On the Security of LWE-based KEMs under Various Distributions: A Case Study of Kyber

Mingyao Shao[1,3,4], Yuejun Liu[2*], Yongbin Zhou[1,2,3,4*] & Yan Shao[2]

[1]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China;*
shaomingyao@iie.ac.cn;
[2]*School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing, China;*
[3]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China;*
[4]*Key Laboratory of Cyberspace Security Defense, Beijing, China*

Abstract
Evaluating the security of LWE-based KEMs involves two crucial metrics: the hardness of the underlying LWE problem and resistance to decryption failure attacks, both significantly influenced by the secret key and error distributions. To mitigate the complexity and timing vulnerabilities of Gaussian sampling, modern LWE-based schemes often adopt either the uniform or centered binomial distribution (CBD).
This work focuses on Kyber to evaluate its security under both distributions. Compared with the CBD, the uniform distribution over the same range enhances the LWE hardness but also increases the decryption failure probability, amplifying the risk of decryption failure attacks. We introduce a *majority-voting*-based key recovery method, and carry out a practical decryption failure attack on Kyber512 in this scenario with a complexity of 2^{37}.
Building on this analysis, we propose uKyber, a variant of Kyber that employs the uniform distribution and parameter adjustments under the asymmetric module-LWE assumption. Compared with Kyber, uKyber maintains comparable hardness and decryption failure probability while reducing ciphertext sizes. Furthermore, we propose a *multi-value sampling* technique to enhance the efficiency of rejection sampling under the uniform distribution. These properties make uKyber a practical and efficient alternative to Kyber for a wide range of cryptographic applications.

Keywords LWE-based KEMs, Kyber, hardness, decryption failure, centered binomial distribution, uniform distribution

Citation Mingyao Shao, Yuejun Liu, Yongbin Zhou, et al. Title for citation. , for review

1 Introduction

Recent advancements in quantum computing have heightened the urgency to develop post-quantum cryptography (PQC) capable of resisting both classical and quantum attacks [1]. In 2016, the National Institute of Standards and Technology (NIST) launched a process to develop and standardize PQC. Among the candidate algorithms, lattice-based cryptographies (e.g., Kyber [2], Dilithium [3]) are considered highly competive, offering both strong provable security and remarkable efficiency.

The Learning with Errors (LWE) problem, introduced by Regev [4], has been widely used to construct lattice-based cryptographic schemes. The LWE assumption states that it is computationally difficult to distinguish $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$ from uniformly random samples over $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and $\mathbf{e} \leftarrow \chi_\alpha^m$. It has been extended to the Ring-LWE (RLWE) problem [5] and the Module-LWE (MLWE) problem [6], which aim to enhance computational efficiency and reduce the public key size. Notably, MLWE-based Kyber has emerged as a standard for Key Encapsulation Mechanisms (KEMs), culminating in the publication of the standard for Module-Lattice-Based KEM, designated as the Federal Information Processing Standards (FIPS) 203 [7].

In early LWE-based schemes, the error term $\mathbf{e}$ was typically sampled from a (discrete) Gaussian distribution [4,8], motivated by the existence of worst-case to average-case reduction. However, Gaussian sampling has been shown to be either computationally inefficient [9] or susceptible to timing attacks [10–12]. To address these challenges, Döttling et al. [13] and Micciancio et al. [14] introduced the worst-to-average case reduction for LWE with uniformly distributed errors, which can be sampled very efficiently. Cabarcas et al. [15] proposed the first provably secure LWE-based public key encryption (PKE) scheme,

* Corresponding author (email: ,)

Table 1 NIST PQC submissions based on LWE with the CBD or uniform distribution. δ represent the decryption failure probability corresponding to the parameter set with the claimed NIST security category 5.

Scheme	Functionality	Hard problem	secret key/error distributions	δ
Kyber [2]	KEM/PKE	MLWE	CBD	2^{-174}
NewHope [16]	KEM/PKE	RLWE	CBD	2^{-216}
LAC [17]	KEM/PKE	RLWE	CBD	2^{-122}
uSaber [18]	KEM/PKE	RLWR	uniform distribution	2^{-152}
KINDI [19]	KEM/PKE	MLWE	uniform distribution	2^{-216}
Dilithium [20]	Signature	MLWE	uniform distribution	-

in which the secret key and error are uniformly sampled from a relatively small set. Similarly, Alkim et al. [16] introduced NewHope, a key exchange scheme based on LWE, utilizing the centered binomial distribution (CBD) for error sampling. Building on these advancements, many submissions to the NIST PQC standardization process now adopt the CBD or uniform distribution for efficient sampling of secret keys and errors, as summarized in Table 1.

The security of LWE-based KEMs involves two crucial metrics: the hardness of the underlying LWE problem and resistance to decryption failure attacks. Due to the limited number of available samples, lattice reduction-based attacks, such as primal and dual attacks [21,22], remain the most effective methods for solving the LWE problem [2]. These strategies typically reduce the LWE problem to a Shortest Vector Problem (SVP), which is then addressed using lattice reduction algorithms like BKZ-b and its variants. Besides, lattice-based KEMs often lack perfect correctness [23]. Specifically, certain ciphertexts, though correctly generated using the public key, may fail to decrypt correctly with the secret key. These ciphertexts, known as decryption failures or just failures, not only lead to incorrect results but also expose information about the secret key [23–28]. Adversaries can exploit these failures to recover the secret key, enabling decryption failure attacks.

The distributions of the secret key $\mathbf{s}$ and error $\mathbf{e}$ play a crucial role in determining the hardness of the underlying LWE problem and resistance to decryption failure attacks. In this work, we focus on Kyber to investigate the specific impacts of different distributions on these two aspects of security. While the default choice for the secret key and error distributions in Kyber is the CBD, we aim to qualitatively and quantitatively analyze its security under the uniform distribution and propose a Kyber variant under the uniform distribution as an appropriate alternative. Our main contributions are as follows:

For the hardness of the underlying LWE problem, the performance of lattice reduction-based attacks is influenced not by the exact distribution of the secret key and error, but by their standard deviation. The goal of lattice reduction is to find a short vector in the lattice. As the norms of the secret key and error increase, the corresponding short vector in the lattice becomes longer, making it more difficult to find the target vector. Compared with CBD, the uniform distribution over the same range has a larger standard deviation, resulting in larger norms for $\mathbf{s}$ and $\mathbf{e}$. This gives the uniform distribution certain advantages in enhancing the hardness of LWE-based KEMs. Specifically, for Kyber at different security levels, this increased variance raises the hardness of both primal and dual attacks by approximately 20 bits under the core-SVP method.

On the other hand, in the decryption process of LWE-based KEMs, terms like $\langle \mathbf{s}, \mathbf{e} \rangle$ (the inner product of the secret key and the error vector) play a critical role. Minimizing $\langle \mathbf{s}, \mathbf{e} \rangle$ is essential to reducing the likelihood of decryption failures. However, replacing the CBD with a uniform distribution over the same range leads to a larger $\langle \mathbf{s}, \mathbf{e} \rangle$, significantly increasing the probability of decryption failure. This drawback is particularly evident in Kyber512, where the failure probability reaches $2^{-25.4}$. To exploit this vulnerability, we propose a *majority-voting*-based key recovery method that enables full recovery of the secret key within 3,000 decryption failures, with an overall complexity of 2^{37}. Furthermore, this method is applicable to other decryption failure attack scenarios, such as those described in [26], reducing the required number of failures by approximately 50%.

The above analysis indicates that the choice of secret key and error distributions in LWE-based KEMs represents a trade-off between LWE hardness and decryption failure attack vulnerabilities. Based on this analysis, we propose uKyber, a variant of Kyber under the uniform distribution. Compared with Kyber, uKyber achieves comparable levels of LWE hardness and decryption failure probability while reducing ciphertext size. Additionally, we propose a *multi-value sampling* method to enhance the efficiency of rejection sampling under the uniform distribution. This approach makes uKyber faster in implementation than Kyber. These properties make uKyber a practical and efficient alternative to Kyber for a wide range

of cryptographic applications.

1.1 Organization

In Section 2, we introduce some basic preliminaries. Sections 3 and 4 analyze the impact of replacing the CBD with a uniform distribution over the same range on Kyber's hardness and decryption failure probability. In Section 5, we propose uKyber, an appropriate variant of Kyber designed for the uniform distribution. Section 6 discusses the applications of our proposed *majority-voting* key recovery strategy and *multi-value uniform sampling* method in other scenarios. Finally, Section 7 concludes the paper.

2 Preliminaries

2.1 Notation

We denote the ring of integers modulo $q \in \mathbb{Z}^+$ by $\mathbb{Z}_q$, the ring $\mathbb{Z}[x]/(x^n+1)$ as R and the ring $\mathbb{Z}_q[x]/(x^n+1)$ by R_q. Regular font letters denote elements in R or R_q and bold lower-case letters represent vectors with coefficients in R or R_q. By default, all vectors will be column vectors. A matrix of polynomials in $R_q^{k\times\ell}$ is denoted using bold upper-case letters. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^T$. For $\mathbf{A} \in R_q^{k\times 1}$, the vector $\overline{\mathbf{A}} \in \mathbb{Z}_q^{kn\times 1}$ is formed by concatenating the coefficients of each polynomial in $\mathbf{A}$.

For a set S, we write $s \leftarrow S$ to indicate that s, or each of its coefficients is chosen uniformly at random from S. If S is a probability distribution, then this denotes that s is chosen according to the distribution S. The extendable output function Sam takes an input x and then produces y of any desired length that is distributed according to distribution S. We write $y \sim S := \mathrm{Sam}(x)$.

2.2 Definition

Definition 1 (Sizes of elements)**.** For $w \in R$, the ℓ_∞ and ℓ_2 norms of a ring element $w = w_0 + w_1x + ... + w_{n-1}x^{n-1}$ are defined as: $\|w\|_\infty = \max_i |w_i|, \quad \|w\| = \sqrt{|w_0|^2 + \ldots + |w_{n-1}|^2}$.

Definition 2 (Modular reductions)**.** For an even (resp. odd) integer α, we define $r' = r \ mod^{\pm}\ \alpha$ to be the unique element r' in the range $-\frac{\alpha}{2} < r' \leqslant \frac{\alpha}{2}$ (resp. $-\frac{\alpha-1}{2} \leqslant r' \leqslant \frac{\alpha-1}{2}$) such that $r' \equiv r \pmod{\alpha}$. For any positive integer α, we define $r' = r \ mod^{+}\ \alpha$ to be the unique element r' in the range $0 \leqslant r' < \alpha$ such that $r' \equiv r \pmod{\alpha}$.

