Tailoring two-dimensional codes for structured lattice-based KEMs and applications to Kyber

Thales B. Paiva[1], Marcos A. Simplicio Jr[1,2], Syed Mahbub Hafiz[1], Bahattin Yildiz[1], and Eduardo L. Cominetti[1]

[1] Future Security Team, LG Electronics, USA
{thalespaiva, msimplicio, ecominetti}@larc.usp.br
{syedmahbub.hafiz, bahattin.yildiz}@lge.com
[2] Universidade de Sao Paulo, Brazil

August 6, 2024

Abstract. Kyber is a post-quantum lattice-based key encapsulation mechanism (KEM) selected by NIST for standardization as ML-KEM. The scheme is designed to ensure that the unintentional errors accumulated during decryption do not prevent the receiver to correctly recover the encapsulated key. This is done by using a simple error-correction code independently applied to each bit of the message, for which it is possible to show that the decryption failure rate (DFR) is negligible. Although there have been other proposals of more complex error-correction codes for Kyber, these have important limitations. Some proposals use independence assumptions on the noise distribution that do not hold. Others require significant changes in Kyber's core parameters, which make them unpractical. In this work, we propose a family of 2-dimensional codes that can, in principle, be applied to any lattice-based scheme. Even though our 2D codes have a rather simple construction, they can be tailored for the specific noise distribution observed for different Kyber parameters, and reduce Kyber's DFR by factors of $2^{4.8}$, $2^{5.4}$, and $2^{9.9}$, for security levels 1, 3, and 5, respectively, without requiring independence assumptions. Alternatively, the proposed codes allow for up to 6% ciphertext compression in Kyber Level 5 while maintaining the DFR lower than 2^{-160}, which is the target value defined in Kyber's specification. Furthermore, we provide an efficient isochronous implementation of the encoding and decoding procedures for our 2D codes. Compared with Kyber's reference implementation, the performance impact of the 2D codes in the decapsulation time is negligible (namely, between 0.08% to 0.18%, depending on the security level).

1 Introduction

Post-quantum cryptography (PQC) refers to the study of cryptographic algorithms whose underlying security properties rely on computational problems believed to be hard both for classical and quantum computers. The importance of post-quantum (also called quantum-resistant) cryptographic schemes has been

growing in recent years due to continuous advances in the construction of quantum computers, led by multiple players from private companies and government agencies [30]. While existing symmetric primitives are relatively easy to adapt for operating in a quantum setting, the concern of the cryptography community lies on classical and widely adopted public-key schemes based on discrete logarithms (e.g., ECDSA [21]) or integer factorization (e.g., RSA [34]), which can be broken by large-scale quantum computers running the Shor algorithm [39].

Aiming to mitigate this threat, the National Institute for Standards and Technology (NIST) launched in 2016 the PQC Standardization Program to identify suitable post-quantum schemes to be standardized by the institute [29]. After evaluating dozens of submissions over six years, NIST finally announced the initial set of algorithms selected for standardization in July 2022. Among the proposals for key encapsulation mechanisms (KEM), which enables the secure transmission of symmetric key material, NIST chose Kyber [5], a lattice-based scheme, to be standardized as FIPS203 [28].

One relevant property of Kyber, shared with other efficient KEMs built upon lattices or linear codes, is that the decryption mechanism recovers a shared key from its noisy representation, a procedure prone to failure with some probability. At first sight, decryption failures may seem to be only a minor inconvenience, which were overcome simply by having the sender re-encrypt the message with different randomness. However, decryption failures can be exploited in the so-called *reaction attacks* [19], which are today a major concern in both code-based [12, 17] and lattice-based schemes [7, 8, 18]. This prompted designers to adopt different error-correction strategies, aiming to minimize the decryption failure rates (DFR) to negligible levels in practice.

One challenge in this task, though, lies in the nature of the noise to be corrected: since it is typically close to a bell-shaped distribution, traditional error-correction codes that assume uniform bit-error models are not ideal in this scenario. Kyber, for example, uses a 1-dimensional encoding mechanism that facilitates the computation of its DFR without relying on any independence assumptions on the coefficients of the noise polynomial. Nevertheless, this simple encoding is suboptimal with respect to error correction, as it is well-known that better codes can be found in higher dimensions [11, 23, 26, 36, 37].

Recently, there have been studies aimed at adapting Kyber for using higher-dimensional lattice codes [23,36,37]. Unfortunately, existing proposals come with severe limitations. For example, the recent work by Liu and Sakzad [23] assumes the coefficients of the noise polynomial accumulated during decryption are independent, which does not hold in practice [7]. Meanwhile, although the work by Saliba et al. [36, 37] does not require independence assumptions, the resulting Kyber variant has a larger ciphertext size. Moreover, all of these approaches [23,36,37] require changing at least one of Kyber's core parameters: the polynomial degree n and/or the modulo q. Consequently, the resulting constructions are unable to take advantage of Kyber's NTT-based efficient polynomial multiplications, a main feature behind the scheme's high performance.

Contributions. We present a new framework for obtaining suitable error-correction codes in two dimensions for lattice-based schemes. While most existing approaches first define an error-correction code to be used, and only then compute its decryption failure rate given a noise distribution, our framework works in the reverse order. More precisely, we start by showing how to compute the 2-dimensional noise distribution for pairs of coefficients in the noise polynomial accumulated during decryption. Then we provide a family of error-correction codes that can be efficiently explored, finding the one providing the minimum decryption failure rate for the observed noise distribution.

When compared to previous work, our proposal has the following benefits:

1. *Concrete DFR improvements upon Kyber, without independence assumptions.* Unlike Liu and Sakzad [23], our work does not rely on additional independence assumptions. Furthermore, different than Saliba's et al. [23,36,37], our codes are able to provide both ciphertext compression and lower DFR. For Kyber's highest security parameters, our approach lowers the DFR from 2^{-175} to 2^{-185} without changing any parameters. Moreover, our proposed codes allow 2% to 6% ciphertext compression while maintaining Kyber's DFR close to the values targeted by the current standard.
2. *Emphasis on crypto-agility.* All previous encoding proposals for Kyber require changes in its core parameters n and q, which define the polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, used in all operations. This undermines crypto-agility since most optimizations for NTT-based multiplication and modular operations from existing implementations would not be directly applicable. In contrast, this work only proposes changes to the parameters related to ciphertext compression, namely $d_{\mathbf{u}}$ and d_v, while n and q are kept unchanged.
3. *Simpler security analysis.* Our construction is rather straightforward compared with previous work, which heavily relies on the theory and practice of lattice codes. This makes our proposal more suitable for real-world adoption since complex schemes are commonly more prone to errors both in their security analyses and implementation.
4. *Negligible performance impact, while avoiding timing side-channels.* Unlike previous work, we evaluate our proposal's performance via an isochronous implementation – i.e., one built to ensure that the number of operations does not depend on any secret information. Even so, the performance impact obtained remains between 0.08% and 0.18%.
5. *Fully reproducible.* To facilitate independent verification, we have prepared the code and data that allow anyone to reproduce the results presented in this work. As soon as we get authorization from our legal team, the whole code will be made publicly available at `https://github.com/thalespaiva/kyber2d`.

Paper organization. Section 2 reviews background concepts and our notation. Section 3 describes Kyber and discusses related works on alternative encoding methods. Section 4 describes how to compute the distribution of pairs of coefficients of the accumulated noise polynomial. Section 5 describes our proposed

2-dimensional codes and how good parameters can be chosen to minimize the decryption failure rate based on the distribution computed in the previous section. Section 6 shows how more aggressive compression factors and algorithms can be used to obtain smaller ciphertexts for Kyber. Section 7 summarizes our proposed parameters and discusses how they impact crypto-agility. Section 8 concludes the discussion with open questions and ideas for future work.

2 Background and notation

For any prime q, we write $\mathbb{Z}_q$ to denote the field of integers modulo q. When n is a fixed positive integer, we let R_q denote the polynomial ring $\mathbb{Z}_q[x]/(x^n+1)$. Then, R_q^k is the free module[1] of rank k whose scalars are polynomials in R_q. Polynomials $a \in R_q$ are denoted using lowercase letters. Vectors $\mathbf{a} \in R_q^k$ and matrices $\mathbf{A} \in R_q^{k\times k}$ are denoted in bold using lowercase and uppercase, respectively, where $k \geqslant 1$. When $\mathbf{u}, \mathbf{v} \in R_q^k$, we let $\langle \mathbf{u}, \mathbf{v} \rangle \in R_q$ denote their dot product.

To get the vector-equivalent of a polynomial, we define the `poly_to_vec` function, which, given a polynomial $a \in R_q$, returns its n coefficients as a vector in $\mathbb{Z}_q^n$. In other words, given the polynomial $a = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}$, we have $\mathbf{a} = \texttt{poly\_to\_vec}(a) = \big[a_0, a_1, \ldots, a_{n-1}\big]$. With a slight abuse of notation, we denote the i-th coefficient of a polynomial $a \in R_q$, associated with the power x^i, by either a_i (when discussing the polynomial form of a) or by $a[i]$ (when discussing its vector equivalent), where $0 \leqslant i < n$. Analogously, pairs of coefficients from polynomial a are denoted by $a[i,j] = (a[i], a[j])$. If a polynomial has n coefficients, the circular distance between the coefficients associated with x^i and x^j is $\min(i-j \bmod n, j-i \bmod n)$.

We represent discrete probability distributions as key-value mappings, and, accordingly, if X is a variable whose distribution follows $\mathcal{D}$, then $\mathcal{D}[k]$ denotes the probability that $X = k$. We denote by $\mathcal{B}_\eta$ the centered binomial distribution (CBD) with range $[-\eta, \eta]$.

Let $\texttt{negashift}_i$ be the function that returns a negacyclic shift of the vector-equivalent of a polynomial a by i positions. For example, if $a = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}$, then $\texttt{negashift}_i(a) = \big[a_i, \ldots, a_0, -a_{n-1}, \ldots, -a_{i+1}\big]$. With this notation, we can represent the product of polynomials a and b in the negacyclic ring R_q using its vector form, whose i-th coefficient is given by

$$\texttt{poly\_to\_vec}(ab)[i] = \langle \texttt{poly\_to\_vec}(a), \texttt{negashift}_i(b) \rangle \,. \tag{1}$$

If $x \in \mathbb{Z}_q$, then $|x|$ denotes the minimum between the absolute values of x and $x - q$. Furthermore, $y \leftarrow \text{Compress}(x, d)$ denotes the lossy compression of x to d bits, where $d < \lceil \log_2 q \rceil$. The compression function is defined as $\text{Compress}(x, d) = \big\lfloor (2^d/q)x \big\rceil \bmod 2^d$, where $\lfloor \cdot \rceil$ denotes the rounding function that rounds up on ties. The decompression is defined as $x' = \text{Decompress}(y, d) = \big\lfloor (q/2^d)y \big\rceil$. The error $|x' - x|$ caused by compression and decompression is then

[1] Modules are generalizations of vector spaces that allow scalars to be members of a ring instead of requiring a field.

approximately uniform over the set $\{-\lfloor q/2^{d+1} \rfloor, \ldots, \lceil q/2^{d+1} \rceil\}$, with possibly some slight skewness due to q not being a power of 2.

3 Kyber

This section briefly reviews Kyber's main procedures and selection of security parameters. We also discuss previous work on alternative encoding mechanisms proposed for Kyber that are related to our construction.

3.1 Parameters and algorithms

Kyber is a lattice-based key encapsulation mechanism (KEM) whose security relies on the intractability of the module learning with errors (MLWE) problem. Essentially, it enables two parties to establish a shared 256-bit secret. In what follows, we present a slightly simplified version of Kyber that, although lacking some details, is enough for the purpose of our discussion. In particular, we describe only the underlying algorithms that make the core of Kyber secure against chosen-plaintext attacks (CPA), ignoring the implicit-rejection Fujisaki-Okamoto (FO) transformation that makes Kyber secure against adaptive chosen-ciphertext attacks (CCA) [14, 20]. Furthermore, we omit the optimizations based on the number theory transform (NTT) that are part of the original Kyber specification.

Setup. Kyber supports three (out of the five) security levels defined by NIST, namely levels 1, 3, and 5. For all security levels, Kyber fixes parameters $q = 3329$ and $n = 256$. These parameters define the polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, over which most of the operations are performed. This is interesting from a crypto-agility point of view, as it allows any optimizations or hardware acceleration for operations in R_q to be reused across all security settings.

