- `KAT/kyber768-90s/PQCkemKAT_2400.rsp`,
- `KAT/kyber1024/PQCkemKAT_3168.req`,
- `KAT/kyber1024/PQCkemKAT_3168.rsp`,
- `KAT/kyber1024-90s/PQCkemKAT_3168.req`, and
- `KAT/kyber1024-90s/PQCkemKAT_3168.rsp`,

4 Expected security strength

4.1 Security definition

KYBER.CCAKEM (or short, KYBER) is an IND-CCA2-secure key encapsulation mechanism, i.e., it fulfills the security definition stated in Section 4.A.2 of the Call for Proposals.

4.2 Rationale of our security estimates

Our estimates of the security strength for the three different parameter sets of KYBER—and consequently the classification into security levels as defined in Section 4.A.5 of the Call for Proposals—are based on the cost estimates of attacks against the underlying module-learning-with-errors (MLWE) problem as detailed in Subsection 5.1.

To justify this rationale, we will in the following give two reductions from MLWE: a tight reduction in the random-oracle model (ROM) in Theorem 2 and a non-tight reduction in the quantum-random-oracle model (QROM) in Theorem 3. With those reductions at hand, there remain two avenues of attack that would break KYBER without solving the underlying MLWE problem, namely

1. breaking one of the assumptions of the reductions, in particular attacking the symmetric primitives used in KYBER; or
2. exploiting the non-tightness of the QROM reduction.

We briefly discuss 1.) in Subsection 5.4. The discussion of 2.) requires considering two separate issues, namely

- a (quadratic) non-tightness in the decryption-failure probability of KYBER.CPAPKE, and
- a (quadratic) non-tightness between the advantage of the MLWE attacker and the quantum attacker against KYBER.

In Subsection 5.5 we discuss quantum attacks exploiting decryption failures and in the presentation of the non-tight QROM reduction we explain why the non-tightness between quantum attacks against MLWE and quantum attacks against KYBER is unlikely to matter in practice. More specifically, we show how to eliminate this non-tightness if we allow the reasonable, but non-standard, assumption that KYBER.CPAPKE ciphertexts are pseudorandom, even if all randomness is generated pseudorandomly from a hash of the encrypted message.

4.3 Security Assumption

The hard problem underlying the security of our schemes is Module-LWE [27, 66]. It consists in distinguishing uniform samples $(\mathbf{a}_i, b_i) \leftarrow R_q^k \times R_q$ from samples $(\mathbf{a}_i, b_i) \in R_q^k \times R_q$ where $\mathbf{a}_i \leftarrow R_q^k$ is uniform and $b_i = \mathbf{a}_i^T\mathbf{s} + e_i$ with $\mathbf{s} \leftarrow B_\eta^k$ common to all samples and $e_i \leftarrow B_\eta$ fresh for every sample. More precisely, for an algorithm A, we define $\mathbf{Adv}^{\mathrm{mlwe}}_{m,k,\eta}(\mathsf{A}) =$

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

4.3.1 Tight reduction from MLWE in the ROM

We first note that KYBER.CPAPKE is tightly IND-CPA secure under the Module-LWE hardness assumption.

Theorem 1. *Suppose* XOF *and G are random oracles. For any adversary* A*, there exist adversaries* B *and* C *with roughly the same running time as that of* A *such that* $\mathbf{Adv}^{\mathrm{cpa}}_{\text{KYBER.CPAPKE}}(\mathsf{A}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathsf{B}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathsf{C})$.

The proof of this theorem is easily obtained by noting that, under the MLWE assumption, public-key and ciphertext are pseudo-random.

KYBER.CCAKEM is obtained via a slightly tweaked Fujisaki-Okamoto transform [54, 46] applied to KYBER.CPAPKE. The following concrete security statement proves KYBER.CCAKEM's IND-CCA2-security when the hash functions G and H are modeled as random oracles. It is obtained by combining the generic bounds from [54] with Theorem 1 (and optimizing the constants appearing in the bound).

Theorem 2. *Suppose* XOF*, H, and G are random oracles. For any classical adversary* A *that makes at most* q_{RO} *many queries to random oracles* XOF*, H and G, there exist adversaries* B *and* C *of roughly the same running time as that of* A *such that*

$$\mathbf{Adv}^{\mathrm{cca}}_{\text{KYBER.CCAKEM}}(\mathsf{A}) \leq 2\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathsf{B}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathsf{C}) + 4q_{RO}\delta.$$

Note that the security bound is tight. The negligible additive term $4q_{RO}\delta$ stems from KYBER.CPAPKE's decryption-failure probability δ.

4.3.2 Non-tight reduction from MLWE in the QROM

As for security in the quantum random oracle model (QROM), [54, 96] proved that KYBER.CCAKEM is IND-CCA2 secure in the QROM, provided that KYBER.CPAPKE is IND-CPA secure. A slightly tighter reduction can be obtained by requiring the base scheme KYBER.CPAPKE to be pseudo-random. Pseudo-randomness [96] requires that, for every message m, a (randomly generated) ciphertext $(c_1, c_2) \leftarrow \text{KYBER.CPAPKE.Enc}(pk, m)$ is computationally indistinguishable from a random ciphertext of the form $(\mathsf{Compress}_q(\mathbf{u}, d_u), \mathsf{Compress}_q(v, d_v))$, for uniform $(\mathbf{u}, v)$. (We also require the property of "statistical disjointness" [96] which is trivially fulfilled for KYBER.CPAPKE.) The proof of KYBER.CPAPKE's IND-CPA security indeed shows that KYBER.CPAPKE is tightly pseudo-random under the Module-LWE hardness assumption.

Theorem 3. *Suppose* XOF*, H, and G are random oracles. For any quantum adversary* A *that makes at most* q_{RO} *many queries to quantum random oracles* XOF*, H and G, there exist quantum adversaries* B *and* C *of roughly the same running time as that of* A *such that*

$$\mathbf{Adv}^{\mathrm{cca}}_{\text{KYBER.CCAKEM}}(\mathsf{A}) \leq 4q_{RO} \cdot \sqrt{\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathsf{B})} + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathsf{C}) + 8q^2_{RO}\delta.$$

Unfortunately, the above security bound is non-tight and therefore can only serve as an asymptotic indication of KYBER.CCAKEM's CCA-security in the quantum random oracle model.