Definition 3 (Polynomial multiplication)**.** For two polynomials $f(x) \in R_q$ and $g(x) \in R_q$, represented as $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$, $g(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n-1}x^{n-1}$, their product $h(x) = f(x) \cdot g(x) = \sum_{k=0}^{n-1} c_k x^k$, where $c_k = \sum_{i+j=k \bmod n} a_i b_j$.

Definition 4 (Centered binomial distribution)**.** The centered binomial distribution β_η with a positive integer η is defined as: $\beta_\eta = \sum_{i=1}^{\eta}(a_i - b_i)$, where $(a_1, \ldots, a_\eta, b_1, \ldots, b_\eta) \leftarrow \{0,1\}^{2\eta}$. The standard deviation of β_η is $\sigma = \sqrt{\eta/2}$.

Definition 5 (Centered uniform distribution)**.** The centered uniform distribution $\mathcal{U}_u$ with a positive integer u is defined as the discrete uniform distribution over $[-u, u]$. The standard deviation of $\mathcal{U}_u$ is $\sigma = \sqrt{\frac{(2u+1)^2-1}{12}}$.

2.3 CCA-Secure Version of Kyber

Kyber [2] is based on the MLWE problem. The CCA-secure version of Kyber KEM is constructed from a simple CPA-secure PKE, denoted as Kyber.CPAPKE, along with the Fujisaki-Okamoto (FO) transformation [29, 30]. In Kyber, the coefficients of the secret key and error vectors are sampled from the CBD. The specific parameter sets are listed in Table 2. For clarity, we provide a simplified version of Kyber.CPAPKE as Algorithm 1, 2, 3.

Table 2 Kyber parameter sets in round 3 of NIST PQC standardization [2]

	n	k	q	η_1	η_2	(d_u, d_v)	δ	Security
Kyber512	256	2	3329	3	2	(10, 4)	2^{-139}	1 (AES-128)
Kyber768	256	3	3329	2	2	(10, 4)	2^{-164}	3 (AES-192)
Kyber1024	256	4	3329	2	2	(11, 5)	2^{-174}	5 (AES-256)

Algorithm 1 Kyber.CPAPKE.KeyGen()

$\rho, \sigma \leftarrow \{0,1\}^{256}$
$\mathbf{A} \sim R_q^{k \times k} := \mathrm{Sam}(\rho)$
$(\mathbf{s}, \mathbf{e}) \sim \beta_{\eta_1}^k \times \beta_{\eta_1}^k := \mathrm{Sam}(\sigma)$
$\mathbf{t} := \mathbf{As} + \mathbf{e}$
return $(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$

Algorithm 2 Kyber.CPAPKE.Enc(pk, m)

$r \leftarrow \{0,1\}^{256}$
$\mathbf{A} \sim R_q^{k \times k} := \mathrm{Sam}(\rho)$
$(\mathbf{r}, \mathbf{e_1}, e_2) \sim \beta_{\eta_1}^k \times \beta_{\eta_2}^k \times \beta_{\eta_2} := \mathrm{Sam}(r)$
$\mathbf{u} := \mathbf{A}^T\mathbf{r} + \mathbf{e_1}$
$v := \mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rceil \cdot m$
$\mathbf{c_1} := \mathrm{Compress}_q(\mathbf{u}, d_u)$
$c_2 := \mathrm{Compress}_q(v, d_v)$
return $c := (\mathbf{c_1}, c_2)$

Algorithm 3 Kyber.CPAPKE.Dec(sk, c)

$\mathbf{u}' := \mathrm{Decompress}_q(\mathbf{c_1}, d_u)$
$v' := \mathrm{Decompress}_q(c_2, d_v)$
return $m' := \mathrm{Compress}_q(v' - \mathbf{s}^T\mathbf{u}', 1)$

Definition 6 (Compress and Decompress functions)**.** To reduce the size of ciphertexts, Kyber defines $\mathrm{Compress}_q$ and $\mathrm{Decompress}_q$ functions as follows:

$$\begin{aligned} \mathrm{Compress}_q(x, d) &= \lceil (2^d/q) \cdot x \rfloor \bmod^+ 2^d, \\ \mathrm{Decompress}_q(x, d) &= \lceil (q/2^d) \cdot x \rfloor. \end{aligned} \tag{1}$$

This pair of functions ensures that $x' = \mathrm{Decompress}_q(\mathrm{Compress}_q(x, d), d)$ is an approximation of x, satisfying the following condition: $|x' - x \ mod^{\pm} \ q| \leqslant \lceil \frac{q}{2^{d+1}} \rfloor$.

2.4 Decryption Failure

For Kyber, the occurrence of decryption failures depends on the secret key and error vectors $\mathbf{s}, \mathbf{r}, \mathbf{e}, \mathbf{e_1}, e_2$ in combination with the rounding errors $\epsilon_{\mathbf{u}}$, ϵ_v of the (de)compress functions, which are defined as:

$$\begin{aligned} \epsilon_{\mathbf{u}} &= \mathbf{u}' - \mathbf{u} = \mathrm{Decompress}_q(\mathrm{Compress}_q(\mathbf{u}, d_u), d_u) - \mathbf{u}, \\ \epsilon_v &= v' - v = \mathrm{Decompress}_q(\mathrm{Compress}_q(v, d_v), d_v) - v. \end{aligned} \tag{2}$$

Based on the decryption process in Algorithm 3, the recovered message m' is given by:

$$\begin{aligned} m' &= Compress_q(v' - \mathbf{s}^T\mathbf{u}', 1) = \lceil \frac{2}{q}(v' - \mathbf{s}^T\mathbf{u}') \rfloor \mod 2 \\ &= m + \lceil \frac{2}{q}(\mathbf{e}^T\mathbf{r} - \mathbf{s}^T(\mathbf{e_1} + \epsilon_{\mathbf{u}}) + e_2 + \epsilon_v) \rfloor \mod 2. \end{aligned} \tag{3}$$

Let $\mathbf{S}$ be a vector formed by concatenating $-\mathbf{s}$ and $\mathbf{e}$, $\mathbf{C}$ be a vector formed by concatenating $\mathbf{e_1} + \epsilon_{\mathbf{u}}$ and $\mathbf{r}$, and G be $e_2 + \epsilon_v$. Then:

$$\mathbf{S} = \begin{bmatrix} -\mathbf{s} \\ \mathbf{e} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{e_1} + \epsilon_{\mathbf{u}} \\ \mathbf{r} \end{bmatrix}, \quad G = e_2 + \epsilon_v. \tag{4}$$

The message m can be recovered correctly if and only if: $\|\mathbf{S^T C} + G\|_\infty = \|\mathbf{e}^T\mathbf{r} - \mathbf{s}^T(\mathbf{e_1} + \epsilon_{\mathbf{u}}) + e_2 + \epsilon_v\|_\infty \leqslant \frac{q}{4}$.

Definition 7 (Rotations)**.** For $r \in \mathbb{Z}$, the rotation of $\mathbf{C} \in R_q^{k \times 1}$ is defined as:

$$\mathbf{C}^r := x^r \cdot \mathbf{C}(x^{-1}) \mod (x^n + 1).$$

Correspondingly, $\overline{\mathbf{C}^r} \in \mathbb{Z}_q^{kn \times 1}$ denotes its coefficient vector.

It can be easily shown that $\overline{\mathbf{C}^r}$ is constructed to ensure that, for $r \in [0, n-1]$, the r^{th} coordinate of $\mathbf{S}^T\mathbf{C}$ is given by the scalar product $\overline{\mathbf{S}}^T\overline{\mathbf{C}^r}$. In other words, $\mathbf{S}^T\mathbf{C}$ can be decomposed as a sum of scalar products:

$$\mathbf{S}^T\mathbf{C} := \sum_{r \in [0, n-1]} \overline{\mathbf{S}}^T\overline{\mathbf{C}^r} \cdot x^r.$$

A brief example illustrating this decomposition is provided in Appendix A.

3 The Hardness of Kyber under the Uniform Distribution

We analyze the impact of replacing the CBD with a uniform distribution over the same range on Kyber's hardness. Since the best-known attacks on the underlying MLWE problem in Kyber do not exploit the lattice structure, we analyze Kyber's hardness by treating it as an LWE problem.

3.1 The Core-SVP Hardness

Many algorithms have been proposed to solve the LWE problem [31], but many of these are not applicable to the parameter set of Kyber. In particular, for Kyber, since the attacker only has $m = (k+1)n$ LWE samples available, the main remaining approaches are two lattice reduction attacks, commonly referred to as primal and dual attacks [21, 22]. Both attacks primarily rely on lattice reduction algorithms, with the BKZ-b algorithm and its variants being the most prominent.

The BKZ-b algorithm operates by leveraging a SVP oracle in a reduced dimension b, where the number of oracle calls is polynomially bounded [32]. Cryptanalysis typically disregards this polynomial factor and focuses on the *core-SVP* hardness, which refers to the computational cost of solving a single SVP instance in dimension b. Advances in lattice sieving algorithms have significantly reduced the heuristic complexity. Using Locality-Sensitive Hashing (LSH) techniques [33,34], the complexity has been reduced to approximately $2^{0.292b+o(b)}$. Furthermore, when combined with Grover's quantum search algorithm, the complexity is further reduced to $2^{0.265b+o(b)}$ [35, 36].

3.2 Primal Attack

The primal attack typically transforms the LWE problem into a unique SVP (uSVP) through embedding techniques and subsequently resolves it using lattice reduction algorithms.

Specifically, given an LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e} \bmod q)$, it can be reformulated as an inhomogeneous Short Integer Solution (ISIS) problem: $\mathbf{b} = (\mathbf{A}|\mathbf{I}_m)\binom{\mathbf{s}}{\mathbf{e}} \bmod q$. By employing embedding techniques, we can frame this problem as a uSVP within the lattice: $\Lambda = \{\mathbf{x} \in \mathbb{Z}^{m+kn+1} : (\mathbf{A}|\mathbf{I}_m|\text{-}\mathbf{b})\mathbf{x} = \mathbf{0} \mod q\}$. It is easy to verify that the column vectors of the matrix

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ -\mathbf{A} & q\mathbf{I}_m & \mathbf{b} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix} \in \mathbb{Z}^{(m+kn+1)\times(m+kn+1)}$$

constitute a basis for the lattice Λ. This lattice has dimensionality $d = m + kn + 1$, volume q^m, and contains a solution $\mathbf{v} = (\mathbf{s}, \mathbf{e}, 1)$ with a norm approximated as $\|\mathbf{v}\| \approx \sigma\sqrt{m + kn + 1}$, where σ is the standard deviation of the secret key and error coefficients.