Given a desired security level, the setup takes public parameters $k, \eta_1, \eta_2, d_{\mathbf{u}}$, and d_v from Table 1. Parameter k defines the sizes of the modules used in the scheme. Parameters η_1 and η_2 define the centered binomial distributions $\mathcal{B}_{\eta_1}$ and $\mathcal{B}_{\eta_2}$ used to generate coefficients with small norm in $\mathbb{Z}_q$. Integers $d_{\mathbf{u}}$ and d_v are the number of bits into which coefficients from the two parts of the ciphertext are compressed. Table 1 also provides an upper bound on the decryption failure rate (DFR) for each parameter set according to Kyber's security analysis.

Kyber comprises three operations for every security level: Key Generation, Encryption, and Decryption, which are detailed next.

Key generation. Let $\mathbf{A}$ be a $k \times k$ matrix of polynomials sampled uniformly at random from R_q. Sample two vectors $\mathbf{s}$ and $\mathbf{e}$ from $\mathcal{B}_{\eta_1}(R_q^k)$, i.e., the coefficients of their polynomials are sampled according to the centered binomial distribution $\mathcal{B}_{\eta_1}$. Compute vector $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \in R_q^k$. The resulting public key is the pair $(\mathbf{A}, \mathbf{t})$, while the private key is vector $\mathbf{s} \in R_q^k$.

Table 1. Kyber parameters for each security level [5].

NIST security	Parameter set	k	η_1	η_2	$d_{\mathbf{u}}$	d_v	Ciphertext size (bytes)	DFR
Level 1	Kyber512	2	3	2	10	4	768	$2^{-139.1}$
Level 3	Kyber768	3	2	2	10	4	1088	$2^{-165.2}$
Level 5	Kyber1024	4	2	2	11	5	1568	$2^{-175.2}$

Encryption. Let $\mathbf{m}$ be an n-bit message to be encrypted using public-key $(\mathbf{A}, \mathbf{t})$. Sample two vectors $\mathbf{r}$ and $\mathbf{e}_1$ from $\mathcal{B}_{\eta_1}\left(R_q^k\right)$ and $\mathcal{B}_{\eta_2}\left(R_q^k\right)$, respectively. Similarly, sample a polynomial e_2 from $\mathcal{B}_{\eta_2}(R_q)$. Let $\mathbf{u} = \mathbf{A}^{\mathrm{T}}\mathbf{r} + \mathbf{e}_1$. Compute polynomial $z = \langle \mathbf{t}, \mathbf{r} \rangle + e_2$. Let $v = \text{Encode}\,(\mathbf{m}) + z$, where the encoding function, when applied to each bit b of $\mathbf{m}$, behaves as follows

$$\text{Encode}\,(b) = \begin{cases} 0, \text{ if } b = 0, \text{ and} \\ \lceil q/2 \rceil, \text{ if } b = 1. \end{cases}$$

Compress the coefficients of vector $\mathbf{u}$ and polynomial v to $d_{\mathbf{u}}$ and d_v bits, respectively, obtaining $\mathbf{c}_{\mathbf{u}} = \text{Compress}(\mathbf{u}, d_{\mathbf{u}})$ and $c_v = \text{Compress}(v, d_v)$. Finally, return the ciphertext $(\mathbf{c}_{\mathbf{u}}, c_v)$.

Decryption. To decrypt a ciphertext $(\mathbf{c}_{\mathbf{u}}, c_v)$ using secret key $\mathbf{s}$, first decompress the ciphertext components obtaining $\mathbf{u}' = \text{Decompress}(\mathbf{c}_{\mathbf{u}}, d_{\mathbf{u}})$ and $v' = \text{Decompress}(c_v, d_v)$. Then compute $m' = v' - \langle \mathbf{u}', \mathbf{s} \rangle$. If we let

$$\begin{cases} \Delta\mathbf{u} & = \text{Decompress}(\mathbf{c}_{\mathbf{u}}, d_{\mathbf{u}}) - \mathbf{u}, \text{ and} \\ \Delta v & = \text{Decompress}(c_v, d_v) - v, \end{cases}$$

then expanding m' gives us $m' = \text{Encode}\,(\mathbf{m}) + \Delta m$, where the accumulated noise polynomial Δm is given by

$$\Delta m = \langle \mathbf{e}, \mathbf{r} \rangle - \langle \mathbf{s}, \mathbf{e}_1 + \Delta\mathbf{u} \rangle + e_2 + \Delta v.$$

Kyber's parameters are carefully chosen to ensure that polynomial Δm has only relatively small coefficients. Therefore, the message can be recovered by computing $\hat{m} = \text{Decode}\,(m')$, where the decoding function, applied to each coefficient $m'[i]$ of the noisy message polynomial m', returns

$$\text{Decode}\,(m'[i]) = \begin{cases} 0, \text{ if } |m'[i]| < \lceil q/4 \rceil, \text{ and} \\ 1, \text{ otherwise.} \end{cases}$$

In the next section, we briefly discuss how Kyber parameters are chosen to guarantee security and a negligible decryption failure rate.

3.2 Security and decryption failure rate

Kyber's design and security analysis revolve around finding parameters that ensure the MLWE problems protecting the secret key and the ciphertext are hard to solve while maintaining a negligible DFR. To facilitate the scheme's security analysis, the Kyber team provides public scripts[2] that compute the complexity of known attacks and the resulting DFR for a given parameter set.

The parameters having the most impact on the MLWE security are the modulus q, the degree n, and the module dimension k, together with the parameters η_1 and η_2 that control the noise added to the LWE samples. Though not as much, the ciphertext compression parameters $d_{\mathbf{u}}$ and d_v can also affect security. For example, lowering $d_{\mathbf{u}}$ and d_v compresses the ciphertext, which results in some additional noise that must be handled during decompression, translating to a harder MLWE instance for the ciphertext.

A necessary condition for a KEM to provide chosen-ciphertext attack (CCA) security is to resist attacks exploiting decryption failures [8,18]. In Kyber's round 2 specification, the DFR target was defined as 2^{-160} for all security levels [4, §1.5]. However, in Kyber's most recent specification, the authors relaxed the DFR requirement to 2^{-128} for level 1, while keeping the 2^{-160} target for levels 3 and 5 [5, §1.4 and §4.4]. In this work, we only propose parameter sets that meet the DFR targets required by Kyber's latest specification [5], for each security level.

For a given parameter set, the DFR is algorithmically computed as follows. Since the distribution of all coefficients $\Delta m[i]$ are individually the same, we can start by computing the distribution of $\Delta m[0]$. This is done by considering the sums of the distributions corresponding to the right-hand side of the following equation

$$\Delta m[0] = \langle \mathbf{e}, \mathbf{r} \rangle [0] - \langle \mathbf{s}, \mathbf{e}_1 + \Delta \mathbf{u} \rangle [0] + e_2[0] + \Delta v[0],$$

which are easy to compute. Then, an upper bound on the DFR is computed using the union bound:

$$\begin{aligned} \Pr\left(\text{Decryption fails}\right) &= \Pr\left(|\Delta m[i]| \geqslant q/4 \text{ for any } 0 \leqslant i < n\right) \\ &\leqslant n \Pr\left(|\Delta m[0]| \geqslant q/4\right). \end{aligned}$$

Notice that, although the coefficients of $\Delta m[i]$ are not independent, the fact that each message bit is encoded into one polynomial coefficient allows us to eliminate any dependence problems with the union bound. In contrast, we show in Section 4 how encoding pairs of bits into two polynomial coefficients requires a more complicated analysis. In particular, to compute the DFR for such 2-dimensional encoding schemes, one needs to accurately compute the probability distribution of pairs of coefficients $\Delta m[i,j] = (\Delta m[i], \Delta m[j])$ of the noise polynomial Δm.

We remark that, within all of our work, we do not propose changes to parameters q, n, k, η_1, or η_2. However, we consider ciphertext compression by proposing

[2] https://github.com/pq-crystals/security-estimates

smaller choices for $d_{\mathbf{u}}$ or d_v while keeping the DFR close to the values originally proposed by Kyber for each security level. Therefore, the resulting scheme should provide better MLWE-related security without significantly lowering Kyber's resilience against attacks that rely on decryption failures.

3.3 Previous work on alternative encoding methods for Kyber

Following Regev's [33] work, most of the efficient lattice-based schemes, including Kyber, use the same encoding scheme during encryption: each bit b of the message is encoded into $\mathbb{Z}_q$ as $b\lceil q/2 \rceil$. However, some schemes take additional steps to achieve better error correction. For example, some lattice-based candidates in the first round of NIST's post-quantum standardization process [1] apply distinct error-correction codes to the message before encryption: LAC [25] uses well-known BCH codes; Round5 [6] uses a custom code named XEf [35]; and NewHope [2] uses repetition codes. Interestingly, a previous version of NewHope [3] used more complex, 4-dimensional lattice codes, but those were superseded in favor of the simpler repetition codes, which are easier to understand and to analyze.

There are also more recent proposals [23,36,37] that, similarly to this work, propose the use of higher-dimensional codes in Kyber. One example is Liu and Sakzad [23], who propose using lattice codes with dimensions 16 and 24. While their construction requires a different polynomial degree n, they claim results that improve both the DFR and the ciphertext size. Unfortunately, their work, like most proposals for error-correction in lattice-based schemes, requires independence assumptions on the coefficients of the noise polynomial Δm, which do not hold in practice [7]. In particular, these assumptions would break Kyber's DFR arguments from Section 3.2, so it would be hard (if at all feasible) to adapt Liu and Sakzad's [23] work to Kyber's design.

More closely related to our work is the approach taken by Saliba et al. [37], which is explained in depth in Saliba's PhD thesis [36]. Their work proposes a variant of Kyber based on reconciliation, which in lattice-based schemes refers to a procedure in which the sender and receiver produce the same shared string from different noisy versions of it. This contrasts with the encoding-decoding paradigm, where the intended shared message is predefined. Their construction uses 8-dimensional lattice codes and does not require independence assumptions, so it can be seen as an extension of the original NewHope's DFR analysis [3,31] to Kyber. While Saliba et al. were able to obtain between 10 to 15 extra bits of LWE security for Kyber's 3 security levels, their proposal has a few practical shortcomings that can be seen in Table 2. One of the main shortcomings of their proposal is that the values of the modulo q are powers of two, which means they cannot use the NTT for polynomial multiplication. Furthermore, their scheme increases the ciphertext size in all security levels, while the DFR is increased in both levels 1 and 5. For example, when compared with Kyber, there is a noticeable increase in the DFR for level 5, by a factor of 2^{37}.

In summary, since the approaches found in the literature [23,36,37] on alternative Kyber encoding mechanisms require either changing n or q, they do not

Table 2. The DFR and ciphertext sizes obtained by Saliba et al. [36,37].

Security	q	Ciphertext size (bytes)	DFR	Relative ciphertext size	Relative DFR	Advantages (ciphertext size and DFR)
Level 1	2^{11}	832	2^{-133}	108.3%	2^{6}	**None**
Level 3	2^{11}	1184	2^{-174}	108.8%	2^{-10}	Better DFR
Level 5	2^{11}	1600	2^{-137}	102.0%	2^{37}	**None**

benefit from a core feature in Kyber: the fast NTT-based multiplication. For instance, Saliba et al.'s proposal [37] requires different values of q for each security level, hindering Kyber's crypto-agility properties. Furthermore, their proposal's performance impact is not reported, and we were unable to find any publicly available implementation for conducting an independent evaluation.

4 The joint distribution of two noisy coefficients

In this section, we describe the initial step of our work, which is the computation of the joint distribution of pairs of coefficients within the noise polynomial Δm. These results are then used in an optimization step that aims to find the best possible 2-dimensional code without changing Kyber's original parameters.

As discussed in Section 3, Kyber uses exactly one coefficient of the message polynomial to encode each message bit. This choice avoids the problem of dealing with the natural dependence between the coefficients of the accumulated noise polynomial Δm and allows using the union bound to compute an upper bound on the decryption failure probability. However, the cost to pay for this simplicity is a suboptimal encoding strategy [11,23,26,36,37].

Conversely, this section shows that it is possible to efficiently compute the joint distribution of two coefficients of Δm when their circular distance is $n/2$. In other words, we characterize the joint probability distribution

$$\Pr\left(\Delta m[i, i + n/2 \bmod n]\right) = \Pr\left(\Delta m[i], \Delta m[i + n/2 \bmod n]\right).$$

Our main motivation is that, with this distribution at hand, we can search for an optimal encoding scheme for Kyber using two coefficients of the message polynomial to encode a pair of bits from the message $\mathbf{m}$ to be encrypted.

