High-speed key encapsulation from NTRU

Andreas Hülsing[1], Joost Rijneveld[2], John Schanck[3,4], and Peter Schwabe[2] *

[1] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, The Netherlands
andreas@huelsing.net
[2] Digital Security Group, Radboud University, The Netherlands
joost@joostrijneveld.nl, peter@cryptojedi.org
[3] Institute for Quantum Computing, University of Waterloo, Canada, and
[4] Security Innovation, Wilmington, MA, USA
jschanck@uwaterloo.ca

Abstract. This paper presents software demonstrating that the 20-year-old NTRU cryptosystem is competitive with more recent lattice-based cryptosystems in terms of speed, key size, and ciphertext size. We present a slightly simplified version of textbook NTRU, select parameters for this encryption scheme that target the 128-bit post-quantum security level, construct a KEM that is CCA2-secure in the quantum random oracle model, and present highly optimized software targeting Intel CPUs with the AVX2 vector instruction set. This software takes only 307 914 cycles for the generation of a keypair, 48 646 for encapsulation, and 67 338 for decapsulation. It is, to the best of our knowledge, the first NTRU software with full protection against timing attacks.
Keywords. Post-quantum crypto, lattice-based crypto, NTRU, CCA2-secure KEM, QROM, AVX2.

1 Introduction

In December 2016, NIST issued a call for proposals for "post-quantum cryptography" [43] to select schemes for standardization. More specifically, NIST requests algorithms in three categories: public-key encryption, key exchange or key encapsulation mechanisms (KEMs), and digital signatures. Obviously, the central requirement is that proposed schemes are indeed "post-quantum", i.e., that they resist attacks by a large quantum computer.

For encryption and key encapsulation, it seems that the most promising approach in terms of speed, key size, and ciphertext size is lattice-based cryptography. It is no coincidence that Google chose a lattice-based scheme, more specifically the NEWHOPE Ring-LWE-based key exchange [2], for their post-quantum

* This work has been supported by the European Commission through the ICT program under contract ICT-645622 (PQCRYPTO), and by the Netherlands Organisation for Scientific Research (NWO) through Veni 2013 project 13114. This work has also been supported by Canada's NSERC CREATE program. The Institute for Quantum Computing is supported in part by the Government of Canada and the Province of Ontario. Permanent ID of this document: 65dcfe39848495fe9b2423ac0a563d43. Date: June 28, 2017

TLS experiment [9]. It is also not surprising that various recent papers propose constructions and parameters, often together with implementations, for lattice-based encryption schemes and KEMs. See, for example, [2, 3, 7, 8, 14, 15, 19, 44].

These schemes differ in terms of security notions (e.g., passive vs. active security), underlying hard problems (e.g., learning-with-errors vs. learning with rounding), structure of the underlying lattices (standard vs. ideal lattices), cryptographic functionality (encryption vs. key encapsulation), and performance in terms of speed and sizes.

Contributions of this paper. In this paper we take a step back and turn our attention to the "grandfather of lattice-based encryption schemes", namely NTRU [26], with the goal of constructing a CCA2-secure key encapsulation mechanism (KEM).

We start by reconsidering the textbook OW-CPA-secure NTRU encryption scheme and show how a restriction on parameters leads to a considerably simpler and more efficient key generation algorithm. We also reconsider the sample spaces for private key and message vectors. We avoid the commonly used fixed-weight sample spaces and propose a sampling algorithm that produces independent and identically distributed coefficients. These changes make constant-time noise sampling much more efficient without significantly impacting security.

We then carefully optimize NTRU parameters to achieve 128 bits of post-quantum security while at the same time eliminating the possibility of decryption failures. Next, we transform this optimized OW-CPA-secure scheme into a CCA2-secure KEM in the quantum-accessible random oracle model (QROM). To this end, we tweak a known transform by Dent [20] such that security can also be shown in the QROM without notably sacrificing efficiency.

We illustrate the performance of our NTRU-based KEM by providing a highly optimized implementation targeting Intel processors with AVX2 vector instructions, the same architectures targeted by the optimized NEWHOPE software described in [2]. To the best of our knowledge, our software is the first NTRU software with full timing-attack protection.

KEM vs. PKE and passive vs. active security. Achieving CCA2 security for an NTRU-based public key encryption scheme appears to require the use of complex padding mechanisms [31]. However, already [53] and [45] showed that most of this complexity can be avoided when constructing an NTRU-based CCA2-secure *KEM*. CCA2-secure KEMs are very versatile building blocks. Together with a CCA2-secure symmetric "data encapsulation mechanism" (DEM) they can be used for CCA2-secure public-key encryption of messages of arbitrary length [18]. They are, furthermore, the central building block in (authenticated) key exchange constructions (see, e.g., [36]), in particular those that do not rely on signatures for authentication. We note that the NTRU-based KEM we describe in this paper could be used in place of NEWHOPE in the key exchange setting considered in Google's post-quantum TLS experiment. As a potential benefit, the CCA2 security allows busy servers to cache and reuse ephemeral keys to reduce the number of CPU cycles spent on key generation. This is a common

optimization in TLS libraries, but passively secure schemes like NEWHOPE or Frodo may not be secure when this optimization is deployed. See [2, Sec. 2].

Hasn't NTRU been superseded? From some recent papers on lattice-based cryptography one might get the impression that NTRU has been "superseded" by public-key encryption based on Ring-LWE [39] or by NTRU Prime [3]. For example, Kirchner and Fouque write in [35]: *"Since the practical cost of transforming a [sic] NTRU-based cryptosystem into a Ring-LWE-based cryptosystem is usually small, especially for key-exchange [...], we recommend to dismiss the former, in particular since it is known to be weaker."* Bernstein, Chuengsatiansup, Lange, and van Vredendaal write in [3]: *"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 classic NTRU cryptosystem, and have none of our recommended defenses."*

The statement by Kirchner and Fouque about NTRU being weaker than Ring-LWE is an asymptotic statement. It is actually not surprising that Ring-LWE is asymptotically a better choice than NTRU, because Ring-LWE-based (and LWE-based) cryptography has been designed, to a large extent, with asymptotic security statements in mind. However, these asymptotic results say little or nothing about the *concrete* security of parameters that have been proposed for actual use. See for example [11, Section 6]. NTRU on the other hand was designed with concrete security for concrete efficient parameters in mind and does not attempt to make asymptotic statements.

NTRU Prime can be seen as a variation of NTRU that uses a different ring (or, as the authors phrase it, that avoids *"rings with worrisome structure"*). Whether or not this choice of ring and other choices made in the design of NTRU Prime lead to a more or less secure scheme will need careful investigation. This is acknowledged by the authors, who state that they *"caution potential users that many details of Streamlined NTRU Prime are new and require careful security review"*.

To summarize, in this paper we do not argue for an order of preference among NTRU, NTRU Prime, and Ring-LWE. For concrete parameters aiming at a similar level of security and efficiency it is unclear which of the three will prove optimal in the long run. At the moment there are good reasons for and against choosing any of them. We focus on NTRU, the oldest of these schemes, with a track record of surviving 20 years of cryptanalysis.

A note on patents. One reason that NTRU is not more widely deployed is that there have been patents restricting its usage for most of its lifetime. The NTRU cryptosystem was patented in [27], and NTRU with "product-form keys" was patented in [28]. The former patent was due to expire on August 19, 2017, but in March of this year Security Innovation released both patents [49], placing NTRU into the public domain. Neither the present work nor the accompanying software makes use of product-form keys.

Availability of software. We place all software presented in this paper into the public domain to maximize reusability of our results. It is available for download at `https://joostrijneveld.nl/papers/ntrukem`.

2 Preliminaries

Minimal representatives. In describing NTRU it is useful to refer to quotient rings such as $\mathbb{Z}[x]/(8192, x^n - 1)$ and $\mathbb{Z}[x]/(3, x^n - 1)$. However, the scheme involves computations that are not well defined as maps on quotient rings. To avoid technical pitfalls around this issue, we describe all operations in $\mathbb{Z}[x]$ and introduce a "minimal representative" map to enact reduction modulo an ideal.

Let I be an ideal of $\mathbb{Z}[x]$ with $\mathbb{Z}[x]/I \cong (\mathbb{Z}/\ell)^m$ for some m, possibly ∞. The minimal representative map $[\cdot]_I : \mathbb{Z}[x] \to \mathbb{Z}[x]$ is defined such that $[a]_I \equiv a \pmod{I}$, $\deg [a]_I < m$, and $[a]_I$ has coefficients in $[-\ell/2, \ell/2)$. When ℓ is even we use the convention that $[\ell/2]_I = -\ell/2$. We write $[1/a]_I$ to denote the minimal b for which $[ab]_I = 1$, if such an element exists.