Hardness: For an attacker, the optimal scenario occurs when the reduced basis obtained from the BKZ-b algorithm satisfies the Geometric Series Assumption (GSA) [37]. Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ denote the reduced basis and $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*)$ its orthogonalized matrix. Under the GSA, the following holds:

$$\|\mathbf{b}_1\| = \delta^d \det(\Lambda)^{1/d}, \quad \|\mathbf{b}_i^*\| = \delta^{-2d(i-1)/(d-1)}\|\mathbf{b}_1\|,$$

where $\delta = ((\pi b)^{1/b} \cdot b/2\pi e)^{1/(2(b-1))}$ [31, 38]. In this context, the BKZ-b algorithm successfully recovers the unique short vector $\mathbf{v}$ if the norm of $\mathbf{v}$'s projection onto the subspace spanned by the last b orthogonal vectors $(\mathbf{b}_{d-b+1}^*, \ldots, \mathbf{b}_d^*)$ is smaller than $\|\mathbf{b}_{d-b+1}^*\|$. The projected norm can be approximated as $\ell\sqrt{b/d}$, where $\ell = \|\mathbf{v}\| \approx \sigma\sqrt{d}$. Therefore, the hardness of the LWE problem under the primal attack is determined by the smallest b satisfying:

$$\ell\sqrt{b/d} = \sigma\sqrt{b} \leqslant \delta^{(-d^2+2db-d)/(d-1)} \cdot \det(A)^{1/d}.$$

It is evident that as the norm $\|\mathbf{v}\| \approx \sigma\sqrt{d}$ increases, the block size b required to solve the problem also rises. Consequently, an increase in the standard deviation σ of the secret key and error distributions results in a larger norm for the target vector $\mathbf{v}$, thereby amplifying the difficulty of solving the LWE using the primal attack.

3.3 Dual Attack

The dual attack fundamentally transforms the LWE problem into a SIS problem, and leverages the solution of this SIS problem to address the decision-LWE problem.

In the dual lattice $\Lambda = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}^T\mathbf{x} = \mathbf{0} \mod q\}$, define $\mathbf{A}^T = (\mathbf{A}_1 \| \mathbf{A}_2) \in \mathbb{Z}_q^{n\times m}$, where $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times n}$ is a square matrix formed by the first n columns of $\mathbf{A}^T$. Without loss of generality, assume $\mathbf{A}_1$ is invertible. By defining $\mathbf{A}' = (\mathbf{I}_n | \mathbf{A}_1^{-1}\mathbf{A}_2)$, the column vectors of $\Lambda = \{\mathbf{x} \in \mathbb{Z}^m | \mathbf{A}^T\mathbf{x} = 0 \mod q\} = \{\mathbf{x} \in \mathbb{Z}^m | \mathbf{A}'\mathbf{x} = 0 \mod q\}$ are identified, and the matrix

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_n & -\mathbf{A}_1^{-1}\mathbf{A}_2 \\ 0 & \mathbf{I}_{m-n} \end{pmatrix} \in \mathbb{Z}^{m\times m}$$

establishes a lattice basis for Λ, with $\det(\Lambda) = q^n$.

Upon discovering a vector $\mathbf{v}$ in lattice Λ, we calculate $u = \langle \mathbf{v}, \mathbf{b} \rangle = \mathbf{v}^T(\mathbf{A}\mathbf{s} + \mathbf{e}) = \langle \mathbf{v}, \mathbf{e} \rangle$ to distinguish between the Gaussian distribution and the uniform distribution. Specifically, if the norm $\ell = \|\mathbf{v}\|$ is small, then the value of $\langle \mathbf{v}, \mathbf{e} \rangle$ is also small and u can be treated as a random variable drawn from the Gaussian distribution with standard variance $\ell\sigma$. Conversely, if $\mathbf{b}$ is uniformly selected from $\mathbb{Z}_q$, then $u = \langle \mathbf{v}, \mathbf{b} \rangle$ follows a uniform distribution within $\mathbb{Z}_q$.

Hardness: The computational cost of the dual attack is primarily dominated by solving the SIS problem and transforming distinguishing attacks into secret recovery attacks. The probability of distinguishing the subGaussian distribution with standard variance $\ell\sigma$ from the uniform distribution is denoted as $\epsilon = 4\exp(-2\pi^2\tau^2)$, where $\tau = \ell\sigma/q$ [22]. To achieve a success probability greater than $1/2$, at least $1/\epsilon^2$ short vectors are needed. Given that the sieving algorithm provides $2^{0.2075b}$ vectors, the process must be repeated $R = \max(1, 1/(2^{0.2075b}\epsilon^2))$ times. Thus, the complexity of the dual attack is estimated as:

$$R \cdot \text{cost}_b = \max(1, 1/(2^{0.2075b} \cdot 16\exp(-4\pi^2\tau^2))) \cdot \text{cost}_b,$$

where cost_b is the estimated complexity of classical and quantum algorithms for solving the SVP problem on b-dimensional lattices, as in the primal attack.

Clearly, for the LWE problem, as the value of σ increases, the value of ϵ decreases, making it more difficult to distinguish u from elements uniformly distributed in $\mathbb{Z}_q$. Therefore, an increase in the standard deviation σ of the secret key and error distributions amplifies the difficulty of solving the LWE using the dual attack.

We have completed the experimental validation, and the hardness of the primal and dual attacks for Kyber under both the CBD and the uniform distribution are presented in Table 3.

Table 3 Classical and quantum hardness of the primal and dual attacks for Kyber, under the CBD and uniform distribution. σ represents the standard deviation of the secret key and error distributions.

	Kyber512		Kyber768		Kyber1024	
secret key/error distributions	CBD	Uniform	CBD	Uniform	CBD	Uniform
σ of secret key/error	1.225	2.0	1.0	1.414	1.0	1.414
Core-SVP methodology, Primal attack						
BKZ-blocksize β	406	472	626	688	878	961
core-SVP classical hardness	118	138	183	201	256	281
core-SVP quantum hardness	107	125	166	182	232	254
Core-SVP methodology, Dual attack						
BKZ-blocksize β	403	469	620	688	868	953
core-SVP classical hardness	117	137	181	199	253	278
core-SVP quantum hardness	106	124	164	181	230	252

From the perspective of the hardness of Kyber's underlying MLWE problem, the uniform distribution over the same range exhibits a larger variance than CBD, resulting in larger hardness. Specifically, this increased variance raises the hardness of both primal and dual attacks by approximately 20 bits under the core-SVP method. However, this increased variance also raises the probability of decryption failure, thereby increasing the risk of decryption failure attacks. In the next section, we will specifically discuss the threat of decryption failure attacks for Kyber under the uniform distribution.

4 Decryption Failure Attacks on Kyber under the Uniform Distribution

Lattice-based KEMs often lack perfect correctness, which not only leads to incorrect decryption results but also exposes information about the secret key. Adversaries can exploit these failures to recover the secret key, enabling decryption failure attacks. This attack typically involves collecting decryption failures, identifying their signs and positions, and leveraging this information to recover the secret key.

4.1 Collecting Decryption Failures

Firstly, we calculate the decryption failure probability of Kyber when the distribution of the secret key and error is replaced with a uniform distribution over the same range. Specifically, we use convolution technique to determine the distribution of the coefficients in $\mathbf{S^T C}+G = \mathbf{e}^T\mathbf{r}-\mathbf{s}^T(\mathbf{e_1}+\epsilon_\mathbf{u})+e_2+\epsilon_v$ according to the distributions of $\mathbf{s}, \mathbf{r}, \mathbf{e}, \mathbf{e_1}, e_2, \epsilon_\mathbf{u}, \epsilon_v$. We then calculate the probability of $|(\mathbf{S^T C}+G)[i]| > \frac{q}{4} \approx 3329$, denoted as δ_1. Assuming that the decryption failures for each coefficient are independent, the probability that at least one coefficient fails to decrypt is given by $\delta = 1-(1-\delta_1)^n \approx n\delta_1$. These failure probabilities were computed via a Python implementation[1)], with the results shown in Table 4.

Table 4 Decryption failure probability of Kyber under the CBD and uniform distribution. σ represents the standard deviation of the secret key and error distributions.

	Kyber512		Kyber768		Kyber1024	
secret key/error distributions	CBD	Uniform	CBD	Uniform	CBD	Uniform
σ of secret key/error	1.225	2.0	1.0	1.414	1.0	1.414
decryption failure probability δ	$2^{-139.1}$	$2^{-25.4}$	$2^{-165.2}$	$2^{-50.3}$	$2^{-175.2}$	$2^{-47.5}$

The results show that replacing the CBD in Kyber with a uniform distribution over the same range significantly increases the probability of decryption failure. In particular, the decryption failure probability for Kyber512 increases to $2^{-25.4}$. This indicates that a substantial number of decryption failures will occur within a practically feasible time.

For our experiment, we collected decryption failures by randomly generating ciphertexts and requesting decryption. A total of 30,000 decryption failures were collected within one week for Kyber512 under the uniform distribution. The experiment was conducted on a Linux server with a 2.7 GHz Intel Xeon CPU, 1TB of memory, and 56 cores.

4.2 Identifying the Sign and Position of Decryption Failure

Next, we identify both the sign and specific position of the decryption failure. The probability of a single coefficient failing to decrypt is given by $\delta' = n\delta_1(1-\delta_1)^{n-1}$. When $\delta_1 \ll \frac{1}{n}$, it follows that $\delta' \approx \delta$. That implies that, in the event of a decryption failure, there is likely exactly one coefficient responsible for the failure. We denote the coefficient index by r, meaning $|(\mathbf{S}^T\mathbf{C}+G)[r]| = |\overline{\mathbf{S}}^T\overline{\mathbf{C}^r}+G[r]| > \frac{q}{4}$. If $(\mathbf{S}^T\mathbf{C}+G)[r] > \frac{q}{4}$, we classify it as a positive failure; otherwise, it is classified as a negative failure.

Identifying the sign of decryption failure: For Kyber512, $d_v = 4$ implies that the rounding error $\epsilon_v[i]$ is distributed over $\mathcal{U}_{104}$. Thus, the value of $G[i] = e_2[i]+\epsilon_v[i]$ has an important impact on the failure. A positive $G[i]$ indicates a higher probability of a positive failure, while a negative $G[i]$ suggests a higher probability of a negative failure. Statistical analysis of the collected decryption failures shows a 99.8% match rate between the sign of $G[i]$ and the failure sign. Therefore, we can use the sign of $G[i]$ to identify the failure sign with an extremely high probability.