4.1 The source of dependence among entries of the noise polynomial

Consider the noise polynomial $\Delta m = \langle \mathbf{e}, \mathbf{r} \rangle - \langle \mathbf{s}, \mathbf{e}_1 + \Delta \mathbf{u} \rangle + e_2 + \Delta v$. Notice that, by definition, all coefficients from e_2 and Δv are independent. However, the coefficients of the polynomials resulting from the two dot products $\langle \mathbf{e}, \mathbf{r} \rangle$ and $\langle \mathbf{s}, \mathbf{e}_1 + \Delta \mathbf{u} \rangle$ cannot be assumed to be independent, because they are computed through sums of polynomial multiplications. If ignored, this dependence is known to cause issues when estimating the DFR in scenarios where error-correction

Table 3. Possible outcomes considering the equiprobable possibilities of polynomials $a, b \in \mathbb{F}_2[x]/\left(x^2+1\right)$, and $c = ab$.

a_0	a_1	b_0	b_1	c_0	c_1
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0

a_0	a_1	b_0	b_1	c_0	c_1
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	1	1	1

a_0	a_1	b_0	b_1	c_0	c_1
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	1

a_0	a_1	b_0	b_1	c_0	c_1
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	0	0

codes are used, leading to theoretical and practical effects studied in detail by D'Anvers et al. [7].

For a concrete demonstration of this dependence, consider, as a toy example, the case when a and b are random polynomials with $n = 2$ binary coefficients, say $a, b \in \mathbb{Z}_2[x]/\left(x^2+1\right)$. If we let $c = ab = c_0 + c_1 x$, then $c_0 = a_0 b_0 - a_1 b_1$ and $c_1 = a_0 b_1 + a_1 b_0$. Consider Table 3, which shows all equiprobable outcomes for $c = ab$. We notice that c_0 and c_1 are not independent because $\Pr(c_1 = 1) = 3/8$ but $\Pr(c_1 = 1 | c_0 = 0) = 1/5$.

Now, let us focus on $\langle \mathbf{e}, \mathbf{r} \rangle$, which is the simplest of the two dot products that define Δm in Kyber. Since both $\mathbf{e}$ and $\mathbf{r}$ are vectors of polynomials whose coefficients are taken from the centered binomial distribution $\mathcal{B}_{\eta_1}$, their dot product can be written as

$$\langle \mathbf{e}, \mathbf{r} \rangle = \mathbf{e}[0]\mathbf{r}[0] + \ldots + \mathbf{e}[k-1]\mathbf{r}[k-1].$$

We start by noticing that every product of polynomials $\mathbf{e}[i]\mathbf{r}[i]$ is independent of $\mathbf{e}[j]\mathbf{r}[j]$ for $j \neq i$. Also, because all $\mathbf{e}[i]$ and $\mathbf{r}[i]$ are sampled from the same $\mathcal{B}_{\eta_1}$, every product $\mathbf{e}[i]\mathbf{r}[i]$ follows the same distribution for all i. Therefore, in what follows, we focus our attention on the joint distribution of $\mathbf{e}[i]\mathbf{r}[i]$ for any particular i to analyze the distribution of $\langle \mathbf{e}, \mathbf{r} \rangle$.

4.2 The joint distribution of coefficients in a polynomial product

In this section, we present a mathematical result that allows us to separate the joint probability distribution of two coefficients of a polynomial product in $R_q = \mathbb{Z}_q[x]/(x^n+1)$ into a sum of independent distributions, provided that three conditions are met: (*i*) n must be a power of 2; (*ii*) the circular distance between the two coefficients is n/2; and (*iii*) the probability distribution for the coefficients of at least one of the polynomials must be symmetric.

Let $a, b \in R_q$ and consider their product $c = ab$. Suppose we want to compute the joint distribution of the two coefficients $c[i, j]$. Using a direct, naive approach, we would have to consider all possible values of a and b, as well as their products. At first sight, that task might appear to be reasonably easy in Kyber because the k polynomials that build $\mathbf{e}$ and $\mathbf{r}$ come from the centered binomial distribution with a small parameter η_1. However, since there are $(2\eta_1 + 1)^n$ possibilities for each polynomial, such a naive approach ends up being highly impractical even for small values of η_1.

To handle this issue, we proceed as follows. First, we build upon Lemma 1 to obtain a complete characterization of the probability distribution of $c[i, i + n/2 \bmod n]$ as the sum of smaller, efficiently computable distributions. We prove the lemma for the joint distribution of $c[0, n/2]$, aiming for a cleaner presentation. Then, in Proposition 1, we show that, under the same conditions required by Lemma 1, the distributions of $c[i, i + n/2 \bmod n]$ and $c[0, n/2]$ are the same.

Lemma 1. Let $n \geqslant 4$ be a power of 2. Consider the distribution $\mathcal{P}_n$ over pairs $(c_0, c_{n/2})$ defined by the following experiment. Let $\mathcal{D}_1$ be any distribution over $\mathbb{Z}_q$, and $\mathcal{D}_2$ be a symmetric distribution over the same set. Choose polynomials a and b in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ by taking its coefficients from distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. Compute their product $c = ab \in R_q$, and output the pair $(c_0, c_{n/2})$ consisting of the coefficients of c associated with powers x^0 and $x^{n/2}$, correspondingly. Then $\mathcal{P}_n$ can be split as $\mathcal{P}_n = \mathcal{P}_{n/2} + \mathcal{P}_{n/2}$.

Proof. Let $n = 2^\ell$, for some $\ell \geqslant 2$. Take polynomials a and b from R_q. If $c = ab$, then, using Equation 1 from Section 2, we can write each coefficient of the product as

$$c_i = \texttt{poly\_to\_vec}(c)[i] = \langle \texttt{poly\_to\_vec}(a), \texttt{negashift}_i(b) \rangle .$$

Let $\mathbf{a} = \texttt{poly\_to\_vec}(a)$ and $\mathbf{b} = \texttt{negashift}_0(b)$, i.e., $\mathbf{b}$ is the first column of the negacyclic matrix generated by the coefficients of b. Notice that, since $\mathcal{D}_2$ is symmetric, then $\mathbf{b} = (b_0, -b_{n-1}, \ldots, -b_1)$ has the same distribution as b.

Since $n \geqslant 4$ is a power of 2, then $\mathbf{a}$ and $\mathbf{b}$ can be split into 4 parts of equal length such that

$$\begin{aligned} c_0 &= \left\langle \left[\mathbf{a}_{\text{I}} \ , \ \mathbf{a}_{\text{II}} \ , \ \mathbf{a}_{\text{III}} \ , \ \mathbf{a}_{\text{IV}}\right], \left[\mathbf{b}_{\text{I}} \ , \ \mathbf{b}_{\text{II}} \ , \ \mathbf{b}_{\text{III}} \ , \ \mathbf{b}_{\text{IV}}\right] \right\rangle , \text{ and} \\ c_{n/2} &= \left\langle \left[\mathbf{a}_{\text{I}} \ , \ \mathbf{a}_{\text{II}} \ , \ \mathbf{a}_{\text{III}} \ , \ \mathbf{a}_{\text{IV}}\right], \left[-\mathbf{b}_{\text{III}} \ , \ -\mathbf{b}_{\text{IV}} \ , \ \mathbf{b}_{\text{I}} \ , \ \mathbf{b}_{\text{II}}\right] \right\rangle . \end{aligned}$$

We can then write $c_0 = c'_0 + c''_0$ and $c_{n/2} = c'_{n/2} + c''_{n/2}$, where

$$\begin{cases} c'_0 = \langle \mathbf{a}_{\text{I}}, \mathbf{b}_{\text{I}} \rangle + \langle \mathbf{a}_{\text{III}}, \mathbf{b}_{\text{III}} \rangle , \\ c'_{n/2} = -\langle \mathbf{a}_{\text{I}}, \mathbf{b}_{\text{III}} \rangle + \langle \mathbf{a}_{\text{III}}, \mathbf{b}_{\text{I}} \rangle , \end{cases} \quad \begin{cases} c''_0 = \langle \mathbf{a}_{\text{II}}, \mathbf{b}_{\text{II}} \rangle + \langle \mathbf{a}_{\text{IV}}, \mathbf{b}_{\text{IV}} \rangle , \\ c''_{n/2} = -\langle \mathbf{a}_{\text{II}}, \mathbf{b}_{\text{IV}} \rangle + \langle \mathbf{a}_{\text{IV}}, \mathbf{b}_{\text{II}} \rangle . \end{cases}$$

In other words, we have $(c_0, c_{n/2}) = \left(c'_0, c'_{n/2}\right) + \left(c''_0, c''_{n/2}\right)$. But notice that the pairs $\left(c'_0, c'_{n/2}\right)$ and $\left(c''_0, c''_{n/2}\right)$ are independent, and both follow distribution $\mathcal{P}_{n/2}$. Therefore, $\mathcal{P}_n$ is the sum of equal distributions $\mathcal{P}_{n/2} + \mathcal{P}_{n/2}$. □

Proposition 1. Let a and b be two polynomials in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, for some even n. Also, suppose that the coefficients of a and b are sampled from two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively, where $\mathcal{D}_2$ is symmetric. If $c = ab$ is their product, then, for all $0 \leqslant i < n - 1$, we have $\Pr(c[0, n/2]) = \Pr(c[i, n/2 + i])$.

Proof. The proof is done in two steps. First we observe that $c[0, n_2]$ and $c[i, n/2 + i]$ share a common structure. Then, we use the symmetry of $\mathcal{D}_2$ to show that they are identically distributed.

Let $(\mathbf{u}_{\text{I}}, \mathbf{u}_{\text{II}})$ and $(\mathbf{v}_{\text{I}}, \mathbf{v}_{\text{II}})$ denote the two halves of vectors $\texttt{negashift}_0(b)$ and $\texttt{negashift}_{\text{I}}(b)$, respectively. If we let $\mathbf{a} = \texttt{poly\_to\_vec}(a)$, then

$$\begin{cases} c_0 = \langle \mathbf{a}, [\mathbf{u}_{\text{I}} \ , \ \mathbf{u}_{\text{II}}] \rangle \, , \\ c_{n/2} = \langle \mathbf{a}, [\mathbf{u}_{\text{II}} \ , \ -\mathbf{u}_{\text{I}}] \rangle \, , \end{cases} \quad \begin{cases} c_i = \langle \mathbf{a}, [\mathbf{v}_{\text{I}} \ , \ \mathbf{v}_{\text{II}}] \rangle \, , \\ c_{n/2+i} = \langle \mathbf{a}, [\mathbf{v}_{\text{II}} \ , \ -\mathbf{v}_{\text{I}}] \rangle \, . \end{cases}$$

Now notice that both $(\mathbf{u}_{\text{I}}, \mathbf{u}_{\text{II}})$ and $(\mathbf{v}_{\text{I}}, \mathbf{v}_{\text{II}})$ contain all the coefficients of b, although possibly reordered and with different signs. But since the distribution $\mathcal{D}_2$ is symmetric, then $(\mathbf{u}_{\text{I}}, \mathbf{u}_{\text{II}})$ and $(\mathbf{v}_{\text{I}}, \mathbf{v}_{\text{II}})$ follow the same distribution. Therefore $\Pr(c[0, n/2]) = \Pr(c[i, n/2 + i])$. □

We remark that Lemma 1 can be seen as a consequence of the *polynomial splitting for recovery* used in NewHope's original analysis [3, Section C]. The main difference is that we present the result in a way that allows for a more direct construction of the joint distribution, which, in previous approaches, did not need to be effectively computed. In particular, in the next section, we show how to use this lemma to efficiently compute the joint probability distribution of $\Delta m[i, i + n/2]$.

4.3 Computing the joint distribution of the accumulated noise's coefficients

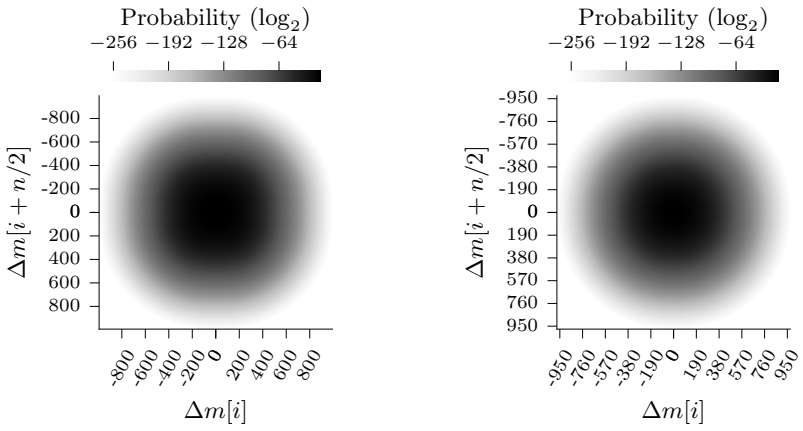
Let us first briefly discuss the applicability of Lemma 1 to Kyber. Consider the two dot products $\langle \mathbf{e}, \mathbf{r} \rangle$ and $\langle \mathbf{s}, \mathbf{e}_1 + \Delta\mathbf{u} \rangle$ that appear in the computation of Δm. The first one is done with polynomials whose coefficients are taken from the centered binomial $\mathcal{B}_{\eta_1}$, which is symmetric. In the second one, elements from $\mathbf{s}$ are also drawn from $\mathcal{B}_{\eta_1}$. Therefore, we can swap the operands of the commutative product so that all of the lemma's hypotheses are satisfied.