Tight reduction under non-standard assumption. We can use [54, 96] to derive a tight security bound in the QROM from a non-standard security assumption, namely that a deterministic version of KYBER.CPAPKE, called DKYBER.CPAPKE, is pseudo-random in the QROM. Deterministic KYBER.CPAPKE is defined as KYBER.CPAPKE, but the random coins r used in encryption are derived deterministically from the message m as $r := \mathrm{G}(m)$. Pseudo-randomness for deterministic encryption states that an encryption (c_1, c_2) of a randomly chosen message is computationally indistinguishable from a random ciphertext $(\mathsf{Compress}_q(\mathbf{u}, d_u), \mathsf{Compress}_q(v, d_v))$, for uniform $(\mathbf{u}, v)$. In the classical ROM, pseudo-randomness of DKYBER.CPAPKE is tightly equivalent to MLWE but in the QROM the reduction is non-tight (and is the reason for the term $q_{RO} \cdot \sqrt{\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathsf{B})}$ in Theorem 3). Concretely, we obtain the following bound:

$$\mathbf{Adv}^{\mathrm{cca}}_{\text{KYBER.CCAKEM}}(\mathsf{A}) \leq 2\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathsf{B}) + \mathbf{Adv}^{\mathrm{pr}}_{\text{DKYBER.CPAPKE}}(\mathsf{C}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathsf{D}) + 8q^2_{RO}\delta.$$

We remark that we are not aware of any quantum attack on deterministic KYBER.CPAPKE that performs better than breaking the MLWE problem.

Table 4: Classical and quantum hardness of the different proposed parameter sets of KYBER together with the claimed security level as defined in Section 4.A.5 of the Call for Proposals.

	KYBER512	KYBER768	KYBER1024
NIST Security level	1	3	5

Core-SVP methodology, Primal attack only (Sec. 5.1.4)

lattice attack dim. d	1003	1424	1885
BKZ-blocksize β	403	625	877
core-SVP classical hardness	118	182	256
core-SVP quantum hardness	107	165	232

Refined estimate for classical attacks using [40, 36, 5] (Sec. 5.2)

lattice attack dim. d	1025	1467	1918
BKZ-blocksize β	413	637	894
sieving dimension $\beta' = \beta - d_{4f}$	375	586	829
$\log_2$(gates)	151.5	215.1	287.3
$\log_2$(memory in bits)	93.8	138.5	189.7

4.4 Estimated security strength

Table 4 lists the security levels according to the definition in Section 4.A.5 of the Call for Proposals for the different parameter sets of KYBER. Our claims are based on the cost estimates of the best known attacks against the MLWE problem underlying KYBER as detailed in Subsection 5.1. Specifically we list the classical and the quantum *core-SVP hardness* and use those to derive security levels.

All parameter sets of KYBER have some probability of decryption failure. These failures are a security concern (see Section 5.5) and so the probabilities with which they occur need to be small. But because in the classical random oracle model, the decryption failure probability is information-theoretic, we do not see a need for it to decrease with the security parameter. In particular, decryption failure for our level 3 and 5 parameter sets is less than 2^{-160}, which means that if 2^{80} instances of Kyber were run every second from now until our sun becomes a white dwarf, the odds still heavily favor there never being a decryption failure. We therefore exclude these attacks from our claims regarding the NIST security estimates.

The impact of the deterministic noise caused by $\mathsf{Compress}_q$ on KYBER512. Each coefficient of $\mathbf{e}_1$ (and $\mathbf{e}_2$) in Algorithm 5 is distributed as a binomial distribution with parameter $\eta_2 = 2$, which has variance $\eta_2/2 = 1$. The parameter $d_u = 10$ implies that the $\mathsf{Compress}_q$ function maps elements modulo q to a set of size 2^{10} where the difference between every two elements in the latter set is either 3 or 4. This implies that the error added by the $\mathsf{Compress}_q$ function for each coefficient is either uniform over $\{-1, 0, 1\}, \{-1, 0, 1, 2\}$, or $\{-2, -1, 0, 1\}$. It therefore has variance at least as large as the uniform distribution over the set $\{-1, 0, 1\}$, which is $2/3$. This makes the total variance of each coefficient of $\mathbf{e}_1$ plus the deterministic error at least $1 + 2/3 = 5/3$. The other secret and noise terms have binomial distributions with parameter $\eta_1 = 3$ for a variance of $\eta_1/2 = 3/2 < 5/3$. When accounting for the errors added by $\mathsf{Compress}_q$, we therefore calculate the hardness of KYBER512 assuming that the variance of each secret/error coefficient is $3/2$.

The impact of MAXDEPTH. The best known quantum speedups for the sieving algorithm, which we consider in our cost analysis (see Subsection 5.1.1), are only mildly affected by limiting the depth of a quantum circuit, because it uses Grover search on sets of small size (compared to searching through the whole keyspace of AES). For the core-SVP-hardness operation estimates to match the quantum gate cost of breaking AES at the respective security levels, a quantum computer would need to support a maximum depth of 70–80. When limiting the maximum depth to smaller values, or when considering classical attacks, the core-SVP-hardness estimates are smaller than the gate counts for attacks against AES. Recent study [5] on the concrete cost of

sieving suggest that the quantum speed-ups of these algorithms are tenuous, independently of the value of MAXDEPTH.

4.5 Additional security properties

4.5.1 Forward secrecy.

Kyber has a very efficient key-generation procedure (see also Section 2) and is therefore particularly well suited for applications that use frequent key generations to achieve forward secrecy.

4.5.2 Side-channel attacks.

Timing attacks. Neither straight-forward reference implementations nor optimized implementations of Kyber use any secret-dependent branches or table lookups[3].This means that typical implementations of Kyber are free from the two most notorious sources of timing leakage. Another possible source of timing leakage are non-constant-time multipliers like the `UMULL` instruction on ARM Cortex-M3 processors, which multiplies two 32-bit integers to obtain a 64-bit result. However, multiplications in Kyber have only 16-bit inputs, and most non-constant-time multipliers show timing variation only for larger inputs. For example, on ARM Cortex-M3 processors the obvious way to implement multiplications in Kyber is through the constant-time `MUL` instruction, which multiplies two 32-bit integers, but returns only the bottom 32-bits of the result. What remains as a source of timing leakage are modular reductions, which are sometimes implemented via conditional statements. However, timing leakage in modular reductions is easily avoided by using (faster) Montgomery [78] and Barrett reductions [16] as illustrated in our reference and AVX2 implementations.

We note that the 90s variant is only really attractive to use if AES hardware support is available. If hardware support is not available, then table-based implementations of AES are notorious for leaking secrets trough cache timing. In our C reference implementation of the 90s variant we use a constant-time bitsliced implementation of AES, which is based on code from BearSSL [90].