Cyclotomic rings. We denote the d^{th} cyclotomic polynomial by Φ_d. Note $\Phi_1 = x - 1$ and if d is prime $\Phi_d = 1 + x + x^2 + \cdots + x^{d-1}$. These are the only two cases we consider. We define

$$\begin{aligned} S_d &:= \mathbb{Z}[x]/(\Phi_d), \\ R_n &:= \mathbb{Z}[x]/(x^n - 1). \end{aligned}$$

For prime n we have $x^n - 1 = \Phi_1 \Phi_n$ and $R_n \cong S_1 \times S_n$. We will occasionally need to lift elements of S_n/p to R_n for a fixed prime p. We do this by solving the system of congruences

$$\begin{aligned} \mathsf{Lift}_p(v) &\equiv 0 \pmod{\Phi_1} \\ \mathsf{Lift}_p(v) &\equiv v \pmod{(p, \Phi_n)}. \end{aligned}$$

Solutions are guaranteed to exist by the Chinese remainder theorem. We fix a particular solution

$$\mathsf{Lift}_p(v) := \left[\Phi_1 \left[v/\Phi_1 \right]_{(p,\Phi_n)} \right]_{(x^n-1)}.$$

An efficient algorithm for Lift_p may be found in Appendix B.

Coefficient embedding of R_n. The coefficient embedding identifies the monomial basis $\{1, x, x^2, \ldots, x^{n-1}\}$ of R_n as an orthonormal basis of $\mathbb{R}^n$. We write v_i for the i^{th} coefficient of v and allow arithmetic modulo n in the index. We write $\langle \cdot, \cdot \rangle$ for the inner product on $\mathbb{R}^n$ and define the 2-norm and max-norm as usual: $|v|_2 = \sqrt{\langle v, v \rangle}$, and $|v|_\infty = \max_i |v_i|$.

For $a \in R_n$ we write $\overline{a}$ to denote the element with $\overline{a}_i = a_{-i}$ for all i. This "reversal" map reveals a connection between the multiplicative structure of R_n and the geometry of the coefficient embedding that we use in Lemma 1. Namely,

$$ab = \sum_{k=0}^{n-1} \sum_{i=0}^{n-1} a_{k-i} b_i x^k = \sum_{k=0}^{n-1} \langle x^k \overline{a}, b \rangle x^k.$$

Min-entropy. If ρ is a probability distribution on a finite set X, then the min-entropy of ρ is $\min\{-\log_2 \rho(x) : x \in X\}$.

eXtendable Output Functions (XOF). Our constructions make use of an extendable output function $\mathsf{XOF}(X, L, S)$, where X is the input bitstring, L is the desired output length in bits, and S is a context string (domain separator). As the XOF will be modeled as (quantum-accessible) random oracle in our security arguments, we require the instantiation of the XOF to be indistinguishable from a random oracle. The XOF can be instantiated, for example, using sponge constructions like SHAKE [5]. We often need a length value that is consistent with the security level for the scheme. We denote this μ. One may assume $\mu = 256$.

3 OW-CPA-secure NTRU encryption

In this section we describe the key generation, encryption, and decryption routines for our OW-CPA-secure NTRU encryption scheme. We make several departures from previous instantiations. First, we work directly with S_n to avoid common security issues associated with the S_1 subring of R_n. While it is possible to instantiate NTRU directly in S_n, and not use R_n at all, we still lift elements of S_n to R_n to take advantage of convenient computational and geometric features of R_n. Second, we choose parameters so that decryption failures are completely eliminated, and we do this without restricting the key and message spaces. Finally, we eliminate any need for fixed-weight distributions like those used in [3, 16, 19, 23, 24, 26, 32]. All of our sampling routines are chosen to admit simple and efficient constant time implementations.

3.1 Parameters

NTRU is parameterized by an odd prime n and coprime positive integers p and q. The parameter n indexes R_n and S_n, hereafter denoted R and S. We define $\mathfrak{p} = (p, \Phi_n)$ and $\mathfrak{q} = (q, x^n - 1)$. Ciphertexts and public keys belong to the set of minimal representatives of $R/q = \mathbb{Z}[x]/\mathfrak{q}$. Messages, blinding polynomials, and private keys belong to the set of minimal representatives of $S/p = \mathbb{Z}[x]/\mathfrak{p}$, denoted

$$\mathcal{T} = \{a \in \mathbb{Z}[x] : a = [a]_{\mathfrak{p}}\}.$$

Private keys are restricted to non-negatively correlated elements of $\mathcal{T}$:

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

The correlation restriction is new to this work. It comes from the proof of correctness in Section 3.5.

In Section 3.5 we prove that our instantiation of NTRU is correct, i.e. that decryption failures are impossible, with $p = 3$ and $q = q(n)$ where $\log_2 q(n) = \lceil 7/2 + \log_2(n)\rceil$. With $p = 3$ and q as the smallest power of two providing correctness, n is our only free parameter.

A final restriction on parameters is that Φ_n must be irreducible modulo both p and q. This obviates invertibility tests during key generation and makes the process more amenable to a constant time implementation. A similar condition has been recommended since the original description of NTRU [25, 26], but has not previously been a requirement. Streamlined NTRU Prime has an analogous requirement for q, but not for p [3]. This leaves us with only a handful of valid n in the range typical of recent NTRU and LWE instantiations. They are: 509, 557, 653, 677, 701, 773, 797, 821, 859, 907, 941, and 1061. All of these satisfy $q(n) \in \{8192, 16384\}$. The largest for which $q(n) = 8192$ is $n = 701$. In Section 4 we show that the corresponding parameter set, $n = 701, p = 3, q = 8192$, is expected to provide 128-bit security in a post-quantum setting.

3.2 Key Generation

A private key for our OW-CPA-secure encryption scheme is a non-zero element $f \in \mathcal{T}$. A corresponding public key is $h \in R$ such that $[fh]_q$ has small coefficients. We generate h by sampling g in $\mathcal{T}$ and computing $h = [\Phi_1 g f_q]_q$ where $f_q = [1/f]_{(q,\Phi_n)}$. This ensures $[fh]_q = \Phi_1 g$.

Previous instantiations of NTRU have taken f to be a short element of R with an inverse in both R/p and R/q. With the parameters of the previous section, every non-zero element of $\mathcal{T}$ is invertible as an element of both S/p and S/q. Invertibility in S/p and S/q is sufficient for our decryption procedure, so we can forego tests for invertibility in R/p and R/q. Inverses must still be computed, but the process never fails.

The factor of Φ_1 in the definition of h ensures that h is equivalent to zero modulo (q, Φ_1). Previous instantiations of NTRU have taken $h = [g/f]_q$ and the value of h modulo (q, Φ_1) has been a security concern – one that is typically mitigated by sampling f and g from sets of fixed-weight vectors. To avoid complicated sampling routines, we allow f and g to take any value in $\mathcal{T}_+$.

Of course the exact distribution from which f and g are drawn affects security. Algorithm 1 makes use of a generic subroutine $\mathsf{Sample}\mathcal{T}_+$ that may be thought of as sampling from the uniform distribution on $\mathcal{T}_+$. In Section 3.4 we describe a non-uniform $\mathsf{Sample}\mathcal{T}_+$ routine that admits simple and efficient constant time implementation. Our security claims in Section 4 are relative to this non-uniform distribution. Implementations that sample from the uniform distribution on $\mathcal{T}_+$ may be able to claim a slightly higher security level.

Algorithm 1 $\mathsf{KeyGen}(coins)$

1: $g = \mathsf{Sample}\mathcal{T}_+(\mathsf{XOF}(coins, \mu, \texttt{randg}))$
2: $f = \mathsf{Sample}\mathcal{T}_+(\mathsf{XOF}(coins, \mu, \texttt{randf}))$
3: $f_q = [1/f]_{(q,\Phi_n)}$
4: $h = [\Phi_1 g f_q]_q$

Output: Private key f, Public key h

3.3 OW-CPA Encryption

An NTRU public key determines an R-module of rank 2 that we denote by

$$L_h = \{(u, v) \in R^2 : v \equiv uh \pmod{q}\}. \tag{1}$$

Clearly $(1, h) \in L_h$, so $[L_h]_{\mathsf{q}}$ is a set of exactly q^n distinct points in R^2. Elements of R^2 of the form $r(1, h) + (0, m) = (r, rh + m)$ will generally not be in L_h. The essential idea behind NTRU is that with suitable restrictions on r and m it is possible to recover m uniquely from $rh + m$.

In previous instantiations of NTRU, r and m have been chosen to have coefficients in $\{-1, 0, 1\}$ with a prescribed number of coefficients taking each value. We depart from this by letting r and m take arbitrary values in $\mathcal{T}$.

We also ensure that all ciphertexts are identical modulo (q, Φ_1). Toward this end we take encryption to be the map

$$(r, m) \mapsto [prh + \mathsf{Lift}_p(m)]_{\mathsf{q}}.$$

Since h and $\mathsf{Lift}_p(m)$ are equivalent to 0 modulo (q, Φ_1) this achieves our goal.

Complete encryption and decryption routines are given by Algorithms 2 and 3. As with $\mathsf{Sample}\mathcal{T}_+$ in the previous section, $\mathsf{Sample}\mathcal{T}$ in Algorithm 2 is a generic sampling routine that may be thought of as sampling from the uniform distribution on $\mathcal{T}$. However, our security claims in Section 4 are with respect to the $\mathsf{Sample}\mathcal{T}$ instantiation described in Section 3.4, which does not sample the uniform distribution on $\mathcal{T}$.

Line 2 of Algorithm 2 is equivalent to the original NTRU encryption primitive [26] on the subring corresponding to S. The OW-CPA security [20, Definition 3] of this scheme is (trivially) equivalent to the assumption that random NTRU ciphertexts are hard to invert.

