Tight bound on NewHope failure probability

Thomas Plantard *, Arnaud Sipasseuth †, Willy Susilo ‡ and Vincent Zucca §
Institute of Cybersecurity and Cryptology, University of Wollongong, Australia
Email: *thomaspl@uow.edu.au, †as447@uowmail.edu.au, ‡wsusilo@uow.edu.au, §vzucca@uow.edu.au

***Abstract*—`NewHope` Key Encapsulation Mechanism (KEM) has been presented at USENIX 2016 by Alchim et al. and is one of the remaining lattice-based candidates to the post-quantum standardisation initiated by the NIST. However, despite the relative simplicity of the protocol, the bound on the decapsulation failure probability resulting from the original analysis is not tight. In this work we refine this analysis to get a tight upper-bound on this probability which happens to be much lower than what was originally evaluated. As a consequence we propose a set of alternative parameters, increasing the security and the compactness of the scheme. However using a smaller modulus prevent the use of a full NTT algorithm to perform multiplications of elements in dimension 512 or 1024. Nonetheless, similarly to previous works, we combine different multiplication algorithms and show that our new parameters have competitive execution times on a constant time vectorized implementation. Our most compact parameters bring a speed-up of 9.5% (resp. an overcost of 3.2%) in performance but allow to gain more than 19% over the bandwidth requirements and to increase the security of 6% (resp. 4%) in dimension 512 (resp. 1024).**

***Index Terms*—Post-Quantum Cryptography, Karatsuba, Number Theoretic Transform, AVX implementation, Lattice Based Cryptography, Ring-LWE**

I. Introduction

The recent call of the National Institute of Standards and Technology (NIST) for a post-quantum standardization aims at selecting the best protocols resilient to the quantum computer for encryption, key exchange and digital signature. The first round of this process has been completed recently and only 26 candidates remain over the 69 initially proposed. The algorithms chosen at the end of this process are supposed to become cryptographic standards for the next decade(s). Candidates whose security is based on lattice related problems represent almost half of the remaining schemes and tend to be among the most efficient. The majority of these lattice candidates are based on the Learning With Errors (LWE) ([Reg05]) problem and its variants over rings: Ring-Learning With Errors (Ring-LWE) ([LPR10]) and modules: Module-LWE ([LS15]). Among them, `NewHope` protocol [ADPS16b] is a KEM whose security is based on the Ring-LWE problem and has been reported by the NIST as competitive in term of bandwidth and clock-cycles despite quite conservative parameters[1]. It allows two users to exchange a common 256 bits key in order to secure their future communications. However the small errors ensuring the security of the protocol can add together and make it fail, i.e. the two users will not have the same key at the end. In order to ensure this only happens with very low probability, the modulus q must be chosen to be large enough in comparison to the size of the errors. In [ADPS16b] the authors have shown that the modulus $q = 12289$ was large enough for this purpose. This modulus has not been chosen randomly since it is the smallest one allowing to use the Number Theoretic Transform (NTT) algorithm in dimension $n = 512$ or $n = 1024$, which permits to compute efficiently multiplications of elements. Considering that the multiplication is the bottleneck of the arithmetic in this scheme, and that vectorized implementations of NTT are the most efficient way to perform this task on these dimensions ([Sei18]), it is crucial to ensure good performance.

In this work, we review the probability error analysis made in [ADPS16b], also used in the slightly different version [ADPS16a] submitted to the NIST, and show that the actual bound is much lower than what was initially evaluated. This allows to justify the use of a smaller modulus q while maintaining a very low probability failure. Reducing the size of the modulus benefits directly to the bandwidth requirements since elements will be represented on less bits, and to the security of the scheme which is increased. However, given that the original modulus was the smallest one allowing the use of the NTT in dimensions 512 and 1024, one needs to find an alternative strategy to compute the product of elements efficiently. We propose different alternative moduli and, similarly to previous works ([Moe76], [ZXZ+18], [LS19], [ZPL19],...), we show that by combining Karatsuba and school-book multiplication with NTTs of smaller dimensions we obtain competitive performance with the state-of-the-art. More precisely, two of these moduli increase both the compactness and the performance up to 15% without reducing significantly the level of security (−1% in the worst-case). The detailed parameters can be found in Table III. Note that our vectorized implementation works in constant time and is based on Seiler's work ([Sei18]) which is, at the best of our knowledge the current state-of-the-art.

II. Related Art

Optimising the efficiency of the submissions to the NIST is a hot topic. In a certain sense, `Kyber` ([BDK+18]), which is essentially the adaptation of `NewHope` to the Module-RLWE setting, has been created to achieve the best possible performance from `NewHope`. However the parameters of `NewHope` remain, even now, very conservative. It has been mentionned recently that one could reduce the modulus size of `NewHope` to improve the compactness of the scheme ([ZXZ+18]) while performing the arithmetic with a classical trick: combining a naive multiplication algorithm with smaller NTTs ([Moe76]). This work has been recently improved by using a Karatsuba pattern instead of the naive algorithm ([ZPL19]). However

[1]https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf

since the authors use the same failure probability analysis than in [ADPS16b], the parameters they have derived are far from optimal. Moreover the design of their arithmetic technics has lead to slightly worst performance than in the original setting.

Using the idea of [ZXZ+18], Lyubaschevsky and Seiler have proposed a different arithmetic design. The idea is to perform an uncomplete – i.e. when the defining polynomial does not factorize in linear terms, NTT first and then perform the products modulo the small, not linear, factors ([LS19]). This has lead to significantly better performance and has been included in `Kyber`'s implementation since then ([RJL+19]).

`NewHope` implementation of the NTT ([ADPS16b]) which smartly combines Montgomery reductions ([Mon85]) and floating point operations, was considered, to the best of our knowledge, as the reference implementation at the time and was also used in `Kyber`. However in a remarkable work ([Sei18]), Seiler showed that by using a slightly different Montgomery reduction algorithm, and vectorized integer instructions for the NTT, one could gain a factor 4 (resp. 6) over the polynomial multiplication of `NewHope` (resp. `Kyber`). His work has been included in `Kyber`'s implementation recently ([RJL+19]).

III. Background

Let m be a power of two, and let $\mathbf{\Phi}_m(X) = X^n + 1$ be the cyclotomic polynomial of index m with $n = \varphi(m) = m/2$ with φ the Euler's totient function. The associated cyclotomic ring will be denoted $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ and for $q \geq 1$ we will denote the quotient ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} \cong (\mathbb{Z}/q\mathbb{Z})[X]/(X^n + 1)$. Bold lower-case letters $\boldsymbol{a}$ will denote elements in $\mathcal{R}$ which can be seen as polynomials of degree strictly smaller than n with integer coefficients. For an element $\boldsymbol{a} \in \mathcal{R}$ we denote, the L_∞ norm $\|\boldsymbol{a}\|_\infty = \max\{|a_i|, 0 \leq i < n\}$, the L_1 norm $\|\boldsymbol{a}\|_1 = \sum_{i=0}^{n-1} |a_i|$ and the L_2 norm $\|\boldsymbol{a}\|_2 = \sqrt{\sum_{i=0}^{n-1} a_i^2}$.

