CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM

Joppe Bos*, Léo Ducas†, Eike Kiltz‡, Tancrède Lepoint§, Vadim Lyubashevsky¶,
John M. Schanck‖, Peter Schwabe**, Gregor Seiler††, Damien Stehlé‡‡,
*NXP Semiconductors, Belgium. Email: joppe.bos@nxp.com
†CWI Amsterdam, The Netherlands. Email: ducas@cwi.nl
‡Ruhr-University Bochum, Germany. Email: eike.kiltz@rub.de
§SRI International, USA. Email: tancrede.lepoint@sri.com
¶IBM Research Zurich, Switzerland. Email: vad@zurich.ibm.com
‖University of Waterloo, Canada. Email: jschanck@uwaterloo.ca
**Radboud University, The Netherlands. Email: peter@cryptojedi.org
††IBM Research Zurich, Switzerland. Email: grs@zurich.ibm.com
‡‡ENS de Lyon, France. Email: damien.stehle@ens-lyon.fr

***Abstract*—Rapid advances in quantum computing, together with the announcement by the National Institute of Standards and Technology (NIST) to define new standards for digital-signature, encryption, and key-establishment protocols, have created significant interest in post-quantum cryptographic schemes.**

This paper introduces Kyber (part of CRYSTALS – *Cryptographic Suite for Algebraic Lattices* – a package submitted to NIST post-quantum standardization effort in November 2017), a portfolio of post-quantum cryptographic primitives built around a key-encapsulation mechanism (KEM), based on hardness assumptions over module lattices. Our KEM is most naturally seen as a successor to the NEWHOPE KEM (Usenix 2016). In particular, the key and ciphertext sizes of our new construction are about half the size, the KEM offers CCA instead of only passive security, the security is based on a more general (and flexible) lattice problem, and our optimized implementation results in essentially the same running time as the aforementioned scheme.

We first introduce a CPA-secure public-key encryption scheme, apply a variant of the Fujisaki–Okamoto transform to create a CCA-secure KEM, and eventually construct, in a black-box manner, CCA-secure encryption, key exchange, and authenticated-key-exchange schemes. The security of our primitives is based on the hardness of Module-LWE in the classical and quantum random oracle models, and our concrete parameters conservatively target more than 128 bits of post-quantum security.

1. Introduction

There has been an increased interest in post-quantum cryptographic schemes triggered by recent advances in quantum computing [35] and the announcement by the National Institute of Standards and Technology (NIST) to define new standards for digital-signature, encryption, and key-establishment protocols [28]. Constructions based on the hardness of lattice problems are considered to be one of the leading candidates to replace the currently used schemes based on the believed hardness of the traditional number-theoretic problems such as integer factorization and discrete logarithms.

Lattice cryptography initially gained a lot of interest in the theoretical community due to the fact that the designs for cryptographic constructions were accompanied by security proofs based on *worst-case* instances of lattice problems. The first lattice-based encryption scheme was proposed by Ajtai and Dwork [1]. This scheme was later simplified and improved upon by Regev in [67], [68]. One of the major achievements of Regev's work was the introduction of an intermediate problem – the Learning With Errors (LWE) Problem – which was relatively simple to use in cryptographic constructions and asymptotically at least as hard as some standard worst-case lattice problems [61], [25].

The LWE assumption states that it is hard to distinguish from uniform the distribution $(\mathbf{A}, \mathbf{As} + \mathbf{e})$, where $\mathbf{A}$ is a uniformly-random matrix in $\mathbb{Z}_q^{m \times n}$, $\mathbf{s}$ is a uniformly-random vector in $\mathbb{Z}_q^n$, and $\mathbf{e}$ is a vector with random "small" coefficients chosen from some distribution. Applebaum et al. [6] showed that the secret $\mathbf{s}$ in the LWE problem does not need to be chosen uniformly at random: the problem remains hard if $\mathbf{s}$ is chosen from the same narrow distribution as the errors $\mathbf{e}$. Based on the idea from the NTRU cryptosystem [43] of working with elements over polynomial rings rather than over the integers, and following a series of works on this topic [58], [56], [63], [71], Lyubashevsky et al. [57] showed that it is also hard to distinguish a variant of the LWE distribution from the uniform one over certain polynomial rings, thus defining the Ring-LWE assumption.

The combination of all of the above results finally led

to the cryptosystem in Section 3.[1] Setting the parameter k to 1 and defining $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ makes the scheme a Ring-LWE cryptosystem as originally defined in [57], whereas setting the ring R_q to $\mathbb{Z}_q$, makes the scheme an LWE-based one.[2] If one sets the ring R_q to some polynomial ring of dimension greater than 1 and sets $k > 1$, then the scheme is based on the hardness of the Module-LWE problem [24], [54]. The number of bits that can be transmitted is related to the dimension of the ring, thus using a ring R_q of larger degree n allows one to transmit more bits, and this is the main reason that Ring-LWE encryption is more efficient than LWE encryption. On the other hand, having a smaller k implies more algebraic structure, making the scheme potentially susceptible to more avenues of attack. Nevertheless, at this point in time, it is unknown how to exploit the algebraic structure of Ring-LWE and concrete parameters are chosen according to the corresponding LWE problem of dimension $k \cdot n$.

This cryptosystem design was also applied to build a CPA-secure KEM by Ding et al. [36] and Peikert [62]. The main difference between this KEM and the encryption scheme is in how the parameter v is defined in line 6 of the encryption algorithm (Algorithm 2). The advantage of the constructions in [36], [62] is that if one would like to construct a CPA-secure KEM transmitting a b-bit key, then the ciphertext is b bits shorter, which is about a 3% savings for typical parameters.[3] If one wishes to construct a CCA-secure KEM, however, this advantage disappears since typical transformations from CPA-secure KEMs to CCA-secure ones implicitly go through a CPA-secure encryption scheme, which will result in adding b bits to the KEM. Since in this paper we are only concerned with CCA-secure constructions, we find it simpler to start directly from the CPA-secure encryption scheme design in Section 3.

The above designs based on Ring-LWE have resulted in many recent concrete proposals accompanied by practical implementations. The instantiation presented in [21] is based on Ring-LWE and was subsequently improved in [4], [55], which resulted in an experiment by Google where they used this key-exchange protocol in their Chrome Canary browser from July to November 2016 [23], [53]. Although the Ring-LWE problem results in very practical key-sizes and protocol communication, the additional algebraic structure might inspire less confidence in the underlying security. This was the motivation to study a very similar practical instantiation of a key-exchange protocol but based on LWE in [20], or to propose an efficient implementation of a CCA-secure KEM over a different ring [13].

1. It should be noted that this cryptoscheme design, as well as the result from [6] applied to the Learning Parity with Noise (LPN) problem, was already present much earlier in the work of Alekhnovich [3] in which he constructed a cryptosystem based on the hardness of the LPN problem. The LWE problem is a generalization of LPN and results in more efficient cryptosystems.

2. The original cryptosystem was not optimized and did not include the "bit-dropping" $\mathsf{Compress}_q$ function in the key generation and encryption algorithms; but this is considered folklore. Dropping bits to reduce ciphertext size in (Ring)-LWE cryptosystems was first (to our knowledge) mentioned in [61] and subsequently used in a variety of concrete instantiations of (Ring)-LWE cryptosystems and key exchange schemes derived from [68] and [57] (c.f. [65], [4], [48], [5], [30]). Our $\mathsf{Compress}_q$ function uses the original definition from [61].

3. It was mentioned in [62, Sec. 4] that the ciphertext in the KEM goes down by a factor of two compared to encryption schemes. However, this applies only to the naive instantiations of encryption schemes where the "bit-dropping" $\mathsf{Compress}_q$ function is not applied to v in line 6 of Algorithm 2.

1.1. Our contribution

Our main contribution is a highly-optimized instantiation of a CCA-secure KEM called Kyber, which is based on the hardness of Module-LWE.[4] More precisely, we instantiate a CPA-secure PKE scheme Kyber.CPA in Section 3, then apply a variant of the Fujisaki–Okamoto transform to create a CCA-Secure KEM Kyber in Section 4. The security reduction from the CPA-secure scheme is tight in the random-oracle model, but non-tight is the quantum-random-oracle model [44]. From a CCA-secure KEM, one can construct, in a black-box manner, CCA-secure encryption (Kyber.Hybrid), key exchange (Kyber.KE), and authenticated-key-exchange (Kyber.AKE) schemes. Our resulting schemes are as efficient as ones that are based on Ring-LWE, but have additional flexibility and security advantages.

Flexibility. One of the most expensive operations in lattice-based schemes over rings is polynomial multiplication. If a scheme is based on the Ring-LWE assumption (i.e., with $k = 1$ in Algorithm 2), then if one wants to vary the security parameter related to the scheme, one would need to change the ring R_q and re-implement all the ring operations. With our design, where we only work over the ring $R_q = \mathbb{Z}_{7681}[X]/(X^{256} + 1)$, there is only one ring over which operations need to be optimized. Increasing and decreasing the security of the scheme can then be done simply by changing the dimension k of the matrix. Our proposed conservative parameters, which we believe have very generous margins for 128-bit post-quantum security, use $k = 3$. This is the scheme we recommend using for long-term security. But if one only needs short-term security, we believe that today (and probably for the near future) one can safely use $k = 2$ for which we conservatively estimate 102-bit post quantum security. This latter parameter set will reduce the communication size of the key exchange by around 33% and considerably speed up the scheme. The main building blocks of the two schemes are exactly the same, and any optimized software / hardware used for efficient multiplication in R_q can be re-used.

Security. There have been recent attacks exploiting the algebraic structure of cyclotomic ideal lattices [26], [17], [32], [33], and others that exploit the presence of dense sublattices in NTRU lattices [2], [50]. In these attacks, it appears that the dimension of the module makes a big difference. In particular, the quantum attacks on finding short vectors in ideals currently do not extend to Ring-LWE [26], [17],

4. Our scheme is in fact an optimization that slightly deviates from the Module-LWE assumption. We discuss this in Section 3.

[32], [33]. The obstacle seems to be that solutions to the shortest vector problem in ideal lattices are ring elements, whereas solutions to the Ring-LWE problem are elements in a module of dimension 2. In that respect, solutions to Module-LWE are in a module of dimension $k+1$. Similarly, the larger module dimension also decreases the relative dimension of the dense sub-lattice, making the attack of [50] inapplicable. Based on the recent cryptanalytic progress, it therefore seems that practical attacks are less likely to appear against Module-LWE than against Ring-LWE or NTRU.