Algorithm 2 $\mathcal{E}(m, coins, h)$

1: $r = \mathsf{Sample}\mathcal{T}(coins)$
2: $e = [prh + \mathsf{Lift}_p(m)]_{\mathsf{q}}$
Output: Ciphertext e.

Algorithm 3 $\mathcal{D}(e, f)$

1: $m' = \left[[ef]_{\mathsf{q}} f^{-1}\right]_{\mathsf{p}}$
Output: m'

3.4 Simplified sampling

Sampling from the uniform distribution on $\mathcal{T}$ or $\mathcal{T}_+$ in constant time may be difficult or slow. In this section we present alternative distributions that admit simple and efficient constant time sampling routines. Our security analysis in Section 4 assumes that these simplified sampling routines are used. Implementations that sample from the uniform distribution on these spaces may be able to claim a slightly higher security level.

We first note that any routine for sampling from $\mathcal{T}$ can be transformed into a routine for sampling from $\mathcal{T}_+$ with at most a one bit loss in the min-entropy of its output distribution. Let v be an element of $\mathcal{T}$ and let w be the element obtained by flipping the signs of the even index coefficients of v. With the exception of $w_{n-1}w_0$, each product in the expansion of $\langle xw, w\rangle$ contains one even index term and one odd index term. Hence $\langle xw, w\rangle = 2v_{n-1}v_0 - \langle xv, v\rangle$. However, since $v \in \mathcal{T}$ we have $v_{n-1} = 0$ and in fact $\langle xw, w\rangle = -\langle xv, v\rangle$.

Our simplified $\mathsf{Sample}\mathcal{T}_+$ routine (Algorithm 4) draws v from $\mathcal{T}$ and then conditionally applies an even index sign flip to v if $\langle xv, v\rangle < 0$. While Algorithm 4 does not preserve the distribution of its $\mathsf{Sample}\mathcal{T}$ subroutine, in the way that rejection sampling would, it does preserve expected length. Also note that the even index sign flip is an involution on $\mathcal{T}$, so the min-entropy of the output of $\mathsf{Sample}\mathcal{T}_+$, over a uniform choice of coins, is at most one bit less than the min-entropy of the output of $\mathsf{Sample}\mathcal{T}$.

Algorithm 4 $\mathsf{Sample}\mathcal{T}_+(coins)$

$v = \mathsf{Sample}\mathcal{T}(coins)$
$s = \mathrm{sign}(\langle xv, v\rangle)$
/* $s = \pm 1$, $\mathrm{sign}(0) = 1$ */
for $i = 0$ **to** $(n-1)/2$ **do**
    $v_{2i} = s \cdot v_{2i}$
end for
Output: $v \in \mathcal{T}_+$

Algorithm 5 $\mathsf{Sample}\mathcal{T}(coins)$

$b = \mathsf{XOF}(coins, 4n-4, \texttt{expand})$
$v = 0$
for $i = 0$ **to** $n-2$ **do**
    $v_i = [b_{4i} + b_{4i+1} - b_{4i+2} - b_{4i+3}]_p$
end for
Output: v

Our simplified $\mathsf{Sample}\mathcal{T}$ routine (Algorithm 5) draws $n-1$ coefficients independently from a centered binomial distribution[5] of parameter $t = 2$ and then reduces these coefficients modulo p. The process always consumes exactly $2t(n-1)$ random bits. The resulting distribution is centrally symmetric (for any value of p and t) and tends to the uniform distribution as t is increased. With $p = 3$ and $t = 2$, a coefficient drawn from this distribution is $-1, 0$, or 1 with probability $\frac{5}{16}$, $\frac{6}{16}$, and $\frac{5}{16}$ respectively. The expected length of the output is therefore $\sqrt{\frac{5}{8}(n-1)}$.

3.5 Correctness

The following lemma determines the parameters for which we can prove that $(\mathsf{KeyGen}, \mathcal{E}, \mathcal{D})$ is a correct probabilistic encryption scheme. It also explains our use of $\mathcal{T}_+$. A similar statement with $g \in \mathcal{T}$ would require a factor of 2 rather than $\sqrt{2}$.

[5] A centered binomial distribution of parameter t is defined as $\sum_{i=1}^{t} b_i - b_{t+i}$ where $b_1, b_2, \ldots, b_{2t}$ are uniform random bits.

Lemma 1. *For $r \in \mathcal{T}$ and $g \in \mathcal{T}_+$,*

$$|\mathsf{Lift}_p(r)g|_\infty \leq \sqrt{2} \max_{a \in \mathcal{T}} |a|_2^2.$$

Proof. We may write $\mathsf{Lift}_p(r) = [(x-1)\overline{v}]_{(x^n-1)}$ where $\overline{v} = [r/\Phi_1]_\mathfrak{p} \in \mathcal{T}$. The quantity in question satisfies

$$|\mathsf{Lift}_p(r)g|_\infty = |\overline{v}(xg - g)|_\infty = \max_i |\langle x^i v, xg\rangle - \langle x^i v, g\rangle|.$$

To simplify the indexing we will assume wlog that the maximum is attained at $i = 0$. Let $\gamma = \langle v, g\rangle / |g|_2^2$, and let $\widetilde{v}$ denote the projection of v orthogonal to g, $\widetilde{v} = v - \gamma g$. Let $\eta = \langle g, xg\rangle / |g|_2^2$. Crucially, note that $g \in \mathcal{T}_+$ implies $\eta \in [0, 1]$. This gives us

$$\begin{aligned}
|\langle v, xg\rangle - \langle v, g\rangle| &= |\langle \widetilde{v}, xg\rangle + \gamma\langle g, xg\rangle - \langle v, g\rangle| \\
&\leq |\widetilde{v}|_2 |xg|_2 + |\gamma\langle g, xg\rangle - \langle v, g\rangle| \\
&= |\widetilde{v}|_2 |g|_2 + |\eta\langle v, g\rangle - \langle v, g\rangle| \\
&\leq |\widetilde{v}|_2 |g|_2 + |\langle v, g\rangle|.
\end{aligned}$$

For an upper bound we may assume that $|v|_2 = |g|_2 = \max\{|a|_2 : a \in \mathcal{T}\}$. Then with θ as the angle between v and g we have $|\widetilde{v}|_2 = \sin(\theta)|v|_2$, hence

$$\begin{aligned}
|\widetilde{v}|_2 |g|_2 + |\langle v, g\rangle| &\leq (\sin(\theta) + \cos(\theta)) \max_{a \in \mathcal{T}} |a|_2^2 \\
&\leq \sqrt{2} \max_{a \in \mathcal{T}} |a|_2^2
\end{aligned}$$

as claimed. □

Theorem 1 (Correctness). *The algorithms* KeyGen, $\mathcal{E}$, *and* $\mathcal{D}$ *with parameters $p = 3$ and $q > 8\sqrt{2}n$ are a correct probabilistic encryption scheme.*

Proof. Let f, g, and h be as in Algorithm 1, and let h' be such that $h = [\Phi_1 h']_\mathfrak{q}$. Fix a message $m \in \mathcal{T}$ and coins $c \in \{0,1\}^\mu$. Let $e = \mathcal{E}(m, c, h)$. Note that we may write $e = [p\mathsf{Lift}_p(r)h' + \mathsf{Lift}_p(m)]_\mathfrak{q}$ for some $r \in \mathcal{T}$. The claim is that $[[ef]_\mathfrak{q} f^{-1}]_\mathfrak{p} = m$. It suffices to show that $[[ef]_\mathfrak{q}]_\mathfrak{p} = [\mathsf{Lift}_p(m)f]_\mathfrak{p}$. Toward this end, note that

$$\begin{aligned}
ef &= [p\mathsf{Lift}_p(r)h' + \mathsf{Lift}_p(m)]_\mathfrak{q} f \\
&\equiv p\mathsf{Lift}_p(r)g + \mathsf{Lift}_p(m)f \pmod{\mathfrak{q}}.
\end{aligned}$$

By definition of the minimal representative map, the claim holds if

$$[p\mathsf{Lift}_p(r)g + \mathsf{Lift}_p(m)f]_\mathfrak{q} = [p\mathsf{Lift}_p(r)g + \mathsf{Lift}_p(m)f]_{(x^n-1)}.$$

Only the reduction modulo q can obstruct this since $(x^n - 1) \subset \mathfrak{q}$. Hence it is sufficient to have

$$|p\mathsf{Lift}_p(r)g + \mathsf{Lift}_p(m)f|_\infty < q/2.$$

With $p = 3$ an element of $\mathcal{T}$ is of norm at most $n - 1$. By Lemma 1 we have

$$|3\mathsf{Lift}_p(r)g + \mathsf{Lift}_p(m)f|_\infty < 4\sqrt{2}n < q(n)/2,$$

and the claim follows. □

4 NTRU parameters for 128-bit post-quantum security

We claim that our $n = 701$ parameter set offers 128-bit security in a post-quantum setting. Recall that we have defined our KEM so that n is the only free parameter; for $n = 701$ we have $p = 3$ and $q = 8192$. The claim of 128-bit post quantum security is based on two separate numerical analyses. First, an analysis of the "known quantum" primal attack described in [2] with the cost model of the same paper. Second, an analysis of the hybrid attack [30] using the cost model of [24]. In Appendix A we review the cryptanalytic literature around NTRU and provide some insight into how security analyses of NTRU have evolved since 1996.