Differential attacks. We expect that any implementation of Kyber without dedicated protection against differential power or electromagnetic radiation (EM) attacks will be vulnerable to such attacks. This is true for essentially any implementation of a cryptographic scheme that uses long-term (non-ephemeral) keys. Deployment scenarios of Kyber in which an attacker is assumed to have the power to mount such an attack require specially protected—typically masked—implementations. In [82], Oder, Schneider, Pöppelmann, and Güneysu present such a masked implementation of Ring-LWE decryption with a CCA transform very similar to the one used in Kyber. The implementation targets Cortex-M4F microcontrollers; the conclusion of the work is that protecting the decryption (decapsulation) step against first-order DPA incurs an overhead of about about a factor of 5.5. The techniques presented in that paper also apply to Kyber and we expect that the overhead for protecting Kyber against differential attacks is in the same ballpark.

Template attacks. Protections against differential attacks do not help if an attacker is able to recover even ephemeral secrets from a single power or EM trace. At CHES 2017, Primas, Pessl, and Mangard presented such a single-trace attack against an implementation of Ring-LWE on a Cortex-M4F microcontroller [91]. The attacker model in this attack is rather strong: it is the typical setting of template attacks, which assumes an attacker who is able to generate template traces on known inputs on a device with leakage very similar to the actual target device. In [91], the authors used *the same device* for generating target traces and in the attack. The attack was facilitated (maybe even enabled) by the fact that the implementation under attack used variable-time modular reductions. Consequently, the paper states that *"One of the first measures to strengthen an implementation against SPA attacks is to ensure a constant runtime and control flow"*. This is the case for all implementations of Kyber. The attack from [91] would thus certainly not straight-forwardly apply to implementations of Kyber, but more research is required to investigate whether also constant-time implementations of Kyber (and other lattice-based schemes) succumb to template attacks, and what the cost of suitable countermeasures is.

[3]Note that the rejection sampling in generating the matrix $\mathbf{A}$ does not involve any secret data.

4.5.3 Multi-target attacks

Our security analysis makes no formal claims about security bounds in the multi-target setting. However, in the design of Kyber we made two decisions that aim at improving security against attackers targeting multiple users:

- We adopt the "against-all-authority" approach of re-generating the matrix $\mathbf{A}$ for each public key from NewHope [10]. This protects against an attacker attempting to break *many keys at the cost of breaking one key.*
- In the CCA transform (see Alg. 8) we hash the public key into the pre-key $\bar{K}$ and the coins r. Making the coins r dependent of the public key protects against precomputation attacks that attempt to break *one out of many keys.* For details, see Subsection 5.5.

4.5.4 Misuse resilience

The first, and most important, line of defense against misuse is the decision to make IND-CCA2 security non-optional. As discussed in Subsection 1.5, it would have been possible to achieve slightly shorter public keys and ciphertexts, and faster decapsulation, in a CPA-secure variant of Kyber. Using IND-CCA2 security by default makes it safe to use Kyber with static keys and as a consequence also to re-use ephemeral keys for some time. What is *not* safe, is to reuse the same randomness in encapsulation, but that randomness is also not exposed to the outside by the API. The CCA transform has a second effect in terms of robustness: it protects against a broken implementation of the noise sampling. A rather peculiar aspect of LWE-based cryptography is that it will pass typical functional tests even if one communication partner does not add any noise (or by accident samples all-zero noise). The deterministic generation of noise via SHAKE-256 during encapsulation and the re-encryption step during decapsulation will reveal such an implementation mistake immediately.

An additional line of defense against misuse is to hash the public-key into the "pre-key" $\bar{K}$ and thus make sure that the KEM is contributory. Only few protocols require a KEM to be contributory and those protocols can always turn a non-contributory KEM into a contributory one by hashing the public key into the final key. Making this hash part of the KEM design in Kyber ensures that nothing will go wrong on the protocol level if implementers omit the hash there.

A similar statement holds for additionally hashing the ciphertext into the final key. Several protocols need to ensure that the key depends on the complete view of exchanged protocol messages. This is the case, for example, for the authenticated-key-exchange protocols described in the Kyber paper [24, Sec. 5]. Hashing the full protocol view (public key and ciphertext) into the final key already as part of the KEM makes it unnecessary (although of course still safe) to take care of these hashes on the higher protocol layer.

5 Analysis with respect to known attacks

5.1 Attacks against the underlying MLWE problem

MLWE as LWE. The best known attacks against the underlying MLWE problem in Kyber do not make use of the structure in the lattice. We therefore analyze the hardness of the MLWE problem as an LWE problem. We briefly discuss the current state of the art in *algebraic attacks*, i.e., attacks that exploit the structure of module lattices (or ideal lattices) at the end of this subsection.

5.1.1 Attacks against LWE

Many algorithms exist for solving LWE (for a survey see [7]), but many of those are irrelevant for our parameter set. In particular, because there are only

$$m = (k+1)n$$

LWE samples available to the attacker, we can rule out BKW type of attacks [59] and linearization attacks [13]. This essentially leaves us with two BKZ [97, 32] attacks, usually referred to as primal and dual attacks that we will recall in Subsections 5.1.2 and 5.1.3.

The algorithm BKZ proceeds by reducing a lattice basis using an SVP oracle in a smaller dimension b. It is known [52] that the number of calls to that oracle remains polynomial, yet concretely evaluating the number of calls is rather painful, and this is subject to new heuristic ideas [32, 31, 11].

We start with an analysis that ignores this polynomial factor, i.e. only considers the cost of *a single call* to an SVP oracle in dimension b. We will also use for now a very simple cost estimate for the hardness of SVP. This *core-SVP hardness* methodology was introduced in [10, Sec. 6], as a simple way of estimating security. In the light of the cryptanalytic progress that happened during the NIST evaluation Rounds 1 and 2, we consider that this method remains informative, but too coarse to produce accurate security estimates, especially the for security against classical attackers. This will be addressed by the new Section 5.2, added for the Round 3.

Enumeration vs. sieving. There are two algorithmic approaches for the SVP oracle in BKZ: enumeration and sieving algorithms. These two classes of algorithms have very different performance characteristics and, in particular for sieving, it is hard to predict how *practical performance* scales from lattice dimensions that have been successfully tackled to larger dimensions that are relevant in attacks against cryptosystems like Kyber. The starting point of such an analysis is the fact that enumeration algorithms have super-exponential running time, while sieving algorithms have only exponential running time. Experimental evidence from typical implementations of BKZ [48, 32, 38] shows that enumeration algorithms are more efficient in "small" dimensions, so one question is at what dimension sieving becomes more efficient. At the beginning of the NIST PQC project, the best known sieving techniques were slower in practice for accessible dimensions of up to $b \approx 130$.

