ and $tseed$;
$\mathbf{m}_{mask} =$ Sampler
$\mathbf{m} = \mathbf{m}' + \mathbf{m}_{mask} \pmod p$
$rseed = \text{HASH}(\mathbf{m}|\mathbf{h})$
Instantiate Sampler with $\mathcal{T}$ and $rseed$;
$\mathbf{r} \leftarrow$ Sampler
$msg, mlen = \text{Extract}(\mathbf{m})$
if $p \cdot \mathbf{r} * \mathbf{h} = \mathbf{t}$ **then**
    $result = msg, mlen$
else
    $result = \perp$
end if

Output: $result$

The Extract() operation in Algorithm 3 is the inverse of Pad() so we omit the details. It outputs a message m and its length $mlen$.

Remark 1. Since NIST's API does not have an input field for the public key $\mathbf{h}$, we have encoded the public key in the secret key to make the algorithm compatible with the existing API.

The ntru-kem algorithms We recommend that ntru-kem to be used for ephemeral key establishments via the following algorithms. ntru-kem uses a same key generation algorithm as ntru-pke, namely, Algorithm 2. Here we present the encapsulation and decapsulation algorithms in Algorithms 4 and 5.

In a nutshell, the ntru-kem scheme uses an ntru-pke scheme to transport an encapsulated secret, and uses both this secret and the public key to derive a shared secret via a secure Key Derivation Function (KDF).

Algorithm 4 ntru-kem.ENCAP

Input: Public key $\mathbf{h}$, a parameter set PARAM, and a *seed*

1: encaped_secret $\leftarrow \{0,1\}^{8\times \text{CRYPTO\_BYTES}}$
2: $\mathbf{c}$ =ntru-pke.ENCRYPT($\mathbf{h}$, encaped_secret, CRYPTO_BYTES, PARAM, *seed*)
3: ss = KDF(encaped_secret, $\mathbf{h}$).

Output: A ciphertext $\mathbf{c}$ and the shared secret ss.

Algorithm 5 ntru-kem.DECAP

Input: Secret key $\mathbf{f}$ and a parameter set PARAM

1: encaped_secret =ntru-pke.DECRYPT($\mathbf{f}, \mathbf{h}, \mathbf{c}$, PARAM);
2: ss = KDF(encaped_secret, $\mathbf{h}$).

Output: The shared secret ss.

The ss-ntru-pke algorithms The ss-ntru-pke schemes use Algorithms 6, 7 and 8.

Algorithm 6 ss-ntru-pke.KEYGEN

Input: Parameter sets PARAM = $\{N, p, q, \sigma\}$ and a *seed*

1: Instantiate Sampler with χ_σ^N and *seed*;
2: $\mathbf{f} \leftarrow$ Sampler, $\mathbf{g} \leftarrow$ Sampler;
3: $\mathbf{h} = \mathbf{g}/(p\mathbf{f}+1) \bmod q$

Output: Public key $\mathbf{h}$ and secret key $(p\mathbf{f}, \mathbf{g})$

ss-ntru-pke uses a similar key generation algorithm as ntru-pke. The major difference is that $\mathbf{f}$ and $\mathbf{g}$ are sampled from a Gaussian with deviation σ, rather than from $\mathcal{T}(d, d+1)$. In addition, since ss-ntru-pke works over the polynomial ring $\mathbb{Z}_q[x]/(x^N+1)$, where every element has an inverse, we are not required to check if $\mathbf{f}$ and $\mathbf{g}$ has an inverse.