The ring of integers modulo q will be denoted $\mathbb{Z}_q$. The notation $[\cdot]_q$ will represent the centred reduction of an integer modulo q in $[-q/2, q/2)$ and can be extended to polynomials by applying it coefficient-wise. The flooring (resp. the rounding to the nearest integer) of a rational number will be denoted $\lfloor \cdot \rfloor$ (resp. $\lfloor \cdot \rceil$) and these notations will be used for polynomials similarly.

ψ_k will denote a centered binomial distribution of parameter k (elements sampled in $[-k; k] \cap \mathbb{Z}$). Sampling from ψ_k can be done by computing $\sum_{i=1}^{k}(b_i - b'_i)$ where b_i and $b'_i \in \{0, 1\}$ are independent uniformly sampled bits. ψ_k^n will denote the distribution over $\mathcal{R}$ where all the coefficients of a polynomial are sampled independently from ψ_k. Finally, for a distribution $\mathcal{D}$ (resp. a set $\mathcal{I}$), $a \xleftarrow{\$} \mathcal{D}$ (resp. $a \xleftarrow{\$} \mathcal{I}$) will mean that a is sampled randomly from $\mathcal{D}$ (resp. uniformly in $\mathcal{I}$).

A. NewHope

For the sake of completeness, `NewHope` protocol is recalled in Figure 1. We also present and discuss the main points of the protocol to ease our next analysis. For further details concerning the protocol the interested reader can refer to [ADPS16b].

Alice (server)		**Bob (client)**
$s \xleftarrow{\$} \{0, \ldots, 255\}^{32}$		
$\boldsymbol{a} \xleftarrow{\$} \text{Parse}(\texttt{SHAKE128}(s))$		
$\boldsymbol{s}_1, \boldsymbol{e}_1 \xleftarrow{\$} \psi_k^n$		
$\boldsymbol{b} \leftarrow [\boldsymbol{a} \cdot \boldsymbol{s}_1 + \boldsymbol{e}_1]_q$		$\boldsymbol{s}_2, \boldsymbol{e}_2, \boldsymbol{e}_3 \xleftarrow{\$} \psi_k^n$
$m_a \leftarrow \text{encodeA}(s, \boldsymbol{b})$	$\xrightarrow{m_a}$	$(s, \boldsymbol{b}) \leftarrow \text{decodeA}(m_a)$
		$\boldsymbol{a} \xleftarrow{\$} \text{Parse}(\texttt{SHAKE128}(s))$
		$\boldsymbol{u} \leftarrow [\boldsymbol{a} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_2]_q$
		$\nu \leftarrow \{0, 1\}^{256}$
		$\nu' \leftarrow \texttt{SHA3-256}(\nu)$
		$\boldsymbol{\kappa} \leftarrow \texttt{NHSEncode}(\nu')$
		$\boldsymbol{c} \leftarrow [\boldsymbol{b} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_3 + \boldsymbol{\kappa}]_q$
		$\hat{\boldsymbol{c}} \leftarrow \texttt{NHSCompress}(\boldsymbol{c})$
$(\boldsymbol{u}, \hat{\boldsymbol{c}}) \leftarrow \text{decodeB}(m_b)$	$\xleftarrow{m_b}$	$m_b \leftarrow \text{encodeB}(\boldsymbol{u}, \hat{\boldsymbol{c}})$
$\boldsymbol{c}' \leftarrow \texttt{NHSDecompress}(\hat{\boldsymbol{c}})$		$\mu \leftarrow \text{SHA3-256}(\nu')$
$\boldsymbol{\kappa}' = [\boldsymbol{c}' - \boldsymbol{u} \cdot \boldsymbol{s}_1]_q$		
$\nu' \leftarrow \texttt{NHSDecode}(\boldsymbol{\kappa}')$		
$\mu \leftarrow \texttt{SHA3-256}(\nu')$		

Fig. 1. `NewHope` Protocol (without reconciliation) [ADPS16a]

1) Encoding and compression functions: As shown in Figure 1, the key $\nu' \in \{0, 1\}^{256}$ is encoded by Bob as an element $\boldsymbol{\kappa} \in \mathcal{R}_q$ through an encoding function `NHSEncode` and Alice recovers it through `NHSDecode`. Additionally, a compression function `NHSCompress` is used in order to reduce the size of the elements sent by Bob to Alice. The way these functions work is detailed below.

Let $\boldsymbol{c} \in \mathcal{R}_q$ and $t \ll q$ be the compression parameter, the compression function consists in keeping only the $\log_2 t$ most significant bits of each coefficient, thus:

$$\texttt{NHSCompress}(\boldsymbol{c}) = \hat{\boldsymbol{c}} \in \mathcal{R}_t$$
$$\text{with } \hat{c}_i = \lfloor c_i \cdot t/q \rceil \bmod t \text{ for } i \in \{0, \ldots, n-1\}.$$

The decompression function consists in applying the reverse transformation:

$$\texttt{NHSDecompress}(\hat{\boldsymbol{c}}) = \boldsymbol{c}' \in \mathcal{R}_q$$
$$\text{with } c'_i = \lfloor \hat{c}_i \cdot q/t \rceil \text{ for } i \in \{0, \ldots, n-1\}$$

which allows to recover the original polynomial with an error $\boldsymbol{\epsilon}_{comp} \in \mathcal{R}$, such that $\|\boldsymbol{\epsilon}_{comp}\|_\infty < q/2t + 1/2$, due to the successive roundings.

The 256-bit key ν' is encoded in $\mathcal{R}_q$ thanks to a repetition code which encodes each bit of ν' in $n/256$ coefficients of $\boldsymbol{\kappa}$:

$$\begin{array}{rccc} \texttt{NHSEncode}: & \{0;1\}^{256} & \to & \mathcal{R}_q \\ & \nu' & \mapsto & \boldsymbol{\kappa} = \sum_{i=0}^{n-1} \kappa_i \cdot X^i \end{array}$$

with $\kappa_i = \nu'_i \cdot \lfloor q/2 \rfloor$ for $0 \leq i \leq 256$ and $\kappa_{i+256 \cdot j} = \kappa_i$ for $0 \leq j < n/256$.