High performance. As we previously mentioned, the main reason that Ring-LWE is preferred to LWE in practical applications is because it allows for a larger message to be transmitted in the same amount of communication. We show that the flexibility and security improvements by moving from Ring-LWE to Module-LWE come at almost no cost. In particular, since public-key protocols only need to transmit 256 bits of information, it is unnecessary to work with rings that are greater than dimension 256 in order to be able to transmit one bit per coefficient of a ring element. Thus the key and message sizes of our protocols versus those based on Ring-LWE are not affected.

The one part where using a $k > 1$ is less efficient than $k = 1$ is when dealing with the $k \times k$ random matrix $\mathbf{A}$. If one uses $k = 1$ and a ring of dimension n, then the representation of $\mathbf{A}$ is $k^2 n = n$ elements in $\mathbb{Z}_q$. On the other hand, if one uses $k = 3$ and a ring of dimension $n/3$ (thus keeping the lattice-reduction security the same), then $\mathbf{A}$ requires $k^2 n = 3n$ elements in $\mathbb{Z}_q$ to represent. Since the matrix $\mathbf{A}$ is never stored, but rather expanded from some seed ρ using an extendable output function (XOF), this disadvantage only manifests in the slight increase in the running time used in the expansion. This is to some extent mitigated because the k^2 entries of the matrix $\mathbf{A}$ can be expanded independently, which enables very efficient vectorization of the XOF computation.

Take away. In this paper, we propose and implement a portfolio of post-quantum cryptographic primitives (CPA-secure encryption, CCA-secure KEM, CCA-secure public-key encryption, key exchange and authenticated key exchange) based on the hardness of Module-LWE in the classical and quantum random-oracle models. Our schemes are as efficient as the ones based on Ring-LWE, but also feature flexibility and security advantages.

Availability of software. We place all software described in this paper into the public domain to maximize reusability of our results. It is available for download on GitHub: https://github.com/pq-crystals/kyber.

Full version of the paper. The full online version of the paper is available at http://pq-crystals.org/kyber/resources.shtml.

2. Preliminaries

All our algorithms are probabilistic. If b is a string, then $a \leftarrow \mathsf{A}(b)$ denotes the output of algorithm A when run on input b; if A is deterministic, then a is a fixed value and we write $a := \mathsf{A}(b)$. We use the notation $b := \mathsf{A}(b; r)$ to make the randomness r of a probabilistic algorithm A explicit.

2.1. Cryptographic definitions

A public-key encryption scheme $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a triple of probabilistic algorithms together with a message space $\mathcal{M}$. The key-generation algorithm KeyGen returns a pair (pk, sk) consisting of a public key and a secret key. The encryption algorithm Enc takes a public key pk and a message $m \in \mathcal{M}$ to produce a ciphertext c. Finally, the deterministic decryption algorithm Dec takes a secret key sk and a ciphertext c, and outputs either a message $m \in \mathcal{M}$ or a special symbol $\perp$ to indicate rejection. Following [44], we say that PKE is $(1-\delta)$-correct if $\mathbf{E}[\max_{m \in \mathcal{M}} \Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m]] \geq 1 - \delta$, where the expectation is taken over $(pk, sk) \leftarrow \mathsf{KeyGen}()$ and the probability is taken over the random coins of Enc.

We recall the standard security notions for public-key encryption of indistinguishability under chosen-ciphertext and chosen-plaintext attacks (IND-CCA and IND-CPA) [66]. The advantage of an adversary A is defined as $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{PKE}}(\mathsf{A}) =$

$$\left| \Pr\left[b = b' \ : \ \begin{array}{l} (pk, sk) \leftarrow \mathsf{KeyGen}(); \\ (m_0, m_1, s) \leftarrow \mathsf{A}^{\textsc{Dec}(\cdot)}(pk); \\ b \leftarrow \{0,1\}; c^* \leftarrow \mathsf{Enc}(pk, m_b); \\ b' \leftarrow \mathsf{A}^{\textsc{Dec}(\cdot)}(s, c^*) \end{array} \right] - \frac{1}{2} \right|,$$

where the decryption oracle is defined as $\textsc{Dec}(\cdot) := \mathsf{Dec}(sk, \cdot)$. We further require that $|m_0| = |m_1|$ and that in the second phase A is not allowed to query $\textsc{Dec}(\cdot)$ with the challenge ciphertext c^*. The advantage $\mathbf{Adv}^{\mathrm{cpa}}_{\mathsf{PKE}}(\mathsf{A})$ of an adversary A is defined as $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{PKE}}(\mathsf{A})$, with the modification that A cannot query the decryption oracle.

A key-encapsulation scheme $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ is a triple of probabilistic algorithms together with a key space $\mathcal{K}$. The key-generation algorithm KeyGen returns a pair (pk, sk) consisting of a public key and a secret key. The encapsulation algorithm Encaps takes a public key pk to produce a ciphertext c and a key $K \in \mathcal{K}$. Finally, the deterministic decapsulation algorithm Decaps takes a secret key sk and a ciphertext c, and outputs either a key $K \in \mathcal{K}$ or a special symbol $\perp$ to indicate rejection. We say that KEM is $(1-\delta)$-correct if $\Pr[\mathsf{Decaps}(sk, c) = K : (c, K) \leftarrow \mathsf{Encaps}(pk)] \geq 1 - \delta$, where the probability is taken over $(pk, sk) \leftarrow \mathsf{KeyGen}()$ and the random coins of Encaps.

We recall the standard security notion for key encapsulation of indistinguishability under chosen-ciphertext attack. The advantage of an adversary A is defined as $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{KEM}}(\mathsf{A}) =$

$$\left| \Pr\left[b = b' \ : \ \begin{array}{l} (pk, sk) \leftarrow \mathsf{KeyGen}(); \\ b \leftarrow \{0,1\}; \\ (c^*, K_0^*) \leftarrow \mathsf{Encaps}(pk); \\ K_1^* \leftarrow \mathcal{K}; \\ b' \leftarrow \mathsf{A}^{\textsc{Decaps}(\cdot)}(pk, c^*, K_b^*) \end{array} \right] - \frac{1}{2} \right|,$$

where the DECAPS oracle is defined as $\text{DECAPS}(\cdot) \coloneqq \mathsf{Decaps}(sk, \cdot)$. We further require that A is not allowed to query $\text{DECAPS}(\cdot)$ with the challenge ciphertext c^*.

In the random oracle model [11], the adversary A is additionally given access to a random oracle that it can query up to q_H times. If the adversary has access to a quantum computer, it is realistic to model its access to all "offline primitives" (such as hash functions) in a quantum setting. Concretely, in the quantum random oracle model [19] the adversary has access to a quantum random oracle (also called quantum accessible random oracle) that can be queried up to q_H times on arbitrary quantum superpositions of input strings.

2.2. Rings and distributions

Let R and R_q denote the rings $\mathbb{Z}[X]/(X^n+1)$ and $\mathbb{Z}_q[X]/(X^n+1)$, respectively, where $n = 2^{n'-1}$ such that X^n+1 is the $2^{n'}$-th cyclotomic polynomial. Throughout this paper, the values of n, n' and q are 256, 9 and 7681, respectively. Regular font letters denote elements in R or R_q (which includes elements in $\mathbb{Z}$ and $\mathbb{Z}_q$) and bold lower-case letters represent vectors with coefficients in R or R_q. By default, all vectors will be column vectors. Bold upper-case letters are matrices. For a vector $\mathbf{v}$ (or matrix $\mathbf{A}$), we denote by $\mathbf{v}^T$ (or $\mathbf{A}^T$) its transpose.

Modular reductions. For an even (resp. odd) positive integer α, we define $r' = r \bmod^{\pm} \alpha$ to be the unique element r' in the range $-\frac{\alpha}{2} < r' \leq \frac{\alpha}{2}$ (resp. $-\frac{\alpha-1}{2} \leq r' \leq \frac{\alpha-1}{2}$) such that $r' = r \bmod \alpha$. For any positive integer α, we define $r' = r \bmod^{+} \alpha$ to be the unique element r' in the range $0 \leq r' < \alpha$ such that $r' = r \bmod \alpha$. When the exact representation is not important, we simply write $r \bmod \alpha$.

Rounding. For an element $x \in \mathbb{Q}$ we denote by $\lceil x \rfloor$ rounding of x to the closest integer with ties being rounded up.

Sizes of elements. For an element $w \in \mathbb{Z}_q$, we write $\|w\|_\infty$ to mean $|w \bmod^{\pm} q|$. We now define the ℓ_∞ and ℓ_2 norms for $w = w_0 + w_1 X + \ldots + w_{n-1}X^{n-1} \in R$:

$$\|w\|_\infty = \max_i \|w_i\|_\infty, \|w\| = \sqrt{\|w_0\|_\infty^2 + \ldots + \|w_{n-1}\|_\infty^2}.$$

Similarly, for $\mathbf{w} = (w_1, \ldots, w_k) \in R^k$, we define

$$\|\mathbf{w}\|_\infty = \max_i \|w_i\|_\infty, \quad \|\mathbf{w}\| = \sqrt{\|w_1\|^2 + \ldots + \|w_k\|^2}.$$

Distributions. For a set S, we write $s \leftarrow S$ to denote that s is chosen uniformly at random from S. If S is a probability distribution, then this denotes that s is chosen according to the distribution S.

Extendable output function. Suppose that Sam is an extendable output function, that is a function on bit strings in which the output can be extended to any desired length. If we would like Sam to take as input x and then produce a value y that is distributed according to distribution S (or uniformly over a set S), we write $y \sim S \coloneqq \mathsf{Sam}(x)$. It is important to note that this procedure is completely deterministic: a given x will always produce the same y. For simplicity we assume that the output distribution of Sam is perfect, whereas in practice Sam will be implemented using random oracles and produces an output that is statistically close to the perfect distribution.

Binomial distribution. We define the centered binomial distribution B_η for some positive integer η as follows:

$$\text{Sample } \{(a_i, b_i)\}_{i=1}^{\eta} \leftarrow (\{0,1\}^2)^\eta \text{ and output } \sum_{i=1}^{\eta}(a_i - b_i).$$

If v is an element of R, we write $v \leftarrow \beta_\eta$ to mean that $v \in R$ is generated from a distribution where each of its coefficients is generated according to B_η. Similarly, a k-dimensional vector of polynomials $\mathbf{v} \in R^k$ can be generated according to the distribution β_η^k.