Identifying the position of decryption failure: D'Anvers et al. [25, 26] observed that failure vectors exhibit correlation, which can be exploited to distinguish failure vectors from success vectors. However, as the decryption failure probability increases, this correlation weakens. In our study, we find that for Kyber512 under the uniform distribution, the decryption failure probability of $\delta \approx 2^{-25.4}$ is insufficient to reliably distinguish between failure and success vectors. Experimental verification indicates that, using their method alone, the success rate of identifying the failure position r is only 0.04. However, we observe that there is still a strong correlation between the failure vectors and the secret key, as shown in Figure 1. Based on this, we propose an *iterative-enhancement* method to accurately identify the decryption failure positions. The procedure is detailed in Algorithm 4.

1) https://github.com/pq-crystals/security-estimates.

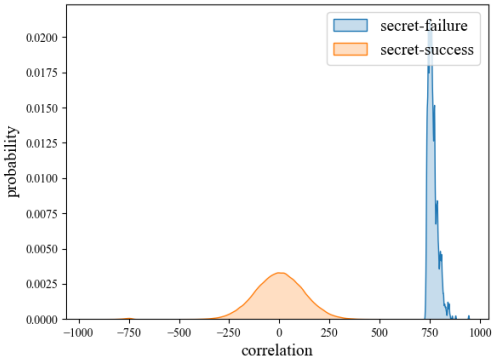

Figure 1 Correlation between the secret key and the failure or success vectors for Kyber512 under the uniform distribution.

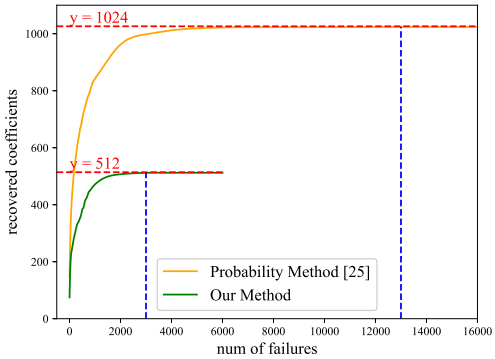

Figure 2 Number of failures required for secret key recovery of Kyber512 under the uniform distribution. The blue vertical line represents the required failures for a 100% success rate.

Algorithm 4 Identifying the failure positions through iterative enhancement

Input: m decryption failures
Output: the decryption failure coefficient for each failure

$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_m\}$ // all decryption failures
$\mathbf{F} = \{\mathbf{C}_1, \cdots, \mathbf{C}_k\}$, $\mathbf{P} = \{r_1, \cdots, r_k\}$ ▷ Initialization phase
$\overline{\mathbf{E}} = \overline{\mathbf{C}_1^{r_1}} + \cdots + \overline{\mathbf{C}_k^{r_k}}$
for $j \in [1, m-k]$ **do** ▷ Enhancement phase
  $\mathbf{C}, r := \underset{\mathbf{C}_i \in \mathbf{C}/\mathbf{F}, r_i \in [0,n)}{\operatorname{argmax}} |\overline{\mathbf{E}} \cdot \overline{\mathbf{C}_i^{r_i}}|$
  $\mathbf{F} = \mathbf{F} \cup \{\mathbf{C}\}$, $\mathbf{P} = \mathbf{P} \cup \{r\}$
  $\overline{\mathbf{E}} = \overline{\mathbf{E}} + \overline{\mathbf{C}^r}$
end for
for $d \in [1, ite]$ **do** ▷ Iteration phase
  for $i \in [1, m]$ **do**
    $r := \underset{r \in [0,n)}{\operatorname{argmax}} |\overline{\mathbf{E}} \cdot \mathbf{C}_i^r|$
    if $r \neq r_i$ **then**
      $\overline{\mathbf{E}} = \overline{\mathbf{E}} - \overline{\mathbf{C}_i^{r_i}}$
      $\overline{\mathbf{E}} = \overline{\mathbf{E}} + \overline{\mathbf{C}_i^r}$
    end if
  end for
end for
return $r_{i, i \in [1,m]}$

In the initialization phase, we assume that k decryption failures $\mathbf{F} = \{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ and their corresponding positions $\mathbf{P} = \{r_1, \ldots, r_k\}$ have already been identified. Based on the geometric perspective described in [26], we can estimate the direction of the secret key by computing the sum of the failure vectors: $\overline{\mathbf{E}} = \sum_{i \in [1,k]} \overline{\mathbf{C}_i^{r_i}} / \|\overline{\mathbf{C}_i^{r_i}}\|_2$. In the enhancement phase, we evaluate all possible rotations of the remaining decryption failures. For each rotation $\overline{\mathbf{C}_j^{r_j}}$, we compute the inner product between the rotated failure vector and the estimated secret key $\overline{\mathbf{E}}$. The failure $\mathbf{C}_j$ and its corresponding position r_j, which yield the largest inner product, are selected and added to the set of identified failures $\mathbf{F}$ and positions $\mathbf{P}$. The estimated secret key $\overline{\mathbf{E}}$ is then updated accordingly. As the number of selected failures increases, the accuracy of the secret key direction estimate improves. This process is then repeated iteratively, performing error correction and updating the selected failure positions, until no further updates occur or the iteration limit (typically set to 3) is reached.

Complexity: In the initialization phase, for each failure $\mathbf{C}_i$, we evaluate its 256 possible rotations $\mathbf{C}_i^r$ in descending order of $G[r]$ values. Experimental results show that, on average, 48 attempts are required to identify the correct failure position. Therefore, for k decryption failures, 48^k attempts are needed to identify all failure positions. Given that k positions are known, the probability of successfully identifying all failure positions is given by $\beta = P[r_i = \delta_i,\ i \in [1, m]]$, where δ_i denotes the correct failure position for $\mathbf{C}_i$. The average number of attempts required to recover all decryption failure positions is $48^k \times \frac{1}{\beta}$.

Table 5 Success rate β and average number of attempts to identify failure positions for 3,000 collected failures with k known positions for Kyber512 under the uniform distribution.

k known failure positions	1	2	3	4
success rate $\beta = P[r_i = \delta_i,\ i \in [1, m]]$	10%	64%	94%	100%
num of attempts: $48^k \times \frac{1}{\beta}$	$2^{8.9}$	$2^{11.8}$	$2^{16.8}$	$2^{22.8}$

Experimental verification: For Kyber512 under the uniform distribution, we collected 3,000 decryption failures and used the *iterative-enhancement* method to identify their positions. As shown in Table 5, the success rate β of recovering all failure positions increases as the number of known failure positions k increases. However, the number of attempts 48^k required during the initialization phase increases exponentially with k. To reduce the overall computational complexity $48^k \times \frac{1}{\beta}$, we set $k = 1$, assuming one known failure position, and applied the above method to identify the remaining failure positions, with a success probability of 10%. The total number of attempts required is $2^{8.9}$.

4.3 Recovering the Secret Key Using Decryption Failures

Once the signs and positions of the decryption failures are identified, we leverage this information to recover the secret $\mathbf{S}$. By applying Bayes' theorem, we can compute the conditional probabilities for different secret key values based on these failures, facilitating effective recovery of the secret key [25].

One decryption failure: Consider the case where only one failure $(\mathbf{C}, G)$ is known, i.e. $\overline{\mathbf{S}}^T\overline{\mathbf{C}^0} + g > q_t$. Using Bayes' theorem, the probability can be expressed as:

$$\begin{aligned} P(\overline{\mathbf{S}}_i|\overline{\mathbf{S}}^T\overline{\mathbf{C}^0} > q_t - g) &= \frac{P(\overline{\mathbf{S}}^T\overline{\mathbf{C}^0} > q_t - g|\overline{\mathbf{S}}_i)P(\overline{\mathbf{S}}_i)}{P(\overline{\mathbf{S}}^T\overline{\mathbf{C}^0} > q_t - g)} \\ &= \frac{P\left(\sum_{j\neq i}\overline{\mathbf{S}}_j\overline{\mathbf{C}^0}_j > q_t - g - \overline{\mathbf{S}}_i\overline{\mathbf{C}^0}_i\right)P(\overline{\mathbf{S}}_i)}{P(\overline{\mathbf{S}}^T\overline{\mathbf{C}^0} > q_t - g)}. \end{aligned} \tag{5}$$

Multiple decryption failures: Consider the case where m failures $\mathbf{C}_1, \cdots, \mathbf{C}_m$ are known. Without loss of generality, assume that $r_i = 0$ and all signs are positive, with corresponding values $g_1, \cdots, g_m$. Assuming the failure vectors $\mathbf{C}_i$ are independent given $\mathbf{S}$, we can express the probability as:

$$\begin{aligned} P(\overline{\mathbf{S}}_i|\overline{\mathbf{S}}^T\overline{\mathbf{C}_k^0} > q_t - g_{k,k\in[1,m]}) &= \frac{P(\overline{\mathbf{S}}^T\overline{\mathbf{C}_k^0} > q_t - g_{k,k\in[1,m]}|\overline{\mathbf{S}}_i)P(\overline{\mathbf{S}}_i)}{P(\overline{\mathbf{S}}^T\overline{\mathbf{C}_k^0} > q_t - g_{k,k\in[1,m]})} \\ &= \frac{P(\overline{\mathbf{S}}_i)\prod_{k=1}^m P(\overline{\mathbf{S}}^T\overline{\mathbf{C}_k^0} > q_t - g_k|\overline{\mathbf{S}}_i)}{\prod_{k=1}^m P(\overline{\mathbf{S}}^T\overline{\mathbf{C}_k^0} > q_t - g_k)}. \end{aligned} \tag{6}$$

The term $P\left(\sum_{j\neq i}\overline{\mathbf{S}}_j\overline{\mathbf{C}^0}_j > q_t - g - \overline{\mathbf{S}}_i\overline{\mathbf{C}^0}_i\right)$, denoted as $P_{fail}[i]$, involves the convolution of $n-1$ random variables $\overline{\mathbf{S}}_j$, which is computationally expensive. To enhance efficiency, we apply the central limit theorem (CLT) to approximate the probability distribution. Given the large number of summed variables, the probability mass function (PMF) of the sum can be approximated by a normal distribution.

$$\begin{aligned} P_{fail}[i] &= P\left(\frac{\sum_{j\neq i}\overline{\mathbf{S}}_j\overline{\mathbf{C}^0}_j - \sum_{j\neq i}\overline{\mathbf{C}^0}_j\,\mathrm{E}[\overline{\mathbf{S}}_j]}{\sqrt{\sum_{j\neq i}\overline{\mathbf{C}^0}_j{}^2\,\mathrm{Var}[\overline{\mathbf{S}}_j]}} > \frac{q_t - g - \overline{\mathbf{S}}_i\overline{\mathbf{C}^0}_i - \sum_{j\neq i}\overline{\mathbf{C}^0}_j\,\mathrm{E}[\overline{\mathbf{S}}_j]}{\sqrt{\sum_{j\neq i}\overline{\mathbf{C}^0}_j{}^2\,\mathrm{Var}[\overline{\mathbf{S}}_j]}}\right) \\ &\approx F_{norm}\left(\frac{\overline{\mathbf{S}}_i\overline{\mathbf{C}^0}_i + \sum_{j\neq i}\overline{\mathbf{C}^0}_j\,\mathrm{E}[\overline{\mathbf{S}}_j] - q_t + g}{\sqrt{\sum_{j\neq i}\overline{\mathbf{C}^0}_j{}^2\,\mathrm{Var}[\overline{S}_j]}}\right). \end{aligned} \tag{7}$$

For each coefficient $\overline{\mathbf{S}}_i$, we compute its probability distribution based on m failures, and select the value with the highest probability as its estimate. For Kyber512, secret $\mathbf{S}$ is a vector formed by concatenating $-\mathbf{s}$ and $\mathbf{e}$, containing a total of $512 + 512$ coefficients. The experimental results, summarized as "Probability Method" in Table 6 and Figure 2, show that for Kyber512 under the uniform distribution, applying the conditional probability method described above allows the full 1024 secret $\mathbf{S}$ coefficients to be recovered with a 100% success rate using 13,000 failures.