The work [40] showed (in the classical setting) that sieving techniques can be sped up *in practice* for exact-SVP, becoming less than an order of magnitude slower than enumeration already in dimension 60 to 80. One reason for the improvement is the "dimensions-for-free" technique, allowing to solve SVP in dimension b by sieving in a sligthly smaller dimension $b' = b - d_{4\mathrm{f}}$. Further algorithmic and implementation efforts have finally managed to get sieving to outperform enumeration in practice [4], with a cross-over around dimension 80.

The hardness estimation is complicated by the fact that sieving algorithms are much more *memory intensive* than enumeration algorithms. Specifically, sieving algorithms have exponential complexity not only in time, but also in memory, while enumeration algorithms require only small amounts of memory. In practice, the cost of access to memory increases with the size of memory, which typically only becomes noticeable once the memory requirement exceeds fast local memory (RAM). There is no study, yet, that investigates the algorithmic optimization and practical performance of sieving using slow background storage.

We follow the approach of [10, Sec. 6] to obtain a conservative lower bound on the performance of both sieving and enumeration for the dimensions that are relevant for the cryptanalysis of Kyber. This approach works in the RAM model, i.e., it assumes that access into even exponentially large memory is free.

A lot of recent work has pushed the efficiency of the original lattice sieving algorithms [81, 77], improving the heuristic complexity from $(4/3)^{b+o(b)} \approx 2^{0.415b+o(b)}$ down to $\sqrt{3/2}^{b+o(b)} \approx 2^{0.292b+o(b)}$ using *locality-sensitive hashing* (LSH) techniques [62, 17]. Without the dimensions-for-free technique, the hidden sub-exponential factors are typically much greater than 1 in practice [76, 40].

Most of the sieving algorithms have been shown [64, 61] to benefit from Grover's quantum search algorithm, bringing the complexity down to $2^{0.265b+o(b)}$. However, a concrete analysis shows that, in practice, the quantum speed-up remains tenuous [5].

For our first analysis, we will use $2^{0.292b}$ as the classical and $2^{0.265b}$ and the quantum cost estimate of both the primal and dual attacks with block size (dimension) b. We recall those two attacks in the following.

5.1.2 Primal attack.

The primal attack consists of constructing a unique-SVP instance from the LWE problem and solving it using BKZ. We examine how large the block dimension b is required to be for BKZ to find the unique solution. Given the matrix LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s}+\mathbf{e})$ one builds the lattice $\Lambda = \{\mathbf{x} \in \mathbb{Z}^{m+kn+1} : (\mathbf{A}|\mathbf{I}_m|-\mathbf{b})\mathbf{x} = \mathbf{0} \bmod q\}$

of dimension $d = m+kn+1$, volume q^m, and with a unique-SVP solution $\mathbf{v} = (\mathbf{s}, \mathbf{e}, 1)$ of norm $\lambda \approx \varsigma\sqrt{kn+m}$, where ς is the standard deviation (i.e., the square root of the variance) of the individual secret / error coefficients. Note that the number of used samples m may be chosen between 0 and $(k+1)n$ in our case and we numerically optimize this choice.

Success condition. We model the behavior of BKZ using the geometric series assumption (which is known to be optimistic from the attacker's point of view), that finds a basis whose Gram-Schmidt norms are given by $\|\mathbf{b}_i^\star\| = \delta^{d-2i-1} \cdot \mathrm{Vol}(\Lambda)^{1/d}$, where $\delta = ((\pi b)^{1/b} \cdot b/2\pi e)^{1/2(b-1)}$ [31, 7]. The unique short vector $\mathbf{v}$ will be detected if the projection of $\mathbf{v}$ onto the vector space spanned by the last b Gram-Schmidt vectors is shorter than $\mathbf{b}_{d-b}^\star$. Its projected norm is expected to be $\varsigma\sqrt{b}$, that is, the attack is successful if and only if

$$\varsigma\sqrt{b} \leq \delta^{2b-d-1} \cdot q^{m/d}. \tag{6}$$

We note that this analysis introduced in [10] differs and is more conservative than prior works, which were typically based on the hardness of unique-SVP estimates of [47]. The validity of the new analysis has been confirmed by further analysis and experiments in [6].

5.1.3 Dual attack

The dual attack consists of finding a short vector in the dual lattice $\mathbf{w} \in \Lambda' = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^m \times \mathbb{Z}^{kn} : \mathbf{A}^t\mathbf{x} = \mathbf{y} \bmod q\}$. Assume we have found a vector $(\mathbf{x}, \mathbf{y})$ of length ℓ and compute $z = \mathbf{v}^t \cdot \mathbf{b} = \mathbf{v}^t\mathbf{A}\mathbf{s} + \mathbf{v}^t\mathbf{e} = \mathbf{w}^t\mathbf{s} + \mathbf{v}^t\mathbf{e} \bmod q$, which is distributed as a Gaussian of standard deviation $\ell\varsigma$ if $(\mathbf{A}, \mathbf{b})$ is indeed an LWE sample (otherwise it is uniform mod q). Those two distributions have maximal variation distance bounded by $\epsilon = 4\exp(-2\pi^2\tau^2)$, where $\tau = \ell\varsigma/q$, that is, given such a vector of length ℓ one has an advantage ϵ against decision-LWE.

The length ℓ of a vector given by the BKZ algorithm is given by $\ell = \|\mathbf{b}_0\|$. Knowing that Λ' has dimension $d = m + kn$ and volume q^{kn} we get $\ell = \delta^{d-1}q^{kn/d}$. Therefore, obtaining an ϵ-distinguisher requires running BKZ with block dimension b, where

$$-2\pi^2\tau^2 \geq \ln(\epsilon/4). \tag{7}$$

Note that small advantages ϵ are not relevant since the agreed key is hashed: an attacker needs an advantage of at least $1/2$ to significantly decrease the search space of the agreed key. He must therefore amplify his success probability by building about $1/\epsilon^2$ many such short vectors. Because the sieving algorithms provides $2^{0.2075b}$ vectors, the attack must be repeated at least R times where

$$R = \max(1, 1/(2^{0.2075b}\epsilon^2)).$$

This makes the conservative assumption that all the vectors provided by the Sieve algorithm are as short as the shortest one.