Algorithm 7 ss-ntru-pke.ENCRYPT

Input: Public key $\mathbf{h}$, message msg of length $mlen$, PARAM and a *seed*

$\mathbf{m} = \text{Pad}(msg, seed)$
$rseed = \text{HASH}(\mathbf{m}|\mathbf{h})$
Instantiate Sampler with χ_σ^N and $rseed$;
$\mathbf{r} \leftarrow$ Sampler, $\mathbf{e} \leftarrow$ Sampler
$\mathbf{t} = p \cdot \mathbf{r} * \mathbf{h}$
$tseed = \text{HASH}(\mathbf{t})$
Instantiate Sampler with $\mathcal{B}$ and $tseed$;
$\mathbf{m}_{mask} \leftarrow$ Sampler
$\mathbf{m}' = \mathbf{m} - \mathbf{m}_{mask} \pmod p$
$\mathbf{c} = \mathbf{t} + p \cdot \mathbf{e} + \mathbf{m}'$

Output: Ciphertext $\mathbf{c}$

Algorithm 8 ss-ntru-pke.DECRYPT

Input: Secret key $\mathbf{f}$, public key $\mathbf{h}$, ciphertext $\mathbf{c}$, and a parameter PARAM.

$\mathbf{m}' = \mathbf{f} * \mathbf{c} \pmod p$
$\mathbf{t} = \mathbf{c} - \mathbf{m}$
$tseed = \text{HASH}(\mathbf{t})$
Instantiate Sampler with $\mathcal{B}$ and $tseed$;
$\mathbf{m}_{mask} \leftarrow$ Sampler
$\mathbf{m} = \mathbf{m}' + \mathbf{m}_{mask} \pmod p$
$rseed = \text{HASH}(\mathbf{m}|\mathbf{h})$
Instantiate Sampler with χ_σ^N and $rseed$;
$\mathbf{r} \leftarrow$ Sampler
$\mathbf{e} = p^{-1}(\mathbf{t} - \mathbf{r} * \mathbf{h})$
if $|\mathbf{e}|_\infty \geq \tau\sigma$ **then**
    $result = \perp$
else
    $result = \text{Extract}(\mathbf{m})$
end if

Output: $result$

The ss-ntru-kemalgorithms The algorithms for ss-ntru-kem are described in 9 and 10. It uses the same method as ntru-kem to convert a public key encryption scheme into a key encapsulation mechanism.

Algorithm 9 ss-ntru-kem.ENCAP

Input: Public key $\mathbf{h}$, message msg of length $mlen$, a parameter set Param and a $seed$

1: encaped_secret $\leftarrow \{0,1\}^{8\times \text{CRYPTO\_BYTES}}$
2: $\mathbf{c}$ =ss-ntru-pke.ENCRYPT($\mathbf{h}$, encaped_secret, CRYPTO_BYTES, PARAM, $seed$)
3: ss = KDF(encaped_secret, $\mathbf{h}$).

Output: A ciphertext $\mathbf{c}$ and the shared secret ss.

Algorithm 10 ss-ntru-kem.DECAP

Input: Secret key f and a parameter set PARAM

1: encaped_secret =ss-ntru-pke.DECRYPT($\mathbf{f}, \mathbf{h}, \mathbf{c}$, PARAM);
2: ss = KDF(encaped_secret, $\mathbf{h}$).

Output: The shared secret ss.

3 Design Rationale

3.1 Hardness assumption

Overview We first give an overview of the hardness assumptions in this proposal.

- For ntru-pke and ntru-kem schemes:
  - The CPA security is based on the NTRU assumption;
  - We use the NAEP transformation [31] to convert the scheme into a CCA-2 secure encryption scheme.
- For ss-ntru-pke and ss-ntru-kem schemes:
  - The CPA security is based on the ring learning with error (R-LWE) problem [36];
  - We use the NAEP transformation [31] to convert the scheme into a CCA-2 secure encryption scheme.

All the above problems and notions are well studied in the literature, except perhaps for the NAEP transform. Therefore we give a high level description of NAEP and show its connections to the well-known Fujisaki-Okamoto transform.

NAEP transform In [17], the authors proposed a generic method to transform a CPA secure encryption algorithm into a CCA-2 secure version. This method is usually referred to as Fujisaki-Okamoto transform. At a high level, it works as follows. During the encryption, one first chooses a salt, and appends it to the message. Then one hashes the appended message into a random element that is to be used in the encryption. During the decryption, after one has recovered the padded message, one re-encrypts the message with the same salt, and compares the resulting ciphertext with the received one. If those two does not match, abort the decryption.