Compression and Decompression. We now define a function $\mathsf{Compress}_q(x, d)$ that takes an element $x \in \mathbb{Z}_q$ and outputs an integer in $\{0, \ldots, 2^d - 1\}$, where $d < \lceil \log_2(q) \rceil$. We furthermore define a function $\mathsf{Decompress}_q$, such that

$$x' = \mathsf{Decompress}_q\big(\mathsf{Compress}_q(x, d), d\big) \tag{1}$$

is an element close to x – more specifically

$$|x' - x \bmod^{\pm} q| \leq B_q \coloneqq \left\lceil \frac{q}{2^{d+1}} \right\rfloor .$$

The functions satisfying these requirements are defined as:

$$\mathsf{Compress}_q(x, d) = \lceil (2^d/q) \cdot x \rfloor \bmod^{+} 2^d ,$$
$$\mathsf{Decompress}_q(x, d) = \lceil (q/2^d) \cdot x \rfloor .$$

If x' is a function of x as in Eq. (1), then for a randomly chosen $x \leftarrow \mathbb{Z}_q$, the distribution of

$$x' - x \bmod^{\pm} q$$

is almost uniform over the integers of magnitude at most B_q. In particular, this distribution has equal weight over integers of magnitude at most $B_q - 1$ and has a smaller weight on the integer(s) of magnitude B_q.

When $\mathsf{Compress}_q$ or $\mathsf{Decompress}_q$ is used with $x \in R_q$ or $\mathbf{x} \in R_q^k$, the procedure is applied to each coefficient individually.

The main reason for defining the $\mathsf{Compress}_q$ and $\mathsf{Decompress}_q$ functions is to be able to discard some low-order bits in the public key and the ciphertext which do not have much effect on the correctness probability of decryption – thus making the parameters smaller. The $\mathsf{Compress}_q$ function is also used in one other place where its intuitive purpose is not to "compress". In line 3 of the decryption procedure (Algorithm 3), the function is used to decrypt to a 1 if $v - \mathbf{s}^T\mathbf{u}$ is closer to $\lceil q/2 \rfloor$ than to 0, and decrypt to a 0 otherwise.

2.3. Module-LWE

Let k be a positive integer parameter. The hard problem underlying the security of our schemes is Module-LWE. It consists in distinguishing uniform samples $(\mathbf{a}_i, b_i) \leftarrow R_q^k \times R_q$ from samples $(\mathbf{a}_i, b_i) \in R_q^k \times R_q$ where $\mathbf{a}_i \leftarrow R_q^k$ is uniform and $b_i = \mathbf{a}_i^T\mathbf{s} + e_i$ with $\mathbf{s} \leftarrow \beta_\eta^k$ common to all samples and $e_i \leftarrow \beta_\eta$ fresh for every sample.[5] More precisely, for an algorithm A, we define $\mathbf{Adv}^{\mathrm{mlwe}}_{m,k,\eta}(\mathsf{A}) =$

$$\left|\Pr\left[b'=1 \ : \ \begin{array}{l}\mathbf{A}\leftarrow R_q^{m\times k}; (\mathbf{s},\mathbf{e})\leftarrow \beta_\eta^k\times\beta_\eta^m; \\ \mathbf{b}=\mathbf{A}\mathbf{s}+\mathbf{e}; b'\leftarrow \mathsf{A}(\mathbf{A},\mathbf{b})\end{array}\right]\right.$$
$$\left. - \Pr\left[b'=1 \ : \ \mathbf{A}\leftarrow R_q^{m\times k}; \mathbf{b}\leftarrow R_q^m; b'\leftarrow \mathsf{A}(\mathbf{A},\mathbf{b})\right]\right|.$$

3. Kyber's IND-CPA-secure encryption

Let k, d_t, d_u, d_v be positive integer parameters, and recall that $n = 256$. Let $\mathcal{M} = \{0,1\}^{256}$ denote the message space, where every message $m \in \mathcal{M}$ can be viewed as a polynomial in R with coefficients in $\{0,1\}$. Consider the public-key encryption scheme $\mathsf{Kyber.CPA} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ as described in Algorithms 1 to 3. Note that cipheretxts are of the form $(\mathbf{u}, v) \in \{0,1\}^{256\cdot kd_u} \times \{0,1\}^{256\cdot d_v}$.

Algorithm 1 $\mathsf{Kyber.CPA.KeyGen}()$: key generation

1: $\rho, \sigma \leftarrow \{0,1\}^{256}$
2: $\mathbf{A} \sim R_q^{k\times k} := \mathsf{Sam}(\rho)$
3: $(\mathbf{s}, \mathbf{e}) \sim \beta_\eta^k \times \beta_\eta^k := \mathsf{Sam}(\sigma)$
4: $\mathbf{t} := \mathsf{Compress}_q(\mathbf{A}\mathbf{s} + \mathbf{e}, d_t)$
5: **return** $(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$

Algorithm 2 $\mathsf{Kyber.CPA.Enc}(pk = (\mathbf{t}, \rho), m \in \mathcal{M})$: encryption

1: $r \leftarrow \{0,1\}^{256}$
2: $\mathbf{t} := \mathsf{Decompress}_q(\mathbf{t}, d_t)$
3: $\mathbf{A} \sim R_q^{k\times k} := \mathsf{Sam}(\rho)$
4: $(\mathbf{r}, \mathbf{e}_1, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := \mathsf{Sam}(r)$
5: $\mathbf{u} := \mathsf{Compress}_q(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, d_u)$
6: $v := \mathsf{Compress}_q\left(\mathbf{t}^T\mathbf{r} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot m, d_v\right)$
7: **return** $c := (\mathbf{u}, v)$

Algorithm 3 $\mathsf{Kyber.CPA.Dec}(sk = \mathbf{s}, c = (\mathbf{u}, v))$: decryption

1: $\mathbf{u} := \mathsf{Decompress}_q(\mathbf{u}, d_u)$
2: $v := \mathsf{Decompress}_q(v, d_v)$
3: **return** $\mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$

5. While the exact distribution shape does not seem to play any role in the hardness of (Module)-LWE encryption schemes, we mention that it is possible to show with a simple Rényi divergence-based analysis a la [9], [4] that one can substitute β_η with the n-dimensional rounded Gaussian distribution of standard deviation $\sqrt{\eta/2}$, which was the one considered in [54].

Correctness. We show below the correctness of the encryption scheme described in Algorithms 1 to 3. We will select parameters in Section 6 to make the decryption error negligible, i.e., so that Kyber.CPA is $(1-\delta)$-correct with $\delta < 2^{-128}$.

Theorem 1. *Let k be a positive integer parameter. Let $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e}_1, e_2$ be random variables that have the same distribution as in Algorithms 1 and 2. Also, let $\mathbf{c}_t \leftarrow \psi_{d_t}^k, \mathbf{c}_u \leftarrow \psi_{d_u}^k, c_v \leftarrow \psi_{d_v}$ be distributed according to the distribution ψ defined as follows:*
Let ψ_d^k be the following distribution over R:

1: *Choose uniformly-random* $\mathbf{y} \leftarrow R^k$
2: ***return*** $\left(\mathbf{y} - \mathsf{Decompress}_q\left(\mathsf{Compress}_q(\mathbf{y}, d), d\right)\right)$ *mod*$^{\pm}$ q.

Denote

$$\delta = \Pr\left[\left\|\mathbf{e}^T\mathbf{r} + e_2 + \mathbf{c}_v - \mathbf{s}^T\mathbf{e}_1 + \mathbf{c}_t^T\mathbf{r} - \mathbf{s}^T\mathbf{c}_u\right\|_\infty \geq \lceil q/4 \rfloor\right].$$

Then Kyber.CPA *is $(1-\delta)$-correct.*

Remark 1. *We provide with our software a Python script that allows to compute a tight upper bound on δ; the parameter set we recommend for* Kyber *in Table 1 yields $\delta = 2^{-142}$.*

Proof. The value of $\mathbf{t}$ in line 6 of Algorithm 2 is:

$$\mathbf{t} = \mathsf{Decompress}_q\left(\mathsf{Compress}_q(\mathbf{A}\mathbf{s} + \mathbf{e}, d_t), d_t\right) = \mathbf{A}\mathbf{s} + \mathbf{e} + \mathbf{c}_t,$$

for some $\mathbf{c}_t \in R^k$. The value of $\mathbf{u}$ in Algorithm 3 is

$$\begin{aligned}\mathbf{u} &= \mathsf{Decompress}_q\left(\mathsf{Compress}_q(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, d_u), d_u\right) \\ &= \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 + \mathbf{c}_u,\end{aligned}$$

for some $\mathbf{c}_u \in R^k$. And the value of v is

$$\begin{aligned}v &= \mathsf{Decompress}_q\left(\mathsf{Compress}_q(\mathbf{t}^T\mathbf{r} + e_2 + \lceil q/2 \rfloor \cdot m, d_v), d_v\right) \\ &= \mathbf{t}^T\mathbf{r} + e_2 + \lceil q/2 \rfloor \cdot m + \mathbf{c}_v \\ &= (\mathbf{A}\mathbf{s} + \mathbf{e} + \mathbf{c}_t)^T\mathbf{r} + e_2 + \lceil q/2 \rfloor \cdot m + c_v \\ &= (\mathbf{A}\mathbf{s} + \mathbf{e})^T\mathbf{r} + e_2 + \lceil q/2 \rfloor \cdot m + c_v + \mathbf{c}_t^T\mathbf{r},\end{aligned}$$

for some $c_v \in R$. In all of the above, we can safely assume that the values $\mathbf{c}_t, \mathbf{c}_u$, and c_v are distributed according to the distribution ψ defined in the theorem statement. The reason is that all of these are of the form $\left(\mathbf{y} - \mathsf{Decompress}_q\left(\mathsf{Compress}_q(\mathbf{y}, d), d\right)\right)$ $\mathrm{mod}^{\pm}$ q where $\mathbf{y}$ is pseudo-random based on the hardness of Module-LWE.