5.1.4 Core-SVP hardness of Kyber

In Table 4 we list the classical and quantum core-SVP-hardness of the three parameter sets of Kyber. The lower bounds of the cost of the primal and dual attack were computed following the approach outlined above using the analysis script `Kyber.py` that is available online at https://github.com/pq-crystals/kyber/tree/master/scripts/.

5.2 Beyond core-SVP hardness

At the time the core-SVP hardness measure was introduced by [10], the best implementations of sieving [17, 76] had performance significantly worse than the $2^{.292b}$ CPU cycles proposed as a conservative estimate by this methodology. This was due to substantial polynomial or even sub-exponential overheads hidden in the complexity analysis of $2^{.292b+o(b)}$ given in [17]. Before this [10] proposed the core-SVP approach, security estimates of lattice schemes were typically based on the cost of SVP via enumeration given in [32, 7], leading to much more aggressive parameters. Beyond the cost of SVP-calls, this methodology also introduced a different prediction of when BKZ solves LWE, which was later confirmed [6] and refined [36].

While doubts were expressed to whether sieving would ever outperform the super-exponential, yet practically smaller, costs of enumeration [32] for relevant cryptographic dimensions, significant progress on sieving algorithms [40, 4] has brought down the cross-over point down to dimension about $b = 80$. In fact, the current SVP records are now held by algorithms that employ sieving[4]. This progress mandates a revision and refinement of Kyber security estimates, especially regading classical attacks. In particular, while it was pretty clear from experiments than the costs hidden in the $o(b)$ before those improvement were positive both in practice and asymptotically, the dimensions-for-free technique of [40] offers a sub-exponential speed-up, making it a priori unclear whether the total $o(b)$ term is positive or negative, both asymptotically and concretely.

In summary, while the core-SVP methodology introduced 5 years ago has pushed designers to be more conservative than previously, it now appears this estimation technique is too coarse to produce accurate security estimates. In the following, we give a refined analysis based on the latest developments described in the literature. We complement this with a discussion of all the approximations, simplifications and foreseeable developments that remain to be explored.

We also note that the choice of the gate count metric was recently discussed in the NIST PQC forum mailing list, and in the case of the algorithm of [17], we do not believe this metric to be very realistic. However, it appears that alternative metrics unavoidably involve physical and technological constants (speed of light, density of information, energy efficiency of gates and data transfers). There seems to be no clear consensus on what some those constants would be with current technology, let alone future technologies. These other metrics would also greatly increase the complications involved in the already delicate tuning and analysis of sieving.

In the following subsection we therefore discuss the current understanding of attacks against Kyber in the gate-count metric. We focus the discussion on the concrete case of Kyber512. This preliminary analysis gives a cost of 2^{151} gates, which is a 2^8 factor margin over the targeted security of the 2^{143} gates required for attacks against AES. Our discussion of the 'known unknowns' conclude that this number could be affected by a factor of up to 2^{16} in either direction. While there is a risk to see the security claim drop below the 2^{143} bar in the gate count metric, one should consider the choice of the gate count metric itself as a substantial margin. We do not think that even a drop as large as 2^{16} would be catastrophic, in particular given the massive memory requirements that are ignored in the gate-count metric. Having listed and documented the sources of uncertainties, we hope that many of them can be tackled in the months to come, to narrow the "confidence interval".

5.2.1 A tentative gate-count estimate accounting for recent progress

For concreteness, we focus this discussion to the case of Kyber512. Let us start by defining the *progressivity overhead* $C = 1/(1 - 2^{-.292}) = 5.46$, that is the limit of ratio between $\sum_{i \leq b} 2^{.292i+o(i)}$ and $2^{.292b+o(b)}$ as b grows.

Primal Attack Only. Our first point is that, while the core-SVP hardness methodology suggest that the dual attack is slightly cheaper than the primal one, it is in fact significantly more expensive. Indeed, the analysis of the dual attack of [10] (repeated above) assumes that one gets exponentially many vectors from sieving in the first block that will be as short as the shortest one. In fact, most of them will have a length $\sqrt{4/3}$ larger. Furthermore, the assumption that we obtain exponentially many such short vectors is incompatible with some of the latest sieving improvements, specifically the dimensions-for-free techinque from [40]. We note that the analysis from [2], which does not assujme those extra short vectors, indeed does predict much larger cost For the dual attack than the primal.

BKZ simulation. The analysis of the BKZ success condition from [10, 6] is based on the so called geometric-series assumption, an assumption that has several inaccuracies; in particular it misses a "tail" phenomenon [98]. We instead rely on the simulator provided as part of the leaky-LWE-estimator of [36]. This simulator uses progressive-BKZ [11, 4], which we believe gives somewhat better performance than fixed-block BKZ. It predicts a median success when reaching blocksize $b = 413$. Success grows by a factor of 1.373 from $b = 412$ to $b = 413$, which is larger than the cost factor of $2^{0.292} \approx 1.224$: aiming at lower blocksize and

[4] https://www.latticechallenge.org/svp-challenge/

retrying is not a worthy strategy (it grows even faster for $b < 412$). The overall dimension of the lattice is $n = 1025$.

The cost of progressive BKZ with sieving up to blocksize b is essentially $C \cdot (n - b) \approx 3340$ times the cost of sieving for SVP in dimension b.

Dimensions for free. According to [40], the number of "dimensions for free" is $d_{4f} = \frac{b \ln(4/3)}{\ln(b/(2\pi e))}$, which is $d_{4\mathrm{f}} = 37.3$ (or 38, rounding upward). That is, each SVP oracle call in dimension $b = 413$ will require running sieving in dimension $b' = b - d_{4\mathrm{f}} = 375$.

We note that [4, Fig. 5] obtains a few more dimensions for free in practice than [40] with two tricks: "on-the-fly lifting" and "pump-down sieves". In practice, this saves a bit of memory ($2^{\cdot 5}$), and has no visible effect on time. The trick at hand is really fit for CPUs (as it uses more floating-point arithmetic), and probably less interesting for implementations in dedicated circuits (which favor XOR-popcounts).

Gate cost of sieving. Recent work [5] has proposed an analysis of the cost of sieving with a classical and quantum circuit (with RAM access to data). More specifically, they focus on the analysis of a "AllPairSearch" function, account for exact volumes of spherical caps and wedges, compute explicit gate counts for the innermost loop operations (XOR-popcounts, inner products), and automatically tune parameters to obtain concrete classical and quantum costs. Regarding the best classical algorithm, they conclude with a cost of about $2^{137.4}$ gates for AllPairSearch in dimension 375.[5]

The work in [5] is motivated by the quantum/classical speed-up, therefore it does not consider the required number of calls to AllPairSearch. Naive sieving requires a polynomial number of calls to this routine, however this number of calls appears rather small in practice using progressive sieving [40, 63], and we will assume that it needs to be called only once per dimension during progressive sieving, for a cost of $C \cdot 2^{137.4}$ gates[6].