Let $\mathcal{P}_{\text{prod}}^{(\phi_a, \phi_b)}$ denote the probability distribution of a product $c = ab$ of two polynomials $a, b \in \mathbb{Z}_q/\left(x^2 + 1\right)$, whose coefficients are selected according to distributions ϕ_a and ϕ_b, correspondingly. Let $\mathcal{D}_{\Delta\mathbf{u}}$ denote the distribution of the coefficients of $\Delta\mathbf{u}$. Then, by Lemma 1, we have:

$$\begin{cases} \langle \mathbf{e}, \mathbf{r} \rangle \, [i, i + n/2] & \sim \frac{kn}{2} \mathcal{P}_{\text{prod}}^{\left(\mathcal{B}_{\eta_1}, \mathcal{B}_{\eta_1}\right)}, \\ \langle \mathbf{s}, \mathbf{e}_1 + \Delta\mathbf{u} \rangle \, [i, i + n/2] & \sim \frac{kn}{2} \mathcal{P}_{\text{prod}}^{\left(\mathcal{B}_{\eta_2}, \mathcal{B}_{\eta_1} + \mathcal{D}_{\Delta\mathbf{u}}\right)}. \end{cases}$$

Notice that the base distributions over coefficient pairs, namely $\mathcal{P}_{\text{prod}}^{\left(\mathcal{B}_{\eta_1}, \mathcal{B}_{\eta_1}\right)}$ and $\mathcal{P}_{\text{prod}}^{\left(\mathcal{B}_{\eta_2}, \mathcal{B}_{\eta_1} + \mathcal{D}_{\Delta\mathbf{u}}\right)}$, are easily computed by enumerating the corresponding polynomials in $\mathbb{Z}_q/\left(x^2 + 1\right)$ and computing their products while keeping track of the associated probabilities. Now let $\mathcal{P}_{(\Delta v + e_2)}$ be the probability distribution of pairs $(\Delta v[i, i + n/2] + e_2[i, i + n/2])$. The product rule can directly compute this distribution since the coefficients in both Δv and e_2 are all independent. Then, we have

$$\Delta m[i, i + n/2] \sim \frac{kn}{2} \mathcal{P}_{\text{prod}}^{\left(\mathcal{B}_{\eta_1}, \mathcal{B}_{\eta_1}\right)} + \frac{kn}{2} \mathcal{P}_{\text{prod}}^{\left(\mathcal{B}_{\eta_2}, \mathcal{B}_{\eta_1} + \mathcal{D}_{\Delta\mathbf{u}}\right)} + \mathcal{P}_{(\Delta v + e_2)}.$$



(a) Level 3. **(b)** Level 5.

Figure 1. Joint distributions $\Pr(\Delta m[i, n/2+i])$ for security levels 3 and 5. Notice that the distribution for Level 3 has a slightly more squared shape.

While the Python scripts[3] provided by the Kyber team are fast enough to compute the distribution of a single coefficient of Δm, their approach is highly inefficient when computing sums of joint distributions. An interesting approach to speed up the computation of the convolutions would be to use the 2-dimensional FFT. However, most available implementations use only double- or up to quadruple-precision floating-point arithmetic, which are not reliable when computing negligibly small cryptographic probabilities.

For example, the error accumulated by each FFT computation would be around $8.48\varepsilon \log_2 n$, where ε denotes the machine epsilon [15, 38]. In our setup, the machine epsilon is $\varepsilon = 1.0842 \times 10^{-19}$ for quadruple precision, resulting in errors of the order $7.3 \times 10^{-18} \approx 2^{-56.9}$. These errors are much larger than the negligible probabilities we are interested in computing, which would make the output of the computation to have no practical value.

To address this problem, we implemented a custom 2-dimensional FFT with multiprecision complex arithmetic using the MPC [10] and MPFR [13] libraries. For simplicity, and mainly because we observed acceptable computation time and memory requirements, we did not attempt to incorporate any optimizations that might be applicable – such as taking advantage of the fact that the inputs are real and the distributions are symmetric [32].

We ran the computations on a standard PC, which has an Intel Core i7-8700 CPU at 3.20GHz and 32G of RAM, using its 12 threads. Under this setup, the computation of the joint probability distribution $\Pr(\Delta m[i, i+n/2])$ with 260 bits of precision takes less than 3 minutes for each parameter set.

[3] https://github.com/pq-crystals/security-estimates

Figure 1 shows the joint distribution $\Pr(\Delta m[i, i + n/2])$ considering levels 3 and 5 of Kyber. We can see that the distribution for level 3 has, overall, a slightly more squared shape than the one for level 5. This results from the difference in d_v, which causes the somewhat uniform rounding error components of Δv. Since $d_v = 5$ in level 5, the uniform errors are less noticeable than in level 3, in which $d_v = 4$. Intuitively, since the overall shapes of the distributions are different, then we should not expect that the best error correction code for one case would also be optimal for the other.

4.4 Challenges when extending the results beyond 4 dimensions

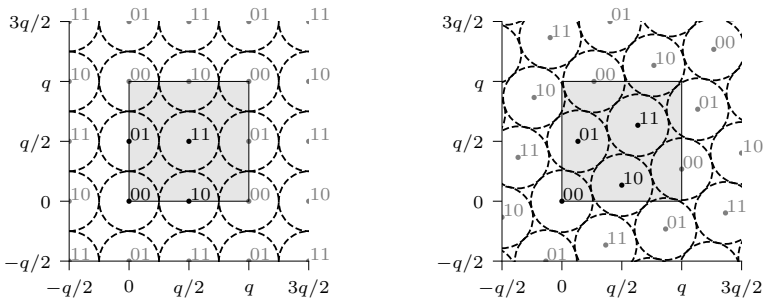
It is possible to extend Lemma 1 and Proposition 1 to any power of two between 2 and n. Indeed, this type of idea has already appeared in previous works on NewHope [3, 31], and also have been applied to a variant of Kyber [36, 37]. However, unlike previous works, our proposal requires the computation of the joint probability distribution of coefficients of the noise polynomial, making it harder to generalize in practice.

For example, for the 4-dimensional setup, we need to compute the joint distribution $\Pr(\Delta m[0, n/4, n/2, 3n/4])$. Using the FFT-based convolutions would require three forward 4-dimensional FFT computations, followed by the inverse transform. Notice that the FFT needs to be computed with padding so that the convolutions do not cause cyclic interference.

Suppose we use ρ-bit precision values for the operations and consider parameters for Kyber Level 5. In particular, let $k = 4$ and $\eta_1 = \eta_2 = 2$. Let us use the value $\mu \approx 2\eta_1 kn = 4096$ as the dimension of each 1-dimensional FFT, with padding. Since this is the same value μ used in our 2D computations, we consider it to be a good starting point for this estimate. The number of operations for each 4-dimensional FFT would be approximately $\mu^4 \log_2(\mu^4) \approx 2^{54}$ operations with ρ-bit complex numbers, which is computationally expensive. In terms of memory usage, without optimizations, we would need at least $2\rho\mu^4$ bits to store the 4D array, where the 2 factor comes from the entries being complex numbers. Such memory requirements could possibly be reduced to $\rho(\mu/2)^4$ bits using FFT symmetries [32], that are applicable since the inputs are real numbers and we deal mostly with symmetric distributions. Unfortunately, for $\rho = 200$ bits, this translates to about 2^{48} bytes ≈ 280 terabytes of memory. Therefore, it appears to be hard to extend our results beyond 4 dimensions without significantly improving how the joint distribution is computed or with a more efficient way to directly compute the DFR.

5 Proposed 2-dimensional codes for Kyber

In this section, we show an important application of the results discussed in Section 4: the construction of new encoding schemes that are provably better at error correction than the one used in Kyber's current specification.



(a) Kyber's code with minimum distance of $\lfloor q/2 \rfloor = 1664$.

(b) A lattice code with minimum distance of about 1722.

Figure 2. Comparison between the original Kyber code seen as a 2-dimensional code and a denser lattice code. The shaded areas represent the $\mathbb{Z}_q^2$ square.

5.1 Motivation

Consider Kyber's mechanism for encoding the message into a polynomial. We can treat it as a two-dimensional code by pretending it encodes a pair (b_0, b_1) of message bits into coefficients $(b_0\lceil q/2 \rceil, b_1\lceil q/2 \rceil) \in \mathbb{Z}_q^2$. This code is illustrated in Figure 2a, where dots denote the codewords, and the circles around them show the radius of minimum distance decoding (i.e., any point falling into the area of a given circle is corrected to the valid codeword at its center). We remark that the minimum distance is computed considering all representatives of the codewords, not only the leading ones in $\mathbb{Z}_q^2$. Notice that the minimum distance of Kyber's code is $\lfloor q/2 \rfloor = 1664$.

The two-dimensional view of Kyber's code in Figure 2a highlights one possible problem: it leaves too much uncovered space under its minimum distance. A notable code family that supports denser codes in 2 or more dimensions are the so-called lattice codes, which are very effective in correcting Gaussian noise. For example, Figure 2b shows a lattice code with minimum distance ≈ 1722.

Even though lattice codes are useful in a variety of contexts, there is an important limitation when trying to employ them in Kyber: the operations are done in $\mathbb{Z}_q$, so the noise polynomial Δm to be corrected is guaranteed to be small only when considering the modulo q representatives of the coefficients centered at 0. Consequently, if we want to use lattice codes in Kyber, it has to be periodic in $\mathbb{Z}_q^2$. This $q \times q$ square is represented as the shaded areas in Figures 2a and 2b.

Previous proposals [36, 37] deal with this issue by changing parameter q to powers of 2, so that one can easily employ an 8-dimensional lattice that is periodic in $\mathbb{Z}_q^8$. While this allows such proposals to exploit the lattice structure when proving the DFR of the resulting scheme, these values of q significantly impact Kyber's performance, because fast NTT multiplication would no longer be available. Furthermore, the resulting scheme requires larger ciphertexts than

Kyber. Conversely, in our proposal (detailed in the next section), we show a core application of the capability of efficiently computing the joint distribution $\Pr(\Delta m[i, i + n/2])$: we can find an optimal 2-dimensional encoding scheme via exhaustive search, without needing it to be a lattice code.

5.2 Optimal 2-dimensional codes for Kyber

Our proposal consists essentially in generating a family of suitable 2-dimensional codes and then evaluate their performance when correcting errors distributed according to the joint probability distribution $\Pr(\Delta m[i, i + n/2])$. Naturally, we want to choose codes that minimize the decryption failure rate (DFR) when plugged into Kyber.

While it may be tempting to simply pick the code with the largest minimum distance, we must be mindful that the best code depends on the nature of the error. For example, if the noise was approximately normally distributed, then a code with the largest minimum distance indeed exists. However, if the noise were uniform in a region, then Kyber's original code would be a better choice. Interestingly, Kyber's errors Δm consist of both an approximately normal factor, coming from $(\langle \mathbf{e}, \mathbf{r} \rangle - \langle \mathbf{s}, \mathbf{e}_1 + \Delta \mathbf{u} \rangle + e_2)$, and a somewhat uniform term yielded from Δv. As shown in Figure 1, this can significantly impact the overall shape of the error distribution, so the optimal 2D code may not be the same for different Kyber parameters. Our goal is, then, to find optimal 2D codes by taking into account the particular shapes of the noise $\Delta m[i, i + n/2]$ when different Kyber parameters are used.

Algorithm 1 shows the steps to compute the DFR for a code $\mathcal{C}$ when the error distribution comes from a known precomputed distribution $\Pr(\Delta m[i, i + n/2])$. First, it computes the error probability p_{failure} for the 2-dimensional code $\mathcal{C}$. This is done by accumulating the probabilities that noise coefficients drawn from $\Pr(\Delta m[i, i + n/2])$ cause decoding failures for each of the 4 possible codewords. Since each codeword appears with probability $1/4$, the error probabilities have to be weighed by this factor when updating the value of p_{failure} in Line 8. Then, the algorithm returns the upper bound on the DFR by considering the union bound over the decoding failure for the $n/2 = 128$ encoded pairs.