Once the message decompressed, Alice has to extract the

key ν' from $\boldsymbol{\kappa}'$ which is equal to the following modulo q:

$$\begin{aligned}\boldsymbol{\kappa}' &= \boldsymbol{c} + \boldsymbol{\epsilon}_{comp} - (\boldsymbol{a}\cdot\boldsymbol{s}_2 + \boldsymbol{e}_2)\cdot\boldsymbol{s}_1 \\ &= (\boldsymbol{a}\cdot\boldsymbol{s}_1 + \boldsymbol{e}_1)\cdot\boldsymbol{s}_2 + \boldsymbol{e}_3 + \boldsymbol{\kappa} + \boldsymbol{\epsilon}_{comp} - (\boldsymbol{a}\cdot\boldsymbol{s}_2 + \boldsymbol{e}_2)\cdot\boldsymbol{s}_1 \\ &= \texttt{NHSEncode}(\nu') + \boldsymbol{e}_1\cdot\boldsymbol{s}_2 - \boldsymbol{e}_2\cdot\boldsymbol{s}_1 + \boldsymbol{e}_3 + \boldsymbol{\epsilon}_{comp}\end{aligned}$$

Thus $\boldsymbol{\kappa}'$ corresponds to the encoding of the original key ν' plus some error terms. Therefore, if the error term is not too large, the key ν' can be extracted from $\boldsymbol{\kappa}'$ through the following decoding function:

$$\begin{array}{rccc}\texttt{NHSDecode}: & \mathcal{R}_q & \to & \{0;1\}^{256} \\ & \boldsymbol{\kappa}' & \mapsto & \nu'\end{array}$$

where for each $0 \le i < 256$, ν'_i is recovered by checking the size of the difference between the coefficients of $\boldsymbol{\kappa}'$ and $\lfloor q/2 \rfloor$, which is the value of the coefficients encoding $\nu'_i = 1$. More precisely we check whether:

$$\sum_{j=0}^{n/256-1} |[\kappa'_{i+256\cdot j}]_q - \lfloor q/2 \rfloor|$$

is smaller than $(n/256 \cdot \lfloor q/2 \rfloor)/2$ ($\nu'_i = 1$) or not ($\nu'_i = 0$). Indeed without the error terms (i.e. if $\boldsymbol{\kappa}' = \text{NHSEncode}(\nu')$) then when $\nu'_i = 1$ (resp. 0), the sum would be equal to 0 (resp. $n/256 \cdot \lfloor q/2 \rfloor$). Therefore to ensure the success of the decoding we must ensure that the error remains smaller, in absolute value, than the median value $B_{\text{dec}} = (n/256 \cdot \lfloor q/2 \rfloor)/2$ with very high probability. This comes to ensure that for any $0 \le i < 256$:

$$\sum_{j=0}^{n/256-1} |(\boldsymbol{e}_1\cdot\boldsymbol{s}_2 - \boldsymbol{e}_2\cdot\boldsymbol{s}_1 + \boldsymbol{e}_3 + \boldsymbol{\epsilon}_{comp})_{i+256\cdot j}| < B_{\text{dec}} \quad (1)$$

2) A word about the parameters: In order to be able to encode a 256 bits key, n must be at least 256. In their submission package[2], the authors propose two instantiations with different dimensions: $n = 512$ and $n = 1024$ denoted as `NewHope512` and `NewHope1024`, respectively. Even though they recommend to use `NewHope1024` for security reasons, they show that `NewHope512` still ensure a descent security while offering performance roughly twice faster.

Because the bottleneck of the arithmetic in $\mathcal{R}_q$ is by far the multiplication of elements, the usage of the efficient NTT is essential to ensure good performance. Hence, while the protocol can be formally defined for any cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(\boldsymbol{\Phi}_m(X))$, taking m different from a power-of-two would lead to a more complex NTT algorithm and would significantly complicate the generation of the error ([LPR13]). In order to ensure the existence of the roots of unity required for the NTT, q must be congruent to 1 modulo $2n$. Therefore it was set to the smallest prime satisfying this condition which is $q = 12289$ for both $n = 512$ and $n = 1024$.

Finally, for efficiency reasons, the error distribution has been chosen as a centered binomial distribution, instead of a discrete Gaussian as it is usually the case for the (Ring)-LWE problem. Both `NewHope512` and `NewHope1024`, use $k = 8$ as parameter for the binomial distribution. Finally, the compression parameter is chosen as $t = 8$ for both versions.

[2]https://newhopecrypto.org/resources.shtml

B. Number Theoretic Transform (NTT)

Choosing q prime and such that $q \equiv 1 \bmod 2n$ ensures the existence of a primitive $2n$-th roots of unity modulo q that we will denote ζ. Since the odd powers ζ, are also the roots of $X^n + 1$ modulo q, the polynomial factors in linear terms over $\mathbb{Z}_q$. Therefore the Chinese Remainder Theorem (CRT) gives us the ring isomorphism:

$$\begin{array}{ccc}\mathcal{R}_q & \to & \mathbb{Z}_q[X]/(X-\zeta)\times\cdots\mathbb{Z}_q[X]/(X-\zeta^{2n-1}) \\ \boldsymbol{a} & \mapsto & (\boldsymbol{a}(\zeta),\ \boldsymbol{a}(\zeta^3),\ldots,\ \boldsymbol{a}(\zeta^{2n-1}))\end{array}$$

The NTT (resp. inverse NTT) allows to compute this morphism (resp. its inverse) in quasi-linear time, through an FFT algorithm. Once in NTT form additions and multiplications can be performed coefficient-wise, and thus in linear time. Therefore the product of elements $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R}_q$ can be performed by computing $\boldsymbol{c} = \texttt{invNTT}(\texttt{NTT}(\boldsymbol{a}) \odot \texttt{NTT}(\boldsymbol{a}))$, where $\odot$ denotes the product coefficient-wise. Overall a product can thus be computed in quasi-linear time ($O(n \log n)$). Efficient ways of implementating NTTs and its inverse on a vectorized implementation can be found in [Sei18].

IV. Analysis of the failure probability

In this section we review the failure probability made in [ADPS16b] and refine it. In a nutshell, we will follow their proof but instead of bounding the probability twice using the Chernoff-Cramer inequality and a lemma about subgaussian variables, we will bound it only once through Chernoff-Cramer inequality and deduce the rest from numerical simulations.

A. About key-encoding and multiplication

We consider the cyclotomic polynomials $\boldsymbol{r}(X) = X^n + 1$ and $\boldsymbol{s}(X) = X^{n/256} + 1$, in particular we have $\boldsymbol{r}(X) = \boldsymbol{s}(X^{256})$. The associated polynomial rings are denoted $\mathcal{R} = \mathbb{Z}[X]/(\boldsymbol{r})$ and $\mathcal{S} = \mathbb{Z}[X]/(\boldsymbol{s})$.