Both of our numerical analyses attempt to estimate the cost of lattice reduction on a sublattice of L_h (Eq. 1). Specifically, a lattice generated by a subset of the columns of

$$\left(\begin{array}{c|c} q \cdot I_{n-1} & H \\ \hline 0 & I_{n-1} \end{array}\right), \tag{2}$$

where column $0 \leq i < n-1$ of H is given by $[x^i h]_{(q,\Phi_n)}$. The analyses also require estimates for the length of a shortest vector in L_h. For this we assume the distribution on f and g induced by Algorithm 5.

When optimized according to the success criteria of [2], the "known quantum" primal attack applies BKZ with blocksize 466 to the first 1283 columns of Eq. (2)[6]. The cost of BKZ-466 is dominated by a polynomial number of calls to an SVP solver in dimension 466. The quantum version of Laarhoven's hypercone filtering sieve solves SVP in dimension k at a cost of $(13/9)^{k/2+o(k)}$ queries to a quantum search oracle [37, Section 14.2.10]. Following [2] we assume that the $o(k)$ term is positive for relevant values of k. Setting $k = 466$ and suppressing all subexponential factors, including the number of SVP calls made by BKZ, we obtain a cost of $(13/9)^{466/2} > 2^{123}$ queries. In [2] the overhead of converting the query cost into a quantum RAM model cost is absorbed into the $(13/9)^{o(k)}$ term. Our claim of 128 bit post-quantum security follows as long as a query has a quantum RAM model cost $\geq 2^5$. To see that this is the case we will briefly sketch the steps of the hypercone filtering sieve and what these queries involve.

With $k = 466$, each iteration of the sieve involves the enumeration of $> 2^{98}$ lattice points. A subset of these of size 2^{97} is put aside for later use. The remaining lattice points, of which there are $> 2^{97}$, are stored in a database that admits nearest-neighbor queries. Points are stored in a data structure called a filter bucket in order to facilitate these queries. Each point is stored in 2^{26} out of a total of 2^{97} filter buckets. The total number of point representations stored is therefore $> 2^{123}$. Having built this database, the attacker makes a nearest neighbor query for each of the reserved 2^{97} points. Let v be such a point. The search for a neighbor of v involves the construction of a list of points in filter buckets relevant to v. There are expected to be $\gtrsim 2^{26}$ filter buckets relevant to v, each containing $\gtrsim 2^{26}$ points. For the $(13/9)^{k/2}$ query cost estimate, one assumes

[6] If $\mathsf{Sample}\mathcal{T}$ produced the uniform distribution on $\mathcal{T}$, then the attack would apply BKZ with blocksize 470 to the first 1285 columns of Eq. (2).

that this list of $\gtrsim 2^{52}$ points relevant to v is presented by a quantum-accessible oracle, O_v, and moreover, that quantum search finds a nearest neighbor of v after 2^{26} queries to O_v. Each query tests whether a point w is close to v; the test is performed in superposition over relevant w. Accessing the entries of w, in order to compute $||v - w||$, has a (quantum) RAM model cost that is at least linear in k (the dimension of v and w). The nearest neighbor search is repeated for each of the reserved points for a total quantum RAM model cost of at least $466 \cdot 2^{26} \cdot 2^{97} > 2^{131}$ operations.

We will now consider the cost of Howgrave-Graham's hybrid attack [30]. This analysis allows for a more direct comparison with recent security estimates for NTRU [24] and NTRU Prime [3]. As in [3, 24] we use the BKZ 2.0 simulator of [13] to estimate the quality of the basis produced by BKZ-β after a fixed number of SVP calls.

In a slight departure from [3, 24], we estimate the cost of solving SVP by enumeration in dimension β using a quasilinear fit to the experimental data of Chen and Nguyen [12]. Following [1] we use the trend line:

$$enum(\beta) = 0.18728 \cdot \beta \log_2(\beta) - 1.0192 \cdot \beta + 16.10. \tag{3}$$

Note that $enum(\beta)$ estimates the logarithm of the RAM cost for one SVP call.

After optimizing the attack parameters subject to the success criteria given in [24], we estimate that hybrid attack makes $> 2^{13}$ SVP calls in dimension 339. Each SVP call has a cost of $2^{enum(\beta)} > 2^{204}$ operations. The attack has a cost of $> 2^{217}$ operations. As described, this is an entirely pre-quantum attack. The meet-in-the-middle stage of the hybrid attack can be replaced by quantum search to reduce the storage requirements, but this does not change the estimated cost.

Following [19], we also consider the effect of a quadratic improvement in the cost of solving SVP by enumeration. We report the resulting cost in the "Quantum Enum" column of Table 1. Lastly, in the column labeled "Quantum Sieve", we report the cost of the hybrid attack when the quantum version of Laarhoven's hypercone filtering sieve is used within BKZ.

Table 1: Cost of the hybrid attack with various SVP subroutines.

SVP routine	**Enum**	**Quantum Enum**	**Quantum Sieve**
Dimension	925	1092	1144
Blocksize β	339	434	464
SVP calls	12250	11462	13128
SVP cost exponent	$enum(\beta)$	$enum(\beta)/2$	$\beta \log_2(13/9)$
Cost	2^{217}	2^{156}	2^{136}

Note that the cost estimates in Table 1 do not have the same units. The enumeration column has units of "bit operations." The quantum enumeration and

quantum sieve columns have units of "quantum queries." Furthermore the queries required for quantum enumeration could potentially be replaced with polynomial space algorithms, while the quantum sieve requires exponential space.

5 CCA2-secure key-encapsulation mechanism

We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM. Toward this end, we make use of a generic transform by Dent [20, Table 5]. Similar transforms have already been used for the NTRU-based KEMs described in [45, 53] and [3]. This transform comes with a security reduction in the random oracle model. As we are interested in post-quantum security, we have to deal with the quantum-accessible random oracle model (QROM) [6]. As it turns out, Dent's transform can be viewed as the KEM version of the Fujisaki-Okamoto transform (FO) [21]. For this FO-transform there exists a security reduction in the QROM by Targhi and Unruh [54]. It just requires appending a hash to the ciphertext.

The basic working of the KEM-transform is as follows. First, a random string m is sampled from the message space of the encryption scheme. This string is encrypted using random coins, deterministically derived from m using a hash function, later modeled as a random oracle (RO) in the proof (we use a XOF to instantiate all ROs). The session key is derived from m by applying another RO. Finally, the ciphertext and the session key are output.

The decapsulation algorithm decrypts the ciphertext to obtain m, derives the random coins from m, and re-encrypts m using these coins. If the resulting ciphertext matches the received one, it generates the session key from m.

In the QROM setting, Targhi and Unruh had to add the hash of m to the ciphertext to make the proof go through. The reason is that in the QROM setting a reduction simulating the RO has no way to learn the actual content of adversarial RO queries. This issue can be circumvented by this additional hash using a length preserving RO. In the proof, the reduction simulates this RO using an invertible function. When it later receives a classical output of this RO, it can recover the corresponding input, inverting the function.

An unfortunate detail in our case is that message space elements are strictly larger than a single hash value. Appending the output of a length preserving function to the ciphertext would therefore significantly increase the encapsulation size. One might think of several ways to circumvent this issue, sadly all straight forward approaches fail. A first approach would be to append a hash of the coins used for $\mathsf{Sample}\mathcal{T}$ instead of using its output. This does not work in the given setting as $\mathsf{Sample}\mathcal{T}$ is not invertible. Hence, the receiver has no way to check the validity of the hash. A second approach would be as follows. Instead of deriving everything from the message, one could first compute a message digest using a XOF parameterized to be compressing. Then the coins used in the encryption, the encapsulated key, and the appended hash are all computed from this message digest. This makes the security reduction fail, as it becomes impossible for the reduction to verify if a given decapsulation query contains

a valid ciphertext. The reduction would always return a valid decapsulation as it does not use decryption for this. Hence, the behavior of the reduction would significantly differ from the real security game. As none of these straightforward approaches work, we accept the increase of 141 bytes, which "only" accounts for 11% of the final encapsulation size. Users that do not consider a QROM proof necessary, can just omit this hash value. Alternatively, one could replace $\mathsf{Sample}\mathcal{T}$ with an efficiently invertible function. In that case the first approach described above becomes viable.

Algorithm 6 $\mathsf{Encaps}\,(h)$

1: $c_0 \leftarrow \{0,1\}^\mu$
2: $m = \mathsf{Sample}\mathcal{T}(c_0)$
3: $c_1 = \mathsf{XOF}(m, \mu, \texttt{coins})$
4: $k = \mathsf{XOF}(m, \mu, \texttt{key})$
5: $e_1 = \mathcal{E}(m, c_1, h)$
6: $e_2 = \mathsf{XOF}(m, \mu, \texttt{qrom})$

Output: Ciphertext (e_1, e_2), session key k.