We want to define a family $\mathcal{F}$ of codes with two main properties: (i) codes in $\mathcal{F}$ should be efficiently decodable; and (ii) the set $\mathcal{F}$ must not be intractably large. Then, we can do an exhaustive search for the best code $\mathcal{C}_{\text{best}}$ in $\mathcal{F}$ as the one with the lowest DFR. Formally, we would have

$$\mathcal{C}_{\text{best}} = \arg\min_{\mathcal{C} \in \mathcal{F}} \{\text{DFR}(\mathcal{C}, \Pr(\Delta m[i, i + n/2]))\}.$$

Now, we can use heuristics to define the code family with the desired properties. Since we want to have an efficient encoding, it is a good idea to have a linear encoding procedure. More precisely, if we let $\mathbf{z} \in \mathbb{Z}_2^2$ be the pair of bits to be encoded, we would like to encode it as $\mathbf{Cz} \in \mathbb{Z}_q^2$, using a basis matrix

procedure DFR(code $\mathcal{C}$, distribution $\Pr(\Delta m[i, i+n/2])$)
  $p_{\text{failure}} \leftarrow 0$ ▷ Accumulates the decoding error probability for $\mathcal{C}$
  for each codeword $\mathbf{c} \in \mathcal{C}$ **do**
    for each possible error $\mathbf{e} \in \mathbb{Z}_q^2$ **do**
      $\mathbf{c}' = \mathbf{c} + \mathbf{e}$
      if Decode $(\mathbf{c}') \neq \mathbf{c}$ **then**
        ▷ Updates p_{failure} considering that $\mathbf{c}$ appears with probability 1/4
        $p_{\text{failure}} \leftarrow p_{\text{failure}} + \frac{1}{4}\Pr(\Delta m[i, i+n/2])[\mathbf{e}]$
  return $\frac{n}{2} p_{\text{failure}}$ ▷ Union bound over the $n/2 = 128$ pairs.

Algorithm 1. Computation of the decryption failure rate (DFR) for a given 2-dimensional code $\mathcal{C}$.

$\mathbf{C} \in \mathbb{Z}_q^{2\times 2}$. We can also make decoding more efficient by exploiting symmetries[4], so we can require that $\mathbf{C}$ is symmetric.

Using these constraints, we propose the family of codes $\mathcal{F}$ where each code is described by parameters α and β in $\mathbb{Z}_q$, which define the basis matrix $\mathbf{C}$. Formally, we can write

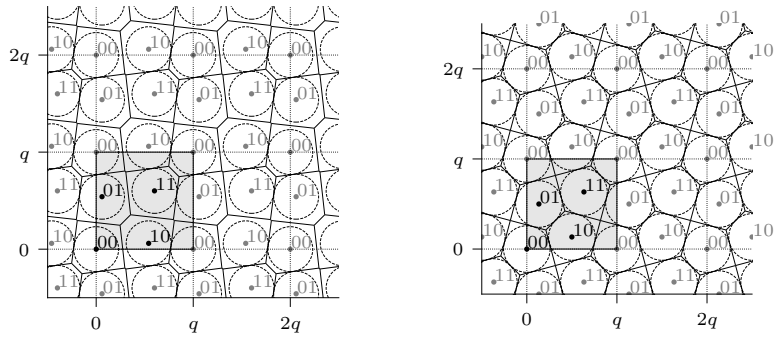
$$\mathcal{F} = \left\{\mathcal{C}_{(\alpha,\beta)} : (\alpha,\beta) \in \mathbb{Z}_q^2\right\}, \quad \text{where} \quad \mathcal{C}_{(\alpha,\beta)} = \left\{ \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix} \mathbf{z} : \mathbf{z} \in \mathbb{Z}_2^2 \right\}.$$

For example, when viewed in two dimensions, we notice that $\mathcal{C}_{(1664,0)}$ is equivalent to Kyber's original code[5]. As extra examples, Figures 3a and 3b show two codes from $\mathcal{F}$, namely $\mathcal{C}_{(1800,200)}$ and $\mathcal{C}_{(1664,446)}$. Notice that, in addition to the radius, the figures also show the polygons representing the Voronoi cells[6] associated with each codeword. With $\mathcal{C}_{(1800,200)}$, the minimum distance is $1542 < 1664$, so it can be considered reasonably worse than Kyber's original code. In contrast, $\mathcal{C}_{(1664,446)}$ maximizes the minimum distance in $\mathcal{F}$, so it appears to be an interesting intermediate between Kyber's code and the lattice code from Figure 2b: it has the minimum distance of 1722 while being periodic in $\mathbb{Z}_q^2$.

To find the actual best codes $\mathcal{C}_{(\alpha,\beta)}$ in $\mathcal{F}$ for each of Kyber's security levels, we simply performed an exhaustive search over all possible parameters. The best parameter α for all levels was $1664 = \lfloor q/2 \rfloor$. However, the best value for β varied, as shown in Figure 4. As an important sanity check, notice that we observe essentially the same DFR values as the original Kyber for $\mathcal{C}_{(1664,0)}$.

The best codes obtained for each security level are shown in Table 4. It is interesting to notice that none of the best codes achieves the maximum minimum distance of 1722 seen in Figure 3b. Even though this may seem counterintuitive initially, the explanation is related to the nature of the error factor Δv. Recall that coefficients in Δv are errors caused by compression and decompression, which are approximately uniform in $\left\{-\lfloor q/2^{d_v+1} \rfloor, \ldots, \lceil q/2^{d_v+1} \rceil\right\}$. Furthermore,

[4] We explicitly show in Section 5.3 how to exploit symmetries for decoding.
[5] While $\mathcal{C}_{(1665,0)}$ would be a more precise definition than $\mathcal{C}_{(1664,0)}$ for Kyber's original code viewed in 2D, they are equivalent with respect to error correction and DFR.
[6] The Voronoi cell of a codeword $\mathbf{c}$ is the set of points that are closer to $\mathbf{c}$ than to any other codeword.



(a) Code $\mathcal{C}_{(1800,200)}$ with minimum distance ≈ 1542.

(b) Code $\mathcal{C}_{(1664,446)}$ that maximizes the minimum distance over $\mathcal{F}$, which is ≈ 1722.

Figure 3. Examples of codes in $\mathcal{F}$. The radiuses of the circles are half of the code's minimum distance, and polygons represent the Voronoi cells of each codeword.

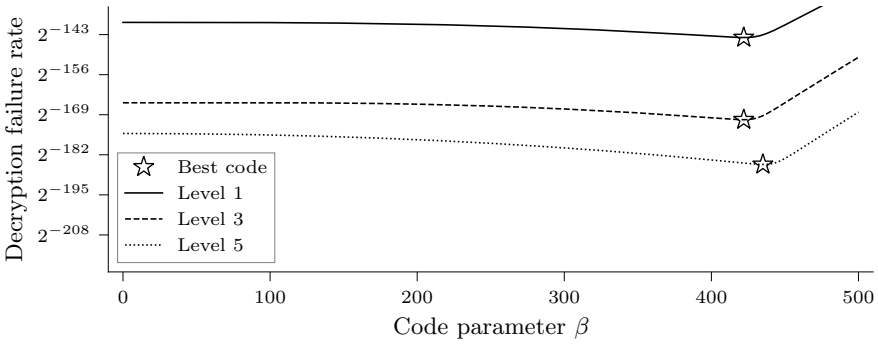


Figure 4. The DFR of codes $\mathcal{C}_{(1664,\beta)}$, considering Kyber parameters for the different security levels.

when the impact of the uniform factor Δv in the coefficients of Δm is high, the overall shape of $\Pr(\Delta m[i, i + n/2])$ looks more like a squared circle, as illustrated in Figure 1. In this case, the best code is not the one that maximizes the minimum Euclidean between codewords, but a code that balances between maximizing the Euclidean and Manhattan distances among its codewords.

In our proposed code family $\mathcal{F}$, when $\alpha = 1664$, the balance between Euclidean and Manhattan distances is controlled by parameter β. When $\beta = 0$, the minimum Manhattan distance between codewords is maximized, which makes the code better at correcting errors with a stronger uniform component. Conversely, when β approaches 446, the code gets better at correcting bell-shaped

```
void poly_frommsg(poly *r, const uint8_t msg[32]) {
  size_t base = 0;  // Invariant: base = 8*i + j/2
  for(size_t i = 0; i < KYBER_N/8; i++) {
    for(size_t j = 0; j < 8; j += 2) {
      // The masks below are 0xffff if bit is 1, and 0x0000 if 0.
      uint16_t bit0_mask = -((msg[i] >> j) & 1);
      uint16_t bit1_mask = -((msg[i] >> (j + 1)) & 1);
      uint16_t coeff0 = (ALPHA & bit1_mask) + (BETA & bit0_mask);
      uint16_t coeff1 = (BETA & bit1_mask) + (ALPHA & bit0_mask);
      r->coeffs[base] = coeff0;
      r->coeffs[base + KYBER_N/2] = coeff1;
      base++;
    }
  }
}
```

Algorithm 2. Isochronous implementation of proposed encoding function.

errors. Now, since $d_v = 4$ in levels 1 and 3, and $d_v = 5$ in level 5, we observed the same value for β in levels 1 and 3, but a slightly larger β for level 5.

Table 4. The best codes in $\mathcal{F}$ for each security level, together with their DFRs.

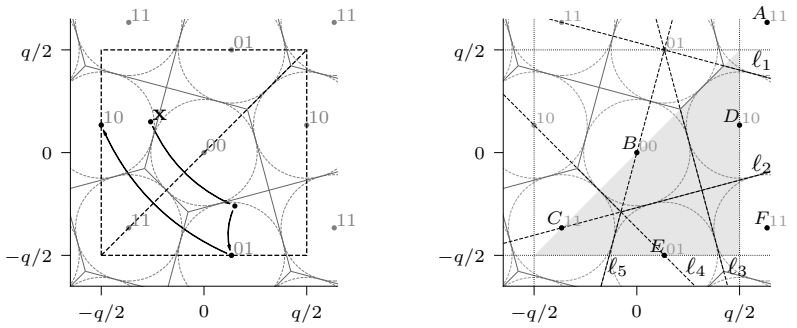
Security	Best code in $\mathcal{F}$	Minimum distance	DFR
Level 1	$\mathcal{C}_{(1664,422)}$	1716	$2^{-143.9}$
Level 3	$\mathcal{C}_{(1664,422)}$	1716	$2^{-170.6}$
Level 5	$\mathcal{C}_{(1664,435)}$	1720	$2^{-185.1}$

5.3 Implementation and performance

We now discuss how the optimal codes found in Section 5.2 can lead to efficient isochronous implementations – i.e., one where the number of operations does not depend on any secret information. Since the encoding and decoding algorithms can be listed using just a few lines of code, we present the algorithms in C. This is useful because it facilitates showing how the proposal can be implemented in an isochronous manner, following programming practices usually employed to protect against timing attacks. In addition, for any interested developer, the encoding and decoding procedures can be directly applied to Kyber's existing C implementations quite easily.

Encoding. The encoding function is shown in Algorithm 2. The function iterates through adjacent pairs $\mathbf{z}$ of message bits, outputting the encoded coefficients as $c_0 = \alpha\mathbf{z}[1] + \beta\mathbf{z}[0]$ and $c_1 = \beta\mathbf{z}[1] + \alpha\mathbf{z}[0]$. These values are then put into polynomial coefficients separated by a circular distance of $n/2$, which is required by our analysis of the joint probability distribution from Section 4.

Decoding based on approximate Voronoi cells. In Kyber's implementation, the input polynomial to the decoding function is represented by a sequence



(a) Symmetry used for decoding a point $\mathbf{x}$ in the upper triangle.

(b) Lines $\ell_1, \ldots, \ell_5$ used for characterizing the closest codewords of points in the lower triangles.

Figure 5. The geometric properties of our codes used during decoding.

of n coefficients modulo q, with the representatives centered around 0. Therefore, for better compatibility, we now use the $[-q/2, q/2]^2$ square instead of the $\mathbb{Z}_q^2$ square employed so far while discussing 2D codes.