For any $\boldsymbol{a} = a_0 + a_1\cdot X\cdots + a_{n-1}\cdot X^{n-1} \in \mathcal{R}$ and $0 \le i \le 255$, $\boldsymbol{a}'_i(X) \in \mathcal{S}$ will denote:

$$\begin{cases} a_i + a_{i+256}\cdot X & \text{if } n = 512; \\ a_i + a_{i+256}\cdot X + a_{i+512}\cdot X^2 + a_{i+768}\cdot X^3 & \text{if } n = 1024. \end{cases}$$

Hence, any $\boldsymbol{a} \in \mathcal{R}$ can be decomposed in the following way:

$$\boldsymbol{a}(X) = \sum_{i=0}^{255} \boldsymbol{a}'_i(X^{256})\cdot X^i.$$

This decomposition is somehow preserved by the product in $\mathcal{R}$, since for any $\boldsymbol{a}, \boldsymbol{b}$ in $\mathcal{R}$ and $0 \le i < 256$ we have:

$$(\boldsymbol{a}\cdot\boldsymbol{b})'_i = \left[\sum_{j=0}^{i}(\boldsymbol{a}'_j\cdot\boldsymbol{b}'_{i-j}) + \sum_{j=i+1}^{255}\pi_{\mathcal{S}}(\boldsymbol{a}'_j\cdot\boldsymbol{b}'_{256+i-j})\right] \in \mathcal{S}$$

where $\pi_{\mathcal{S}}$ denotes the cyclic shift over $\mathcal{S}$ given by the multiplication by X corresponding to:

$$\pi_{\mathcal{S}}(\boldsymbol{a}') = -a'_1 + a'_0 \cdot X \quad \text{if } n = 512;$$
$$\pi_{\mathcal{S}}(\boldsymbol{a}') = -a'_3 + a'_0 \cdot X + a'_1 \cdot X^2 + a'_2 \cdot X^3 \quad \text{if } n = 1024.$$

If we assume that the coefficients of $\boldsymbol{a}$ and $\boldsymbol{b}$ are sampled independently then for each i, $(\boldsymbol{a} \cdot \boldsymbol{b})'_i$ is a sum of 256 independent polynomials of $\mathcal{S}$:

- the $i+1$ first polynomials follow the distribution of a polynomial product in $\mathcal{S}$;
- the $255 - i$ last follow the distribution of a shifted polynomial product in $\mathcal{S}$, through $\pi(\mathcal{S})$.

B. Distribution of the key-encoding error

For any $\boldsymbol{a}'$ and $\boldsymbol{b}'$ in $\mathcal{S}$ we have:

$$\begin{aligned} &\text{if } n = 512 \\ &(\boldsymbol{a}' \cdot \boldsymbol{b}')(X) = \quad (a'_0 b'_0 - a'_1 b'_1) + (a'_0 b'_1 + a'_1 b'_0) \cdot X \end{aligned}$$

$$\begin{aligned} &\text{if } n = 1024 \\ &(\boldsymbol{a}' \cdot \boldsymbol{b}')(X) = \quad (a'_0 b'_0 - a'_1 b'_3 - a'_2 b'_2 - a'_3 b'_1) \\ &\qquad + (a'_0 b'_1 + a'_1 b'_0 - a'_2 b'_3 - a'_3 b'_2) \cdot X \\ &\qquad + (a'_0 b'_2 + a'_1 b'_1 + a'_2 b'_0 - a'_3 b'_3) \cdot X^2 \\ &\qquad + (a'_0 b'_3 + a'_1 b'_2 + a'_2 b'_1 + a'_3 b'_0) \cdot X^3 \end{aligned}$$

From now, we will denote χ_k the distribution of $\boldsymbol{a}' \cdot \boldsymbol{b}' \in \mathcal{S}$ where $\boldsymbol{a}'$ and $\boldsymbol{b}'$ are sampled from $\psi_k^{n/256}$ independently. From the above expression it is straightforward to notice that χ_k is symmetric because $\psi_k^{n/256}$ is. Moreover, since $\psi_k^{n/256}$ is a centered distribution it is invariant under $\pi_{\mathcal{S}}$ and because $\pi_{\mathcal{S}}(\boldsymbol{a}' \cdot \boldsymbol{b}') = \boldsymbol{a}' \cdot \pi_{\mathcal{S}}(\boldsymbol{b}')$, χ_k is also invariant under $\pi_{\mathcal{S}}$. Therefore if $\boldsymbol{a}, \boldsymbol{b} \xleftarrow{\$} \psi_k^n$ then for any $0 \leq i < 256$ $(\boldsymbol{a} \cdot \boldsymbol{b})'_i$ can be seen as a sum of 256 independent random variables following the distribution χ_k.

C. Estimation of the probability failure

In order to estimate the probability than the decryption fails we will need to have an estimation on the tail concentration of a sum of independent identically distributed (i.i.d.) random variables. Similarly to [ADPS16b] we use Chernoff bound for this:

Theorem 1 (Chernoff bound): Let $\mathcal{D}$ be a distribution over $\mathbb{R}$ and X be a sum of ℓ i.i.d. random variables $X_1, \ldots, X_\ell$ of law $\mathcal{D}$, then for any $t > 0$ such that $\mathbb{E}[e^{t(X_i)}] < +\infty$ and for any $a \in \mathbb{R}$ it holds that:

$$\mathbb{P}(X \geq a) \leq \exp(-ta + \ell \ln(\mathbb{E}[e^{tX_i}])).$$

Now, in order to ensure (1), and thus the success of the decapsulation, with a certain probability we need to have:

$$\|(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i + (\boldsymbol{e}_3)'_i + (\boldsymbol{\epsilon}_{comp})'_i\|_1 < B_{\text{dec}}$$

for all $0 \leq i < 256$. Since $\boldsymbol{e}_3 \xleftarrow{\$} \psi_k^n$ and because each coefficients of $\boldsymbol{\epsilon}_{comp}$ follows a law almost uniform over $[-q/2t, q/2t]$, the only difficulty comes from the the evaluation of $(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i$. Similarly to [ADPS16b], we use the fact that for any real vector $\boldsymbol{x} \in \mathbb{R}^d$ we have:

$$\|\boldsymbol{x}\|_1 = \max_{\boldsymbol{y} \in \{\pm 1\}^d} \langle \boldsymbol{x} \ , \ \boldsymbol{y} \rangle .$$