Final gate count. Overall, we conclude with a gate count of

$$G = (1025 - 413) \cdot C^2 \cdot 2^{137.4} = 2^{151.5}.$$

We can also estimate the memory requirement following the analysis of [5], maybe working under the assumption that each coordinate of a lattice vector used during the sieving can be represented within a byte. Automatizing the above calculations,[7] we obtain Table 4, summarizing the refined estimates for all three parameter sets.

We also note that a similar refined count of quantum gates seems essentially irrelevant for our security claim: the work of [5] concluded that quantum speed-up of sieving are rather tenuous, while the quantum security target for each category is significantly lower than the classical target.

5.3 Approximations, overheads, and foreseeable improvements

We now propose a list of research direction to refine the best attack or its cost estimation, together with an educated guess of their potential concrete impact. We hope that this list of open problems may motivate the community towards a more informed consensus on the actual cost of lattice attacks, that would apply not only to Kyber, but to all lattice-based candidates in the NIST PQC standardization project.

Q1: Idealized Angles and Lengths. A first approximation made in the concrete analysis of [5] is that all vectors have the same length, and therefore that a pair of vectors leads to a reduction precisely if they have an angle below $\pi/3$. However, both heuristically and experimentally, this is not exactly true. Indeed, the length of the sum $\|x \pm y\|$ may be smaller than $\max(\|x\|, \|y\|)$ despite an angle slightly larger than $\pi/3$ when $\|x\|$ and $\|y\|$ differ a bit. Furthermore, implementations such as [4] even relax the reduction condition, and perform a reduction $z \leftarrow x \pm y$ where z is neither x nor y, but the current worst vector of the database.

[5] This was interpolated from the datafile `cost-estimate-list_decoding-classical.csv` from the version of May 2020 of [5], available at https://eprint.iacr.org/eprint-bin/getfile.pl?entry=2019/1161&version=20200520:144757&file=1161.pdf . This datafile can be extracted from the pdf via the linux tool `pdfdetach`. Another version of that datafile is more easily accessible at https://github.com/jschanck/eprint-2019-1161/blob/main/data/cost-estimate-list_decoding-classical.csv, and differs from the one we used by less than a bit.

[6] This is using the fact inside BKZ, doing a single pump per SVP instance seems sufficient [2], while an isolated SVP instance requires an extra progressive loop, called "work-out" on top of the "pump". That is, an isolated SVP instance should have an extra C factor.

[7] Scripts available at https://github.com/lducas/leaky-LWE-Estimator/tree/NIST-round3

This approximation and relaxation may explain the discrepancy of about a factor of 8 between the number of vectors required in the analysis [5] and the experiments up to dimension $b = 120$ [4]. However, these reducing pairs at higher angles are a bit harder to detect with near-neighbors-search techniques.
Expected influence on memory estimate: A factor of 2^{-3}.
Expected influence on gate-count estimate: A factor between 2^{-3} and 2^{-1}.

Q2: Idealized near-neighbors search. The analysis of [5] assumes that the BDGL [17] sieve behaves as if the buckets are perfectly uniformly distributed over the sphere; however the actual algorithm [17] must resort to some structure in the way these buckets are distributed. The original analysis shows that this structure does not affect the success probability of finding each pair by a factor more than subexponential $2^{\tilde{O}(\sqrt{b})}$ [17]. However, this asymptotic analysis may not be that tight, and indeed, preliminary experiments suggest that this success-probability loss may not be so large in practice.

Furthermore, depending on the parametrization, this algorithm has overheads compared to the idealization of [5]: minimizing memory requirement induces a $2^{O(b/\log(b))}$ overhead on time, which can in principle be traded for an asymptotically similar memory overhead.
Expected influence on gate-count estimate: A factor of up to 2^5. (part of which tradeable for more memory)

Q3: Saturation, collisions, and dimensions for free. Another question that affects the concrete performance of the sieving is the delicate choice of when to stop it. In practice the latest implementations [40, 4] chose to stop the sieving when a fraction $s = 1/2$ of the expected vectors of length $\sqrt{4/3} \cdot \mathrm{GH}$ has been found.[8] This means that the success probability of actually solving SVP with dimensions for free can decrease by a factor s (or maybe a bit less because in practice the sieving outputs about 6 times more vectors, the remaining of which are a sligthly longer than $\sqrt{4/3}\,\mathrm{GH}$, and also have some probability of lifting to the shortest vector).

On the other hand, stopping the sieving with a larger saturation value s also means that we observe more collisions: a pair does reduces, but result more often in an already known vector, making it useless. This must be compensated by enlarging the database, and/or running more reduction rounds. Preliminary experiments showed that the frequency of collision can be significantly larger than s; this may be explained by repeated reduction of the same pair, and the fact that the database tends to not be uniform over the ball, and biased toward shorter vectors.
Expected influence on gate-count estimate: A factor between 2^{-1} and 2^2.

Q4: Number of calls to AllPairSearch**.** The number of calls to AllPairSearch at each step of Progressive Sieving was assumed to be 1. This could be tested experimentally. It is not unthinkable that this is in fact a fractional number less than 1, as, in progressive-sieving, the previous calls in lower dimension may already provide a close to target set of vectors. Preliminary experiments suggest 1/3 or 1/2 in very small dimensions; these experiments have been carried without NNS to avoid interference, and deserve to be extended.
Expected influence on gate-count estimate: A factor between 2^{-2} and 2^2.

Q5: Routing and Congestion (from RAM to circuit). As far as we understand, the sieving circuit studied in [5] still considers RAM access, which, even in the gate count model, and without considering speed-of-light delays easily underestimates cost. For example, once it has been determined in which bucket a vectors belongs, there must be some routing logic to send it there; if there are 2^x addresses, then the sent bits must traverse at least $O(x)$ gates. This gets quite more involved as several cores may want to send data to the same bucket at the same time, inducing overheads for queuing logic and/or packet drops. Extra logistic costs are also required in the bucket itself, and if the targeted bucket size is small, variations in the actual numbers of vectors that reach a bucket may induce further overheads or packet drops.