Algorithm 7 $\mathsf{Decaps}\,((e_1, e_2), (f, h))$

1: $m = \mathcal{D}(e, f)$
2: $c_1 = \mathsf{XOF}(m, \mu, \texttt{coins})$
3: $k = \mathsf{XOF}(m, \mu, \texttt{key})$
4: $e_1' = \mathcal{E}(m, c_1, h)$
5: $e_2' = \mathsf{XOF}(m, \mu, \texttt{qrom})$
6: **if** $(e_1', e_2') \neq (e_1, e_2)$ **then**
7: $\quad k = \perp$
8: **end if**

Output: Session key k

6 Implementation

With this work, we provide a portable reference implementation of the scheme described above, as well as an optimized implementation using vector instructions from the AVX2 instruction set. Both implementations run in constant time. The AVX2 implementation is tailored to the $n = 701, q = 8192, p = 3$ parameter set. This section highlights some of the relevant building blocks to consider when implementing the scheme, focusing on the AVX2 implementation. Recall that the AVX2 extensions provide 16 vector registers of 256 bits that support a wide range of SIMD instructions.

6.1 Polynomial multiplication

It will come as no surprise that the most crucial implementation aspect is polynomial multiplication. As is apparent from the definition of the scheme, we require multiplication in R/q during key generation and during encryption and decryption. Additionally, decryption uses multiplication in S/p. Furthermore, we use multiplication of binary polynomials in order to perform inversion in S/q, which we will describe in Section 6.2.

Multiplication in R/q. The multiplication can be composed into smaller instances by combining Toom-Cook multiplication with Karatsuba's method[7]. Consider that elements of R/q are polynomials with 701 coefficients in $\mathbb{Z}/8192$; 16 such coefficients fit in a vector register. With this in mind, we look for a sequence of decompositions that result in multiplications best suited for parallel computation.

By applying Toom-Cook to split into 4 limbs, we decompose into 7 multiplications of polynomials of degree 176. We decompose each of those by recursively applying two instances of Karatsuba to obtain 63 multiplications of polynomials of 44 coefficients. Consider the inputs to these multiplications as a matrix, rounding the dimensions up to 64 and 48. By transposing this matrix we can efficiently perform the 63 multiplications in a vectorized manner. Using three more applications of Karatsuba, we decompose first into 22 and 11 coefficients, until finally we are left with polynomials of degree 5 and 6. At this point a straight-forward schoolbook multiplication can be performed without additional stack usage.

The full sequence of operations is as follows. We first combine the evaluation step of Toom-4 and the two layers of Karatsuba. Then, we transpose the obtained 44-coefficient results by applying transposes of 16x16 matrices, and perform the block of 63 multiplications. The 88-coefficient products remain in 44-coefficient form (i.e. aligned on the first and 45^{th} coefficient), allowing for easy access and parallelism during interpolation; limbs of 44 coefficients are the smallest elements that interact during this phase, making it possible to operate on each part individually and keep register pressure low.

A single multiplication in R/q costs 11 722 cycles. Of this, 512 cycles are spent on point evaluation, 3 692 cycles are used for the transposes, 4 776 are spent computing the 64-way parallel multiplications, and the interpolation and recomposition takes 2 742 cycles.

Multiplication in S/p. In this setting it appears to be efficient to decompose the multiplication by applying Karatsuba recursively five times, resulting in 243 multiplications of polynomials of degree 22. One could then bitslice the two-bit coefficients into 256-bit registers with only very minimal wasted space, and perform schoolbook multiplication on the 22-register operands, or even decide to apply another layer of Karatsuba.

For our implementation, however, we instead decide to use our R/q multiplication as though it were a generic $\mathbb{Z}[x]/(x^n - 1)$ multiplication. Even though in general these operations are not compatible, for our parameters it works out. After multiplication and summation of the products, each result is at most $701 \cdot 4 = 2804$, staying well below the threshold of 8192. While a dedicated

[7] Note that, as is observed in [3], popular choices for the ring in Ring-LWE schemes typically make it convenient to use the NTT to perform multiplication. As was also the case in [3], however, our ring of choice is particularly unsuitable. In our case this is caused by q being a power of two, and the polynomials being of prime degree.

S/p multiplication would out-perform this use of R/q multiplication, the choice of parameters makes this an attractive alternative at a reasonable cost.

Multiplication in $(\mathbb{Z}/2)[x]$. Dedicated processor instructions have made multiplications in $(\mathbb{Z}/2)[x]$ considerably easier. As part of the CLMUL instruction set, the `PCLMULQDQ` instruction computes a carry-less multiplication of two 64-bit quadwords, performing a multiplication of two 64-coefficient polynomials over $\mathbb{Z}/2$.

We set out to efficiently decompose into polynomials of degree close to 64, and do so by recursively applying a Karatsuba layer of degree 3 followed by a regular Karatsuba layer and a schoolbook multiplication. This reduces the full multiplication to 72 multiplications of 59-bit coefficients, which we perform using `PCLMULQDQ`. By interweaving the evaluation and interpolation steps with the multiplications, we require no intermediate loads and stores, and a single multiplication ends up measuring in at only 244 cycles.

6.2 Inverting polynomials

Computing the inverse of polynomials plays an important role in key generation. We compute $[1/f]_{(q,\Phi_n)}$ when producing the public key, but also pre-compute $[1/f]_{(p,\Phi_n)}$ as part of the secret key, to be used during decryption.

Inversion in S/q. We compute $[1/f]_{(2,\Phi_n)}$ and then apply a variant of Newton iteration [51] in R/q to obtain $f_q \equiv f^{-1} \pmod{(q,\Phi_n)}$. It may not be the case that $f_q = [1/f]_{(q,\Phi_n)}$, however the difference this makes in the calculation of h is eliminated after the multiplication by Φ_1 in Line 4 of Algorithm 1. The Newton iteration adds an additional cost of eight multiplications in R/q on top of the cost of an inversion in $S/2$.

Finding an inverse in $S/2$ is done using $f^{2^{n-1}-1} \equiv 1 \pmod{(2,\Phi_n)}$, and thus $f^{2^{700}-2} \equiv f^{-1} \pmod{(2,\Phi_{701})}$ [34]. This exponentiation can be done efficiently using an addition chain, resulting in twelve multiplications and thirteen multi-squarings.

Performing a squaring operation in $(\mathbb{Z}/2)[x]$ is equivalent to inserting 0-bits between the bits representing the coefficients: the odd-indexed products cancel out in $\mathbb{Z}/2$. When working modulo (x^n-1) with odd n, the subsequent reduction of the polynomial causes the terms with degree exceeding x^n to wrap around and fill the empty coefficients. This allows us to express the problem of computing a squaring as performing a permutation on bits. More importantly: repeated squarings can be considered repeated permutations, which compose into a single bit permutation.

Rewording the problem to that of performing bit permutations allows for different approaches; both generically and for specific permutations. In order to aid in constructing routines that perform these permutations, we have developed a tool to simulate a subset of the assembly instructions related to bit movement.

Rather than representing the bits by their value, we label them by *index*, making it significantly easier to keep track. The assembly code corresponding to the simulated instructions is generated as output. While we have used this tool to construct permutations that represent squarings, it may be of separate interest in a broader context — the source code is also available as part of this work.

We use two distinct generic approaches to construct permutation routines, based on `pext`/`pdep` (from the BMI2 instruction set), and on `vshufb`.

The first approach amounts to extracting and depositing bits that occur within the same 64-bit block in both the source and destination bit sequence, under the constraint that their order remains unchanged. By relabeling the bits according to their destination and using the patience sorting algorithm, we iteratively find the longest increasing subsequence in each block until all bits have been extracted. Note that the number of required bit extractions is equal to the number of piles patience sort produces. In order to minimize this, we examine the result for each possible rotated input, and rotate it according to the rotation that produces the least amount of disjunct increasing subsequences. Heuristically keeping the most recently used masks in registers allows us to reduce the number of load operations, as the BMI2 instructions do not allow operands from memory. Further improvements could include dynamically finding the right trade-off between rotating registers and re-using masks, as well as grouping similar extractions together. For the permutations we faced, these changes did not immediately seem to hold any promises for significant improvements.

The second approach uses byte-wise shuffling to position the bits within 256-bit registers. We examine all eight rotations of the input bytes and use `vshufb` to reposition the bytes (as well as `vpermq` to cross over between `xmm` lanes). The number of required shuffles is minimized by gathering bytes for all three destination registers at the same time, and where possible, rotation sequences are replaced by shifts (as the rotated bits often play no role in the bit deposit step, and shifts are significantly cheaper). Whereas the bit extraction approach works for well-structured permutations, it is beaten by the (somewhat more constant) shuffling-based method for the more dispersed results. While there is some potential for gain when hand-crafting permutations, it turns out to be non-trivial to beat the generated multi-squarings.

The multi-squaring routines vary around 235 cycles, with a single squaring taking only 58. Including converting from R/q to $S/2$, an inversion in $S/2$ costs 10 332 cycles. Combining this with the multiplication in R/q described above, the full procedure takes 107 726 cycles.

Inversion in S/p. Inversion in S/p is done using the 'Almost Inverse' algorithm described in [47] and [51]. However, the algorithm as described in [51] (listed as Algorithm 9 in Appendix C) does not run in constant time. Notably, it performs a varying number of consecutive divisions and multiplications by x depending on the coefficients in f, and halts as soon as f has degree zero. We eliminate this issue by iterating through every term in f (i.e. including potential zero terms, up to the n^{th} term), and always performing the same operations for each term

(i.e. constant-time swaps and always performing the same addition, multiplied with a flag fixing the sign). See Algorithm 10 in Appendix C for a listing in pseudo-code. Note that `fmadd(f, g, s)` is the operation $f_i + s \cdot g_i \mod 3$, for all coefficients f_i and g_i in f and g.