Using the above, we obtain

$$\begin{aligned}v - \mathbf{s}^T\mathbf{u} = \mathbf{e}^T\mathbf{r} &+ e_2 + \lceil q/2 \rfloor \cdot m \\ &+ \mathbf{c}_v + \mathbf{c}_t^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1 - \mathbf{s}^T\mathbf{c}_u\end{aligned}$$

If $\left\|\mathbf{e}^T\mathbf{r} + e_2 + \mathbf{c}_v + \mathbf{c}_t^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1 - \mathbf{s}^T\mathbf{c}_u\right\|_\infty < \lceil q/4 \rfloor$, then we can write $v - \mathbf{s}^T\mathbf{u} = w + \lceil q/2 \rfloor \cdot m$ where $\|w\|_\infty < \lceil q/4 \rfloor$. Define $m' = \mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$. We then know that

$$\begin{aligned}\lceil q/4 \rfloor &\geq \|v - \mathbf{s}^T\mathbf{u} - \lceil q/2 \rfloor \cdot m'\|_\infty \\ &= \|w + \lceil q/2 \rfloor \cdot m - \lceil q/2 \rfloor \cdot m'\|_\infty.\end{aligned}$$

By the triangle inequality and the fact that $\|w\|_\infty < \lceil q/4 \rfloor$, we obtain

$$\|\lceil q/2 \rfloor \cdot (m - m')\|_\infty < 2 \cdot \lceil q/4 \rfloor,$$

which (for all odd q) implies that $m = m'$, and proves the correctness of Kyber.CPA. □

Security of a modified scheme. We will prove that the encryption scheme defined above *without* compressing $\mathbf{t}$ in Line 4 of Algorithm 1 and without Line 2 in Algorithm 2 (called Kyber.CPA′) is IND-CPA secure under the Module-LWE hardness assumption.

Theorem 2. *For any adversary* A*, there exists an adversary* B *such that* $\mathbf{Adv}^{\text{cpa}}_{\mathsf{Kyber.CPA}'}(\mathsf{A}) \leq 2 \cdot \mathbf{Adv}^{\text{mlwe}}_{k+1,k,\eta}(\mathsf{B})$.

Proof. Let A be an adversary that is executed in the IND-CPA security experiment which we call game G_0, i.e., $\mathbf{Adv}^{\text{cpa}}_{\mathsf{Kyber.CPA}'}(\mathsf{A}) = |\Pr[b = b' \text{ in game } G_0] - 1/2|$. In game G_1, the value $\mathbf{t}' := \mathbf{As} + \mathbf{e}$ which is used in KeyGen is substituted by a uniform random value. It is possible to verify that there exists an adversary B with the same running time as that of A such that $|\Pr[b = b' \text{ in game } G_0] - |\Pr[b = b' \text{ in game } G_1]| \leq \mathbf{Adv}^{\text{mlwe}}_{k,k,\eta}(\mathsf{B}) \leq \mathbf{Adv}^{\text{mlwe}}_{k+1,k,\eta}(\mathsf{B})$. In game G_2, the values $\mathbf{u}' := \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$ and $v' := \mathbf{t}^T\mathbf{r} + e_2$ used in the generation of the challenge ciphertext are simultaneously substituted with uniform random values. Again, there exists an adversary B with the same running time as that of A with $|\Pr[b = b' \text{ in game } G_1] - |\Pr[b = b' \text{ in game } G_2]| \leq \mathbf{Adv}^{\text{mlwe}}_{k+1,k,\eta}(\mathsf{B})$. Note that in game G_2, the value v from the challenge ciphertext is independent of bit b and therefore $\Pr[b = b' \text{ in game } G_2] = 1/2$. Collecting the probabilities yields the required bound. □

Security of the real scheme. In the real scheme, $\mathbf{t} := \mathsf{Decompress}_q(\mathsf{Compress}_q(\mathbf{t}, d_u), d_u)$ in Line 2 of Algorithm 2 is no longer uniform in R_q^k, and so one cannot conclude that the distribution $(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, \mathbf{t}^T\mathbf{r} + e_2)$, for $\mathbf{r}, \mathbf{e}_1 \sim \beta_\eta^k$ and $e_2 \sim \beta_\eta$ is indistinguishable from uniform based on the hardness of the Module-LWE problem. One way to fix this issue is for the encryptor to add a small random (possibly even public) noise $\mathbf{e}' \in R^k$ to $\mathbf{t}$ such that $\mathbf{t} + \mathbf{e}'$ is uniformly random in R_q^k. The distribution of this noise would have to depend on $\mathbf{t}$ and d_u. For our value of $d_u = 11$, this would involve selecting the coefficients of $\mathbf{e}'$ from the set $\{-1, 0, 1, 2\}$ with a probability distribution that depends on $\mathbf{t}$.[6] To achieve the same distribution more easily, one could instead define the $\mathsf{Compress}_q$ function as truncating the last 2 bits, and the $\mathsf{Decompress}_q$ function as a multiplication by 4. Then the coefficients of $\mathbf{e}'$ can be chosen uniformly at random from $\{0, 1, 2, 3\}$. The main disadvantage of adding the extra $\mathbf{e}'$ is that the term $\mathbf{e}'^T\mathbf{r}$ will appear in the decryption and will add to the decryption error. For the recommended parameters (see Table 1), the decryption error will increase from 2^{-142} to 2^{-121}.

6. We thank Jan-Pieter D'Anvers for these observations.

Due to the increase in the decryption error and the cumbersome nature of adding the extra error term, we choose to define Kyber to include the compression of the public key $\mathbf{t}$ but not add any noise. There are several reasons why we strongly believe that this choice does not affect security. First, note that in the encryption Algorithm 2, the decompressed value of $\mathbf{t}$ is used in Line 6 and is compressed with $d_v = 3$ (see Table 1 and Table 2). This means that only approximately the highest 3 bits of $\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m$ are output per coefficient. In particular, if the modulus q is large enough, then the two distributions are statistically close because

$$\mathsf{Compress}_q\left((\mathbf{t} + \mathbf{e}')^T\mathbf{r} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot m, d_v\right) = \mathsf{Compress}_q\left(\mathbf{t}^T\mathbf{r} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot m, d_v\right),$$

where $\mathbf{e}'$ is chosen so as to make $\mathbf{t} + \mathbf{e}'$ uniform as discussed above. This is exactly the same relationship that exists between the Module-LWE and Module-LWR problems.[7] For certain parameters, the two problems are equivalent (see [18] for the most "liberal" reduction between the two), yet for ones used in practice (c.f. [10] and many submissions to the NIST post-quantum call), it is still assumed that the distribution of Module-LWR is pseudorandom despite the fact that the proof in [18] is no longer applicable.

Additionally, our proposal for the KEM is the CCA-transformation in Algorithm 4 which does not even require that the output of the CPA-secure scheme be pseudo-random since the shared key is formed by using the message m inside a random oracle. Because the entropy of $\mathbf{r}$ is larger than the entropy of v (by a factor larger than 2.5), it implies that it is not enough to only look at the v term to recover m (or perhaps to even distinguish it from uniform), but one must also somehow use the "properly formed" Module-LWE component $\mathbf{u}$ at the same time. We believe that it is extremely unlikely that anything about the message m can be recovered from this information assuming that Module-LWE is hard.

4. The CCA-secure KEM

Let $\mathrm{G}\colon \{0,1\}^* \to \{0,1\}^{2\times 256}$ and $\mathrm{H}\colon \{0,1\}^* \to \{0,1\}^{256}$ be hash functions. Consider the public-key key encapsulation mechanism $\mathsf{Kyber} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ as described in Algorithms 1, 4 and 5, where KeyGen is the same as the one of Kyber.CPA from the previous section, with the difference that sk also contains $pk = (\mathbf{t}, \rho)$ and a secret 256-bit random value z. It is obtained by applying a KEM variant with "implicit rejection" [44] of the Fujisaki–Okamoto transform [38] to the Kyber.CPA encryption scheme. Note that we make explicit the randomness r in the Enc algorithm.

We stress that Kyber.Decaps never returns $\perp$. Instead, in case re-encryption fails, it returns a pseudo-random key $K := \mathrm{H}(z, c)$, where z is a random, secret seed.

7. The Module-LWR (Learning with Rounding) problem outputs $(\mathbf{A}, \mathsf{Compress}_q(\mathbf{As}, d))$ for $\mathbf{A} \leftarrow R_q^{m\times k}, \mathbf{s} \leftarrow \beta_\eta^k$ and asks to distinguish this distribution from $(\mathbf{A}, \mathsf{Compress}_q(\mathbf{b}, d))$ for $\mathbf{b} \leftarrow R_q^m$.

Algorithm 4 $\mathsf{Kyber.Encaps}(pk = (\mathbf{t}, \rho))$

$m \leftarrow \{0,1\}^{256}$
$(\hat{K}, r) := \mathrm{G}(\mathrm{H}(pk), m)$
$(\mathbf{u}, v) := \mathsf{Kyber.CPA.Enc}\,((\mathbf{t}, \rho), m; r)$
$c := (\mathbf{u}, v)$
$K := \mathrm{H}(\hat{K}, \mathrm{H}(c))$
return (c, K)

Algorithm 5 $\mathsf{Kyber.Decaps}(sk = (\mathbf{s}, z, \mathbf{t}, \rho), c = (\mathbf{u}, v))$

$m' := \mathsf{Kyber.CPA.Dec}(\mathbf{s}, (\mathbf{u}, v))$
$(\hat{K}', r') := \mathrm{G}(\mathrm{H}(pk), m')$
$(\mathbf{u}', v') := \mathsf{Kyber.CPA.Enc}\,((\mathbf{t}, \rho), m'; r')$
if $(\mathbf{u}', v') = (\mathbf{u}, v)$ **then**
  return $K := \mathrm{H}(\hat{K}', \mathrm{H}(c))$
else
  return $K := \mathrm{H}(z, \mathrm{H}(c))$
end if

Correctness. If Kyber.CPA is $(1-\delta)$-correct and G is a random oracle, then Kyber is $(1-\delta)$-correct [44].

Security. The following concrete security statement proves Kyber's CCA-security when the hash functions G and H are modeled as random oracles. We provide the concrete security bounds from [44] which considers the KEM variant of the FO transformation and also takes a non-zero correctness error δ into account.

Theorem 3. *For any classical adversary* A *that makes at most q_{RO} many queries to random oracles H and G, and q_D queries to the decryption oracle, there exists an adversary* B *such that*

$$\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{Kyber}}(\mathsf{A}) \leq 3\mathbf{Adv}^{\mathrm{cpa}}_{\mathsf{Kyber.CPA}}(\mathsf{B}) + q_{RO} \cdot \delta + \frac{3q_{RO}}{2^{256}}.$$

We remark that there exists an alternative security reduction from the weaker notion of ONE-WAY CPA-security [44] of Kyber.CPA which is, however, not tight as it loses a multiplicative factor q_{RO}.