Table 6 Number of recovered secret key coefficients given some failures for Kyber512 under the uniform distribution. "Probability Method" corresponds to the conditional probability method in [25] described in Section 4.3, and "Our Method" corresponds to our majority-voting method described in Section 4.4. "ratio" represents the success ratio of recovering the full secret key.

Num of failures		0	300	600	1000	2000	2500	3000	3500	6000	10000	13000
Probability Method [25]	num	141.4	604.9	739.2	842	962	981.4	998	1011.2	1021.8	1023.8	1024
	ratio	0	0	0	0	0	0	0	0	0.11	0.91	1
Our Method	num	71	331.3	411.1	471.4	508.2	511	512	512	512	512	512
	ratio	0	0	0	0	0.1	0.4	1	1	1	1	1

Algorithm 5 Accelerating the secret key recovery by majority-voting

Input: m decryption failures
Output: the estimated secret key using these failures

$\mathbf{C} = \{C_1, C_2, \cdots, C_m\}$ // all decryption failures
// Divide into h blocks, recovering the secret key for each block
$\mathbf{B}_1 = \{C_{1,1}, C_{1,2}, \cdots, C_{1,m/2}\}, \cdots, \mathbf{B}_h = \{C_{h,1}, C_{h,2}, \cdots, C_{h,m/2}\}$
for $i \in [1, h]$ **do**
  for $j \in [0, kn-1]$ **do**
    for $l \in [1, m/2]$ **do**
      $P_{fail}[l, j] \leftarrow C_{i,l}$ using Eq. (7)
    end for
    $P_{fail}[j] = P(\overline{S}_j) \cdot \prod_{l=1}^{m/2} P_{fail}[l, j]$
  end for
  $\overline{S}_{j,i} = \text{argmax}(P_{fail}[j])$ for $j \in [0, kn-1]$
end for
for $j \in [0, kn-1]$ **do**
  $E(\overline{S}_j) = \text{mode}(\overline{S}_{j,i})$ for $i \in [1, h]$
end for
// Substitute the selected 512 coefficients into $\boldsymbol{b} = \boldsymbol{As} + \boldsymbol{e}$
sorted_counts = argsort($E(\overline{S}_j)$.count)
index = sorted_counts[:512]
$E_{half} = E(\overline{S})_{\{index\}}$
Substitute E_{half} into $\mathbf{b} = \mathbf{As} + \mathbf{e}$

4.4 Accelerating Secret Key Recovery by Majority-Voting

As shown in "Probability Method" in Table 6 and Figure 2, with 2,000 decryption failures, we are able to recover most of the secret $\mathbf{S}$ coefficients (939 out of 1024) for Kyber512 under the uniform distribution. However, the indices of the recovered coefficients remain unknown. To fully recover all coefficients, the number of failures required increases significantly, reaching 13,000.

To address this challenge, we propose a *majority-voting*-based key recovery method. This method partitions all collected m failures into h blocks, with each block containing m' failures. These blocks may overlap, and in our experiment, we set $h = 23$ and $m' = m/2$. Each block is processed individually to recover the secret key coefficients, and the final value for each coefficient is determined by the mode of the estimates from all blocks. Furthermore, we can optimize the key recovery process by utilizing the relationship $\mathbf{b} = \mathbf{As} + \mathbf{e}$ from the LWE problem. By correctly recovering any 512 coefficients from $\mathbf{S} = (-\mathbf{s}^T, \mathbf{e}^T)$ and knowing their indices, the remaining 512 coefficients can be directly computed by solving the equation $\mathbf{b} = \mathbf{As} + \mathbf{e}$. The detailed process is described in Algorithm 5.

For Kyber512 under the uniform distribution, our experiments show that by selecting the top 512 coefficients with the highest frequency in the mode and substituting them into $\mathbf{b} = \mathbf{As} + \mathbf{e}$, the full secret key can be recovered with 100% accuracy using 3,000 decryption failures. The results are presented as "Our Method" in Table 6 and Figure 2. Compared with "Probability Method" in [25], our approach reduces the required number of failures by 77%, with an average recovery time of 302.45 seconds.

Complexity: In summary, recovering the full secret key for Kyber512 under the uniform distribution requires the collection of 3,000 decryption failures, resulting in a complexity of approximately cost1 $= 3000 \times 2^{25.4} \approx 2^{37}$. The *iterative enhancement* method is then applied to identify the failure signs and positions, requiring an average of $2^{8.9}$ attempts. Finally, the *majority-voting* method is employed to recover the secret key, with the complexity of each recovery denoted as cost2. The total complexity can thus be approximated as cost1 + $2^{8.9}\times$ cost2. Since cost2 is negligible compared with cost1, the overall complexity of the decryption failure attack on Kyber512 under the uniform distribution is dominated by cost1, approximately 2^{37}.

5 uKyber: A Variant of Kyber under the Uniform Distribution

Based on the previous analysis, directly replacing the CBD in Kyber with a uniform distribution over the same range is clearly not feasible. While this substitution enhances the hardness of the underlying LWE problem, it also results in excessively high decryption failure probabilities, thereby exposing Kyber to practical decryption failure attacks.

In this section, we explore using a uniform distribution over other intervals to reduce decryption failure probabilities while maintaining the LWE hardness. Specifically, we propose a variant of Kyber, referred to as uKyber, which samples the secret key and error from the uniform distribution $\mathcal{U}_u = [-u, u]$.

5.1 Construction

In this section, we provide a formal description of uKyber. First, We introduce an intermediate CPA-secure PKE, which is transformed into a CCA-secure KEM using the FO transformation.

uKyber.CPAPKE: Let $n, q, k, u_s, u_e, u_r, u_{e_1}, u_{e_2}, d_u, d_v$ be positive integers, and let Sam be an extendable output function modeled as a random oracle. The uKyber.CPAPKE consists of the following three algorithms:

- uKyber.CPAPKE.KeyGen(): Randomly choose $\rho, \sigma \leftarrow \{0,1\}^n$, and sample $(\mathbf{s}, \mathbf{e}) \sim \mathcal{U}_{u_s}^k \times \mathcal{U}_{u_e}^k := \mathrm{Sam}(\sigma)$. Then compute $\mathbf{A} := \mathrm{Sam}(\rho) \in R_q^{k\times k}$, and $\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$. Then, return the public key $pk = (\mathbf{t}, \rho)$ and the secret key $sk = \mathbf{s}$.
- uKyber.CPAPKE.Enc(pk, m): Given the public key pk and plaintext m, randomly choose $r \leftarrow \{0,1\}^n$, and sample $(\mathbf{r}, \mathbf{e_1}, e_2) \sim \mathcal{U}_{u_r}^k \times \mathcal{U}_{u_{e_1}}^k \times \mathcal{U}_{u_{e_2}} := \mathrm{Sam}(r)$. Then, compute $\mathbf{u} := \mathbf{A}^T\mathbf{r} + \mathbf{e_1}, v := \mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rceil \cdot m$, and return the ciphertext

$$c = (\mathrm{Compress}_q(\mathbf{u}, d_u), \mathrm{Compress}_q(v, d_v)).$$

- uKyber.CPAPKE.Dec(sk, c): Given the secret key $sk = \mathbf{s}$ and a ciphertext $c = (\mathbf{c_1}, c_2)$, decompress $\mathbf{u}' := \mathrm{Decompress}_q(\mathbf{c_1}, d_u)$, $v' := \mathrm{Decompress}_q(c_2, d_v)$, and compute

$$m' := \mathrm{Compress}_q(v' - \mathbf{s}^T\mathbf{u}', 1).$$

uKyber.CCAKEM: Let H: $\mathcal{B}^* \to \mathcal{B}^{32}$ and G: $\mathcal{B}^* \to \mathcal{B}^{32} \times \mathcal{B}^{32}$ be two hash functions modeled as random oracles. By applying a slightly modified FO transformation, the CPA-secure uKyber.CPAPKE is transformed into a CCA-secure uKyber.CCAKEM as follows:

- uKyber.CCAKEM.KeyGen(): Randomly choose $z \leftarrow \mathcal{B}^{32}$, and compute $(pk, sk') =$ uKyber.CPAPK.KeyGen(). Then, return the public key pk and the secret key $sk = (sk' \| pk \| \mathrm{H}(pk) \| z)$.
- uKyber.CCAKEM.Encap(pk): Given the public key pk, randomly choose $m \leftarrow \{0,1\}^{256}$, compute $(\overline{K}, r) := \mathrm{G}(m \| \mathrm{H}(pk))$, $c :=$ uKyber.CPAPKE.Enc(pk, m, r). Finally, compute the encapsulated key $K := \mathrm{KDF}(\overline{K} \| \mathrm{H}(c))$, and return the ciphertext c and encapsulated key K.
- uKyber.CCAKEM.Decap(c, sk): Given the secret key $sk = (sk' \| pk \| \mathrm{H}(pk) \| z)$ and ciphertext c, compute $m' :=$ uKyber.CPAPKE.Dec(sk', c) and $(\overline{K}', r') := \mathrm{G}(m' \| h)$, $c' :=$ uKyber.CPAPKE. Enc (pk, m', r'). If $c' = c$, return $K := \mathrm{KDF}(\overline{K}' \| \mathrm{H}(c))$, otherwise, return $K := \mathrm{KDF}(z \| \mathrm{H}(c))$.