The NEAP [31] transform that we use in Algorithms 2 and 7 is similar to the above Fujisaki-Okamoto transform. In addition, it builds an additional all-or-nothing mask which is also derived from the hash of the padded message. With

this mask, one will either recover all the coefficients of the message polynomial, or no coefficient at all. It also seals the information leakage of $m(1)$ when a polynomial ring of the form $x^N - 1$ is used.

3.2 Parameters

We present our parameter sets in Table 1 and the macros related to NIST's APIs in Table 2. We estimate that

- NTRU-443 provides 128 bits classical security and 84 bits quantum security;
- NTRU-743 provides 256 bits classical security and 159 bits quantum security;
- NTRU-1024 provides $\gg$256 bits classical security and 198 bits quantum security.

The details of the above estimations shall be presented in the next subsection.

Table 1. Parameters

Param	N	q	p	$\mathcal{R}$	d	σ	MaxMSGLen
NTRU-443	443	2048	3	$\frac{\mathbb{Z}_q[x]}{x^N-1}$	143	N/A	33 bytes
NTRU-743	743	2048	3	$\frac{\mathbb{Z}_q[x]}{x^N-1}$	247	N/A	73 bytes
NTRU-1024	1024	$2^{30}+2^{13}+1$	2	$\frac{\mathbb{Z}_q[x]}{x^N+1}$	N/A	724	95 bytes

Table 2. MACRO definitions for NIST's API

Param	NTRU-443		NTRU-743		NTRU-1024	
Scheme	ntru-pke	ntru-kem	ntru-pke	ntru-kem	ss-ntru-pke	ss-ntru-kem
CRYPTO_SECRETKEYBYTES	701		1173		8194	
CRYPTO_PUBLICKEYBYTES	611		1023		4097	
CRYPTO_BYTES	32		48		48	
CRYPTO_CIPHERTEXTBYTES	611		1023		4097	

We address NIST's required security levels as follows:

- Level 1, equivalent to a128-bit block cipher: use ntru-pke and ntru-kem with parameter set NTRU-443 or NTRU-743;
- Level 2, equivalent to a 256-bit hash function: use ntru-pke and ntru-kem with parameter set NTRU-743;
- Level 3, equivalent to a 192-bit block cipher: use ntru-pke and ntru-kem with parameter set NTRU-743;
- Level 4, equivalent to a 384-bit hash function: use ntru-pke and ntru-kem with parameter setNTRU-743, or (for extremely conservative purpose) ss-ntru-pke and ss-ntru-kem with parameter set NTRU-1024;

– Level 5, equivalent to a 256-bit block cipher: use ntru-pke and ntru-kem with parameter set NTRU-743, or (for extremely conservative purpose) ss-ntru-pke and ss-ntru-kem with parameter set NTRU-1024.

3.3 Best known attacks

Summary In this evaluation, we will

1. follow the original BKZ 2.0 analysis [13] with the extreme pruning method to estimate the **classical security**;
2. follow the new analysis in [6] using BKZ 2.0 with quantum sieving to estimate the **quantum security**.

For completeness, we also give the analysis result of

3. the new analysis in [6] using BKZ 2.0 with classical sieving.

However, we will **not** use this result to estimate the classical security, due to the excessive space requirement. We will give more details in the following sections.

Table 3. Best Known attacks and their costs

Param	BKZ + Enum		BKZ + Sieving		BKZ + QSieving	
	uSVP	Hybrid	uSVP	Hybrid	uSVP	Hybrid
NTRU-443	189	**128**	93	89	85	**84**
NTRU-743	443	**268**	176	173	**159**	163
NTRU-1024	**590**	805	218	316	**198**	287

Lattice attacks For an NTRUEncrypt public key polynomial $\mathbf{h}$, let $\mathbf{H}$ be the matrix whose row vectors are the cyclic rotation of $\mathbf{h}$. Then the NTRU lattice associated with $\mathbf{h}$ uses a basis

$$\mathbf{B} = \begin{bmatrix} q\mathbf{I}_N & 0 \\ \mathbf{H} & \mathbf{I}_N \end{bmatrix}$$

where $\mathbf{I}_N$ is an N-dimensional identity matrix. With in this NTRU lattice, there exist unique shortest vectors, namely, the vector form of $\langle \mathbf{f}, \mathbf{g} \rangle$ and its cyclic rotations.