While the number of loop iterations is constant, the final value of the rotation counter k is not; the `done` flag may be set before the final iteration. We compensate for k after the loop has finished by rotating 2^i bits for each bit in the binary representation of k, and subsequently performing a constant-time move when the respective bit is set.

Benefiting from the width of the vector registers, we operate on bitsliced vectors of coefficients. This allows us to efficiently perform the multiplications and additions in parallel modulo 3, and makes register swaps comparatively easy. On the other hand, shifts are still fairly expensive, and two are performed for each loop iteration to multiply and divide by x. With 159 606 cycles, the inversion remains a very costly operation that determines a large chunk of the cost of the key generation operation. There may still be some room for significant improvement, though, considering the fact that each instruction in the critical loop gets executed fourteen hundred times.

7 Results and Comparison

Table 2 gives an overview of the performance of various lattice-based encryption schemes and KEMs. As memory is typically not a big concern on the given platforms, concrete memory usage figures are often not available and we do not attempt to include this in the comparison. In the same spirit, our reference implementation uses almost 11KiB of stack space and our AVX2 software uses over 43KiB, but this should not be considered to be a lower bound. We performed our benchmarks on one core of an Intel Core i7-4770K (Haswell) at 3.5GHz and followed the standard practice of disabling TurboBoost and hyperthreading. We warn the reader that direct comparison of the listed schemes and implementations is near impossible for various reasons: First of all, there are significant differences in the security level; however, at least most schemes aim at a level of around 128 bits of post-quantum security. More importantly, the passively secure KEMs have a very fast decapsulation routine, but turning them into CCA2-secure KEMs via the Targhi-Unruh transform would add the cost of encapsulation to decapsulation. Also, the level of optimization of implementations is different. For example, we expect that Frodo [7] or the spLWE-based KEM from [14] could be sped up through vectorization. Finally, not all implementations protect against timing attacks and adding protection may incur a serious overhead. However, the results show that carefully optimized NTRU is very competitive, even for key generation and even with full protection against timing attacks.

Table 2: Comparison of lattice-based KEMs and public-key encryption. Benchmarks were performed on an Intel Core i7-4770K (Haswell) if not indicated otherwise. Cycles are stated for key generation (**K**), encapsulation/encryption (**E**), and decapsulation/decryption (**D**) Bytes are given for secret keys (**sk**), public keys (**pk**), and ciphertexts (**c**). The column "ct?" indicates whether the software is running in constant time, i.e., with protection against timing attacks.

Scheme	PQ sec.	ct?	Cycles	Bytes
Passively secure KEMs				
BCNS [8]	78[a]	yes	**K**: ≈ 2 477 958	**sk**: 4096
			E: ≈ 3 995 977	**pk**: 4096
			D: ≈ 481 937	**c**: 4224
NewHope [2]	255[a]	yes	**K**: 88 920	**sk**: 1792
			E: 110 986	**pk**: 1824
			D: 19 422	**c**: 2048
Frodo [7] (recommended parameters)	130[a]	yes	**K**: ≈ 2 938 000[b]	**sk**: 11 280
			E: ≈ 3 484 000[b]	**pk**: 11 296
			D: ≈ 338 000[b]	**c**: 11 288
CCA2-secure KEMs				
NTRU Prime [3]	129[a]	yes	**K**: ?[c]	**sk**: 1417
			E: > 51488[c]	**pk**: 1232
			D: ?[c]	**c**: 1141
spLWE-KEM [14] (128-bit PQ parameters)	128[g]	?	**K**: ≈ 336 700[d]	**sk**: ?
			E: ≈ 813 800[d]	**pk**: ?
			D: ≈ 785 200[d]	**c**: 804
NTRU-KEM (this paper)	123[a]	yes	**K**: 307 914	**sk**: 1422
			E: 48 646	**pk**: 1140
			D: 67 338	**c**: 1281
CCA-secure public-key encryption				
NTRU ees743ep1 [24]	159[a]	no	**K**: 1 194 816	**sk**: 1 120
			E: 57 440	**pk**: 1 027
			D: 110 604	**c**: 980
Lizard [15] (recommended parameters)	128[g]	no	**K**: ≈ 97 573 000	**sk**: 466 944[f,h]
			E: ≈ 35 000	**pk**: 2 031 616[h]
			D: ≈ 80 800	**c**: 1 072

[a] According to the conservative estimates obtained by the approach from [2]

[b] Benchmarked on a 2.6GHz Intel Xeon E5 (Sandy Bridge)

[c] The NTRU Prime paper reports benchmarks only for polynomial multiplication

[d] Benchmarked on "PC (Macbook Pro) with 2.6GHz Intel Core i5"

[e] Benchmarked by eBACS [4] on Intel Xeon E3-1275 (Haswell)

[f] Unlike our scheme, the secret key does not include the public key required for decryption in the Targhi-Unruh transform

[g] According to the authors' analysis, i.e., not following [2]

[h] Derived from the implementation – can be compressed to $\frac{10}{16}$ of its size at a marginal increase in cost of **K**, **E** and **D** by representing each coefficient using $\log(q)$ bits

References

1. Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. IACR Cryptology ePrint Archive report 2015/046, 2015. https://eprint.iacr.org/2015/046. 11
2. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange – a new hope. In Thorsten Holz and Stefan Savage, editors, *Proceedings of the 25th USENIX Security Symposium*. USENIX Association, 2016. https://cryptojedi.org/papers/#newhope. 2, 3, 10, 18
3. Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime. IACR Cryptology ePrint Archive report 2016/461, 2016. https://eprint.iacr.org/2016/461. 2, 3, 5, 6, 11, 12, 14, 18
4. Daniel J. Bernstein and Tanja Lange. eBACS: ECRYPT benchmarking of cryptographic systems. http://bench.cr.yp.to. 18
5. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, 2011. http://keccak.noekeon.org/. 5
6. Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, 2011. https://eprint.iacr.org/2010/428. 12
7. Joppe Bos, Craig Costello, Leo 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 Christopher Kruegel, Andrew Myers, and Shai Halevi, editors, *Conference on Computer and Communications Security – CCS '16*, pages 1006–1018. ACM, 2016. https://doi.org/10.1145/2976749.2978425. 2, 17, 18
8. 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 Lujo Bauer and Vitaly Shmatikov, editors, *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE, 2015. https://eprint.iacr.org/2014/599. 2, 18
9. Matt Braithwaite. Experimenting with post-quantum cryptography. Posting on the Google Security Blog, 2016. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html. 2
10. Johannes Buchmann and Christoph Ludwig. Practical lattice basis sampling reduction. In Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Algorithmic Number Theory – ANTS-VII*, LNCS, pages 222–237. Springer, 2006. https://eprint.iacr.org/2005/072. 23
11. Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness ii: Practical issues in cryptography. IACR Cryptology ePrint Archive report 2016/360, 2016. https://eprint.iacr.org/2016/360. 3
12. Yuanmi Chen. *Lattice reduction and concrete security of fully homomorphic encryption*. PhD thesis, l'Université Paris Diderot, 2013. 11
13. 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*, volume 7073 of *LNCS*, pages 1–20. Springer, 2011. http://www.iacr.org/archive/asiacrypt2011/70730001/70730001.pdf. 11, 24
14. Jung Hee Cheon, Kyoohyung Han, Jinsu Kim, Changmin Lee, and Yongha Son. A practical post-quantum public-key cryptosystem based on spLWE. In Seokhie Hong and Jong Hwan Park, editors, *Information Security and Cryptology – ICISC*

2016, volume 10157 of *LNCS*, pages 51–74. Springer, 2017. https://eprint.iacr.org/2016/1055. 2, 17, 18

15. Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! Practical post-quantum public-key encryption from LWE and LWR. IACR Cryptology ePrint Archive report 2016/1126, 2016. https://eprint.iacr.org/2016/1126. 2, 18
16. Consortium for Efficient Embedded Security. EESS #1: Implementation aspects of NTRUEncrypt and NTRUSign v. 2.0, June 2003. http://grouper.ieee.org/groups/1363/lattPK/submissions/EESS1v2.pdf. 5, 23
17. Don Coppersmith and Adi Shamir. Lattice attacks on NTRU. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT '97*, volume 1233 of *LNCS*, pages 52–61. Springer, 1997. http://dx.doi.org/10.1007/3-540-69053-0_5. 23
18. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, 33(1):167–226, 2003. http://www.shoup.net/papers/cca2.pdf. 2
19. Rafael del Pino, Vadim Lyubashevsky, and David Pointcheval. The whole is less than the sum of its parts: Constructing more efficient lattice-based AKEs. In Vassilis Zikas and Roberto De Prisco, editors, *Security and Cryptography for Networks – SCN 2016*, volume 9841 of *LNCS*, pages 273–291. Springer, 2016. https://eprint.iacr.org/2016/435. 2, 5, 11
20. Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *Cryptography and Coding*, volume 2898 of *LNCS*, pages 133–151. Springer, 2003. http://www.cogentcryptography.com/papers/designer.pdf. 2, 7, 12
21. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999. http://dx.doi.org/10.1007/3-540-48405-1_34. 12
22. Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, 2010. http://www.iacr.org/archive/eurocrypt2010/66320257/66320257.pdf. 24
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 Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security – ACNS 2009*, volume 5536 of *LNCS*, pages 437–455. Springer, 2009. https://eprint.iacr.org/2005/045. 5, 23, 24
24. Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. In Helena Handschuh, editor, *Cryptographers' Track at the RSA Conference – CT-RSA 2017*, volume 10159 of *LNCS*, pages 3–18. Springer, 2017. https://eprint.iacr.org/2015/708. 5, 10, 11, 18, 24
25. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A new high speed public key cryptosystem, 1996. draft from at CRYPTO '96 rump session. http://web.securityinnovation.com/hubfs/files/ntru-orig.pdf. 6, 23
26. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory – ANTS-III*, volume 1423 of *LNCS*, pages 267–288. Springer, 1998. http://dx.doi.org/10.1007/BFb0054868. 2, 5, 6, 7, 23

27. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Public key cryptosystem method and apparatus. United States Patent 6081597, 2000. Application filed August 19, 1997, http://www.freepatentsonline.com/6081597.html. 3
28. Jeffrey Hoffstein and Joseph H. Silverman. Speed enhanced cryptographic method and apparatus. United States Patent 7031468, 2006. Application filed August 24, 2001, http://www.freepatentsonline.com/7031468.html. 3
29. Jeffrey Hoffstein, Joseph H Silverman, and William Whyte. Estimated breaking times for NTRU lattices. Technical Report #012, NTRU Cryptosystems, 2003. Version 2. https://assets.onboardsecurity.com/static/downloads/NTRU/resources/NTRUTech012v2.pdf. 23
30. Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *LNCS*, pages 150–169. Springer, 2007. http://www.iacr.org/archive/crypto2007/46220150/46220150.pdf. 10, 11, 23, 24
31. Nick Howgrave-Graham, Joseph H. Silverman, Ari Singer, and William Whyte. NAEP: Provable security in the presence of decryption failures. Cryptology ePrint Archive, Report 2003/172, 2003. https://eprint.iacr.org/2003/172. 2, 23
32. Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3. In Alfred Menezes, editor, *Cryptographers' Track at the RSA Conference – CT-RSA 2005*, volume 3376 of *LNCS*, pages 118–135. Springer, 2005. https://eprint.iacr.org/2005/045. 5, 23
33. IEEE. IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. IEEE Std 1363.1-2008, 2009. http://dx.doi.org/10.1109/IEEESTD.2009.4800404. 24
34. Toshiya Itoh and Shigeo Tsujii. A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases. *Information and computation*, 78(3):171–177, 1988. https://sciencedirect.com/science/article/pii/0890540188900247. 15
35. Paul Kirchner and Pierre-Alain Fouque. Comparison between subfield and straightforward attacks on NTRU. IACR Cryptology ePrint Archive report 2012/387, 2016. https://eprint.iacr.org/2016/717. 3
36. Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *LNCS*, pages 429–448. Springer, 2013. eprint.iacr.org/2013/339. 2
37. Thijs Laarhoven. *Search problems in cryptography*. PhD thesis, Eindhoven University of Technology, 2015. http://www.thijs.com/docs/phd-final.pdf. 10, 24
38. Christoph Ludwig. A faster lattice reduction method using quantum search. In Toshihide Ibaraki, Naoki Katoh, and Hirotaka Ono, editors, *Algorithms and Computation – ISAAC 2003*, volume 2906 of *LNCS*, pages 199–208. Springer, 2003. https://www.cdc.informatik.tu-darmstadt.de/reports/TR/TI-03-03.QSamplingPaper.pdf. 23
39. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, 2010. http://www.di.ens.fr/~lyubash/papers/ringLWE.pdf. 3
40. Alexander May. Cryptanalysis of NTRU, 1999. https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/cryptanalysisofntru.ps. 23

41. Alexander May and Joseph H. Silverman. Dimension reduction methods for convolution modular lattices. In Joseph H. Silverman, editor, *Cryptography and Lattices: International Conference – CaLC 2001*, volume 2146 of *LNCS*, pages 110–125. Springer, 2001. http://dx.doi.org/10.1007/3-540-44670-2_10. 23
42. Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008. ftp://ftp.di.ens.fr/pub/users/pnguyen/JoMC08.pdf. 24
43. NIST. Post-quantum crypto project, 2016. http://csrc.nist.gov/groups/ST/post-quantum-crypto/. 1
44. Markku-Juhani O. Saarinen. Ring-LWE ciphertext compression and error correction: Tools for lightweight post-quantum cryptography. IACR Cryptology ePrint Archive report 2016/461, 2016. https://eprint.iacr.org/2016/1058. 2
45. Halvor Sakshaugh. Security analysis of the NTRUEncrypt public key encryption scheme. Master's thesis, Norwegian University of Science and Technology, 2007. https://brage.bibsys.no/xmlui/handle/11250/258846. 2, 12
46. Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *Symposium on Theoretical Aspects of Computer Science – STACS 2003*, volume 2607 of *LNCS*, pages 145–156. Springer, 2003. https://pdfs.semanticscholar.org/a323/ef7dcaaf2f8d4a52b63393986ba23140faa6.pdf. 23
47. Richard Schroeppel, Hilarie Orman, Sean O'Malley, and Oliver Spatscheck. Fast key exchange with elliptic curve systems. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO '95*, volume 963 of *LNCS*, pages 43–56. Springer, 1995. https://pdfs.semanticscholar.org/edc9/5e3d34f42deabe82ff3e9237266e30adc1a7.pdf. 16
48. Security Innovation. NTRUOpenSourceProject reference NTRUEncrypt implementation, 2013. https://github.com/NTRUOpenSourceProject/ntru-crypto. 24
49. Security Innovation. Security Innovation makes NTRUEncrypt patent-free, 2017. https://www.securityinnovation.com/company/news-and-events/press-releases/security-innovation-makes-ntruencrypt-patent-free. 3
50. Joseph H Silverman. A meet-in-the-middle attack on an NTRU private key. Technical Report #004, NTRU Cryptosystems, 1997. Version 1. 23
51. Joseph H. Silverman. Almost inverses and fast NTRU key creation. Technical Report #014, NTRU Cryptosystems, 1999. Version 1. https://assets.onboardsecurity.com/static/downloads/NTRU/resources/NTRUTech014.pdf. 15, 16
52. Joseph H Silverman. Dimension-reduced lattices, zero-forced lattices, and the NTRU public key cryptosystem. Technical Report #013, NTRU Cryptosystems, 1999. Version 1. https://assets.onboardsecurity.com/static/downloads/NTRU/resources/NTRUTech013.pdf. 23
53. Martijn Stam. A key encapsulation mechanism for NTRU. In Nigel P. Smart, editor, *Cryptography and Coding*, volume 3796 of *LNCS*, pages 410–427. Springer, 2005. 2, 12
54. Ehsan Ebrahimi Targhi and Dominique Unruh. Quantum security of the Fujisaki-Okamoto and OAEP transforms. Cryptology ePrint Archive, Report 2015/1210, 2015. https://eprint.iacr.org/2015/1210. 12
55. Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates. Cryptology ePrint Archive, Report 2016/733, 2016. https://eprint.iacr.org/2016/733. 24

A History of NTRU parameters and security.

We believe the history of cryptanalytic attention given to NTRU also inspires confidence in its security. We summarize parts of this history here.

In the manuscript circulated after their CRYPTO'96 rump session presentation, Hoffstein, Pipher and Silverman expressed optimism that NTRU parameters could be selected so that lattice reduction of L_h would not recover the secret key [25]. Coppersmith and Shamir quickly countered with a demonstration that any sufficiently short vector, not just the secret key, could be used for decryption, and they concluded that improvements in lattice reduction would render the scheme insecure [17]. Lattice reduction techniques have improved markedly in the intervening years, but this conclusion has not been borne out.

The security of early parameter sets was evaluated by extrapolating the runtime of attacks in small dimensions [26,50]. This approach had the obvious downside that new experiments had to be performed for each new attack strategy, but this work was often carried out.

For example, a successful key recovery attack on the toy "NTRU-107" parameter set was reported by Alexander May in [40]. The attack made use of a dimension reduction technique for which early experiments had not accounted. Dimension reduction was analyzed in [41, 52], where it was shown to provide only a small advantage to the attacker, and new experiments and attack time extrapolations were reported in [29]. A more dramatic case comes from Schnorr's introduction of random sampling reduction (RSR) in 2003 [46]. Buchmann and Ludwig proposed a variant of RSR in [10] and presented experiments suggesting that an 80-bit parameter set from the EESS#1v2 standard [16] would provide $\leq$ 74-bit security. Ludwig also developed a quantum RSR and gave post-quantum security estimates for the EESS#1v2 parameter sets[8] [38].

A CCA2-secure padding mechanism was developed in [31], and parameters that completely avoided decryption failures were proposed in [32]. These parameter sets had very small q, often $q < n$, and sparse private keys were needed to avoid decryption failure.