5.2 Choices of Parameters

When using the normal form of LWE, where the secret key and error are sampled from the same distribution [39], it becomes challenging to select suitable parameters without either compromising LWE hardness or significantly increasing the probability of decryption failure, as illustrated in Figure 3.

To allow for more flexible parameter sets that balance the LWE hardness and decryption failure probability, we adopt the asymmetric LWE (ALWE) problem proposed by Zhang et al. [40], where the secret key and error do not need to be sampled from distributions with the same parameters. The ALWE problem, denoted as $\mathrm{ALWE}_{n,m,q,\alpha_1,\alpha_2}$, involves finding $\mathbf{s} \in \mathbb{Z}_q^n$ from samples $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}, \mathbf{s} \xleftarrow{\$} \chi_{\alpha_1}^n, \mathbf{e} \xleftarrow{\$} \chi_{\alpha_2}^m$.

We propose setting $u = 2$ (or $u = 1$) to sample the secret key and error coefficients uniformly from the range $[-2, 2]$ (or $[-1, 1]$), applicable to three security levels, referred to as uKyber512, uKyber768 and uKyber1024. To more precisely match the LWE hardness and decryption failure probability of Kyber,

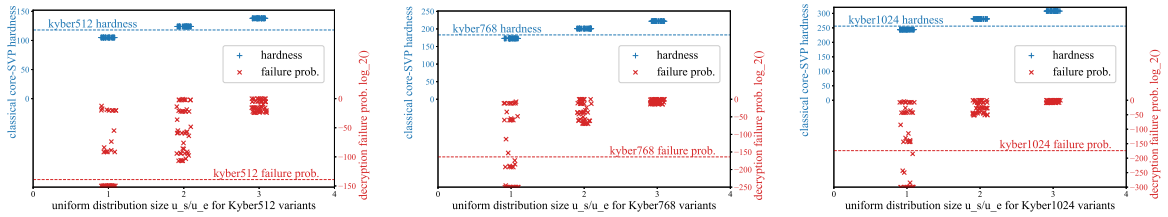

Figure 3 LWE hardness and failure probability of Kyber variants under the uniform distribution, where $\mathbf{s}$ and $\mathbf{e}$ are sampled from same distribution $\mathcal{U}_u$. The horizontal axis denotes the size of the uniform distribution $\mathcal{U}_u$. Blue + indicates the LWE hardness under the classical core-SVP oracle, while red × represents the decryption failure probability. Different values in each column represent the LWE hardness or decryption failure probability under different (de)compression parameters (d_u, d_v).

Table 7 Parameter sets for uKyber in different security levels.

	n	k	q	u_s	u_e	u_r	u_{e_1}	u_{e_2}	(d_u, d_v)
uKyber512	256	2	3329	1	2	1	1	1	$(9, 3)$
uKyber768	256	3	3329	1	2	1	1	1	$(10, 3)$
uKyber1024	256	4	3329	1	2	1	1	1	$(10, 5)$

Table 8 Core-SVP hardness and decryption failure probability for Kyber and uKyber. σ represents the standard deviation of the secret key and error distributions.

	NIST Security Level	σ of secret key/error	Failure Probability	Classical Core-SVP	Quantum Core-SVP
Kyber512	1	1.225	2^{-139}	118	107
uKyber512	1	1.075	2^{-134}	114	103
Kyber768	3	1	2^{-164}	183	166
uKyber768	3	1.075	2^{-156}	186	169
Kyber1024	5	1	2^{-174}	256	232
uKyber1024	5	1.075	2^{-181}	261	237

we adjusted the parameter d_u and d_v of the (de)compression functions, while keeping other parameters n, k, q unchanged. The specific parameter sets are presented in Table 7.

Zhang et al. [40] established an approximate relation between the hardness of ALWE and LWE as follows: $\text{ALWE}_{n,q,m,\alpha_1,\alpha_2} \approx \text{LWE}_{n,q,m,\sqrt{\alpha_1\alpha_2}}$. Leveraging this approximation and the Core-SVP methodology described in Section 3, we evaluate the LWE hardness of uKyber and compute the decryption failure probability under the proposed parameter sets. The results are summarized in Table 8.

As shown in Table 8, compared with Kyber, uKyber achieves comparable LWE hardness and decryption failure probability, making it a suitable variant of Kyber under the uniform distribution from a security perspective.

5.3 Provable Security

Definition 8 (AMLWE Problem)**.** Let k, ℓ, u_1, u_2 be positive integers. The decisional AMLWE problem, denoted as $\text{AMLWE}_{n,q,k,\ell,u_1,u_2}$ asks to distinguish $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$ from uniform samples over $R_q^{k\times\ell} \times R_q^k$, where $\mathbf{A} \leftarrow R_q^{k\times\ell}$, $\mathbf{s} \leftarrow \mathcal{U}_{u_1}^{\ell}$, and $\mathbf{e} \leftarrow \mathcal{U}_{u_2}^{k}$.

Definition 9 (AMLWE Assumption)**.** For an appropriate choice of parameters n, q, k, ℓ, u_1, u_2, there exists no quantum polynomial-time adversary that can solve the $\text{AMLWE}_{n,q,k,\ell,u_1,u_2}$ problem.

Under this assumption, we demonstrate in Appendix C that uKyber.CPAPKE is provably CPA-secure, and uKyber.CCAKEM is provably CCA-secure. Formally, we have the following theorems:

Theorem 1. Suppose Sam is a random oracle. If $\text{AMLWE}_{n,q,k,\ell,u_1,u_2}$ is hard, then the scheme uKyber.CPAPKE is CPA-secure.

Theorem 2. Suppose Sam, H and G are random oracles. If $\text{AMLWE}_{n,q,k,\ell,u_1,u_2}$ is hard, then the scheme uKyber.CCAKEM is CCA-secure.

5.4 Performance Analysis

Compared with Kyber, uKyber offers several performance advantages, including smaller ciphertext sizes, fewer PRBs, and faster implementation speeds. Below, we provide a detailed analysis of these aspects.

Table 9 Comparison between uKyber and Kyber in terms of the size of sk, pk and ct

Schemes	Size (in bytes)		
	sk	pk	ct
Kyber512	1632	800	768
uKyber512	1632	800	672
Kyber768	2400	1184	1088
uKyber768	2400	1184	1056
Kyber1024	3168	1568	1568
uKyber1024	3168	1568	1440

Algorithm 6 Multi-Value Uniform Sampling via Rejection

Sample m elements uniformly from the range $[-u, u]$.
Input: PRBs $\bar{b} = (b_0, b_1, \cdots) \in \{0,1\}^{\lceil \log_2(2u+1)^m \rceil}$
Output: m elements uniformly in $[-u, u]$ or $\perp$

```
d = binary_to_decimal(b̄)
if d < (2u + 1)^m then
    for j ← 0 to m − 1 do
        c ← d mod (2u + 1)
        d ← ⌊d/(2u + 1)⌋
        u_i ← c
    end for
    return u_i − u
else
    return ⊥
end if
```

Smaller ciphertext sizes: The parameters (d_u, d_v) for the (de)compress functions in uKyber are reduced compared with those in Kyber, resulting in shorter ciphertext lengths. The ciphertext length is computed using the formula: $length(c) = (k \cdot d_u + d_v) \cdot n/8$. Specifically, for uKyber512, uKyber768, and uKyber1024, the ciphertext sizes are 672 bytes, 1056 bytes, and 1440 bytes, respectively, as listed in Table 9. These sizes are smaller than the corresponding ciphertext sizes of Kyber512 (768 bytes), Kyber768 (1088 bytes), and Kyber1024 (1568 bytes).

Fewer pseudorandom bits: PRBs are critical for generating the secret key and error. When sampling from the CBD β_u, $2u$ PRBs are required. For sampling from the uniform distribution $\mathcal{U}_u$, at least $\lceil log_2(2u+1) \rceil$ PRBs are necessary. In cases where $2u+1$ is not a power of 2, rejection sampling is employed, similar to the approach used in Dilithium.

Consider the case where $u = 2$. Sampling uniformly from $[-2, 2]$, which includes 5 elements, is equivalent to uniformly sampling d from $[0, 4]$ and subtracting 2. This requires at least $\lceil \log_2(5) \rceil = 3$ PRBs. However, with 3 PRBs, $2^3 = 8$ states can be represented, leaving 3 extra states that must be rejected if $d > 4$. This results in a rejection probability of $\frac{3}{8}$. Consequently, on average, $3 \times \frac{8}{5} = 4.8$ PRBs are required to successfully sample one value from $\mathcal{U}_2$. To reduce PRB usage, we propose a *multi-value uniform sampling via rejection* method, detailed in Algorithm 6.

This method transforms the process of sampling m values from $[0, k-1]$ into sampling a single value c from $[0, k^m - 1]$. The value c is then converted into its base-k representation, $(u_1 u_2 \dots u_m)_{(k)}$, which is evenly mapped to m-digit base-k numbers. The resulting $u_1, u_2, \dots, u_m$ represent m random numbers uniformly distributed across $[0, k-1]$. General rejection sampling is employed to sample c, requiring $\lceil \log_2(k^m) \rceil$ PRBs. The rejection probability for this process is $1 - (k^m / 2^{\lceil \log_2(k^m) \rceil})$.

For $u = 2$, where $k = 2u + 1 = 5$, we set $m = 3$, which corresponds to uniformly sampling 3 numbers from $[0, 4]$. To achieve this, we select $\lceil \log_2(5^3) \rceil = 7$ PRBs and convert them into a decimal number c. If $c < 5^3$, we convert c into a base-5 number $(b_1 b_2 b_3)_{(5)}$; otherwise, c is rejected. The probability of successful sampling is $5^3/2^7 = \frac{125}{128}$, and the average number of PRBs required to sample one value is $(7 \times \frac{128}{125})/3 \approx 2.4$. Thus, on average, 2.4 PRBs are required to uniformly sample a value from $[-2, 2]$. Similarly, for $u = 1$, where $k = 2u + 1 = 3$, we set $m = 5$. We select $\lceil \log_2(3^5) \rceil = 8$ PRBs and convert them into a decimal number c. If $c < 3^5$, we convert c into a base-3 number $(b_1 b_2 b_3 b_4 b_5)_{(3)}$; otherwise, c is rejected. The probability of successful sampling is $3^5/2^8 = \frac{243}{256}$, and the average number of PRBs required to sample one value is $(8 \times \frac{256}{243})/5 \approx 1.7$. Thus, on average, 1.7 PRBs are required to uniformly sample a value from $[-1, 1]$.