This attack was firstly presented in the original NTRU paper [27] circulated during the rump session of Crypto'96. It was later observed in [15] that one does not necessarily need to find the exact secret key to be able to decrypt. An attack is successful if the attacker can locate any vectors in this lattice that are sufficiently small (such as a cyclic rotation of the secret key).

It has been shown in [18] that the ability to locate a unique shortest vector in a lattice depends on the root Hermite factor of the lattice, which is the n-th root of

$$\frac{\text{Gaussian expected length}}{l_2 \text{ norm of the target vector}}$$

where n is the dimension of the lattice.

Here, we give an estimation of the root Hermite factor for the proposed parameter set. This lattice has a dimension of $2N$. The Gaussian expected length of the shortest vector in this lattice is

$$\sqrt{qN/\pi e},$$

while the l_2 norm of the target vectors are $\|\mathbf{f}, \mathbf{g}\|_2$. This gives the root Hermite factor of the lattice as

$$\left(\frac{\sqrt{Nq/\pi e}}{\|\mathbf{f}, \mathbf{g}\|_2}\right)^{\frac{1}{2N}}.$$

For ntru-pke and ntru-kem we have $\|\mathbf{f}, \mathbf{g}\|_2 \approx \sqrt{4N/3}$, while for ss-ntru-pke and ss-ntru-kem we have $\|\mathbf{f}, \mathbf{g}\|_2 \approx \sqrt{2N}\sigma$. The table below gives the root Hermite factor of corresponding parameter sets.

Table 4. Root Hermite Factor for NTRU lattices

N	q	$\|\mathbf{f}, \mathbf{g}\|_2$	rhf
443	2048	$\sqrt{572} \approx 23.92$	1.0030
743	2048	$\sqrt{988} \approx 31.43$	1.0020
1024	$2^{30} + 2^{13} + 1$	32764.5	1.0011

It was believed that the current technique of BKZ 2.0 [13] is only able to find a short vector with a root Hermite factor of at least 1.005. However, in [6], the authors give a conservative analysis of the cost of BKZ 2.0 reduction. As pointed out by the authors themselves, those estimations are very optimistic about the abilities of an attacker. In particular, unlike the analysis of BKZ 2.0 [13], where the cost of shortest vector subroutines is estimated via the cost of enumeration with extremely pruning [19], this analysis assumes that for large dimensional lattices, shortest vector problems can be solved very efficiently using heuristic sieving algorithms, ignoring the sub-exponential to exponential requirement of space.

Giving a few details, the best known classical and quantum sieving algorithms have time costs of $2^{0.292n}$ and $2^{0.265n}$, respectively [7]. The best plausible quantum short vector problem solver costs more than $2^{0.2075n}$ since this is the space required to store the list of vectors. In practice, sieving tends to process much slower than enumeration techniques. Moreover, sieving algorithms require a similar level of space complexity (exponential in n), while the space requirement of enumeration techniques is polynomial.

For the sake of completeness, we present the estimated cost of BKZ with classical and quantum sieving algorithms, following the methodology of [6]. It is easy to see that the space requirement for classical sieving algorithms is far from practical. For example, it is estimated that the world's storage capacity is around 295 exabytes $\approx 2^{68}$ bits [1]; and the number of atoms in the whole earth is around $10^{49} \approx 2^{162}$ [2]. Thus we do not use BKZ with classical sieving to estimate the classical security of our parameters. Nonetheless, we do use BKZ with quantum sieving algorithms to estimate the quantum security, in accounting for unknown effects on data storages with quantum computers.