As for security in the quantum random oracle model (QROM), [69] can be used to prove that Kyber is IND-CCA secure in the QROM, provided that Kyber.CPA is ONE-WAY CPA secure and *sparse pseudo-random*. Sparse pseudo-randomness [69] is a slightly stronger security notation than IND-CPA security and essentially states that (i) a properly generated ciphertext is pseudo-random (i.e., it is computationally indistinguishable from a random high-entropy bit-string) and that (ii) a random bit-string is, with high probability, not a properly generated ciphertext. The proof of Theorem 2 shows that $\mathsf{Kyber.CPA}'$ (i.e., Kyber.CPA without compressing $\mathbf{t}$) is tightly pseudo-random under the Module-LWE hardness assumption. Concretely, the pseudo-randomness advantage is bounded by $\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{Kyber.CPA}'}(\mathsf{A}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathsf{B})$. One can argue again that the same bound holds for Kyber.CPA. The sparseness property is trivially fulfilled for Kyber.CPA since the set of properly generated ciphertexts is a sparse subset of the ciphertext space $\{0,1\}^{256(kd_u)} \times \{0,1\}^{d_v}$.

One can use [69] (in a combination with [45] to account for the correctness error δ) to obtain the following concrete statement in the QROM.

Theorem 4. *For any quantum adversary* A *that makes at most q_{RO} many queries to quantum random oracles H and G, and at most q_D many (classical) queries to the decryption oracle, there exists a quantum adversary* B *such that*

$$\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{Kyber}}(\mathsf{A}) \leq 8q^2_{RO} \cdot \delta + 4q_{RO} \cdot \sqrt{\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{Kyber.CPA}}(\mathsf{B})}.$$

Unfortunately, the above security bound is non-tight and therefore can only serve as an asymptotic indication of Kyber's CCA-security in the quantum random oracle model. We can use [69] to derive a tight security bound in the QROM from a non-standard security assumption, namely that a deterministic version of Kyber.CPA is sparse pseudo-random in the QROM. Deterministic Kyber.CPA is defined as Kyber.CPA, but the randomness r used in encryption is derived deterministically from the message m via $r := \mathrm{G}(m)$. In the classical ROM this assumption is tightly implied by the IND-CPA security of Kyber.CPA but in the QROM the reduction is non-tight (and is the reason for the term $q_{RO} \cdot \sqrt{\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{Kyber.CPA}}(\mathsf{B})}$ in Theorem 4).

Hashing pk into $\hat{K}$. The Kyber CCA transform is essentially the transform from [72], [44], with one small tweak: we hash the public key pk (or more precisely $\mathrm{H}(pk)$) into $\hat{K}$. This tweak has two effects. First, it makes the KEM contributory; the shared key K does not depend only on input of one of the two parties. The second effect is a multi-target protection. Consider an attacker who searches through many values m to find one that is "likely" to produce a failure during decryption. Such a decryption failure of a legitimate ciphertext would leak some information about the secret key. In the pre-quantum setting this attack approach is doomed because of the negligible failure probability δ. In a post-quantum setting, the attacker could use Grover's algorithm to search for such an m. However, the attacker is then facing the problem to encode "likely to produce a decryption failure" in the Grover oracle. This is equivalent to identifying noise vectors that are likely to have a large inner product with $(\mathbf{s}, \mathbf{e})$; probably the best strategy is to search for m that produce noise vectors of large norm. Even though we believe this attack approach is unlikely to result in any better performance than a brute-force Grover search of the 256-bit shared key K, hashing pk into $\hat{K}$ ensures that an attacker would not be able to use precomputed values m against multiple targets.

Supporting non-incremental hash APIs. One might wonder why we use $\mathrm{H}(pk)$ instead of pk as input to G when computing $\hat{K}$ and why we use $\mathrm{H}(c)$ instead of c as input to H when computing K. The reason is that this simplifies implementation with non-incremental hash APIs, such as the ones used in the SUPERCOP benchmarking framework [14] and the Networking and Cryptography library (NaCl) [15].

Furthermore using $\mathrm{H}(pk)$ instead of pk as input to G enables a small speedup for decapsulation at the cost of a slightly increased secret-key size as explained in the Section 7.

CCA-secure public-key encryption. We remark that a CCA-secure public-key encryption scheme can be obtained by combining the CCA-secure KEM Kyber with any CCA-secure symmetric encryption scheme [34] (aka. DEM). We describe the resulting hybrid encryption scheme Kyber.Hybrid in the full online version of the paper in Appendix A.

5. Key Exchange Protocols

Let $\mathsf{Kyber} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ be the IND-CCA secure KEM from the previous section. Figure 1 describes the Kyber key exchange protocol Kyber.KE obtained as a direct application of the key encapsulation mechanism. In key exchange constructions using a KEM, it is common to hash the "view" of each participant (i.e., all received and sent messages) into the final key. In Kyber, the public key pk is hashed into the "pre-key" $\hat{K}$ and the ciphertext is hashed into the final key K; hence the shared key obtained in a key exchange already includes the complete "view" of each participant.

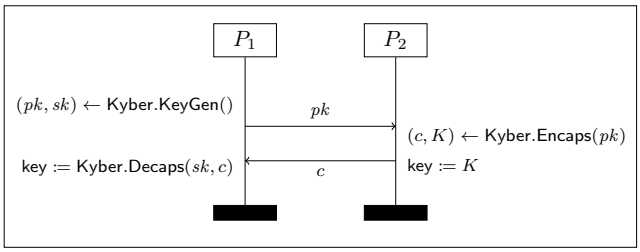

Figure 1. Kyber.KE – Key Exchange protocol using the $\mathsf{Kyber} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ key encapsulation mechanism.

Authenticated key exchanges protocols. Note that the protocol of Fig. 1 *by itself* only provides security against passive adversaries (and in particular fails to protect against man-in-the-middle attacks). Let $\mathrm{H}\colon \{0,1\}^* \to \{0,1\}^{256}$ be a hash function. Figure 2 describes our one-sided (unilateral) authenticated key exchange protocol Kyber.UAKE in which party P_1 knows the static (long-term) key of party P_2, and Fig. 3 describes our authenticated key-exchange protocol Kyber.AKE where each party knows the static (long-term) key of the other party.

The shared key derived at the end of the above protocols not only depends on the ephemeral key and ciphertext (pk, c), but also on the static (long-term) keys pk_i and associated ephemeral ciphertexts c_i (where $i = 2$ and $i = 1, 2$ respectively).

Our authenticated key-exchange protocols follow a generic construction from any CCA-secure encryption scheme. Concretely, security of Kyber.AKE in the Canetti–Krawczyk model with weak forward secrecy [27] follows directly from the generic security bounds of [22], [37]. (Note

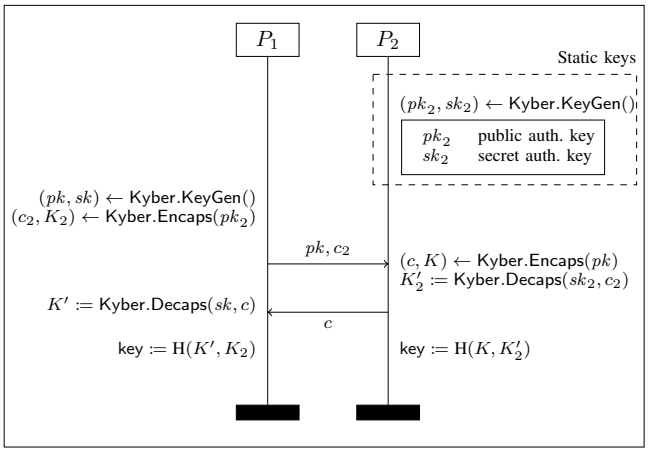

Figure 2. Kyber.UAKE – One-sided authenticated key exchange protocol using Kyber, where P_1 knows the static public key of P_2.

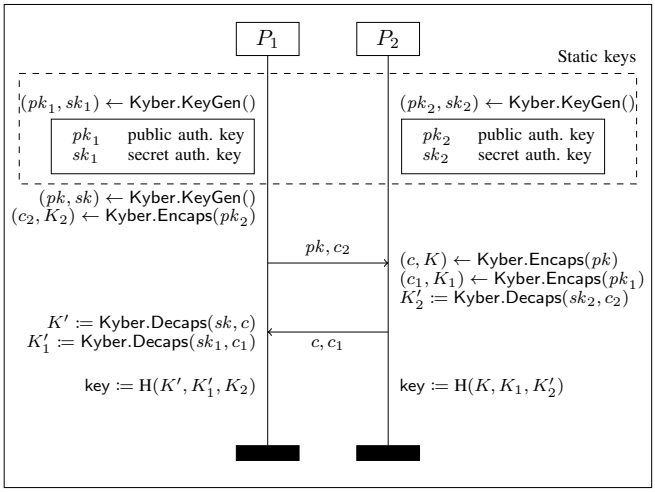

Figure 3. Kyber.AKE – Authenticated key exchange protocol using Kyber, where both parties know each other's static public keys.

that full forward secrecy is not achievable for a two-round authenticated key-exchange protocol [27].)

6. Parameters and Security Analysis

In this section we give the Kyber parameter set that aims at 128 bits of post-quantum (and classical) security, with a generous security margin to account for future improvements in cryptanalysis. We only consider the parameters that are relevant to the underlying lattice problem; instantiations of symmetric primitives are given in Section 7.

TABLE 1. KYBER PARAMETER SET, AIMING AT 128-BIT CLASSICAL AND POST-QUANTUM SECURITY, WITH GENEROUS MARGINS.

	n	k	q	η	(d_u, d_v, d_t)	δ	pq-sec
Kyber	256	3	7681	4	$(11, 3, 11)$	2^{-142}	161

The parameters of Kyber are summarized in Table 1. The first parameter we fixed was $n = 256$, which stems from the fact that we want to encapsulate 256 bits of entropy (targeting a 128-bit security level for symmetric keys [40]) and that we want to encode each of these bits into one polynomial coefficient. We then picked $q = 7681$ as the smallest prime that fulfills $q \equiv 1 \bmod 2n$, which allows us to use fast multiplication in R_q based on the negacyclic number-theoretic transform (NTT). The next parameter we fixed is $k = 3$, which controls the dimension of the lattice, and thereby largely the security. Finally we tuned the parameters η, d_u, d_v, and d_t to balance security, failure probability δ, public-key size, and ciphertext size. We decided to fix $d_u = d_t = 11$, which unifies compression of public keys and the "key component" $\mathbf{u}$ of the ciphertext.

Core-SVP hardness. To analyze the security of Kyber, we follow the methodology introduced in [4, Sec. 6.1]. This means that we assume that the best way to solve the Module-LWE problem underlying Kyber is to treat it as a general LWE problem. Moreover we consider the primal and dual attacks to be the only known attacks relevant to our parameter sets. After optimizing the parameters for the primal attack with respect to the success criteria of [4, Sec. 6.3], we find that the attack would invoke BKZ with blocksize 610 to 615 (depending on whether one uses the primal or dual attack). The cost of BKZ with blocksize 610 is dominated by a polynomial number of calls to a dimension 610 SVP solver. Suppressing this polynomial number of SVP calls and all subexponential factors in the cost of the best known quantum algorithm for SVP [51, Sec. 14.2.10], this implies a cost of $> 2^{161}$ operations in the quantum RAM model. According to this very conservative analysis, Kyber offers 161 bits of security against the best known quantum attacks targeting the underlying lattice problem.