We calculate the PRBs required for secret key and error generation in KeyGen and Encryption operations. Experimental results, shown in Table 10, confirm that uKyber requires approximately half the PRBs needed by Kyber for these operations. This significantly reduces the computational time required for hash operations to generate these PRBs.

Faster implementation speed: The reduction in the number of PRBs required for secret key and error sampling results in a corresponding decrease in the hash operations necessary for generating these PRBs. This improvement enhances the overall efficiency of uKyber's implementation.

Experiments were conducted using the C reference implementation on a 64-bit Ubuntu 22.04 virtual machine equipped with an Intel Core i7-12700 processor (2.10 GHz) and 11 GB of memory. Each algorithm was executed 1,000,000 times, and the average CPU cycles were recorded. As shown in Table

Table 10 Comparison between uKyber and Kyber in terms of the number of PRBs required for secret key/error generation in KeyGen and Encryption, and the running time of KeyGen, Encap and Decap.

	PRBs in KeyGen		PRBs in Enc		Running time (in cycles)		
	theoretical	practical	theoretical	practical	KeyGen	Encap	Decap
Kyber512	6144	6144	6144	6144	44064	53540	62446
uKyber512	2099	2126	2176	2203	40980	51402	60416
Kyber768	6144	6144	7168	7168	75295	83423	94132
uKyber768	3149	3189	3046	3048	73329	80498	91207
Kyber1024	8192	8192	9216	9216	114871	120801	137113
uKyber1024	4198	4252	3917	3944	113100	118668	132391

10, the experimental results demonstrate that uKyber offers some advantages in terms of running time compared with Kyber.

Shared uniform distribution sampler with Dilithium: In the transition to PQC, applications like the TLS protocol require both PKE/KEM for secure key exchange and digital signature schemes for authentication. Most cryptoprocessors in real-world scenarios operate on resource-constrained platforms with limited area and memory. Thus, developing compact and unified design methods for implementing post-quantum PKE/KEM and signature schemes is crucial for practical PQC deployment.

Kyber and Dilithium, both part of the CRYSTALS suite and standardized by NIST as PKE/KEM and signature schemes, are often implemented together on the same processor [41, 42]. Shared modules such as polynomial multiplier, adder and subtractors can significantly reduce the area of the combined implementation [43]. For the combined implementation of uKyber and Dilithium, further sharing of the rejection sampling module for uniform distributions can further enhance the compactness of the combined implementation.

6 Discussion

Our proposed *majority-voting key recovery* method and *multi-value uniform sampling via rejection* method can be applied to other scenarios and cryptographic schemes. Below, we discuss their applications in more detail.

6.1 Application of the *Majority-Voting Key Recovery* Method

The *majority-voting* method can be extended to other decryption failure attack strategies, such as the geometric approach from D'Anvers et al. [26]. This approach leverages the property that, in high-dimensional spaces, two vectors with large inner product values tend to have close directions [44]. The mean of the failure vectors is used as the estimated direction of the secret key.

To demonstrate the effectiveness of the majority-voting method, we use Kyber as an example. Suppose there are m linearly independent failure ciphertexts $\mathbf{C}_1, \ldots, \mathbf{C}_m$ and their corresponding decryption failure positions $r_1, \ldots, r_m$. The direction of the secret key is estimated as $\overline{\mathbf{E}} = \overline{\mathbf{C}_{\text{rav}}}/\|\overline{\mathbf{C}_{\text{rav}}}\|_2$ where

$$\overline{\mathbf{C}_{\text{rav}}} = \sum_{i\in[1,m]} \overline{\mathbf{C}_i^{r_i}}/\|\overline{\mathbf{C}_i^{r_i}}\|_2. \tag{8}$$

The norm of the secret key is then estimated using the approximation: $\overline{\mathbf{E}'}^T \cdot \left(\sum_{i\in[1,m]} \overline{\mathbf{C}_i^{r_i}}\right)/m \approx q_t$, resulting in the estimated value of the secret key:

$$\overline{\mathbf{E}'} := \overline{\mathbf{E}} \cdot m \cdot q_t \cdot \| \sum_{i\in[1,m]} \overline{\mathbf{C}_i^{r_i}}\|_2^{-1}. \tag{9}$$

We replicated this key recovery method in our experiments and improved it using our *majority-voting* approach. As shown in Figure 4, our method successfully recovers the full secret key with less than half the number of failures required by the original geometric method. For example, for Kyber768, the geometric method requires 210 failures to fully recover the secret key with 100% success rate, whereas our method achieves the same result with only 100 failures.

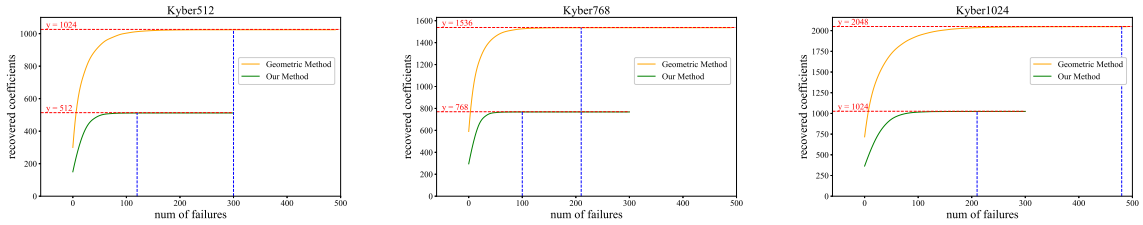

Figure 4 Comparison of the number of failures required for secret key recovery of Kyber using our *majority-voting* method and *geometric* method in [26]. The blue vertical line represents the required failures for a 100% success rate.

Table 11 Comparison of the running time (in cycles) for poly_uniform function, secret key sampling, and KeyGen between Dilithium and Dilithium-M.

scheme	poly_uniform	sk sampling	KeyGen
Dilithium2	1548	11232	106034
Dilithium-M2	929	7413	99870
Dilithium3	1604	17893	200767
Dilithium-M3	920	9970	170684
Dilithium5	1559	20033	297608
Dilithium-M5	961	13912	285973

6.2 Application of the *Multi-Value Uniform Sampling via Rejection* Method

Dilithium employs rejection sampling to generate the secret vectors $\mathbf{s_1}, \mathbf{s_2}$, which follows a uniform distribution over $[-\eta, \eta]$. To enhance sampling efficiency, we adopt the above proposed *multi-value uniform sampling via rejection* method and refer to the modified algorithm as Dilithium-M.

For both Dilithium 2 and Dilithium 5, where $\eta = 2$, we follow the description in Section 5.4, selecting 7 PRBs at a time to generate 3 values uniformly distributed over $[-2, 2]$. For Dilithium 3, where $\eta = 4$, we set $m = 5$, corresponding to sampling 5 values uniformly distributed over $[-4, 4]$. By selecting $\lceil \log_2(9^5) \rceil = 16$ PRBs to generate 5 values, the probability of successful sampling is calculated as $9^5/2^{16} \approx 0.9$, and the average number of PRBs required per value is $(16/0.9)/5 \approx 3.56$.

We evaluated the efficiency of the proposed method by measuring the number of PRBs required for generating sk, as well as the runtime (measured in CPU cycles) of the poly_uniform function, secret key sampling, and KeyGen algorithms in Dilithium and Dilithium-M. The results, summarized in Table 11, indicate that Dilithium-M reduces the runtime for secret key sampling by approximately 40% compared with the original Dilithium.

7 Conclusion

In this work, we conduct a comprehensive security analysis of Kyber under both the CBD and the uniform distribution, examining the trade-offs from both LWE hardness and decryption failure probability. Compared with the CBD, the uniform distribution over the same range increases the variance, enhancing the LWE hardness. However, this improvement comes at the cost of a higher decryption failure probability, making Kyber more susceptible to decryption failure attacks. To design an appropriate variant of Kyber under the uniform distribution, we propose uKyber leveraging the ALWE assumption proposed by Zhang Jiang. Compared with Kyber, uKyber maintains comparable LWE hardness and decryption failure probability, while providing some performance advantages such as smaller ciphertext size, and faster implementation speed. These properties make uKyber a practical and efficient alternative to Kyber for various cryptographic applications.

References

1 IBM. IBM debuts next-generation quantum processor & IBM quantum system two, extends roadmap to advance era of quantum utility. 2023
2 Roberto Avanzi, Joppe Bos, L´eo Ducas, et al. CRYSTALS-Kyber algorithm specifications and supporting documentation (version 3.02). NIST PQC Round 3, 2022
3 Gorjan Alagic, Daniel Apon, David Cooper, et al. Status report on the third round of the nist post-quantum cryptography standardization process. US Department of Commerce, NIST, 2022