Table 5. Lattice strength following analysis of [6]

N	m	b	Known Classical	Known Quantum	Best Plausible	Space Requirement
443	390	321	93	85	66	$> 2^{66}$
743	613	603	176	159	125	$> 2^{125}$
1024	1870	747	218	198	155	$> 2^{155}$

m: the number of used samples
b: block size for BKZ 2.0
Known Classical: using the best known classical SVP solver
Known Quantum: using the best known quantum SVP solver
Best Plausible: using a best plausible quantum SVP solver
Space Requirement: requirement for all 3 sieving algorithms

Search attack For NTRU with trinary keys, since the secret keys are trinary polynomials with df number of 1s and -1s, the search space for the secret key is $\binom{N}{df,df}/N$. For example, with parameter set NTRU-743, we have 2^{1158} candidates. (The factor $1/N$ comes from the fact that an attacker can guess any of N cyclic rotations of the secret key, rather than just the secret key itself.) We remark that this key space for our parameter set is considerably larger than that in [25] due to the switch from product form polynomials to flat form polynomials. This is sufficient even with the presence of meet-in-the-middle attacks [28] and quantum attacks using Grover's algorithm [21].

Hybrid attack The previous best known attack against NTRU, prior to the BKZ with quantum sieving analysis [6], was the hybrid attack [29] which is a hybrid of a lattice attack and a meet-in-the-middle search attack.

The rough idea is as follows. One first chooses $N_1 < N$ and extracts a block, $\mathbf{B}_1$, of $2N_1 \times 2N_1$ coefficients from the center of the matrix $\mathbf{B}$. The rows of $\mathbf{B}_1$ are taken to generate a lattice $\mathcal{L}_1$.

$$\left(\begin{array}{c|c} q\mathbf{I}_N & 0 \\ \hline \mathbf{H} & \mathbf{I}_N \end{array}\right) = \left(\begin{array}{c|c|c} q\mathbf{I}_{r_1} & 0 & 0 \\ \hline * & \mathbf{B}_1 & 0 \\ \hline * & * & \mathbf{I}_{r_2} \end{array}\right) \tag{1}$$

A lattice reduction algorithm is applied to find a unimodular transformation, $\mathbf{U}'$, such that $\mathbf{U}'\mathbf{B}_1$ is reduced, and an orthogonal transformation, $\mathbf{Y}'$, is computed such that $\mathbf{U}'\mathbf{B}_1\mathbf{Y}' = \mathbf{T}'$ is in lower triangular form. These transformations are applied to the original basis to produce a basis for an isomorphic lattice:

$$\mathbf{T} = \mathbf{UBY} = \left(\begin{array}{c|c|c} \mathbf{I}_{r_1} & 0 & 0 \\ \hline 0 & \mathbf{U}' & 0 \\ \hline 0 & 0 & \mathbf{I}_{r_2} \end{array}\right) \left(\begin{array}{c|c|c} q\mathbf{I}_{r_1} & 0 & 0 \\ \hline * & \mathbf{B}_1 & 0 \\ \hline * & * & \mathbf{I}_{r_2} \end{array}\right) \left(\begin{array}{c|c|c} \mathbf{I}_{r_1} & 0 & 0 \\ \hline 0 & \mathbf{Y}' & 0 \\ \hline 0 & 0 & \mathbf{I}_{r_2} \end{array}\right) = \left(\begin{array}{c|c|c} q\mathbf{I}_{r_1} & 0 & 0 \\ \hline * & \mathbf{T}' & 0 \\ \hline * & * & \mathbf{I}_{r_2} \end{array}\right) \quad (2)$$

Notice that $(\mathbf{g}, \mathbf{f})\mathbf{Y}$ is a short vector in the resulting lattice.

By a lemma of Furst and Kannan (Lemma 1 in [29]), if $\mathbf{y} = \mathbf{uT}+\mathbf{x}$ for vectors $\mathbf{u}$ and $\mathbf{x}$ in $\mathbb{Z}^{2N}$, and $-\mathbf{T}_{i,i}/2 < \mathbf{x}_i \leq \mathbf{T}_{i,i}/2$, then reducing $\mathbf{y}$ against $\mathbf{T}$ with Babai's nearest plane algorithm will yield $\mathbf{x}$ exactly. Thus if $\mathbf{v}$ is a shortest vector in $\mathcal{L}$ and $\mathbf{T}$ is well reduced, it is guaranteed that $\mathbf{v}$ can be found by enumerating candidates for its final $K = 2N - r_2$ coefficients. In the initial hybrid attack paper, this enumerating process was done via meet-in-the-middle attacks. To accommodate the quantum attack models, we will use Grover's algorithm to analize the cost of this enumeration.