Resistance to hybrid attacks. Several schemes [13], [42] are potentially vulnerable to a hybrid attack [46], [39], mixing lattice reduction techniques with Meet-in-the-Middle combinatorial search. This attack is particularly difficult to analyze, and recent work [73] suggests that it is often not as competitive as previously thought. We note that this attack is especially relevant when secrets and errors are ternary and sparse, which is not the case for our design.

Algebraic attacks. The main novelty of our design is in the use of Module-LWE rather than Ring-LWE. One of the motivations for this change is to move further away from the recently uncovered weaknesses of ideal lattices [26], [17], [32], [33] – yet without the cost of using completely unstructured LWE. The work of [33] mentions obstacles towards a quantum attack on Ring-LWE from their new techniques, but nevertheless suggests using Module-LWE, as it plausibly creates even more obstacles.

Scaling security and performance. A particularly attractive feature of Module-LWE (as compared to LWE or Ring-LWE) is, that scaling security only needs marginal changes to existing, possibly highly optimized implementations. Specifically, the only parameters that need to change to scale security (and performance) of Kyber, are k and η; note that optimized code for polynomial arithmetic is not affected by changing those parameters. Table 2 lists one "paranoid" parameter set aiming at security similar to NEWHOPE (using dimension $n \cdot k = 1024$) and one "light" parameter set that might become interesting for the 96-bit security level, or, with a tighter security analysis for the 128-bit security level, if continued effort in cryptanalysis does not produce significantly better attacks.

The Core-SVP hardness analysis against the best known quantum attacks yields 218 bits of security for the paranoid parameter set and 102 bits of security for the light parameter set.

A note on passively secure KEMs. We note that in order to support the CCA transformation, we need a negligible (in the cryptographic sense) failure probability. Previous proposals like NEWHOPE [4] or FRODO [20] are designed to only achieve passive security and can live with much higher failure probabilities ($\approx 2^{-60}$ for NEWHOPE and $2^{-38.9}$ for the recommended parameter set of FRODO). If one were to optimize a passively secure KEM from Module-LWE, one could reduce the rounding parameters d_u and d_t to $d_u = d_t = 10$ to further reduce public-key size (to 992 bytes) and ciphertext size (to 1088 bytes) while increasing the failure probability (to $2^{-71.9}$).

7. Implementation

In this section we give all the remaining details of our implementations of Kyber and report on performance of subroutines. An even more detailed description is given in the specification of Kyber submitted to NIST [8]. Both implementations are fully protected against timing attacks. All cycle counts in this section were obtained on one core of an Intel Core-i7 4770K (Haswell) with hyperthreading and TurboBoost turned off running at 3.5 GHz. They are median cycle counts over 1000 measurements.

7.1. Primitives and encodings

Sections 3 and 4 introduce Kyber in abstract terms without fixing concrete instantiations of the functions H, G, and Sam, and without fixing encodings of messages. This subsection details concrete instantiations of these building blocks.

Symmetric primitives. The main symmetric building blocks are the two hash functions H and G, a function that accepts as input the public seed ρ and generates the uniform matrix $\mathbf{A} \in R_q^{k \times k}$, and a function that accepts as input a secret seed r and generates as output noise polynomials sampled from β_η. Note that in passively secure KEMs like BCNS [21], NEWHOPE [4], or FRODO [20], the choice of how noise polynomials are sampled is a local decision: implementations on different platforms can choose whatever

TABLE 2. ALTERNATIVE PARAMETER SETS FOR HIGHER AND LOWER SECURITY LEVELS.

	n	k	q	η	(d_u, d_v, d_t)	δ	pq sec.	$\|pk\|$ in bytes	$\|c\|$ in bytes
Paranoid	256	4	7681	3	(11, 3, 11)	2^{-169}	218	1 440	1 536
Light	256	2	7681	5	(11, 3, 11)	2^{-145}	102	736	832

PRNG is the best option on the respective platform. This is also true for noise generation in Kyber's key generation, but, because of the CCA transform, is no longer true for noise generation in encapsulation.

We decided to instantiate all hash functions with functions derived from Keccak as standardized in FIPS 202 [60]. Specifically, to generate the matrix $\mathbf{A}$ we use SHAKE-128; to generate noise polynomials we use SHAKE-256; and we instantiate H with SHA3-256 and G wit SHA3-512. With this choice, all symmetric primitives in Kyber rely on the same underlying primitive, namely the Keccak-$f1600$ permutation. The only exception is that for key generation, different implementations are free to use whatever PRNG is offering the best performance and security on their respective platform.

We are aware that another choice of symmetric primitives would yield somewhat better performance on most platforms. For example, we could have decided to use SHA256 for all hashes (with output extension for G via MGF1; see [59, App. B.2.1]), and AES in counter mode for the expansion of seeds. This choice would certainly be faster on platforms with hardware AES and SHA256 support. However, on platforms without hardware support, AES implementations are notorious for timing-attack vulnerabilities. Furthermore, as pointed out in [4, Sec. 3], the use of a PRG (which AES in counter mode is), is not helpful to argue security, because in the generation of $\mathbf{A}$, the input is *public*, whereas security of a PRG is only given for secret inputs.

Other possible choices of primitives that would yield better performance are be the ChaCha20 stream cipher [12] that has recently been standardized for TLS [52] or the BLAKE2X extendable output function [7]. Unfortunately, neither of these functions has received a lot of cryptanalytic attention, yet, so we prefer to stick to the conservative choice of Keccak, which was standardized after years of cryptanalytic scrutiny through the course of the SHA-3 competition.

The NTT domain. Computing the discrete Fourier transform on elements from R_q can be done with methods analogous to the fast Fourier transform [31], except that operations on coefficients are defined in a finite field [64]. This is often referred to as the number theoretic transform (NTT). Before being able to define the expansion of the seed ρ into the matrix $\mathbf{A}$, we need to define the NTT domain of polynomials. Let $\omega = 3844 \in \mathbb{Z}_q$ and $\psi = \sqrt{\omega} = 62$, where ψ is chosen as the smallest element of multiplicative order 2^9 in $\mathbb{F}_q^* = \mathbb{F}_{7681}^*$.

For a polynomial $\mathbf{g} = \sum_{i=0}^{255} g_i X^i \in R_q$ we define the polynomial $\hat{\mathbf{g}}$ *in NTT domain* as

$$\mathsf{NTT}(\mathbf{g}) = \hat{\mathbf{g}} = \sum_{i=0}^{255} \hat{g}_i X^i, \text{ with } \hat{g}_i = \sum_{j=0}^{255} \psi^j g_j \omega^{ij}.$$

The inverse NTT^{-1} of the function NTT is essentially the same as the computation of NTT, except that it uses $\omega^{-1} \bmod q = 6584$, multiplies by powers of $\psi^{-1} \bmod q = 1115$ after the summation, and also multiplies each coefficient by the scalar $n^{-1} \bmod q = 7651$, so that

$$\mathsf{NTT}^{-1}(\hat{\mathbf{g}}) = \mathbf{g} = \sum_{i=0}^{255} g_i X^i, \text{ with } g_i = n^{-1}\psi^{-i} \sum_{j=0}^{255} \hat{g}_j \omega^{-ij}.$$

For two polynomials $\mathbf{f}, \mathbf{g} \in R_q$, the product $\mathbf{fg}$ can be computed as $\mathsf{NTT}^{-1}(\mathsf{NTT}(\mathbf{f}) \circ \mathsf{NTT}(\mathbf{g}))$, where $\circ$ denotes the point-wise multiplication.

Generation of A. Generation of the matrix $\mathbf{A} = (a_{i,j}) \in R_q^{k \times k}$ receives as input the public seed ρ. To generate the entry $a_{i,j} \in R_q$ we first expand ρ through cSHAKE-128 with the 2-byte domain separator (i, j). The output of this expansion is considered a stream of 16-bit little-endian integers. On this sequence of 16-bit integers we run rejection sampling to obtain coefficients in $\{0, \ldots, q-1\}$. The resulting polynomial $a_{i,j}$ is assumed to be in NTT domain.

Generation of noise polynomials. Noise polynomials in Kyber are sampled from β_4. To obtain such a noise polynomial we first expand a seed to an array of $n = 256$ uniformly random bytes $(r_0, \ldots, r_{255})$. We then generate coefficient e_i of a noise polynomial $\mathbf{e} = \sum_{i=0}^{255} e_i X^i$ by subtracting the Hamming weight of the most significant nibble of r_i from the Hamming weight of the least significant nibble of r_i.

Encoding of keys and ciphertexts. In NEWHOPE, polynomials in public keys and the ciphertext are in NTT domain; in Kyber all polynomials sent over the channel are in normal domain. This is necessary for the compression through rounding (see Section 3) to work.

A Kyber public key is a tuple $(\mathbf{t}, \rho)$, where $\mathbf{t}$ is a vector of three polynomials with 256 11-bit coefficients each, and ρ is a 32-byte seed. We encode the polynomials in compressed little-endian format to fit it in $(256 \cdot 11)/8 = 352$ bytes, concatenate the compressed three polynomials and finally concatenate ρ to obtain public keys of $3 \cdot 352 + 32 = 1088$ bytes.

A Kyber secret key is a vector of three polynomials in NTT domain with 256 13-bit coefficients each. We store these polynomials in compressed little-endian format resulting in a total of $(3 \cdot 256 \cdot 13)/8 = 1248$ bytes.

For re-encapsulation during decapsulation we additionally need the public key, which we simply concatenate and store as part of the secret key. Finally, we also concatenate $\mathrm{H}(pk)$ to avoid having to compute this hash during decapsulation and concatenates the 32 bytes of the value z that is used to compute the pseudo-random returned key when re-encapsulation fails. This results in a total size of $1248 + 1088 + 32 + 32 = 2400$ bytes for the secret key.

A Kyber ciphertext is a 3-tuple $(\mathbf{u}, v, d)$, where $\mathbf{u}$ is a vector of three polynomials with 256 11-bit coefficients each, v is a polynomial with 256 3-bit coefficients, and d is a 32-byte hash. Using the same compressed little-endian format for polynomials as for keys we obtain ciphertexts with a total size of $3 \cdot 352 + (3 \cdot 256)/8 + 32 = 1184$ bytes.