As explained in Section IV-A, for any $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}^2$ each $(\boldsymbol{a} \cdot \boldsymbol{b})'_i$ is the sum of 256 products of elements $\boldsymbol{a}'_j \cdot \boldsymbol{b}'_j \in \mathcal{S}$ and thus $(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i = (\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i - (\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i$ is the sum of 512 such products. Seeing an element of $\mathcal{S}$ as a vector of $\mathbb{Z}^{n/256}$ it holds that:

$$\|(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i\|_1 = \max_{\boldsymbol{y} \in \{\pm 1\}^{n/256}} \left\langle (\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i - (\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i \ , \ \boldsymbol{y} \right\rangle \tag{2}$$

where $(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i$ and $(\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i$ can both be seen as a sum of 256 independent random variables following the law χ_k (see Section IV-A). As a consequence, $(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i - (\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i$ can be seen as a sum of 512 i.i.d. random variables. Moreover, remark that because of the different symmetries in χ_k, if X is a random variable of law χ_k, and $\boldsymbol{y} \in \{\pm 1\}^{n/256}$ then the distribution of $\langle X \ , \ \boldsymbol{y} \rangle$ is independent of the choice of $\boldsymbol{y}$. Therefore, because of the linearity of the scalar product $\left\langle (\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i - (\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i \ , \ \boldsymbol{y} \right\rangle$ can be seen as a sum of 512 independent random variables of the form: $\langle X \ , \ \boldsymbol{y} \rangle$ where $X \xleftarrow{\$} \chi_k$.

We have simulated the law of such a scalar product in rational arithmetic using the `GMP`[3] library ([Gt15]) in a `C++` script by computing every possible values of $\langle \boldsymbol{a} \cdot \boldsymbol{b} \ , \ \boldsymbol{y} \rangle$, for $\boldsymbol{a}, \boldsymbol{b} \xleftarrow{\$} \Psi_k^{n/256}$ and $\boldsymbol{y} \in \mathbb{Z}^{n/256}$ the vector containing only $+1$, with their associated probabilities. Note that because of the size of the support of the inputs ($(2k+1)^{n/128}$), this takes quite a long time to simulate (around 70 min on a laptop with an intel i7-4810MQ@2.80GHz for $n = 1024$ and $k = 8$). This is not much of a problem since the law needs only to be computed once and can then be saved.

Once this law and the law of $\boldsymbol{\epsilon}_{comp}$ simulated, we can use Chernoff bound to get:

$$\begin{aligned} &\mathbb{P}\left(\left\langle (\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i + (\boldsymbol{e}_3)'_i + (\boldsymbol{\epsilon}_{comp})'_i \ , \ \boldsymbol{y} \right\rangle \geq B_{\text{dec}}\right) \\ &\quad \leq \exp\left(-tB_{\text{dec}} + 512 \ln \mathbb{E}(e^{tX}) + \tfrac{n}{256} \ln \mathbb{E}(e^{t(Y+Z)})\right) \end{aligned}$$

where $X \xleftarrow{\$} \langle \chi_k \ , \ y \rangle$, $Y \xleftarrow{\$} \psi_k$ and Z follow the law of $\boldsymbol{\epsilon}_{comp}$.

From there, similarly to [ADPS16b], we have deduced a bound on the probability that the decapsulation fails by union-bounding over the $2^{n/256}$ choices of $\boldsymbol{y}$ and the 256 possible i. We have computed this bound on the probability using the `MPFR`[4] library ([FHL+07]) for the floating point arithmetic with a precision set to 2048 bits. Our script for computing the probability is available at: https://gitlab.com/zuccav/alternative-parameters-for-new-hope.

Remark 1: As for `NewHope`, `Kyber` and other related works such as [Saa18], our analysis makes implicitly the assumption that each term are independents when union-bounding over the 256 possibles i. This is clearly not true since they all depend on all the coefficients of the $\boldsymbol{e}_i$s and the $\boldsymbol{s}_j$s. Therefore, while the distribution of each of the terms is the same and precisely analysable, they are not fully indepen-

[3] https://gmplib.org/
[4] https://www.mpfr.org

	Decapsulation error probability	
	NewHope512	NewHope1024
Original analysis ([ERJ+18])	2^{-213}	2^{-216}
Our analysis (using Chernoff bound)	2^{-393}	2^{-413}

TABLE I
DECRYPTION FAILURE PROBABILITY OF NEW HOPE WITH BINOMIAL PARAMETER $k = 8$ AND $q = 12289$.

dents. In [DVV19], the authors show that this independency hypothesis leads to underestimations of the failure probability when using an error correcting code, but suits constructions not using one. Althouh NewHope uses a repetition code, in our case we have simulated exactly the law involving the $n/256$ coefficents correlated via the repetition code (law of $\langle X\ ,\ \boldsymbol{y}\rangle$ for $X \xleftarrow{\$} \chi_k$), hence the hypothesis is only applied on the 256 polynomials of $\mathcal{S}$ which do not interfer with each other via the code. Therefore, according to [DVV19], we can expect our bounds not to be underestimated.

Table I summarizes the results we have obtained and compare them to those of NewHope as they are given in its specifications ([ERJ+18]). As one can see, the probability we obtain is much smaller than the original evaluation which means that one can decrease the size of the modulus while keeping a very small probability of failure.

Remark 2: Note that instead of using Chernoff bound, one could completely simulate the final law, as in Kyber, by computing several convolutions instead. However this method is not as efficient in our case. Indeed since Kyber has a different reconciliation mechanism, they only need to compute the convolutions of 256 random variables $X_i = e \cdot s$, where $e, s \xleftarrow{\$} \Psi_k$, whose law support is of size 7 ($k = 2$ in their case). For NewHope we have to compute the convolutions of 512 random variables following the law of $\langle X\ ,\ \boldsymbol{y}\rangle$ whose support is of size 65 for $k = 2$ and $n = 1024$. This takes around 1h20 in C++ on the aforementionned laptop against 0.5 seconds using Chernoff bound and against a few seconds in python for Kyber. In our experiments we have only observed a difference up to 5 bits between the two variants, hence we rather used Chernoff bound when computing failure probabilities for $k \geq 3$.

V. MULTIPLICATION STRATEGIES FOR SMALLER MODULI

A. Choose the moduli

Using a smaller modulus than $q = 12289$ does not allow to use a full NTT for the dimensions of NewHope. However one can still use a modulus allowing to use NTTs of smaller sizes, for instance the original Kyber modulus $q = 7681$ allows to perform NTTs of size 256. Therefore the idea would be to mix small NTT's with other multiplication algorithms. Since NTT's are the most efficient way to perform polynomial products in high dimensions, one needs to choose moduli with as many NTT levels as possible – i.e. the largest power-of-two n such that $q \equiv 1 \bmod 2n$. Table II presents possible moduli with the number of NTT levels they allow to perform.

modulus q	12289	7681	3329	2017	1601	1409
# NTT levels	10	8	7	4	5	6

TABLE II
POSSIBLE MODULI

A small modulus allows to increase the security of the Ring-LWE instantiation and reduce the size of the elements and thus the communication costs. On the other hand it will increase the failure probability and the efficiency of the implementation directly depends on the number of NTT levels available. Therefore one might consider different moduli given the considered trade-off.