Now we are ready to present the cost of the classical hybrid attack and compare it with solving directly the uSVP.

Table 6. BKZ with classical enumeration, hybrid attack vs. uSVP

	Hybrid Attack Parameters				uSVP	
Param	dim	β	K	Cost	β	cost
NTRU-443	620	241	161	> 128	321	> 189
NTRU-743	890	413	338	> 267	602	> 443
NTRU-1024	2047	953	140	> 811	747	> 590

Table 7. BKZ with quantum sieving, hybrid attack vs. uSVP

	Hybrid Attack Parameters				uSVP	
Param	dim	β	K	Cost	β	cost
NTRU-443	411	317	105	> 84	321	> 85
NTRU-743	645	616	205	> 163	602	> 159
NTRU-1024	2047	1901	50	> 289	747	> 198

Subfield attack Subfield attacks against NTRU have been considered in [8]. It was reported in [4] that for certain "over-stretched" NTRU parameters, one can exploit a subfield. This attack was only applicable to the NTRU lattices that are used to instantiate a (fully) homomorphic encryption scheme. The authors of [4] also showed that for our parameters the subfield attack will not be successful.

3.4 Decryption error rates

We summarize the result in Table 8.

Table 8. Decryption error rate

NTRU-443	NTRU-743	NTRU-1024
$< 2^{-196}$	$< 2^{-112}$	$< 2^{-80}$

We note that for all three parameter sets, it is safe to assume that no decryption errors will be observed, assuming the maximum number of key exchange or encryption that one will perform is bounded by 2^{64} as suggested by NIST.

For detailed analysis of decryption rate of NTRU-443 and NTRU-743, see [24]. For NTRU-1024, we give the following analysis.

Recall that in decryption one computes

$$\mathbf{m}' = \mathbf{f} * \mathbf{c} = p \cdot \mathbf{r} * \mathbf{g} + p \cdot \mathbf{e} * \mathbf{f} + \mathbf{m}' * \mathbf{f} \bmod q.$$

A decrypt error will occur if $\|p \cdot \mathbf{r} * \mathbf{g} + p \cdot \mathbf{e} * \mathbf{f} + \mathbf{m}' * \mathbf{f}\|_\infty > q/2$ which will cause a wraparound. It is sufficient to focus on the first two terms since $\mathbf{m}'$ is a lot smaller than $p \cdot \mathbf{r} * \mathbf{g}$ or $p \cdot \mathbf{e} * \mathbf{f}$. Hence we need to compute the probability that

$$\|\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f}\|_\infty > q/(2p)$$

Two simplify the analysis, we know that $\mathbf{r}$, $\mathbf{g}$, $\mathbf{e}$, $\mathbf{f}$ are all sampled from Gaussian with σ, therefore each coefficient of $(\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f})$ is a sum of $2N$ products of two Gaussian integers. The distribution of product of two Gaussian integers with a same deviations σ is a normal product distribution

$$D(z) = \frac{K_0(\frac{z}{\sigma^2})}{\pi\sigma^2}$$

where

$$K_0(z) = \int_0^\infty \frac{\cos(zt)}{t^2+1} dt$$

This allows us to estimate that

$$\text{Prob}[x > q/(4Np)] \lessapprox 2^{-102}$$

for a single integer with normal product form distribution. Since each coefficient of $(\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f})$ is a sum of $2N$ samples from $D(z)$, require the above event happens for the all the $2N$ samples for each coefficients, therefore we estimate that the decryption error rate will be

$$1 - (1 - 2^{-102})^{(2N)^2} \approx 2^{-80}$$

3.5 Advantages and limitations

Most scrutinized lattice-based cryptosystem The NTRU encryption algorithm was created in 1996, and has survived over 20 years of cryptanalysis. It's cryptographic design has been a fertile source of inspiration to other cryptographers, with uses ranging from the notion of ideal lattices [35] to the construction of some fully homomorphic encryption schemes [20,34]. To date, we see many candidate quantum safe algorithms building upon NTRU or related ideas, such as NTRU-prime [9] and NTRU-KEM [32]. The NTRU trapdoor is still the most efficient way to design lattice based signature schemes.