Size-speed tradeoffs. It is possible to use different tradeoffs between secret-key size and decapsulation speed. If secret-key size is critical, it is of course possible to not store $\mathrm{H}(pk)$ and also to not store the public key as part of the secret key but instead recompute it during decapsulation. Furthermore, not keeping the secret key in NTT domain makes it possible to compress each coefficient to only 5 bits, resulting in a total size of only 320 bytes for the three polynomials. Finally, as all randomness in key generation is generated from two 32-byte seeds, it is also possible to only store these seeds and re-run key generation during decapsulation.

In the other direction, if secret-key size does not matter very much and decapsulation speed is critical, one might decide to store the expanded matrix $\mathbf{A}$ as part of the secret key and avoid recomputation from the seed ρ during the re-encapsuation part of decapsulation.

All performance results reported in the following assume the secret-key format described in the previous paragraph; i.e., with polynomials in NTT domain, including the public key and $\mathrm{H}(pk)$, but not including $\mathbf{A}$.

7.2. Reference implementation

Kyber's reference implementation in C follows in the spirit of the NEWHOPE reference implementation described in [4, Sec. 7.2]. In particular, it only relies on 16-bit and 32-bit integer arithmetic (outside of Keccak) and uses the same combination of short Barrett reductions and Montgomery reductions to accelerate the NTT computation. One consequence of the modulus $q = 7681$ is that the short Barrett reduction becomes slightly more efficient; an unsigned 16-bit integer `a` can be reduced to an unsigned integer `r` between 0 and 11768 and congruent modulo q using the formula `r = a - (a >> 13)*KYBER_Q`.

7.3. AVX2 implementation

Modern 64-bit Intel processors feature the AVX2 vector-instruction set that supports operations on 256-bit vectors that can be interpreted as vectors of 8 single-precision or 4-double-precision floating-point numbers, or as vectors of integers of various sizes. The implementation of NEWHOPE described in [4, Sec. 7] made use of the AVX2 vectorized double-precision floating-point operations for fast arithmetic on polynomials; in the AVX2-optimized implementation of Kyberwe use the AVX2 *integer* instructions.

Polynomial arithmetic. We represent polynomials as arrays of 16-bit unsigned integers, but temporarily switch to signed 16-bit integers during the computation of the NTT. The details of the carefully optimized NTT computations are given in [70]; it achieves a speedup of more than a factor of 5 compared to the approach used in NEWHOPE. In order to keep the specification of Kyber simple, we slightly modify the approach from [70] and insist that all polynomials in NTT domain are stored in bitreversed order and all polynomials in non-NTT domain are in normal order. For details, see [8, Sec. 1.1]. One NTT takes 468 cycles; an NTT^{-1} operation takes 492 cycles. We also use vectorized integer arithmetic for pointwise multiplication and polynomial addition and subtraction.

Vectorized Keccak. As mentioned earlier, Keccak has a reputation of not being particularly fast in software. One reason is that Keccak is very hard to vectorize; in fact, according to the eBACS benchmarks, the fastest implementation of Keccak on Intel Haswell processors is the non-vectorized "simple64" implementation.

The picture changes drastically if a protocol can compute multiple independent streams of SHA-3, SHAKE, or cSHAKE on inputs and outputs of the same length. More specifically, the Keccak code package [16] includes an implementation for AVX2 that computes 4 independent streams in parallel. We make use of this 4-way parallel implementation in the expansion of ρ involved in the generation of the matrix $\mathbf{A}$ and also in the generation of noise polynomials during encapsulation. Specifically, for the generation of $\mathbf{A}$, we generate 8 streams of uniformly random 16-bit numbers via two calls to this function, leaving only one sequential SHAKE-128 call. In encapsulation we generate 8 arrays of 256 uniformly random bytes via two calls to 4-way parallel cSHAKE-128 and discard one of those arrays. The speedup from vectorized Keccak is crucial: compared to NEWHOPE, Kyber needs to generate more than twice as many uniformly random polynomial coefficients, yet, with 35 356 cycles, generation of the matrix $\mathbf{a}$ is about as fast as generation of the equivalent value $\mathbf{a}$ in NEWHOPE.

Rejection sampling. Part of the generation of $\mathbf{A}$ is rejection sampling on the stream of 16-bit integers produced by the cSHAKE-128 expansion. We adopt the fast vectorized approach described in [41] for this task. One difference is that we do not need to first conditionally subtract q four times; we simply eliminate the upper 3 bits of each 16-bit integer in a 256-bit vector through one mask instructions and then compare to a constant vector filled with 16-bit copies of q.

7.4. Flexibility of Kyber

One possible use of Kyber is for ephemeral key exchange, for example in TLS 1.2 as illustrated by [21]

and by Google's post-quantum TLS experiment [23] with NEWHOPE.[8] Indeed, the experiment concluded that they "did not find any unexpected impediment to deploying something like NEWHOPE" [53] and Kyber features performances close to the one of NEWHOPE but with smaller sizes.

However, the CCA security of Kyber makes it a much more versatile tool. Not only is it possible to cache ephemeral keys for some time (which would be a security disaster for BCNS, FRODO, or NEWHOPE), we can also use it for classical IND-CCA public-key encryption of messages of arbitrary length [34] (cf. the hybrid CCA-secure scheme described in the full online version of the paper in Appendix A) and for authenticated key exchange protocols, as described in Fig. 3. The Kyber software package includes implementations of the unilaterally authenticated key exchange Kyber.UAKE described in Fig. 2 and the mutually authenticated key exchange Kyber.AKE described in Fig. 3.

8. Performance results and comparison

In this section we report on the performance of our standalone implementations of Kyber and Kyber-based authenticated key exchange.

8.1. Standalone Kyber

In Table 3 we give performance results of the standalone implementations of Kyber and compare them to results from the literature on lattice-based KEMS, key-exchange protocols, and encryption schemes. We restrict the comparison to schemes that were published before or in parallel to Kyber. We compiled the Kyber software with `gcc-6.3.0` with optimization flags `-O3 -fomit-frame-pointer -march=native -fPIC`.

To give an indication of security levels obtained by the different schemes we include the core-SVP hardness estimation ("Sec. estim.") following the approach from [4]. Note that this estimate does not say anything about the applicability of hybrid or algebraic attacks.

8.2. Kyber-based authenticated key exchanges

To illustrate one use case of Kyber and to establish a data point for high-performance post-quantum authenticated key exchanges, the Kyber software package includes implementations of Kyber.AKE and Kyber.UAKE. The performance in terms of message sizes and CPU cycles (for our AVX2 optimized software) is summarized in Table 4. The only paper describing an implementation of lattice-based authenticated key exchange that we are aware of is [74]. Our software outperforms the results of [74] by more than two orders of magnitude.

8. Note that one can easily combine KEMs (e.g., Kyber with a pre-quantum KEM) by hashing the shared secret keys together.

Acknowledgements

The authors would like to thank Isis Lovecruft for suggesting the name Kyber, Andreas Hülsing for very helpful discussions, and Jan-Pieter D'Anvers for finding a flaw in a previous version of the work.

This work is supported by a Veni Innovational Research Grant from NWO under project number 639.021.645 and by Canada's NSERC CREATE program. IQC is supported in part by the Government of Canada and the Province of Ontario. Eike Kiltz was supported by the ERC Consolidator Grant ERC-2013-CoG-615073-ERCC. Vadim Lyubashevsky was supported by the the SNSF ERC Transfer Starting Grant CRETP2-166734-FELICITY and the H2020 Project Safecrypto. Damien Stehlé was supported by the ERC Starting Grant ERC-2013-StG-335086-LATTAC and acknowledges the support of Bpifrance in the context of the national projet RISQ (P141580). Tancrede Lepoint was supported by SRI International's IoT Security and Privacy Center.

References

[1] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th ACM STOC*, pages 284–293. ACM Press, May 1997. 1

[2] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Heidelberg, August 2016. 2

[3] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003. 2

[4] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange – a new hope. In *Proceedings of the 25th USENIX Security Symposium*, pages 327–343. USENIX Association, 2016. http://cryptojedi.org/papers/#newhope. 2, 5, 9, 10, 11, 12, 13

[5] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Newhope without reconciliation. Cryptology ePrint Archive, Report 2016/1157, 2016. http://eprint.iacr.org/2016/1157. 2

[6] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009. 1, 2

[7] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2X, 2016. https://blake2.net/blake2x.pdf. 10

[8] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber – algorithm specifications and supporting documentation. Submission to the NIST post-quantum project, 2017. https://pq-crystals.org/kyber/resources.shtml. 9, 11

[9] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 3–24. Springer, Heidelberg, November / December 2015. 5

[10] Abhishek Banerjee, Hai Brenner, Gaëtan Leurent, Chris Peikert, and Alon Rosen. SPRING: Fast pseudorandom functions from rounded ring products. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 38–57. Springer, Heidelberg, March 2015. 6

TABLE 3. 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	Sec. estim.	Prob.	ct?		Cycles		Bytes
Passively secure KEMs							
BCNS [21]	78[a]	Ring-LWE	yes	**K**:	≈ 2 477 958	**sk:**	4 096
				E:	≈ 3 995 977	**pk:**	4 096
				D:	≈ 481 937	**c:**	4 224
NEWHOPE [4] (AVX2 optimized)	255[a]	Ring-LWE	yes	**K**:	88 920	**sk:**	1 792
				E:	110 986	**pk:**	1 824
				D:	19 422	**c:**	2 048
FRODO [20] (recommended parameters)	130[a]	LWE	yes	**K**:	≈ 2 938 000[b]	**sk:**	11 280
				E:	≈ 3 484 000[b]	**pk:**	11 296
				D:	≈ 338 000[b]	**c:**	11 288
LWR OKCN [49] (recommended parameters)	137[a]	LWR	yes	**K**:	2 664 789[c]	**sk:**	10880
				E:	3 953 182[c]	**pk:**	9 968
				D:	307 404[c]	**c:**	8 224
LWE OKCN [49] (recommended parameters)	131[a]	LWE	yes	**K**:	3 064 789[c]	**sk:**	11392
				E:	4 023 632[c]	**pk:**	9 968
				D:	335 380[c]	**c:**	8 608
CCA-secure KEMs							
Streamlined NTRU Prime 4591^{761} [13]	137[a]	NTRU[k]	yes	**K**:	59 456[e]	**sk:**	1600
				E:	97 684[e]	**pk:**	1218
				D:	> 6 000 000	**c:**	1047
spLWE-KEM [29] (128-bit PQ parameters)	128[i]	spLWE	?	**K**:	≈ 336 700[d]	**sk:**	?
				E:	≈ 813 800[d]	**pk:**	?
				D:	≈ 785 200[d]	**c:**	804
NTRU-KEM [47]	123[a]	NTRU	yes	**K**:	307 914	**sk:**	1 422
				E:	48 646	**pk:**	1 140
				D:	67 338	**c:**	1 281
Kyber (this paper) (C reference)	161[a]	Module-LWE	yes	**K**:	243 004	**sk:**	2 368
				E:	332 616	**pk:**	1 088
				D:	394 424	**c:**	1 184
Kyber (this paper) (AVX2 optimized)	161[a]	Module-LWE	yes	**K**:	85 472	**sk:**	2 400
				E:	112 660	**pk:**	1 088
				D:	108 904	**c:**	1 184
CCA-secure public-key encryption							
NTRUEncrypt ees743ep1[42]	159[a]	NTRU	no	**K**:	1 194 816	**sk:**	1 120
				E:	57 440	**pk:**	1 027
				D:	110 604	**c:**	980
Lizard [30] (recommended parameters)	128[i]	LWE+LWR	no	**K**:	97 573 000[f]	**sk:**	466 944[g,h]
				E:	≈ 35 050[f]	**pk:**	2 031 616[h]
				D:	≈ 80 840[f]	**c:**	1 072