In 2007, Howgrave-Graham recognized that sparsity could be exploited by a hybrid lattice reduction and combinatorial attack based on Schnorr's RSR and Odlyzko's meet-in-the-middle strategy [30]. Schnorr had suggested a similar idea, a "birthday method", as a general tool for basis reduction in [46], but concluded that the memory requirements would be prohibitive. In the context of NTRU with sparse binary secrets this turns out not to be the case; Howgrave-Graham's proposed attack on EESS#1v2 parameters has expected RAM model cost that is below the security level of those parameters [30].

Hirschhorn, Hoffstein, Howgrave-Graham, and Whyte published revised parameters to account for the hybrid attack [23]. They also undertook new lattice reduction experiments from which they derived security estimates. Parameters

[8] These parameters were deprecated at about the same time due to chosen ciphertext attacks. Unfortunately, Ludwig's analysis has not been applied to other parameter sets.

generated according to [23] enforce $q = 2048$ and $p = 3$ but allow n and a private key sparsity parameter to vary subject to constraints imposed by a cost metric. This method was used to derive the parameters in the IEEE 1363.1-2008 standardization document [33], and to derive the product form parameters available in the commercial NTRU library [48].

To the best of our knowledge, there have been no concrete proposals for (pre-quantum) RAM model attacks on the IEEE 1363.1-2008 parameters with cost less than that suggested in [23]. This is in spite of significant advances in algorithms for enumeration (e.g., [22]) and sieving (e.g., [37, 42]).

Hoffstein, Pipher, Schanck, Silverman, Whyte and Zhang have re-evaluated the cost of the hybrid attack on some NTRU parameter sets in light of improved lattice reduction techniques [24]. Their cost estimates rely on the BKZ-2.0 simulator and extrapolation from experiments performed by Chen and Nguyen [13]. A more recent hybrid attack analysis by Wunderer [55] suggests that the analyses of [23, 24, 30] have underestimated the cost of the hybrid attack. This would imply that we have also underestimated the cost of the hybrid attack in Table 1.

It is likely that some of the IEEE 1363.1-2008 parameters remain secure in a post-quantum setting. However, these parameters do not satisfy the conditions on n, p, and q from Section 3.1, do not eliminate decryption failures, and cannot be used with our simplified key generation routine.

B Efficient computation of Lift_p

Recall that $\mathsf{Lift}_p(v) = \left[\Phi_1 \left[v/\Phi_1\right]_{\mathfrak{p}}\right]_{(x^n-1)}$. Multiplication by $\Phi_1 = x - 1$ can clearly be computed with $O(n)$ additions. It is less obvious that multiplication by $[1/\Phi_1]_{\mathfrak{p}}$ can be computed with $O(pn)$ additions. Algorithm 8 uses the near-periodicity of the coefficients of $[1/\Phi_1]_{\mathfrak{p}}$ to achieve this complexity.

Lemma 2. *Algorithm 8 computes* $\mathsf{Lift}_p(v)$ *correctly.*

Proof. Let $z = [1/\Phi_1]_{\mathfrak{p}}$. Observe that $[(x-1)z]_{(x^n-1)} \equiv [1+c\Phi_n]_{(x^n-1)} \pmod p$, and consequently

$$[(x^p-1)z]_{(x^n-1)} \equiv [(1+x+x^2+\cdots+x^{p-1})(1+c\Phi_n)]_{(x^n-1)} \pmod p.$$

Since $[x\Phi_n]_{(x^n-1)} = \Phi_n$ we obtain

$$[x^p z]_{(x^n-1)} \equiv z + (1+x+x^2+\cdots+x^{p-1}) \pmod p.$$

Applying the reversal map and multiplying through by x^p we see

$$[x^p \overline{z}]_{(x^n-1)} \equiv \overline{z} - (x^1+x^2+\cdots+x^p) \pmod p. \tag{4}$$

Let $v \in S_n$ be the input to Lift_p. Observe that $[v/\Phi_1]_{\mathfrak{p}} = [[vz]_{(p,x^n-1)}]_{\Phi_n}$. Let $a = [vz]_{(p,x^n-1)}$. Then $a_i \equiv \langle x^i \overline{z}, v\rangle \pmod p$, and for $i \geq p$ we have

$$a_i \equiv a_{i-p} - \sum_{j=0}^{p-1} v_{i-j} \pmod p$$

by Equation (4).

This proves the correctness of Lines (1-7) of Algorithm 8. After Line (7) the variable a holds $[vz]_{(p,x^n-1)}$. The fact that lines (8-13) correctly compute $b = [\Phi_1[a]_{\mathfrak{p}}]_{(x^n-1)}$ is clear by inspection. □

Algorithm 8 Lift_p

Input: v

1: $z = [1/\Phi_1]_{\mathfrak{p}}$
2: **for** $i = 0$ **to** $i = p-1$ **do**
3: $\quad a_i = \langle x^i \overline{z}, v \rangle$
4: **end for**
5: **for** $i = p$ **to** $i = n-1$ **do**
6: $\quad a_i = a_{i-p} - \sum_{j=0}^{p-1} v_{i-j}$
7: **end for**
8: $a_0 = a_0 - a_{n-1} \bmod p$
9: $b_0 = -a_0$
10: **for** $i = 1$ **to** $i = n-1$ **do**
11: $\quad a_i = a_i - a_{n-1} \bmod p$ /* $a = [v/\Phi_1]_{\mathfrak{p}}$ */
12: $\quad b_i = a_{i-1} - a_i$ /* $b = \Phi_1[v/\Phi_1]_{\mathfrak{p}}$ */
13: **end for**

Output: b

The following lemma is not needed elsewhere, but it gives an explicit expression for $[1/\Phi_1]_{\mathfrak{p}}$ and suggests a particularly efficient strategy for unrolling Lines (1-4) of Algorithm 8.

Lemma 3. *Let* $z = [1/\Phi_1]_{\mathfrak{p}}$ *and let* $t = [-1/n]_p$. *Then*

$$\overline{z} \equiv -(t+1) + \sum_{i=1}^{n-1} t(i-1)x^i \pmod{p}.$$

Proof. Let $a = z(x-1)$. We have $a \equiv 1 + c\Phi_n \pmod{p}$ for some $c \in [-p/2, p/2]$. Equating coefficients we have

$$a_0 = \langle \overline{z}, x-1 \rangle = \overline{z}_1 - \overline{z}_0 \equiv c+1 \pmod{p}, \tag{5}$$

and for $i \neq 0$,

$$a_{-i} = \langle x^{-i}\overline{z}, x-1 \rangle = \overline{z}_{i+1} - \overline{z}_i \equiv c \pmod{p}. \tag{6}$$

Since z is a minimal representative modulo $\mathfrak{p}$ we have $\overline{z}_1 = z_{n-1} = 0$. On the one hand, $\overline{z}_1 = 0$ gives us $\overline{z}_0 \equiv -(c+1) \pmod{p}$, by (5), and $\overline{z}_i = c(i-1)$, by (6). On the other hand, direct application of (6) gives us $a_{-1} = \overline{z}_0 - \overline{z}_{n-1} \equiv c \pmod{p}$. We conclude

$$c \equiv -1/n \mod p.$$

□

C 'Almost inverse' algorithm

Algorithm 9 'Almost Inverse'

Input: $a(X)$
1: $k = 1$, $b(X) = 1$, $c(X) = 0$, $f(X) = a(X)$, $g(X) = X^N - 1$
2: **while** $f_0 = 0$ **do**
3: $f(X) = f(X) \ / \ X$,
4: $c(X) = c(X) \cdot X$
5: $k = k + 1$
6: **end while**
7: **if** $deg(f) < deg(g)$ **then**
8: swap f and g, swap b and c
9: **end if**
10: **if** $f(X) = \pm 1$ **then**
11: **return** $\pm X^{N-k} b(X) \mod X^N - 1$
12: **end if**
13: **if** $f_0 = g_0$ **then**
14: $f(X) = f(X) - g(X) \mod 3$
15: $b(X) = b(X) - c(X) \mod 3$
16: **else**
17: $f(X) = f(X) + g(X) \mod 3$
18: $b(X) = b(X) + c(X) \mod 3$
19: **end if**
Output: $b(X) \equiv a(X)^{-1} \in \mathfrak{p}$

Algorithm 10 'Almost Inverse' in constant time

Input: $a(X)$
1: $k = 1$, $b(X) = 1$, $c(X) = 0$, $f(X) = a(X)$, $g(X) = X^N - 1$, $deg_f = deg_g = N - 1$
2: **for** $i = 0$ **to** $i = 2 \cdot (N-1) - 1$ **do**
3: $s = 2 \cdot f_0 \cdot g_0 \mod 3$
4: $swap = s \wedge \neg done \wedge (deg_f < deg_g)$
5: **cswap**$(f(X), g(X), swap)$
6: **cswap**$(b(X), c(X), swap)$
7: **cswap**$(deg_f, deg_g, swap)$
8: **fmadd**$(f(X), g(X), s \cdot \neg done)$
9: **fmadd**$(b(X), c(X), s \cdot \neg done)$
10: $f(X) = f(X) \ / \ X$
11: $c(X) = c(X) \cdot X$
12: $deg_f = deg_f - \neg done$
13: $k = k + \neg done$
14: $done = (deg_f == 0)$
15: **end for**
16: **return** $f_0 \cdot X^{N-k} b(X) \mod X^N - 1$
Output: $b(X) \equiv a(X)^{-1} \in \mathfrak{p}$