We start by observing a symmetry, illustrated in Figure 5a, that can be exploited for decoding. We can see that, by construction, the codewords of code $\mathcal{C}_{(\alpha,\beta)}$ are symmetric over the identity line, which separates the $[-q/2, q/2]^2$ square in two triangles. Because of this property, if a point is closer to a codeword associated with $(1, 0)$ in the upper triangle, it will be closer to a codeword associated with $(0, 1)$ in the lower triangle and vice-versa – e.g., see point $\mathbf{x}$ in Figure 5a. However, we can also see that closeness to codewords associated with $(0, 0)$ and $(1, 1)$ is preserved by reflection around the identity line. Building upon this symmetry, any point in the upper triangle can be reflected, decoded in the lower triangle, and then reflected once again. Consequently, we only need to devise an efficient decoding mechanism for the lower triangle.

Suppose we are given a point in the lower triangle and want to find the closest codeword to this point. One simple way to accomplish that task would be to compute the distance to all six codewords whose Voronoi cells overlap the lower triangle, and then output the closest one. Although we considered this simple strategy, the resulting isochronous implementation was not very efficient due to the number of comparisons to the closest codeword.

For the sake of building our argument, assume for a moment that q is divisible by 2. In this setting, we can construct the Voronoi cells of each codeword relevant for decoding points in the lower triangle, as illustrated in Figure 5b. Let us now fix $\alpha = q/2$, as this choice typically provides the codes $\mathcal{C}_{(\alpha,\beta)}$ with the best properties. By the definition of $\mathcal{C}_{(q/2,\beta)}$, the points whose Voronoi cells intersect

the lower triangle, which are shown in Figure 5b, are defined as:

$$A = (q/2 + \beta, q/2 + \beta), \quad B = (0, 0), \quad C = (\beta - q/2, \beta - q/2),$$
$$D = (q/2, \beta), \quad E = (\beta, -q/2), \quad F = (q/2 + \beta, -q/2 + \beta).$$

The Voronoi cells intersecting the lower triangle can be defined by the perpendicular bisector lines, which we call ℓ_i, between the codewords and their neighbors. First we define ℓ_1, ℓ_4 and ℓ_5 as the bisectors between pairs $(A, D), (B, C)$, and (C, E), respectively. Now, since we assume q is even, the set of codewords $\{B, D, E, F\}$ forms a square, thus line ℓ_2 is the bisector of the pairs of points (B, E) and (D, F). Similarly, line ℓ_3 is simultaneously the bisector of both pairs (B, D) and (E, F). This means that, for an even q, we can characterize the Voronoi cells of these codewords using only 5 lines. To effectively use these lines to decode a point (x, y) in the lower triangle, we can verify whether (x, y) is above or below ℓ_i for each i. For example, if (x, y) is above lines ℓ_4 and ℓ_2, but below ℓ_3, then it should be decoded as $(0, 0)$.

Now that we have explained how to handle an even q, we have to deal with the real-world q, which is an odd prime. Since q is not divisible by 2, we must use $\alpha = \lfloor q/2 \rfloor$. This impacts the definition of some of the points. In particular, we now have $D = (\lfloor q/2 \rfloor, \beta)$, $E = (\beta, -\lfloor q/2 \rfloor - 1)$, and $F = (\lfloor q/2 \rfloor + \beta, -\lfloor q/2 \rfloor - 1 + \beta)$. Therefore, the set of points $\{B, D, E, F\}$ does not form a rectangle anymore. However, since $q = 3329$ is relatively large, we observe that $\{B, D, E, F\}$ can be relatively well approximated by a square. More specifically, if we define ℓ_3 as the bisector between points (B, D) and ℓ_2 as the bisector of points (B, E), then we can say that lines $\ell_1, \ldots, \ell_5$ give an approximate characterization of the Voronoi cells when q is prime.

We use this approach based on approximate Voronoi cells for decoding in all DFR results presented in this paper. In particular, after comparing this approach with a slower but trivial algorithm based on the exhaustive search for the closest codeword, we observed that the difference in DFR is negligible. Therefore, this approach can be safely used without any significant impact on the DFR.

Implementation of the decoding procedure. Algorithm 3 shows the full algorithm for decoding using these ideas. It builds upon macros `ABOVE_Li`, that return `0xffffffff` if point (x, y) is above ℓ_i and `0x0` otherwise. Notice that the equations that define lines ℓ_i have only integer coefficients because the representatives of all codewords themselves have integer coefficients. Furthermore, since all points are integers, the implementation of `ABOVE_Li` based on the lines' equations uses only 32-bit integer multiplications, which most implementations, including the ones of Kyber, assume to be isochronous. Furthermore, we note that a reflection mask is used to reflect (x, y) in case it is needed, and then to reflect the result in case codewords corresponding to $(0, 1)$ or $(1, 0)$ are found.

One interesting remark regarding this algorithm is that decoding could be made slightly more efficient if a different square of representatives was used. In particular, if we considered the $q \times q$ square whose bottom left point is $(\lfloor -q/2 \rfloor - \epsilon, \lfloor -q/2 \rfloor - \epsilon)$, then comparison with line ℓ_1 would not be necessary.

```c
static __inline__ int decode_msg_pair(int32_t x, int32_t y) {
  // mask_lower_than(x, y) is 0xffffffff if (x < y) and 0x0 otherwise
  uint32_t reflect_mask = mask_lower_than(x, y);
  int32_t x_prime = (x & ~reflect_mask) | (y & reflect_mask);
  int32_t y_prime = (y & ~reflect_mask) | (x & reflect_mask);

  uint8_t abovel1 = ABOVE_L1(x_prime, y_prime);
  uint8_t abovel2 = ABOVE_L2(x_prime, y_prime);
  uint8_t abovel3 = ABOVE_L3(x_prime, y_prime);
  uint8_t abovel4 = ABOVE_L4(x_prime, y_prime);
  uint8_t abovel5 = ABOVE_L5(x_prime, y_prime);
  // It is unnecessary to check for (00), but conceptually:
  // uint8_t c00 = (~abovel3 & abovel2 & abovel4);
  uint8_t c01 = ~abovel2 & ~abovel5 & ~abovel3;
  uint8_t c10 = abovel2 & abovel3 & ~abovel1;
  uint8_t c11 = abovel1 | (abovel3 & ~abovel2) | (abovel5 & ~abovel4);

  c01 &= (1 ^ reflect_mask);
  c10 &= (2 ^ reflect_mask);
  return (c01 | c10 | c11) & 3;
}

void poly_tomsg(uint8_t msg[KYBER_INDCPA_MSGBYTES], const poly *a) {
  size_t base = 0;
  for (size_t i = 0; i < KYBER_N/8; i++) {
    msg[i] = 0;
    for (size_t j = 0; j < 8; j += 2) {
      int x = a->coeffs[base];
      int y = a->coeffs[base + KYBER_N/2];
      base++;
      msg[i] |= (decode_msg_pair(x, y) << j);
    }
  }
}
```

Algorithm 3. Isochronous C implementation of proposed message decoding algorithm using 2-dimensional codes.

However, since this would complicate the explanation, we leave the analysis of possible extra optimizations for future work.

Performance impacts. To evaluate the impact of the proposed 2-dimensional codes over Kyber, we adapted its reference implementation[7] replacing the original encoding and decoding algorithms with the isochronous implementations hereby described. The code was compiled with `gcc` using flags `-march=native`, `-mtune=native` and `-O3`. Then, we tested its performance on a 64-bit Linux PC with an Intel Core i7-8700 CPU with a clock frequency of 3.20GHz.

Table 5 shows the difference in cycles for the affected operations. We can see that, unsurprisingly, decoding 2D codes is indeed more complex, taking 233 more cycles than observed for the original Kyber code. However, since the cycle count for the decapsulation is orders of magnitude larger than this difference, the overall impact of the more complex decoding on the total decapsulation is very small – specifically, between 0.08% and 0.18%. We can also see that, in some cases, the decapsulation is even faster for the 2D codes. However, this likely

[7] We considered commit `b628ba78711bc28327dc7d2d5c074a00f061884e` from https://github.com/pq-crystals/kyber/ (`main` branch).

Table 5. Comparison of the number of cycles for encoding, decoding, and full decapsulation when using the original Kyber code and the proposed 2D codes. These values are the medians of 10,000 runs.

Code	Encoding (`poly_frommsg`)	Decoding (`poly_tomsg`)	Decapsulation Level 1	Level 3	Level 5
Kyber's code	210[†]	171	126,903	203,902	292,359[‡]
2D code	66[†]	404	127,318	203,686	292,081[‡]

[†] Kyber's encoding performance can be slightly improved using partial unrolling of loops, which naturally occur for the encoding with 2D codes.
[‡] The improved performance for 2D codes may be the result of optimizations automatically done by the compiler and does not indicate that our proposal is faster.

stems from fortuitous optimizations by the compiler rather than an indication that our proposal is more efficient in terms of processing time.

6 Shorter ciphertexts for Kyber

This section shows how our proposed 2D codes improve the balance between ciphertext size and DFR compared with Kyber's original encoding strategy.

We start by noticing that the size, in bytes, of a ciphertext $(\mathbf{c_u}, c_v)$ in Kyber is completely determined by the parameters $n, k, d_\mathbf{u}$, and d_v. Namely, it is given by the following equation:

$$\|(\mathbf{c_u}, c_v)\| = \left\lceil \frac{1}{8} n(kd_\mathbf{u} + d_v) \right\rceil = 32(kd_\mathbf{u} + d_v) \text{ bytes.}$$

Hence, to obtain shorter ciphertexts, one has to adjust parameters $d_\mathbf{u}$ and d_v. There are, however, two caveats. The first one is that, if $d_\mathbf{u}$ or d_v gets too low, the increased noise factors $\Delta\mathbf{u}$ and Δv may result in an increased DFR, which is a security concern. Exploring this trade-off is the main focus of this section.

The second (and trickier) caveat is that, while it is easy to argue that decreasing $d_\mathbf{u}$ and d_v cannot make the MLWE scheme less secure (see Section 3.2), it could be the case that increasing one of them leads to lower security. Luckily for us, the security of Kyber depends very lightly on $d_\mathbf{u}$ and d_v. In particular, for all 3 parameter sets adopted by Kyber, picking even the maximal values of $d_\mathbf{u}$ and d_v does not affect the underlying MLWE security, as we explain next.

Security impacts of changing parameters $d_\mathbf{u}$ and d_v. The static security of Kyber is computed as the minimum between the complexity of solving the two MLWE instances responsible for protecting the secret key and the ciphertext. However, notice that the ciphertext compression parameters $(d_\mathbf{u}, d_v)$ only affect the ciphertext MLWE. Using Kyber's security estimation scripts, we observed the following. For Level 1, we can set $(d_\mathbf{u}, d_v)$ to any pair of values such that $d_\mathbf{u} \leqslant 10$ and the resulting LWE security will be the same as the original Kyber.

In contrast, we are free to choose any pair $(d_{\mathbf{u}}, d_v)$ for levels 3 and 5 without any security loss. In any case, there is no reason to use d_v or $d_{\mathbf{u}}$ larger than $\lceil \log_2 q \rceil$ since they are used to compress elements in $\mathbb{Z}_q$.

6.1 The effect of d_v in the DFR of 2D codes

We are now in place to discuss how the proposed codes can enable shorter ciphertexts in Kyber. For that purpose, we start by evaluating an important relation between d_v and the DFR advantage obtained using 2D codes. We then proceed to find a suitable balance between $d_{\mathbf{u}}$ and d_v that allows for reduced ciphertexts. Finally, we present a generalization of Kyber's compression procedure to build parameter sets that are strictly better than its original configurations, in particular for level 5, and discuss the proposal's crypto-agility.

When discussing the best code $\mathcal{C}_{(\alpha=1664,\beta)}$ for each parameter set in Section 5.2, we observed that the parameter minimizing the DFR was $\beta = 422$ for levels 1 and 3 (with $d_v = 4$), but $\beta = 436$ for level 5 (with $d_v = 5$). While these results suggest some correlation between the best β and the compression factor d_v, the exact details regarding this dependence are not necessarily obvious. To further evaluate this correlation, we ran the following experiment. For Kyber Level 5, we changed parameter d_v from 1 to 12, and computed the DFR for codes $\mathcal{C}_{(\alpha=1664,\beta)}$ using $\beta = 0$ to 500.

Figure 6 shows the result of this experiment. We can see that, when $d_v = 3$, the best code is the original Kyber code, where $\beta = 0$. However, when d_v increases, the best β also increases, although not linearly. More interestingly, though, we see that larger d_v values support 2D codes, providing progressively better DFR margin when compared with the original Kyber code. Furthermore, we notice that there are diminishing returns as we increase d_v, i.e., the larger the d_v, the lower the DFR reduction as we move to $d_v + 1$.