[a] According to the conservative "best known quantum attack" estimates from [4].
[b] Benchmarked on a 2.6GHz Intel Xeon E5 (Sandy Bridge).
[c] Benchmarked on a 2.3GHz Intel Core i7-4712MQ.
[d] Benchmarked on "PC (Macbook Pro) with 2.6GHz Intel Core i5".
[e] Benchmarked by eBACS [14] on Intel Xeon E3-1275 (Haswell).
[f] As reported by the software from https://github.com/LizardOpenSource/Lizard_c, compiled with gcc-6.3 with flags `-O3 -fomit-frame-pointer -msse2avx -mavx2 -march=native` on Intel Core i7-4770K.
[g] Unlike our scheme, the paper reports secret-key size without the public key required for decryption in the Targhi-Unruh transform.
[h] Sizes used by the software; those could be compressed by a factor 1.6, incurring only small computational overhead.
[i] According to the conservative "best known quantum attack" estimates from [4], with appropriate adaptations (balanced lattice attacks [30, Sec. 4.2]).
[k] The problem underlying NTRU Prime is subtly different than in NTRU; it uses a different ring than commonly used in NTRU and uses deterministic noise.

TABLE 4. MESSAGE SIZES AND CYCLE COUNTS FOR Kyber.UAKE AND Kyber.AKE.

	Bytes		Cycles		
	$P_1 \to P_2$	$P_2 \to P_1$	P_1(start)	P_2	P_1(end)
UAKE	2 272	1 184	199 300	225 544	110 572
AKE	2 272	2 368	196 840	334 672	219 136

[11] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. 4

[12] Daniel J. Bernstein. ChaCha, a variant of Salsa20. In *Workshop Record of SASC 2008: The State of the Art of Stream Ciphers*, 2008. http://cr.yp.to/papers.html#chacha. 10

[13] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime. Cryptology ePrint Archive, Report 2016/461, 2016. http://eprint.iacr.org/2016/461. 2, 9, 13

[14] Daniel J. Bernstein and Tanja Lange. eBACS: ECRYPT benchmarking of cryptographic systems. http://bench.cr.yp.to (accessed 2017-05-19). 7, 13

[15] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. The security impact of a new cryptographic library. In Alejandro Hevia and Gregory Neven, editors, *Progress in Cryptology – LATINCRYPT 2012*, volume 7533 of *LNCS*, pages 159–176. Springer, 2012. http://cryptojedi.org/papers/#coolnacl. 7

[16] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, , and Ronny Van Keer. Keccak code package, 2017. https://github.com/gvanas/KeccakCodePackage (accessed 2017-05-17). 11

[17] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Robert Krauthgamer, editor, *27th SODA*, pages 893–902. ACM-SIAM, January 2016. 2, 3, 9

[18] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 209–224. Springer, Heidelberg, January 2016. 6

[19] 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, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011. 4

[20] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1006–1018. ACM Press, October 2016. 2, 9, 13

[21] 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 *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society Press, May 2015. 2, 9, 11, 13

[22] Colin Boyd, Yvonne Cliff, Juan Gonzalez Nieto, and Kenneth G. Paterson. Efficient one-round key exchange in the standard model. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP 08*, volume 5107 of *LNCS*, pages 69–83. Springer, Heidelberg, July 2008. 8

[23] 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, 12

[24] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS '12 Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012. 2

[25] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. 1

[26] Peter Campbell, Michael Groves, and Dan Shepherd. Soliloquy: A cautionary tale. In *ETSI 2nd Quantum-Safe Crypto Workshop*, pages 1–9, 2014. 2, 3, 9

[27] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001. 8

[28] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. Report on post-quantum cryptography. NISTIR 8105, 2016. http://dx.doi.org/10.6028/NIST.IR.8105. 1

[29] 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. 13

[30] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yong Soo 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, 13

[31] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965. 10

[32] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 559–585. Springer, Heidelberg, May 2016. 2, 3, 9

[33] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short Stickelberger class relations and application to ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, volume 10210 of *LNCS*, pages 324–348. Springer, 2017. 2, 3, 9

[34] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. 8, 12

[35] M. H. Devoret and R. J. Schoelkopf. Superconducting circuits for quantum information: an outlook. *Science*, 339(6124):1169–1174, 2013. 1

[36] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. http://eprint.iacr.org/2012/688. 2

[37] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 467–484. Springer, Heidelberg, May 2012. 8

[38] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999. 6

[39] Florian Göpfert, Christine van Vredendaal, and Thomas Wunderer. A quantum attack on LWE with arbitrary error distribution. IACR Cryptology ePrint Archive report 2017/221, 2017. https://eprint.iacr.org/2017/221. 9

[40] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM STOC*, pages 212–219. ACM Press, May 1996. 9

[41] Shay Gueron and Fabian Schlieker. Speeding up R-LWE post-quantum key exchange. In Billy Bob Brumley and Juha Röning, editors, *Secure IT Systems*, volume 10014 of *LNCS*, pages 187–198. Springer, 2016. https://eprint.iacr.org/2016/467. 11

[42] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. In Helena Handschuh, editor, *Topic in Cryptology – CT-RSA 2017*, volume 10159 of *LNCS*, pages 3–18. Springer, 2017. 9, 13

[43] Jeffrey Hoffstein, Jull Pipher, and Joseph H. Silverman. NTRU: a ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic number theory*, volume 1423 of *LNCS*, pages 267–288. Springer, 1998. https://www.securityinnovation.com/uploads/Crypto/ANTS97.ps.gz. 1

[44] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. IACR Cryptology ePrint Archive report 2017/604, 2017. https://eprint.iacr.org/2017/604. 2, 3, 6, 7

[45] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Heidelberg, November 2017. 7

[46] 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. 9

[47] Andreas Hülsing, Joost Rijneveld, John Schanck, and Peter Schwabe. High-speed key encapsulation from NTRU. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, LNCS. Springer, 2017 (to appear). http://cryptojedi.org/papers/#ntrukem. 13

[48] Zhengzhong Jin and Yunlei Zhao. Optimal key consensus in presence of noise. arXiv preprint arXiv:1611.06150, 2016. http://arxiv.org/pdf/1509.02374v2. 2

[49] Zhengzhong Jin and Yunlei Zhao. Optimal key consensus in presence of noise. IACR Cryptology ePrint Archive report 2017/1058, 2017. https://eprint.iacr.org/2017/1058. 13

[50] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, volume 10210 of *LNCS*, pages 3–26. Springer, 2017. 2, 3

[51] Thijs Laarhoven. *Search problems in cryptography*. PhD thesis, Eindhoven University of Technology, 2015. http://www.thijs.com/docs/phd-final.pdf. 9

[52] A. Langley, W. Chang, N. Mavrogiannopoulos, J. Strombergson, and S. Josefsson. Chacha20-poly1305 cipher suites for transport layer security (TLS). RFC 7905, 2016. https://tools.ietf.org/html/rfc7905. 10

[53] Adam Langley. Cecpq1 results. Posting on Adam Langley's Personal Blog, 2016. https://www.imperialviolet.org/2016/11/28/cecpq1.html. 2, 12

[54] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015. 2, 5

[55] Patrick Longa and Michael Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. IACR Cryptology ePrint Archive report 2016/504, 2016. https://eprint.iacr.org/2016/504. 2

[56] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact Knapsacks are collision resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 144–155. Springer, Heidelberg, July 2006. 1

[57] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May 2010. 1, 2

[58] Daniele Micciancio. Improved cryptographic hash functions with worst-case/average-case connection. In *34th ACM STOC*, pages 609–618. ACM Press, May 2002. 1

[59] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. PKCS #1: RSA cryptography specifications version 2.2. RFC 8017, 2016. https://tools.ietf.org/html/rfc8017. 10

[60] National Institute of Standards and Technology. SHA-3 standard: Permutation-based hash and extendable-output functions. FIPS PUB 202, 2015. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf. 10

[61] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009. 1, 2

[62] Chris Peikert. Lattice cryptography for the Internet. In Michele Mosca, editor, *Post-Quantum Cryptography*, volume 8772 of *LNCS*, pages 197–219. Springer, 2014. http://web.eecs.umich.edu/~cpeikert/pubs/suite.pdf. 2

[63] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 145–166. Springer, Heidelberg, March 2006. 1

[64] John M. Pollard. The fast Fourier transform in a finite field. *Mathematics of computation*, 25(114):365–374, 1971. 10

[65] Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 68–85. Springer, Heidelberg, August 2014. 2

[66] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992. 3

[67] Oded Regev. New lattice based cryptographic constructions. In *35th ACM STOC*, pages 407–416. ACM Press, June 2003. 1

[68] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. 1, 2

[69] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. IACR Cryptology ePrint Archive report 2017/1005, 2017. https://eprint.iacr.org/2017/1005. 7

[70] Gregor Seiler. Faster avx2 optimized ntt multiplication for ring-lwe lattice cryptography. IACR Cryptology ePrint Archive report 2018/039, 2018. https://eprint.iacr.org/2018/039. 11

[71] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuri Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, 2009. 1

[72] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 192–216. Springer, Heidelberg, October / November 2016. 7

[73] Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates. Cryptology ePrint Archive, Report 2016/733, 2016. http://eprint.iacr.org/2016/733. 9

[74] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. Authenticated key exchange from ideal lattices. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 719–751. Springer, Heidelberg, April 2015. 12