Such logistic costs can be partially mitigated by re-tuning the algorithm: if, once bucketed, data is used often enough, these overheads will become negligible. However, making buckets larger negatively impacts algorithmic performance in the RAM model, and despite mitigation, some loss seems unavoidable. We note that the current parametrization from [5] suggests to target an average bucket size of 32 in the relevant dimensions, leading to each bit of data being used in 32 many XOR-popcounts. We do not think that this parametrization makes logistic overheads negligible.
Expected influence on gate-count estimate: A factor between 2^2 and 2^8.

[8] Here, GH denote the expected length of the shortest vector according to the Gaussian Heuristic.

Q6: Beyond the gate-count metric. While we mostly stick to the gate-count metric in this discussion, we note that the previous research question Q5 might be naturally paired with a similar study in other metrics, maybe having physical and technological constants as parameters. In particular, we would like to mention an ongoing implementation of Sieving with GPU, for which optimal performances are reached for quite large bucket-size (say 2^{15}), and this despite using a single computation server (i.e., one is limited by RAM to GPU-RAM bandwidth, and not network bandwidth).

Q7: Refined BKZ strategies. We note that there are ways to refine a bit the (progressive) BKZ strategy used in the simulator of [36]. Indeed, a standard idea [67] is to run a single SVP call in a dimension that is considerably larger than the BKZ blocksize after the BKZ preprocessing, in the hope to partially amortize the $(n-b)\cdot C$ factor. While this has been used for sieving in practice [4] on LWE challenges, a precise analysis of the gains in large dimensions is still missing. We note that this is a time-memory trade-off, as this last SVP call will require more memory than the BKZ pre-processing.

Another folklore idea which remains to be documented and quantified, is to try to swap the "heads" and "tails" of the BKZ profile [98], by instead running dual-BKZ as the pre-processing.

One could also tweak the progressive steps of the blocksize in progressive-BKZ, but our preliminary experiments with the simulations of [36] was not suggesting that significant gains are to be expected there.
Expected influence on gate-count estimate: A factor between 2^{-8} and 2^{-2}.

Q8: Module-BKZ in practice. It is known that sieving in cyclotomic/cyclic *ideal* lattices can benefit from speed-ups and memory gain, namely a factor d can be saved on memory, and a factor between $O(d)$ and $O(d^2/\log d)$ can be gained depending on whether Near Neighboor Search techniques are used or not. Namely, the symmetries of those lattices allows to get of full orbit of d vectors of the same length for the price of 1.

In principle, the same gains apply to module lattices of dimension $b = kd$, growing with the module dimension d. It was mentioned on the NIST pqc-forum[9] that a module variant of BKZ could exists, and therefore that its internal SVP oracle could benefit from symmetries. This is confirmed by the theoretical work of [79].

However, there are numerous issues that are likely to prevent substantial gain from this approach. The first one is that one is now very constrained in terms of blocksize: to run module-BKZ with blocksize b on Kyber, one must rely on a sub-module structure of dimension $d = \gcd(b, 256)$. This may force one to over or undershoot the optimal block-size for the attack, and leaves less flexibility, especially for progressivity both at the sieving and BKZ levels. This is even more constraining in combination with the dimension for free, where, as far we can see, one would require both the sieving and the lift context to be aligned with the module structure: $d = \gcd(b, d_{4\mathrm{f}}, 256)$. It is also not clear whether, for the same block-size, module-BKZ would give a basis perfectly as good as a BKZ, given that the basis is now constrained by the module structure. One will also need to adjust Kannan embedding to the module structure, which increases the overall dimension of the lattice by d rather than by 1.
Expected influence on gate-count estimate: A factor between 2^{-4} and at most 2^{0}.

Summary. Combining all the discussion above, it appears that further refinements of the analysis of various aspect of sieving, as well as some foreseeable algorithmic improvements, the estimates may move by a factor somewhere between 2^{-16} and 2^{14}. Again, we hope that at least some of these research directions will soon be documented and elucidated, narrowing down this confidence interval. We recall that they essentially affect all lattice-based candidates in similar way (except maybe Q8).

5.3.1 Algebraic attacks.

While the best known attacks against the MLWE instance underlying Kyber do not make use of the structure in the lattice, we still discuss the current state of the art of such attacks. Most noticeably, several recent works propose new quantum algorithms against Ideal-SVP [43, 30, 21, 34, 35], i.e., solving the shortest vector problem in ideal lattices. The work of [35] mentions obstacles towards a quantum attack on Ring-LWE from their new techniques, but nevertheless suggests using Module-LWE, as it plausibly creates even more obstacles. In [1], Albrecht and Deo establish a reduction from MLWE to RLWE, whose implication is that

[9] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/JXN9NWGt9Ys/m/37tgAlU7DAAJ

a polynomial-time algorithm against RLWE with certain parameters would translate to a polynomial-time algorithm against MLWE. In practical terms, however, this attack has a significant slow-down (and this is not just due to the proof) as the dimension of the module increases. This does suggest that increasing the dimension of the module may make the scheme more secure in concrete terms. In particular, going through this reduction to attack Kyber768 would lead to an RLWE problem with quite large modulus and error ($q' = q^3$, $\varsigma' > q^2\varsigma$), and therefore require the attacker to consider more than 1 sample: the underlying lattice remains a module with a rank strictly larger than 2.

5.4 Attacks against symmetric primitives

All symmetric building blocks of Kyber are instantiated with functions derived from Keccak [20]. In the deterministic expansion of $\mathbf{A}$ from ρ we essentially need SHAKE-128 to produce output that "looks uniformly random" and does not create any backdoors in the underlying lattice problem. In the noise generation we require that concatenating a secret and a public input and feeding this concatenation to SHAKE-256 as input results in a secure pseudorandom function. Breaking any of these properties of SHAKE would be a major breakthrough in the cryptanalysis of SHAKE, which would require replacing SHAKE inside Kyber by another XOF.

The security proofs model SHAKE-128, SHA3-256, and SHA3-512 as random oracles, i.e., they are subject to the standard limitations of proofs in the (quantum-)random-oracle model. Turning these limitations into an attack exploiting the instantiation of XOF, H, or G with SHAKE and SHA3 would again constitute a major breakthrough in the understanding of either Keccak or random-oracle proofs in general.

5.5 Attacks exploiting decryption failures

In Theorems 2 and 3 we see that decryption failure probability plays a role in the attacker's advantage: in the classical context in the term $4q_{RO}\delta$ and in the quantum context in the term $8q_{RO}^2\delta$, where q_{RO} is the number of queries to the (classical or quantum) random oracle.