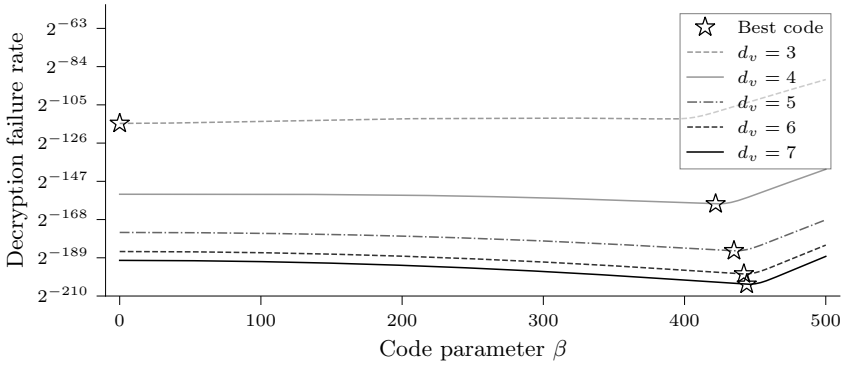


Figure 6. The effect of d_v in the DFR and best code, for Kyber Level 5.

Concerning ciphertext compression, the main conclusion from the above mentioned experiment is that lowering d_v alone is unlikely to yield ciphertext compression even when using 2D codes. Moreover, the 2D codes are better exploited when v is less compressed, which leads to a larger d_v. The main question is, thus, how to balance $d_{\mathbf{u}}$ and d_v to obtain ciphertext compression while maintaining the DFR values as low as those targeted by the original Kyber.

6.2 Balancing the compression factors to obtain shorter ciphertexts

We now consider how changing both $d_{\mathbf{u}}$ and d_v simultaneously impacts the DFR of Kyber's original code and our 2D codes. Since the main goal is to obtain ciphertext compression, we only explore pairs of parameters $(d_{\mathbf{u}}, d_v)$ that yield smaller ciphertexts than the original Kyber parameters.

Figure 7 compares Kyber's code and our proposed 2D codes under multiple security levels and different values of $(d_{\mathbf{u}}, d_v)$ providing ciphertext compression. Considering Level 1, it does not appear to be possible to get some advantage in ciphertext compression from the 2D codes without significantly compromising the DFR, which becomes much higher than 2^{-128} at points $(10, 3)$ and $(9, 4)$. Unfortunately, similar results were observed for Level 3. In particular, for both cases, the second lowest DFR parameters were $(10, 3)$, for which the best 2D code is exactly Kyber's code.

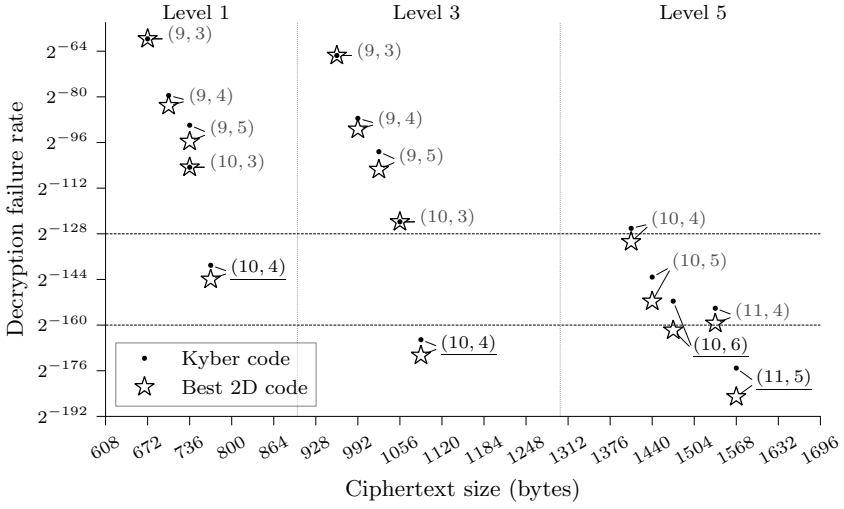


Figure 7. How changing parameters $(d_{\mathbf{u}}, d_v)$ affect the DFR, considering Kyber's original code and our 2D codes, under the security levels 1, 3, and 5.

In contrast, we can see that 2D codes offer a significant advantage in Level 5, creating ciphertext compression opportunities. Unlike any other result with Kyber's original code, the parameter pair $(10, 6)$ used with the 2D codes yields a DFR below the value of 2^{-160}, which is targeted by Kyber levels 3 and 5. These parameters compress the ciphertext from 1568 to only 1472 bytes, which corresponds to a compression factor of about 6.1%.

6.3 Strictly better parameters from generalized compression

During Kyber's encryption, the coefficients of each polynomial in $\mathbf{u}$ are compressed to $d_{\mathbf{u}}$ bits. We propose a simple generalization of this compression strategy, by allowing each polynomial in $\mathbf{u}$ to be compressed to a possibly different number of bits. Formally, we let $\mathbf{d_u} \in \mathbb{Z}^k$ define the compression factors for each of the k polynomials in $\mathbf{u}$. Under this setup, the error caused by compression and decompression of $\mathbf{u}$ have to be defined for each block $i = 0$ to $k - 1$ as

$$\Delta\mathbf{u}[i] = \text{Decompress}(\text{Compress}(\mathbf{u}[i], \mathbf{d_u}[i]), \mathbf{d_u}[i]) - \mathbf{u}[i].$$

The use of multiple factors for compressing $\mathbf{u}$ has only a minor impact on the computation of the joint distribution. This can be shown by evaluating how using multiple factors for compressing $\mathbf{u}$ affects the computation of Δm. We start by writing

$$\begin{aligned}\Delta m &= \langle \mathbf{e}, \mathbf{r} \rangle - \langle \mathbf{s}, \mathbf{e}_1 + \Delta\mathbf{u} \rangle + e_2 + \Delta v \\ &= \langle \mathbf{e}, \mathbf{r} \rangle - \sum_{i=1}^{k} \mathbf{s}[i](\mathbf{e}_1[i] + \Delta\mathbf{u}[i]) + e_2 + \Delta v.\end{aligned}$$

Remember that $\langle \mathbf{s}, \mathbf{e}_1 + \Delta\mathbf{u} \rangle$ is a sum of independent products, which have the same distribution equally distributed. Therefore, we can compute the joint distribution $\Pr(\Delta m[i], \Delta m[i + n/2])$ for this generalization at the cost of computing $(k - 1)$ extra 2-dimensional FFTs.

To understand if this generalization gives us an advantage, we ran computations similar to the ones in the previous section, but now with multiple values of $(\mathbf{d_u}, d_v) \in \mathbb{Z}^k \times \mathbb{Z}$ instead of $(d_{\mathbf{u}}, d_v)$. The results are illustrated in Figures 8 and 9, which show the results for levels 3 and 5, respectively. For conciseness, we omit results for level 1, as they are similar to the ones for level 3.

With respect to level 3, Figure 8 shows that the proposed generalization leads to at least one point below the first DFR target of 2^{-128}, represented by parameters $(\mathbf{d_u}, d_v) = ([10, 10, 9], 4)$. Interestingly, only the point corresponding to our 2D codes crossed the threshold. However, although this point compresses the level 3 ciphertext from 1088 to 1056 bytes, the DFR was not below the target of 2^{-160} defined by Kyber for levels 3 and 5.

On the other hand, when we consider level 5, Figure 9 presents several different points below the 2^{-160} DFR target. Notice that the one that achieves the highest compression of 6% is equivalent to $(d_{\mathbf{u}}, d_v) = (10, 6)$. Interestingly, though, there are now points that, because of the usage of 2D codes, are strictly

better than the original Kyber both concerning ciphertext size and smaller DFR. For example, point $(\mathbf{d_u}, d_v) = ([11, 11, 10, 10], 6)$ provides 2% ciphertext compression with DFR of only $2^{-177.9}$, less than the $2^{-175.2}$ achieved by the original Kyber. Furthermore, point $(\mathbf{d_u}, d_v) = ([11, 11, 11, 10], 6)$ achieves DFR of

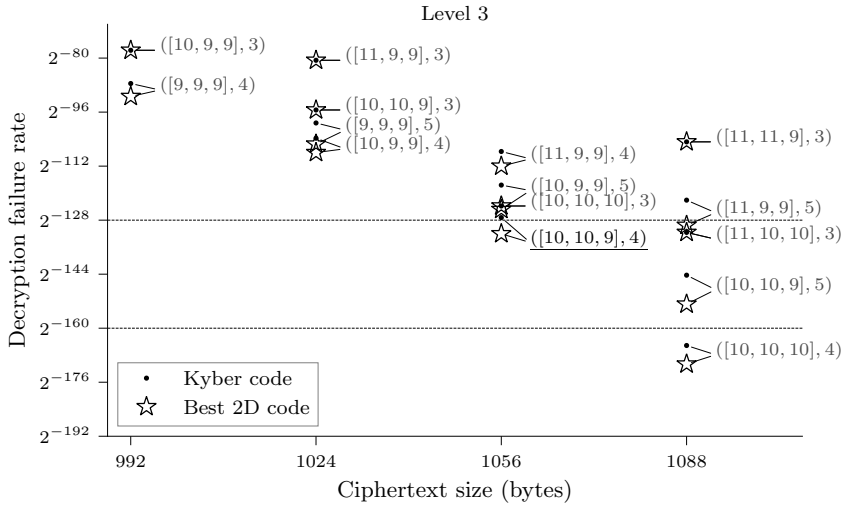


Figure 8. The DFR for generalized compression parameters $(\mathbf{d_u}, d_v) \in \mathbb{Z}^k \times \mathbb{Z}$ considering Kyber security parameters achieving level 3.

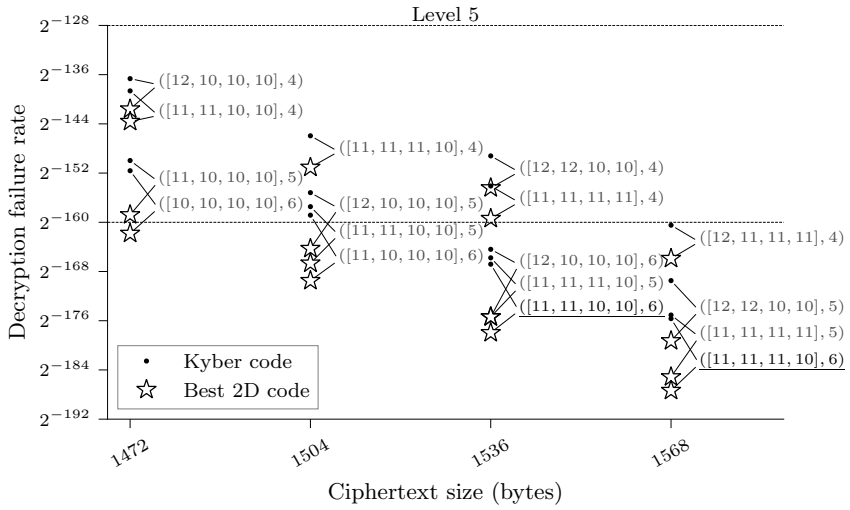


Figure 9. The DFR for generalized compression parameters $(\mathbf{d_u}, d_v) \in \mathbb{Z}^k \times \mathbb{Z}$ considering Kyber security parameters achieving level 5.

$2^{-187.3}$, which is even lower than the DFR of $2^{-185.1}$ previously observed for our 2D codes considering parameters $(d_{\mathbf{u}}, d_v) = (11, 5)$ obtained in Section 5.

7 Proposed parameters and crypto-agility considerations

This section summarizes the most important practical results obtained in this work, and discusses how their adoption may impact crypto-agility.

While the proposed 2D codes can be directly integrated into Kyber by simply replacing functions `poly_frommsg` and `poly_tomsg`, the generalized compression method requires changing Kyber's parameters and compression-related algorithms. Notice, however, that compression and decompression are rather simple algorithms. Furthermore, they are not part of Kyber's main performance bottlenecks, which are hash computations and polynomial multiplications. Since most hardware acceleration and device-dependent optimizations in the literature target those bottleneck operations, we argue that the impact of the generalized compression algorithms should be minor in practice.

All in all, aiming to make the proposed parameters more easily comparable, we can classify the crypto-agility impact of our proposals in the 3 following levels.

1. Drop-in replacement of encoding and decoding functions.
2. Same as 1, and minor change of compression parameters $(d_{\mathbf{u}}, d_v)$.
3. Same as 2, and generalization of the compression algorithms.

Table 6. Proposed parameters for Kyber using 2D codes $\mathcal{C}_{(\alpha=1664,\beta)}$. All settings lead to negligible performance impact. Remember that the DFR values obtained by Kyber are $2^{-139.1}, 2^{-165.2}$, and $2^{-175.2}$ for levels 1, 3, and 5, respectively.