The NTRU algorithm was standardized by IEEE 1363 [3] in 2008 and ANSI X9.98 [37] in 2010. Both standards use the parameters from [23] in 2008. Those parameters have been stable for almost 10 years, despite of the rapid development of lattice cryptanalysis over the last decade, including BKZ 2.0 [13], BKZ 2.0 with sieving [6], subfield attacks [8,5,14,33] and more.

Small package sizes NTRU has the smallest public key and ciphertext sizes, compared to other lattice based solutions, such as NewHope [6] or Kyber [11]. This is particularly interesting for handshake protocols, since in those protocols, it is crucial to fit the handshake payloads in a single Maximum Transmission Unit (MTU). Otherwise, one would expect some package drops that may potentially slow down the protocol and require additional protocol-level features (such as fragmentation management), or even cause the handshake to fail.

The MTU is usually 1.5 kbytes. It allows for a maximum package payload of around 1 kbytes, with the rest reserved for hello messages. The NTRU based solutions, to the best of our knowledge, are the only ones that fit in this model.

Lack of provable security One question that has been asked frequently about NTRU is its lack of provable security. To be precise, one can reduce the security of NTRU encryptions to the unique shortest vector problem over an NTRU lattice. The "lack of security" here means that, unlike some LWE based schemes, where the uSVP problem is over a generic lattice, and may be proven hard for certain parameters (which are often not the same parameters used to instantiate the actual scheme, see [12] for example), the uSVP problem for NTRU lattices has no proof of hardness.

We provide two arguments to address this concern. First, in [36] it was shown that one can establish a reduction to R-LWE problems for certain choices of parameters. Indeed, the ss-ntru-pke algorithms in this submission follows this direction.

On the other hand, in order to provide best performance, we also derive parameters for ntru-pke, based on the best known attacks with a comfortable margin. To validate our understanding of cryptanalytic techniques, we published the NTRU challenge. Only the first few challenges of tiny dimensions have been broken, as we expected, and the running time to break those challenges aligns with our expectation [24].

3.6 Performance and implementations

Benchmark We present the benchmark results in Table 9. We tested our implementation with a dual core Intel i7- 6600U processor @ 2.60GHz. Our operation system was Linux Ubuntu 16.04. We used gcc version 5.4.0. The benchmark result is shown in Table 9.

Table 9. Benchmark results

Param	Key Gen	Encryption	Decryption
ntru-kem-443	440 μs	82 μs	109 μs
ntru-pke-443	472 μs	84 μs	109 μs
ntru-kem-743	1017 μs	140 μs	210 μs
ntru-pke-743	990 μs	121 μs	195 μs
ntru-kem-1024	43.5 ms	67 ms	115 ms
ntru-pke-1024	43.2 ms	67 ms	115 ms

Optimizations not in this submission package. There are two optimizations that we are aware of, that are not included in this submission package. Namely

1. A constant time, AVX2 based optimization for polynomial multiplication [16]; this accelerates polynomial multiplication by 2.3 times.
2. Product form polynomials [24]; this decreases the speed of polynomial multiplications by around 3 times.

We do not provide the first optimization, since it is prohibited by the submission. We also do not provide the second optimization for conservative purposes.

Potential improvements not in this submission package. There are also three potential improvements that we are aware of, that are not included in this submission.

1. A better Gaussian sampler.
2. A security argument against a quantum random oracle.
3. Efficient Number Theoretic Transform (NTT).

The first two items are active research areas. We believe that we shall see many improvements from the PQC community and it is too early to fix on a single solution. We shall include those improvements once they are available, and when a minor revision is allowed. Nonetheless, we remark that our Box-Muller based Gaussian sampler is already quite efficient, and has previously been used in the literature, such as the HElib [22].