Remark 3: In [ZXZ+18] and [ZPL19] the authors proposed to use a modulus as small as 3329, however since they relied on the original probability analysis they could hardly justify to use smaller moduli. As a consequence the binomial and compression parameters proposed in their work were not optimal.

B. Mutiplication strategies

The idea of mixing classical multiplication algorithms with FFT is well-known and was already studied in [Moe76]. The idea is to split the polynomials $\boldsymbol{a}$ and $\boldsymbol{b}$ into smaller ones and perform the product of the small polynomials with the small NTTs before reconstructing the whole product. The splitting is done between the even and the odd coefficients of our polynomials. More precisely we decompose $\boldsymbol{a}$ as:

$$\begin{aligned}\boldsymbol{a}(X) = \sum_{i=0}^{n-1} a_i \cdot X^i &= \sum_{i=0}^{n/2-1} a_{2i} \cdot X^{2i} + \sum_{i=0}^{n/2-1} a_{2i+1} \cdot X^{2i+1} \\ &= \sum_{i=0}^{n/2-1} a_{2i} \cdot Y^i + X \sum_{i=0}^{n/2-1} a_{2i+1} \cdot Y^i \\ &= \boldsymbol{a}_0(Y) + \boldsymbol{a}_1(Y) \cdot X \text{ with } Y = X^2\end{aligned}$$

with $\boldsymbol{a}_0(Y)$ and $\boldsymbol{a}_1(Y) \in \mathbb{Z}_q[Y]/(Y^{n/2}+1)$, and $\boldsymbol{b}$ similarly. Thus we can write:

$$\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b} = \underbrace{\boldsymbol{a}_0 \cdot \boldsymbol{b}_0}_{\boldsymbol{c}_0} + \underbrace{(\boldsymbol{a}_0 \cdot \boldsymbol{b}_1 + \boldsymbol{a}_1 \cdot \boldsymbol{b}_0)}_{\boldsymbol{c}_1} X + \underbrace{\boldsymbol{a}_1 \cdot \boldsymbol{b}_1}_{\boldsymbol{c}_2} X^2.$$

where each product can be performed with NTTs of size $n/2$. Thus one can just compute $\tilde{\boldsymbol{a}}_i = \mathtt{NTT}(\boldsymbol{a}_i)$ and $\tilde{\boldsymbol{b}}_i$ for $i = 0, 1$ requiring 4 NTTs of size $n/2$ (instead of 2 NTTs of size n in the original case). Once in the NTT domain additions and products can be computed coefficient-wise.

At this point we obtain 3 polynomials of degree $n/2$ in the NTT domain: $\tilde{\boldsymbol{c}}_0$, $\tilde{\boldsymbol{c}}_1$ and $\tilde{\boldsymbol{c}}_2$. Note that since $X^2 = Y$ one can precompute $\mathtt{NTT}(Y)$ (which corresponds to the $n/2$-th primitive roots of unity) and multiply it to $\tilde{\boldsymbol{c}}_2$ and add the result to $\tilde{\boldsymbol{c}}_1$. Like this one has only to compute two inverse NTTs of size $n/2$ instead of 3 to recover the coefficients of $\boldsymbol{c}$ (instead of 1 inverse NTT of size n in the original case).

$$\tilde{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_1 \cdot X + \tilde{\boldsymbol{c}}_2 \cdot X^2 = \underbrace{[\tilde{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_2 \odot \mathtt{NTT}(Y)]}_{\mathtt{NTT}(\boldsymbol{c}_0)} + \underbrace{\tilde{\boldsymbol{c}}_1}_{\mathtt{NTT}(\boldsymbol{c}_1)} \cdot X \quad (3)$$

Note that this also allows to send a polynomial in NTT representation, as done in NewHope, without increasing the communication costs.

The complexity of this approach can be reduced in a straightforward-way by using a subquadratic algorithm such as Karatsuba to reduce the number of multiplications at the price of a few extra-additions [ZPL19]. Of course one can iterate this process on several levels in order to use even smaller NTTs. Note that, although computing 2 NTTs of size $n/2$ is more efficient than computing 1 NTT of size n, the extra operations required to expand and reconstruct the Karatsuba pattern tend to make this approach more costly. As a direct consequence, the efficiency of the approach will decrease with the number of Karatsuba levels used.

Overall this approach consists in using one or several levels of classical multiplications algorithms on the top and perform the small products with the NTTs ([ZXZ+18]). This can be done the other way around by performing big, but uncomplete NTTs on the top as in [LS19]:

$$\mathcal{R}_q \xrightarrow{\cong} \mathbb{Z}_q[X]/(X^2-\zeta)\times\cdots\mathbb{Z}_q[X]/(X^2-\zeta^{2n-1}).$$

In this case, since X^n+1 does not factorize in linear terms, the product cannot be performed coefficient wise but modulo the $X^2-\zeta^i$s instead. Once again this can be iterated on ℓ levels so that one would have to perform product modulo $X^{2^\ell}-\zeta^i$ on the bottom. It is shown in [RJL+19], that this approach can be more efficient than a complete NTT for small values of ℓ ($\ell=1$ or $\ell=2$) if correctly implemented. In particular the product modulo the $X^2-\zeta^i$s can be done very efficiently on a vectorized implementation.

C. Mixing the strategies

As shown in Table II, we need to perform up to 6 levels of multiplications without NTTs depending on the chosen modulus. Performing the whole 6 levels of classical multiplications on the top or the bottom would considerably increase the computational cost. Therefore we are going to mix both strategies with ℓ_1 levels of Karatsuba on the top and ℓ_2 levels naive multiplication on the bottom.

Determining the optimal values of ℓ_1 and ℓ_2 is not straightforward. Indeed, the number of operations one can stack on a 16-bit word without performing modular reduction depends on the size of the modulus. Moreover the cost of the modular reductions and their efficiency depend on the shape of the modulus (see [Sei18]). Additionnally, on a vectorized implementation, the cost of an algorithm cannot be restricted only to the number of operations to perform. One must rather consider the number of registers available, of instructions and loadings to perform, etc.... Therefore, the optimal values of ℓ_1 and ℓ_2 has been determinated experimentally for every moduli, by testing every possible combination.

Remark 4: One could think that performing a Karatsuba product would be more efficient than a naive algorithm for the product modulo $X^{2^{\ell_2}}-\zeta^i$ on the bottom. However on a vectorized implementation, for small values of ℓ_2 we have noticed that the naive algorithm was more efficient since it can be implemented very efficiently.