Crypto-agility impact	Security	Compression parameters	β	DFR	Ciphertext compression	Advantages
1	Level 1	$(d_{\mathbf{u}}, d_v) = (10, 4)$	422	$2^{-143.9}$	0%	Lower DFR than $2^{-139.1}$
1	Level 3	$(d_{\mathbf{u}}, d_v) = (10, 4)$	422	$2^{-170.6}$	0%	Lower DFR than $2^{-165.2}$
1	Level 5	$(d_{\mathbf{u}}, d_v) = (11, 5)$	435	$2^{-185.1}$	0%	Lower DFR than $2^{-175.2}$
2	Level 5	$(d_{\mathbf{u}}, d_v) = (10, 6)$	442	$2^{-161.8}$	6%	Smaller ciphertexts and DFR below 2^{-160} target[†]
3	Level 5	$(\mathbf{d_u}, d_v) = \left(\begin{bmatrix}11\\11\\10\\10\end{bmatrix}, 6\right)$	442	$2^{-177.9}$	2%	Smaller ciphertexts and lower DFR than $2^{-175.2}$

[†] Kyber's specification [4, 5] define the DFR target for levels 3 and 5 as 2^{-160} (see Section 3.2).

Table 6 shows our proposed parameters using 2D codes, their impact on crypto-agility, and a brief description of their advantages compared with their original Kyber counterparts. In summary, our 2D codes effectively improve Kyber DFR for all security levels when all parameters remain unchanged. Second, we observed that level 5 is the one most positively affected by the usage of 2D codes. In particular, we can obtain a 6% ciphertext compression for this security level, keeping the DFR below the value targeted by Kyber's designers. Furthermore, the generalized compression algorithms, together with the 2D codes, provide strictly better parameters concerning both DFR and ciphertext size.

8 Conclusion and future work

In this work, we present a novel framework for obtaining better encoding mechanisms for lattice-based schemes, in particular Kyber. Our construction relies on simpler assumptions than related works and provides concrete benefits, both with respect to lowering Kyber's decryption failure rate (DFR) and ciphertext sizes. Our proposed configurations are practical and only require changes in parameters related to ciphertext compression. Also, we provide an efficient isochronous implementation of the encoding and decoding procedures that causes only a minor impact on the full decapsulation execution time.

This work also raises several questions, both in theory and practice. We believe the most important theoretical question is what kind of improvements concerning DFR and ciphertext compression can be obtained using codes with dimensions higher than 2. However, the main challenge of exploring this venue is the increased computational complexity of computing the joint higher-dimensional noise distribution and searching for the best code. Even if these problems can be solved, it would be important to understand whether we can design isochronous decoders for higher-dimensional codes without incurring significant performance overhead.

Although we focus on Kyber, the proposed construction can, at least in theory, be applied to any lattice-based scheme whose dimension n is a power of two. It would be intriguing, for example, to see if our framework can improve prominent lattice-based schemes such as Saber [9] and NewHope [3]. Conversely, while in principle the some of our ideas could be applied to code-based solutions (e.g., HQC [27]), such schemes typically require a prime n to avoid structural attacks [16, 24], which prevents us from using the FFT-based convolutions to compute the joint noise distribution.

Finally, the isochronous decoding algorithm presented in this work was efficient enough to show that our construction is practical. However, we only considered Kyber's reference implementation, which does not leverage either vectorized instructions (e.g., AVX2) or Assembly-optimized code. It would, thus, be interesting to understand the performance impact of our 2-dimensional codes with optimized implementations in different platforms (e.g., integrating the proposal into a fork of *pqm4* [22]).

References

1. Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D.: Status report on the first round of the NIST post-quantum cryptography standardization process. US Department of Commerce, National Institute of Standards and Technology (2019). https://doi.org/10.6028/NIST.IR.8240 Cited on 8.
2. Alkim, E., Avanzi, R., Bos, J., Ducas, L., de la Piedra, A., Pöppelmann, T., Schwabe, P., Stebila, D., Albrecht, M.R., Orsini, E., Osheter, V., Paterson, K.G., Peer, G., Smart, N.P.: NewHope: Algorithm specifications and supporting documentation (2020), https://newhopecrypto.org/data/NewHope_2020_04_10.pdf Cited on 8.
3. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange – a new hope. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 327–343 (2016) Cited on 8, 12, 14, and 29.
4. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber: Algorithm specifications and supporting documentation (version 2.0) (2019), https://pq-crystals.org/kyber/data/kyber-specification-round2.pdf. Cited on 7 and 28.
5. Avanzi, R., Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber: Algorithm specifications and supporting documentation (version 3.02) (2021), https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf. Cited on 2, 6, 7, and 28.
6. Baan, H., Bhattacharya, S., Fluhrer, S., Garcia-Morchon, O., Laarhoven, T., Rietman, R., Saarinen, M.J.O., Tolhuizen, L., Zhang, Z.: Round5: Compact and fast post-quantum public-key encryption. In: Post-Quantum Cryptography: 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers 10. pp. 83–102. Springer (2019). https://doi.org/10.1007/978-3-030-25510-7_5 Cited on 8.
7. D'Anvers, J.P., Vercauteren, F., Verbauwhede, I.: The impact of error dependencies on Ring/Mod-LWE/LWR based schemes. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 103–115. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-25510-7_6 Cited on 2, 8, and 10.
8. D'Anvers, J.P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I.: Decryption failure attacks on IND-CCA secure lattice-based schemes. In: Public-Key Cryptography–PKC 2019: 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II 22. pp. 565–598. Springer (2019). https://doi.org/10.1007/978-3-030-17259-6_19 Cited on 2 and 7.
9. D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: International Conference on Cryptology in Africa. pp. 282–305. Springer (2018). https://doi.org/10.1007/978-3-319-89339-6_16 Cited on 29.
10. Enge, A., Gastineau, M., Théveny, P., Zimmermann, P.: MPC – A library for multiprecision complex arithmetic with exact rounding (Dec 2022), http://www.multiprecision.org/mpc/ Cited on 13.
11. Erez, U., Zamir, R.: Achieving 1/2 log (1 + SNR) on the AWGN channel with lattice encoding and decoding. IEEE Transactions on Information Theory **50**(10), 2293–2314 (2004). https://doi.org/10.1109/TIT.2004.834787 Cited on 2 and 9.

12. Fabšič, T., Hromada, V., Stankovski, P., Zajac, P., Guo, Q., Johansson, T.: A reaction attack on the QC-LDPC McEliece cryptosystem. In: Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8. pp. 51–68. Springer (2017). https://doi.org/10.1007/978-3-319-59879-6_4 Cited on 2.
13. Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., Zimmermann, P.: MPFR: A multiple-precision binary floating-point library with correct rounding. ACM Transactions on Mathematical Software (TOMS) **33**(2), 13–es (2007). https://doi.org/10.1145/1236463.1236468 Cited on 13.
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual International Cryptology Conference. pp. 537–554. Springer (1999) Cited on 5.
15. Gentleman, W.M., Sande, G.: Fast Fourier transforms: for fun and profit. In: Proceedings of the November 7-10, 1966, AFIPS Fall Joint Computer Conference. pp. 563–578 (1966). https://doi.org/10.1145/1464291.1464352 Cited on 13.
16. Guo, Q., Johansson, T., Löndahl, C.: A new algorithm for solving Ring-LPN with a reducible polynomial. IEEE Transactions on Information Theory **61**(11), 6204–6212 (2015). https://doi.org/10.1109/TIT.2015.2475738 Cited on 29.
17. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on MDPC with CCA security using decoding errors. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. pp. 789–815. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_29 Cited on 2.
18. Guo, Q., Johansson, T., Yang, J.: A novel CCA attack using decryption errors against LAC. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 82–111. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_4 Cited on 2 and 7.
19. Hall, C., Goldberg, I., Schneier, B.: Reaction attacks against several public-key cryptosystem. In: Varadharajan, V., Mu, Y. (eds.) Information and Communication Security. pp. 2–12. Springer Berlin Heidelberg, Berlin, Heidelberg (1999). https://doi.org/10.1007/978-3-540-47942-0_2 Cited on 2.
20. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography. pp. 341–371. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_12 Cited on 5.
21. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ECDSA). International journal of information security **1**(1), 36–63 (2001). https://doi.org/10.1007/s102070100002 Cited on 2.
22. Kannwischer, M.J., Petri, R., Rijneveld, J., Schwabe, P., Stoffelen, K.: PQM4: Post-quantum crypto library for the ARM Cortex-M4, https://github.com/mupq/pqm4 Cited on 29.
23. Liu, S., Sakzad, A.: Lattice codes for CRYSTALS-Kyber (Sep 2023), http://arxiv.org/abs/2308.13981 Cited on 2, 3, 8, and 9.
24. Löndahl, C., Johansson, T., Koochak Shooshtari, M., Ahmadian-Attari, M., Aref, M.R.: Squaring attacks on McEliece public-key cryptosystems using quasi-cyclic codes of even dimension. Designs, Codes and Cryptography **80**, 359–377 (2016). https://doi.org/10.1007/s10623-015-0099-x Cited on 29.
25. Lu, X., Liu, Y., Zhang, Z., Jia, D., Xue, H., He, J., Li, B., Wang, K.: LAC: Practical Ring-LWE based public-key encryption with byte-level modulus. Cryptology ePrint Archive, Paper 2018/1009 (2018), https://eprint.iacr.org/2018/1009 Cited on 8.

26. Lyu, S., Liu, L., Ling, C., Lai, J., Chen, H.: Lattice codes for lattice-based PKE. Designs, Codes and Cryptography pp. 1–23 (2023). https://doi.org/10.1007/s10623-023-01321-6 Cited on 2 and 9.
27. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Bos, J., Deneuville, J.C., Dion, A., Gaborit, P., Lacan, J., Persichetti, E., Robert, J.M., Véron, P., Zémor, G.: Hamming Quasi-Cyclic: HQC (2021), https://pqc-hqc.org/doc/hqc-specification_2021-06-06.pdf Cited on 29.
28. National Institute of Standards and Technology: FIPS203: Module-lattice-based key-encapsulation mechanism standard (initial public draft). Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology (2023-08-24 2023), https://doi.org/10.6028/NIST.FIPS.203.ipd Cited on 2.
29. NIST: Post-quantum crypto project. http://csrc.nist.gov/groups/ST/post-quantum-crypto/ (2016) Cited on 2.
30. NIST: Quantum-resistant cryptography technology interoperability and performance report (preliminary draft). Tech. rep., National Institute of Standards and Technology, Department of Commerce, Washington, D.C. (2023), Special Publication (SP 1800-38C) Cited on 2.
31. Plantard, T., Sipasseuth, A., Susilo, W., Zucca, V.: Tight bound on NewHope failure probability. IEEE Transactions on Emerging Topics in Computing **10**(4), 1955–1965 (2022). https://doi.org/10.1109/TETC.2021.3138951 Cited on 8 and 14.
32. Rabiner, L.: On the use of symmetry in FFT computation. IEEE Transactions on Acoustics, Speech, and Signal Processing **27**(3), 233–239 (Jun 1979). https://doi.org/10.1109/TASSP.1979.1163235 Cited on 13 and 14.
33. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009). https://doi.org/10.1145/1568318.1568324 Cited on 8.
34. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978). https://doi.org/10.1145/359340.359342 Cited on 2.
35. Saarinen, M.J.O.: Hila5: On reliability, reconciliation, and error correction for ring-lwe encryption. In: Selected Areas in Cryptography–SAC 2017: 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers 24. pp. 192–212. Springer (2018) Cited on 8.
36. Saliba, C.: Error correction and reconciliation techniques for lattice-based key generation protocols. Ph.D. thesis, CY Cergy Paris Université (2022), https://theses.hal.science/tel-03718212v1/document Cited on 2, 3, 8, 9, 14, and 15.
37. Saliba, C., Luzzi, L., Ling, C.: A reconciliation approach to key generation based on Module-LWE. In: 2021 IEEE International Symposium on Information Theory (ISIT). pp. 1636–1641 (2021). https://doi.org/10.1109/ISIT45174.2021.9517882 Cited on 2, 3, 8, 9, 14, and 15.
38. Schatzman, J.C.: Accuracy of the discrete Fourier transform and the fast Fourier transform. SIAM Journal on Scientific Computing **17**(5), 1150–1166 (1996). https://doi.org//10.1137/S106482759324702 Cited on 13.
39. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing **26**(5), 1484–1509 (1997). https://doi.org/10.1137/S0097539795293172 Cited on 2.