We did not implement item 3 due to time constrains. Our naive NTT algorithm takes roughly $O(N^2/2)$ operations where N is the degree of the polynomial. We are aware of the Cooley-Tukey method which runs in $O(N \log N)$ time, and improves signing speed up to 10 times in practice. We are willing to provide implementation of this during the revision phrase.

3.7 Known Answer Test Values

Please see the *KAT* folder.

4 IPR Statement

Please see the *statement* folder.

References

1. What is the world's data storage capacity?, 2011. available from `http://www.zdnet.com/article/what-is-the-worlds-data-storage-capacity/`.
2. The number of atoms in the World, 2014. available from `http://www.fnal.gov/pub/science/inquiring/questions/atoms.html`.
3. IEEE Std 1363.1-2008. IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices, 2008.
4. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 153–178, 2016.
5. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 153–178, 2016.
6. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343, 2016.
7. Shi Bai, Thijs Laarhoven, and Damien Stehlé. Tuple lattice sieving. *IACR Cryptology ePrint Archive*, 2016:713, 2016.
8. Daniel J. Bernstein. A subfield-logarithm attack against ideal lattices, 2014. available from `https://blog.cr.yp.to/20140213-ideal.html`.
9. Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime. *IACR Cryptology ePrint Archive*, 2016:461, 2016.
10. DanielJ. Bernstein. The salsa20 family of stream ciphers. In Matthew Robshaw and Olivier Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer Berlin Heidelberg, 2008.
11. Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS - kyber: a cca-secure module-lattice-based KEM. *IACR Cryptology ePrint Archive*, 2017:634, 2017.
12. Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness II: practical issues in cryptography. In *Paradigms in Cryptology - Mycrypt 2016. Malicious and Exploratory Cryptology - Second International Conference, Mycrypt 2016, Kuala Lumpur, Malaysia, December 1-2, 2016, Revised Selected Papers*, pages 21–55, 2016.
13. Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, pages 1–20. Springer, 2011.

14. Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without an encoding of zero. *IACR Cryptology ePrint Archive*, 2016:139, 2016.
15. Don Coppersmith and Adi Shamir. Lattice attacks on NTRU. In *EUROCRYPT*, pages 52–61, 1997.
16. Wei Dai, William Whyte, and Zhenfei Zhang. Optimizing polynomial convolution for ntruencrypt. TBA.
17. Eiichiro Fujisaki and Tatsuaki Okamoto. *Secure Integration of Asymmetric and Symmetric Encryption Schemes*, pages 537–554. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.
18. Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 31–51, Berlin, Heidelberg, 2008. Springer-Verlag.
19. Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, 2010.
20. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
21. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.
22. Shai Halevi and Victor Shoup. Algorithms in helib. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 554–571, 2014.
23. Philip S. Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing ntruencrypt parameters in light of combined lattice reduction and MITM approaches. In *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, pages 437–455, 2009.
24. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. In *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, pages 3–18, 2017.
25. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing Parameters for NTRUEncrypt. In *Topics in Cryptology - CT-RSA 2017, The Cryptographers' Track at the RSA Conference 2017*, 2017.
26. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 267–288, 1998.
27. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 267–288, 1998.
28. Jeffrey Hoffstein and Joseph H. Silverman. Meet-in-the-middle Attack on an NTRU private key, 2006. available from `http://www.ntru.com`.
29. Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *CRYPTO*, pages 150–169, 2007.

30. Nick Howgrave-Graham, Joseph H. Silverman, Ari Singer, and William Whyte. NAEP: provable security in the presence of decryption failures. *IACR Cryptology ePrint Archive*, 2003:172, 2003.
31. Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. Choosing parameter sets for ntruencrypt with naep and sves-3. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, pages 118–135, 2005.
32. Andreas Hülsing, Joost Rijneveld, John M. Schanck, and Peter Schwabe. High-speed key encapsulation from NTRU. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 232–252, 2017.
33. Paul Kirchner and Pierre-Alain Fouque. Comparison between subfield and straightforward attacks on NTRU. *IACR Cryptology ePrint Archive*, 2016:717, 2016.
34. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1219–1234, 2012.
35. Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.
36. Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 27–47, 2011.
37. Accredited Standards Committee X9. Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry, 201.