VI. Global parameters and experimental results

A. Security parameters

In practice, NewHope security is evaluated through the difficulty to recover the secret $\boldsymbol{s}$ from a Ring-LWE sample $(\boldsymbol{a},\boldsymbol{b})=(\boldsymbol{a},\boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{e})$. The difficulty of this problem depends on the ratio between the size of the modulus and the standard deviation of the error distribution, in our case $\sqrt{k/2}$, the smaller the ratio, the better the security. Therefore by using a smaller modulus the difficulty of the problem is increased, however not reducing the binomial parameter k would quickly result in a failure probability too large which could lead to serious attacks (see [DGJ+19]).

As a consequence, we have chosen a binomial parameter k as large as possible for the security of the scheme while keeping the failure probability of the protocol around 2^{-200} thus similar to the evaluation of [ERJ+18] over the original parameters. In order to increase the gain over the bandwidth requirement we have also reduced, when possible, the compression parameter t.

Since the conservative analysis of NewHope ([ADPS16b]), is currently used in most of the NIST submissions based on lattices. We have computed the evaluated post-quantum bits of security using the same python script[5] than for the original protocol. However the indicated failure probability has been computed with our analysis (see Section IV).

B. The particular cases of $q=1601$ and $q=1409$

The modulus $q=2017$ fits on 11-bits but only offers 4 levels of NTT and has resulted in poor performance. Hence we have instead selected two other moduli: $q=1601$ and $q=1409$ allowing to perform 1 or 2 extra levels of NTT respectively. However choosing $q=1601$ and $k=2$ would result in a probability of failure around 2^{-129}, which is far too large and choosing $k=1$ would considerably reduce the entropy of the samples and thus lower the security. Finally increasing the compression parameter t would cancel the benefit over the bandwidth requirement.

Since most of the error size comes from the products $\boldsymbol{s}_1\cdot\boldsymbol{e}_2-\boldsymbol{s}_2\cdot\boldsymbol{e}_1$ and because the Ring-LWE security depends essentially on the size of the error rather than the size of the secret. We have chosen to use different binomial parameters k_s and k_e for the secrets and the errors respectively. As a consequence, choosing a binary secret – i.e. $k_s=1$ – and a larger error $k_e=3$ (resp. $k_e=2$) increases the security of the scheme while reducing the error probability to 2^{-170} (resp. 2^{-197}) for $q=1601$ (resp. $q=1409$). Although for $q=1601$ this failure probability is far above the 2^{-200} treshold, it is comparable to the failure probability of Kyber [RJL+19]. Finally, note that when using a different size of errors and secrets, and more particularly with sparse secrets, one needs to adapt the security evaluation by including weights on the lattice to reduce ([BG14], [Alb17]) as done in Round5 [HSS+19]. Hence this is what we have done for our security evaluation of the moduli $q=1601$ and $q=1409$.

[5] available in the NIST submission package: https://newhopecrypto.org/resources.shtml

Eventually note that there are two other smaller moduli which could have been considered for efficient implementation: 1153 and 769 since they offer 6 and 7 levels of NTTs respectively. Both moduli would increase the hardness of the underlying lattice reduction, and $q = 769$ would also improve further the compactness and the performance of the scheme. However their associated failure probability for $n = 1024$ is 2^{-139} and 2^{-122} for $k_s = 1$ and $k_e = 2$, $k_e = 1$ respectively. Nonetheless such a high failure probability would expose the protocol to attacks exploiting decapsulation failures [DVV19]. Therefore, we believe these parameters would weaken the security of the whole protocol and we do not recommend them for an implementation until a precise estimation of the amount of computations required to break them is given.

C. Experimental results

Table III summarizes the different parameters and features of the original version of `NewHope512` and `NewHope1024` together with those of the alternative versions we propose.

In order to estimate the efficiency of our multiplication strategy we have measured the number of cycles required to compute a full product: $\boldsymbol{c} = \texttt{invNTT}(\texttt{NTT}(\boldsymbol{a}) \odot \texttt{NTT}(\boldsymbol{b}))$, and compared it to the classical full NTT approach as proposed in [Sei18]. Our vectorized implementation of NTT is based on [Sei18]. The ℓ_1 levels of Karatsuba have been implemented iteratively so that we could track the size of the coefficients at any point and only perform the modular reductions when needed using similar reductions than [Sei18].

The values presented in Table III were measured on an average of 2^{20} tests run using our vectorized implementation on a laptop endowed with an Intel(R) Core(TM) i7-4810MQ CPU @ 2.80GHz using `AVX2` with Turbo Boost and Hyper-Threading turned-off. Our code was compiled with `gcc` version 7.3.0 using the flags: `-O3 -funroll-loops -fomit-frame-pointer -march=native`.

Our experiments confirm the observations of [LS19]: using smaller NTTs on the top with a naive product on the bottom is more efficient than a full NTT with a gain up to 15% (resp. (13%)) for $n = 512$ (resp. $n = 1024$). Because registers in `AVX2` are 256-bit long, they can only handle 16 coefficients on 16 bits words at a time. In particular, having $\ell_2 > 4$ levels is not interesting since one would have to multiply polynomials of degree $2^{\ell_2} > 16$. Hence it would require two registers to store the coefficients of only one polynomial degrading considerably the performance.

They also highlight the fact that the modulus $q = 2017$ does not lead to good performance +38% (resp. +79%) for $n = 512$ (resp. $n = 1024$), and that the additional level of NTT available with $q = 1601$ and $q = 1409$ improves considerably the performance +7.6% (resp. +35%) for $n = 512$ (resp. $n = 1024$) for $q = 1601$ and −9.5% (resp. +3.2%) for $n = 512$ (resp. $n = 1024$) for $q = 1409$.

Nonetheless we have shown that those parameters allow to obtain competitive performance while gaining around 19% over the bandwidth requirements and up to 8% in security. Moreoever the two moduli 7681 and 3329 do not have any drawback, except a negligible 1% loss of security for 3329. Furthermore the gain of more than 10% on performance comes without having to implement any level of Karatsuba and thus leads to a very simple implementation. If one desires to gain a little bit more on the bandwidth requirements, one can use the modulus $q = 1409$, which brings a speed-up of 9.5% for $n = 512$ and a small overcost of 3.2% for dimensions 1024. On the other hand the modulus $q = 2017$ leads to very poor performance and might not be interesting from a practical point of view.