Attacks exploiting failures. This term in the attacker's advantage is not merely a proof artifact, it can be explained by the following attack: An attacker searches through many different values of m (see line 1 of Alg. 8) until he finds one that produces random coins r (line 3 of Alg. 8) that lead to a decapsulation failure, which will give the attacker information about the secret key. In the quantum setting the search through different values of m is accelerated by Grover's algorithm, which explains the square in the term q_{RO}^2. With this attack in mind note that with 2^{64} ciphertexts (cmp. Section 4.A.2 of the Call for Proposals), there is a chance of 2^{-100} of a decapsulation failure in Kyber768 without any particular effort by the attacker.

To understand what exactly this means for attacks against Kyber, we need to address the following two questions:

1. How hard is it for an attacker to trigger a Kyber decapsulation failure?
2. How hard is it for an attacker equipped with a ciphertext triggering a failure to mount a successful attack against Kyber?

Regarding the first question, the naive approach of an attacker is to try random ciphertexts, which has a success probability of $q_d\delta$, where q_d is the number of decapsulation queries. In the classical random-oracle model, the cost of an attack exploiting failures will also never get lower than that, as it matches the information-theoretic success probability.

Failure boosting using Grover.

A quantum attacker can try to use Grover search to precompute values of m that have a slightly higher chance to produce a failure. The efficacy of Grover search is limited by the fact that an attacker cannot determine offline whether a given value of m, or more specifically, the derived values $\mathbf{r}$ (line 9 of Alg. 5) and $\mathbf{e}_1$ (line 13 of Alg. 5), produce a decapsulation failure. The reason is that the probability of decapsulation failures largely depends on the products $\mathbf{s}^T\mathbf{e}_1$ and $\mathbf{e}^T\mathbf{r}$ and the attacker does not know (the signs of) the coefficients of $\mathbf{s}$ and $\mathbf{e}$.

Original Kyber analysis. The original Kyber submission document gave an analysis of a particular strategy for using Grover's algorithm to search for values of m that produce $\mathbf{e}_1$ and $\mathbf{r}$ with above-average

norm. Intuitively, the larger these values are, the bigger the probability of a decryption failure. The gain achieved through such an approach is, however, limited due to the fact that the distribution of a high-dimensional Gaussian is tightly concentrated around its expected value, while that of a 1-dimensional Gaussian is not as tightly concentrated around its mean. We present this original analysis below, focusing on KYBER768, which has failure probability of 2^{-164}, for concreteness:

The polynomial pair $(\mathbf{e}_1, \mathbf{r})$ can be seen as a vector in $\mathbb{Z}^{1536}$ distributed as a discrete Gaussian with standard deviation $\sigma = \sqrt{\eta_1/2} = 1$ (because $\eta_1 = \eta_2$). By standard tail bounds on discrete Gaussians [14], we know that an m-dimensional vector $\mathbf{v}$ drawn from a discrete Gaussian of standard deviation σ will satisfy

$$\Pr[\|\mathbf{v}\| > \kappa\sigma\sqrt{m}] < \kappa^m \cdot e^{\frac{m}{2}(1-\kappa^2)}, \tag{8}$$

for any $\kappa > 1$.

So for example, the probability of finding a vector which is of length $1.33 \cdot \sigma\sqrt{1536}$ is already as small as 2^{-220}. Even if Grover's algorithm reduces the search space and increases the probability to 2^{-110}, finding such a vector merely increases the chances of getting a decryption error; and the probability increase is governed by the tail-bounds for 1-dimensional Gaussians[10]. For any vector $\mathbf{v}$, if $\mathbf{z}$ is chosen according to a Gaussian with standard deviation σ, then for any κ,

$$\Pr[|\langle \mathbf{z}, \mathbf{v}\rangle| > \kappa\sigma\|\mathbf{v}\|] \leq 2e^{-\kappa^2/2}. \tag{9}$$

If originally, the above probability is set so that decryption errors occur with probability $\approx 2^{-160}$, then $\kappa \approx 15$.[11] If the adversary is then able to increase $\|\mathbf{v}\|$ by a factor of 1.33 (by being able to find larger $(\mathbf{e}_1, \mathbf{r})$), then we can decrease κ by a factor of 1.33 to ≈ 11.25 in (9), which would still give us a probability of a decryption error of less than 2^{-90}. However, finding such a large $\mathbf{v}$ would take at least 2^{110} time, which would make the whole attack cost at least 2^{200}.

Of course one can try to find a slightly smaller $\mathbf{v}$ in the first step so that the entire attack takes less time. If Grover's algorithm really saves a square-root factor, then the optimal value is ≈ 1.05 for κ in (8), which would allow us to lower κ by a factor of 1.05 in (9) to $15/1.05 \approx 14.28$, and would still give a total time to find one decryption error $\approx 2^{-150}$. This makes the attack completely impractical.

Different approaches since 2017. Since the original analysis of failure boosting against KYBER, multiple works have considered multiple aspects of this topic [37, 41]. The latest work on this topic is [22], considers using Grover's algorithm to produce a strategic set of decryption query points which results in a higher chance of triggering a decryption failure *when the number of decryption queries is limited, as in the NIST Call for Proposals.* The overall running time is, however, not less than in the above attack, which didn't restrict itself with a limit on the number of decapsulation queries.

From one to multiple failures. The original KYBER submission document also discussed that an attacker may need more than a single failure to mount a key-recovery attack. For example, the work of [45] required up to 4000 failures, which may suggest that the number of online queries required for a successful attack against KYBER increases by a factor of $\approx 2^{12}$. However, recent work [42] proposed an adaptive strategy, which uses previous failures to significantly lower the cost of the next one. This *directional failure boosting* technique reduces the cost of the full attack to barely more than the cost of triggering the first failure. This means that the answer to the second question in the beginning of this subsection is that one should be on the safe side and make sure that it's hard to trigger even one failure.

Multitarget attacks using failures. Despite the limited gain, an attacker could consider using Grover's algorithm to precompute values of m that produce $\mathbf{r}$ and $\mathbf{e}_1$ with large norm and then use this precomputed set of values of m against many users. This multi-target attack is prevented by hashing the public key pk into the random coins r and thereby into $\mathbf{r}$ and $\mathbf{e}_1$ (line 3 of Alg. 8).

[10] The decryption noise is generated as an inner product of two vectors, and the distribution of this inner product closely resembles the Gaussian distribution.

[11] The above formula only roughly approximates how the decryption error is calculated where $\mathbf{z}$ corresponds to the secret key $(\mathbf{s}, \mathbf{e})$. We should also point out that a part of the decryption error in KYBER is caused by the rounding function Compress, which the adversary has no control over. Therefore this attack will be even less practical than what we describe.