Conclusion

In this work we refine the probability analysis made in [ADPS16b] and show that we can use tighter parameters in order to increase the security, the compactness and sometimes the performance of `NewHope`. We propose five alternative sets of parameters using smaller moduli. However these moduli do not allow to use the full efficient NTT algorithm in the protocol dimensions. Nonetheless we show that by mixing smaller NTTs with different multiplication algorithms we obtain very competitive performance when compared to the state-of-the-art approach (gain of 15% for some of them) on a vectorized, constant time, implementation. While we are competitive in term of performance our alternative parameters allow to increase the security of the protocol up to 8% and reduce the bandwidth requirement by more than 19%. Furthermore some sets of parameters improved the three features of the scheme: compactness, security, performance while the other improve even more the compactness and security but lead to worse performance. For all these sets of parameters we evaluate the failure probability around 2^{-200} as originally evaluated ([ERJ+18]), except for $q = 1601$ where it is as big as 2^{-170}, similarly to `Kyber`, which should be enough to be protected against attacks such as the one mentionned in [DGJ+19].

References

[ADPS16a] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Newhope without reconciliation. Cryptology ePrint Archive, Report 2016/1157, 2016. https://eprint.iacr.org/2016/1157.

[ADPS16b] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum Key Exchange—A New Hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, 2016. USENIX Association.

[Alb17] Martin R. Albrecht. On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 103–129, Cham, 2017. Springer International Publishing.

[BDK+18] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 353–367, April 2018.

[BG14] Shi Bai and Steven D. Galbraith. Lattice Decoding Attacks on Binary LWE. In Willy Susilo and Yi Mu, editors, *Information Security and Privacy*, pages 322–337, Cham, 2014. Springer International Publishing.

[DGJ+19] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 565–598, Cham, 2019. Springer International Publishing.

n	q	k	t	probability of failure	post-quantum bits of security	bandwidth (Bytes)	$\ell_1 \vert \ell_2$	clock cycles
512	12289	8	8	2^{-393} (C)	101	2016	0|0	4253
	7681	5	4	2^{-194} (C)	101 (+0%)	1824 (−9.5%)	0|1	3649 (−15%)
	3329	2	4	2^{-239} (S)	100 (−1%)	1696 (−15.8%)	0|2	3640 (−15%)
	2017	2	8	2^{-197} (S)	108 (+7%)	1632 (−19%)	2|3	5932 (+39%)
	1601	1|3	8	2^{-170} (S)	109 (+8%)	1632 (−19%)	0|4	4578 (+7.6%)
	1409	1|2	8	2^{-197} (S)	107 (+6%)	1632 (−19%)	0|3	3849 (−9.5%)
1024	12289	8	8	2^{-413} (C)	233	4000	0|0	9340
	7681	5	4	2^{-208} (C)	233 (+0%)	3616 (−9.6%)	0|2	8157 (−13%)
	3329	2	4	2^{-249} (S)	230 (−1%)	3360 (−16%)	0|3	8441 (−10%)
	2017	2	8	2^{-204} (S)	245 (+5%)	3232 (−19.2%)	3|3	16656 (+78%)
	1601	1|3	8	2^{-175} (S)	245 (+5%)	3232 (−19.2%)	2|3	12588 (+35%)
	1409	1|2	8	2^{-202} (S)	242 (+4%)	3232 (−19.2%)	1|3	9641 (+3.2%)

TABLE III

PERFORMANCE OF NEWHOPE512 AND NEWHOPE1024 WITH OUR ALTERNATIVE PARAMETERS. (C) AND (S) DENOTE WHETHER THE PROBABILITY OF FAILURE WAS COMPUTED USING CHERNOFF BOUND OR ONLY SIMULATIONS RESPECTIVELY.

[DVV19] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. The Impact of Error Dependencies on Ring/Mod-LWE/LWR Based Schemes. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 103–115, Cham, 2019. Springer International Publishing.

[ERJ+18] Alkim Erdem, Avenzi Roberto, Bos Joppe, Ducas Léo, de la Piedra Antonio, Pöppelmann Thomas, Schwabe Peter, and Stebila Douglas. *NewHope Algorithm Specifications and Supporting Documentation*, 1.0 edition, 2018. https://newhopecrypto.org/data/NewHope_2018_06_14.pdf.

[FHL+07] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. MPFR: A Multiple-precision Binary Floating-point Library with Correct Rounding. *ACM Trans. Math. Softw.*, 33(2), June 2007.

[Gt15] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.1.0 edition, 2015. http://gmplib.org/.

[HSS+19] Baan Hayo, Bhattacharya Sauvik, Fluhrer Scott, Garcia-Morchon Oscar, Laarhoven Thijs, Player Rachel, Rietman Ronald, Saarinen Markku-Juhani O., Tolhuizen Ludo, Torre-Arce José Luis, and Zhenfei Zhang. *Round5: KEM and PKE based on (Ring) Learning with Rounding*, 2019. https://round5.org/Supporting_Documentation/Round5_Submission.pdf.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *On Ideal Lattices and Learning with Errors over Rings*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *A Toolkit for Ring-LWE Cryptography*, pages 35–54. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, Jun 2015.

[LS19] Vadim Lyubashevsky and Gregor Seiler. NTTRU: Truly Fast NTRU Using NTT. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):180–201, May 2019.

[Moe76] Robert T. Moenck. Practical Fast Polynomial Multiplication. In *Proceedings of the Third ACM Symposium on Symbolic and Algebraic Computation*, SYMSAC '76, pages 136–148, New York, NY, USA, 1976. ACM.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[Reg05] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[RJL+19] Avenzi Roberto, Bos Joppe, Ducas Léo, Klitz Eike, Lepoint Tancréde, Lyubaschevsky Vadim, Schanck John M., Schwabe Peter, Seiler Gregor, and Stehlé Damien. *CRYSTALS-Kyber (version 2.0) – Submission to round 2 of the NIST post-quantum project.*, 2.0 edition, 2019. https://pq-crystals.org/kyber/resources.shtml.

[Saa18] Markku-Juhani O. Saarinen. HILA5: On Reliability, Reconciliation, and Error Correction for Ring-LWE Encryption. In Carlisle Adams and Jan Camenisch, editors, *Selected Areas in Cryptography – SAC 2017*, pages 192–212, Cham, 2018. Springer International Publishing.

[Sei18] Gregor Seiler. Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography. Cryptology ePrint Archive, Report 2018/039, 2018. https://eprint.iacr.org/2018/039.

[ZPL19] Yiming Zhu, Yanbin Pan, and Zhen Liu. When NTT Meets Karatsuba: Preprocess-then-NTT Technique Revisited. Cryptology ePrint Archive, Report 2019/1079, 2019. https://eprint.iacr.org/2019/1079.

[ZXZ+18] Shuai Zhou, Haiyang Xue, Daode Zhang, Kunpeng Wang, Xianhui Lu, Bao Li, and Jingnan He. Preprocess-then-ntt technique and its applications to kyber and newhope. In Fuchun Guo, Xinyi Huang, and Moti Yung, editors, *Inscrypt*, volume 11449 of *Lecture Notes in Computer Science*, pages 117–137. Springer, 2018.