4 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM), 2009, 56(6): 1–40
5 Vadim Lyubashevsky, Chris Peikert, Oded Regev. On ideal lattices and learning with errors over rings. In: Proceedings of the Advances in Cryptology–EUROCRYPT 2010, French Riviera, 2010. 1–23
6 Adeline Langlois, Damien Stehl´e. Worst-case to average-case reductions for module lattices. Des. Codes Cryptogr., 2015, 75(3):565–599
7 NIST. FIPS 203, Module-Lattice-based key encapsulation mechanism standard. Department of Commerce, Washington, D.C., Federal Information Processing Standards Publication, 2023
8 Zvika Brakerski, Adeline Langlois, Chris Peikert, et al. Classical hardness of learning with errors. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing, 2013. 575–584
9 Joppe W Bos, Craig Costello, Michael Naehrig, et al. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: Proceedings of 2015 IEEE Symposium on Security and Privacy, 2015. 553–570
10 Leon Groot Bruinderink, Andreas H¨ulsing, Tanja Lange, et al. Flush, gauss, and reload–a cache attack on the BLISS lattice-based signature scheme. In: Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, 2016. 323–345
11 Peter Pessl, Leon Groot Bruinderink, Yuval Yarom. To BLISS-B or not to be: Attacking strongswan's implementation of post-quantum signatures. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017
12 Thomas Espitau, Pierre-Alain Fouque, Benoˆıt G´erard, et al. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017. 1857–1874
13 Nico D¨ottling, J¨orn M¨uller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In: Proceedings of the Advances in Cryptology–EUROCRYPT 2013, Athens, Greece, 2013. 18–34
14 Daniele Micciancio, Chris Peikert. Hardness of sis and lwe with small parameters. In: Proceedings of the Annual cryptology conference, 2013. 21–39
15 Daniel Cabarcas, Florian G¨opfert, and Patrick Weiden. Provably secure LWE encryption with smallish uniform noise and secret. In: Proceedings of the 2nd ACM Wookshop on ASIA Public-Key Cryptography, Kyoto, Japan, 2014. 33–42
16 Erdem Alkim, L´eo Ducas, Thomas P¨oppelmann, et al. Post-quantum key exchange—a New Hope. In: Proceedings of 25th USENIX Security Symposium (USENIX Security 16), 2016. 327–343
17 Xianhui Lu, Yamin Liu, Dingding Jia, et al. LAC: Lattice-based Cryptosystems. NIST PQC Round 2, 2018
18 Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, et al. Saber: Mod-lwr based kem (round 3 submission). NIST PQC Round 3, 2022
19 Rachid El Bansarkhani. KINDI: 20171130 submission. NIST PQC Round 1, 2017.
20 Shi Bai, L´eo Ducas, Eike Kiltz, et al. CRYSTALS-Dilithium: Algorithm specifications and supporting documentation. NIST PQC Round 3. 2020
21 Claus-Peter Schnorr, Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming, 1994, 66:181–199
22 Yuanmi Chen, Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In: Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, 2011. 1–20
23 Nick Howgrave-Graham, Phong Q Nguyen, David Pointcheval, et al. The impact of decryption failures on the security of NTRU encryption. In: Proceedings of the Annual International Cryptology Conference, 2003. 226–246
24 Qian Guo, Thomas Johansson, Jing Yang. A novel CCA attack using decryption errors against LAC. In: Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, 2019. 82–111
25 Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, et al. Decryption failure attacks on IND-CCA secure lattice-based schemes. In: Proceedings of the Public-Key Cryptography–PKC 2019, Beijing, China, 2019. Part II 22, 565–598
26 Jan-Pieter D'Anvers, M´elissa Rossi, Fernando Virdia. (One) failure is not an option: bootstrapping the search for failures in lattice-based encryption schemes. In: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2020. 3–33
27 Nina Bindel, John M Schanck. Decryption failure is more likely after success. In: Proceedings of the International Conference on Post-Quantum Cryptography, 2020. 206–225
28 Jan-Pieter D'Anvers, Senne Batsleer. Multitarget decryption failure attacks and their application to Saber and Kyber. In: Proceedings of the IACR International Conference on Public-Key Cryptography, 2022. 3–33
29 Eiichiro Fujisaki, Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In: Proceedings of the Annual international cryptology conference, 1999. 537–554
30 Dennis Hofheinz, Kathrin H¨ovelmanns, Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In: Proceedings of the Theory of Cryptography Conference, 2017. 341–371
31 Martin R Albrecht, Rachel Player, Sam Scott. On the concrete hardness of learning with errors. Journal of Mathematical Cryptology, 2015, 9(3):169–203
32 Guillaume Hanrot, Xavier Pujol, Damien Stehl´e. Terminating BKZ. Cryptology ePrint Archive, 2011
33 Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In: Proceedings of the Advances in Cryptology–CRYPTO 2015, Santa Barbara, CA, USA, 2015. Part I 35, 3–22
34 Anja Becker, L´eo Ducas, Nicolas Gama, et al. New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms, 2016, 10–24
35 Thijs Laarhoven, Michele Mosca, Joop Van De Pol. Finding shortest lattice vectors faster using quantum search. Designs, Codes and Cryptography, 2015, 77:375–400
36 Thijs Laarhoven. Search problems in cryptography: from fingerprinting to lattice sieving. 2016
37 Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, 2003. 145–156
38 Yuanmi Chen. Lattice reduction and concrete security of fully homomorphic encryption. Dept. Informatique, ENS, Paris, France, PhD thesis, 2013
39 Benny Applebaum, David Cash, Chris Peikert, et al. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Proceedings of the 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, 2009. 595–618
40 Jiang Zhang, Yu Yu, Shuqin Fan, et al. Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of smaller sizes. In: Proceedings of the 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, 2020. Part II 23, 37–65
41 Luke Beckwith, Abubakr Abdulgadir, Reza Azarderakhsh. A flexible shared hardware accelerator for nist-recommended algorithms crystals-kyber and crystals-dilithium with SCA protection. In: Proceedings of the Cryptology - CT-RSA 2023, San Francisco, CA, USA, 2023. 469–490
42 Konstantina Miteloudi, Joppe W. Bos, Olivier Bronchain, Bj¨orn Fay, and Joost Renes. PQ.V.ALU.E: post-quantum RISC-V custom ALU extensions on dilithium and kyber. In: Proceedings of the 22nd International Conference, CARDIS 2023,

Amsterdam, The Netherlands, 2023. 190–209
43 Suraj Mandal, Debapriya Basu Roy. Kid: A hardware design framework targeting unified NTT multiplication for crystalskyber and crystals-dilithium on FPGA. In: Proceedings of the 37th International Conference on VLSI Design and 23rd International Conference on Embedded Systems, Kolkata, India, 2024. 455–460
44 T Tony Cai, Jianqing Fan, Tiefeng Jiang. Distributions of angles in random packing on spheres. Journal of Machine Learning Research, 2013, 14:1837
45 Patrick Billingsley. Probability and measure. John Wiley & Sons, 2017
46 Haodong Jiang, Zhenfeng Zhang, Long Chen, et al. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In: Proceedings of the 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, 2018. Part III 38, 96–125

Appendix A Rotations of polynomial vectors

Definition 10 (Rotations)**.** For $r \in \mathbb{Z}$, the rotation of $\mathbf{C} \in R_q^{k\times 1}$ is defined as:

$$\mathbf{C}^r := x^r \cdot \mathbf{C}(x^{-1}) \mod x^n + 1.$$

Correspondingly, $\overline{\mathbf{C}^r} \in \mathbb{Z}_q^{kn\times 1}$ denotes its coefficient vector.

It is easy to show that $\overline{\mathbf{C}^r}$ is constructed to ensure that for $r \in [0, \ldots, n-1]$, the r^{th} coordinate of $\mathbf{S}^T\mathbf{C}$ is given by the scalar product $\overline{\mathbf{S}}^T\overline{\mathbf{C}^r}$. In other words, one is now able to decompose $\mathbf{S}^T\mathbf{C}$ as a sum of scalar products:

$$\mathbf{S}^T\mathbf{C} := \sum_{r\in[0,n-1]} \overline{\mathbf{S}}^T\overline{\mathbf{C}^r} \cdot x^r.$$

We introduce a brief example to illustrate it in Example 1.

Example 1. For polynomial vectors $\mathbf{S}$ and $\mathbf{C}$ in $\mathbb{Z}_q^{2\times 1}[x]/(x^3+1)$:

$$\mathbf{S} = \begin{bmatrix} s_{0,0} + s_{0,1}x + s_{0,2}x^2 \\ s_{1,0} + s_{1,1}x + s_{1,2}x^2 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} c_{0,0} + c_{0,1}x + c_{0,2}x^2 \\ c_{1,0} + c_{1,1}x + c_{1,2}x^2 \end{bmatrix},$$

we get the following coefficient vectors:

$$\overline{\mathbf{S}} = \begin{bmatrix} s_{0,0} \\ s_{0,1} \\ s_{0,2} \\ s_{1,0} \\ s_{1,1} \\ s_{1,2} \end{bmatrix}, \quad \overline{\mathbf{C^0}} = \begin{bmatrix} c_{0,0} \\ -c_{0,2} \\ -c_{0,1} \\ c_{1,0} \\ -c_{1,2} \\ -c_{1,1} \end{bmatrix}, \quad \overline{\mathbf{C^1}} = \begin{bmatrix} c_{0,1} \\ c_{0,0} \\ -c_{0,2} \\ c_{1,1} \\ c_{1,0} \\ -c_{1,2} \end{bmatrix}, \quad \overline{\mathbf{C^2}} = \begin{bmatrix} c_{0,2} \\ c_{0,1} \\ c_{0,0} \\ c_{1,2} \\ c_{1,1} \\ c_{1,0} \end{bmatrix}.$$

Therefore, the product of the two polynomial vectors $\mathbf{S}^T\mathbf{C} := \sum_{r\in[0,2]} \overline{\mathbf{S}}^T\overline{\mathbf{C}^r} \cdot x^r$.

Appendix B Probability Theory

Theorem B3 (Bayes' theorem)**.** Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

A version of Bayes' theorem for 3 events results from the addition of a third event C on which all probabilities are conditioned. The formula shown is:

$$P(A|B \cap C) = \frac{P(B|A \cap C)P(A|C)}{P(B|C)}.$$

Theorem B4 (Central limit theorem (CLT))**.** Suppose $\{X_1, \ldots, X_n\}$ is a sequence of i.i.d. random variables with $E[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2$, $\overline{X}_n = (X_1 + \cdots + X_n)/n$. Then, as n approaches infinity, the random variables $\sqrt{n}\left(\overline{X}_n - \mu\right)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$:

$$\sqrt{n}(\overline{X}_n - \mu) \to \mathcal{N}(0, \sigma^2). \tag{B1}$$

More generally, X_i has to be independent, but not necessarily identically distributed. Suppose $\{X_1, \ldots, X_n\}$ is a sequence of independent random variables, each with finite expected value μ_i and variance σ_i^2. Define $s_n^2 = \sum_{i=1}^n \sigma_i^2$. If the Lyapunov's condition [45] is satisfied, then a sum of $(X_i - \mu_i)/s_n$ converges in distribution to a standard normal random variable, as n goes to infinity:

$$\frac{1}{s_n}\sum_{i=1}^{n}(X_i - \mu_i) \to \mathcal{N}(0, 1).$$

Appendix C Provable Security of uKyber

In this section, we show that under the MLWE assumption, uKyber.CPAPKE is provably CPA-secure, and uKyber.CCAKEM is provably CPA-secure.

Appendix C.1 IND-CPA Security of uKyber.CPAPKE

Theorem C5. Suppose Sam is random oracle. If $\mathrm{MLWE}_{n,q,k,\ell,u_1,u_2}$ is hard, then the scheme uKyber.CPAPKE is CPA-secure.

Proof. This proof proceeds via a sequence of games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$. In the final game, we show that the advantage of any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is negligible.

Game $\mathbf{G}_0$: This game is the real game for the IND-CPA security. In this game, the adversary $\mathcal{A}$ has access to a random oracle Sam, and is given a public key $pk = (\mathbf{t}, \rho)$. Adversary $\mathcal{A}$ chooses two plaintexts m_0, m_1, and then obtains a challenge ciphertext $c = (\mathbf{c_1}, c_2)$ on message m_b, where b is a random bit. Finally, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ as the guess of $b \in \{0,1\}$.

Game $\mathbf{G}_1$: This game is the same as $\mathbf{G_0}$, except that picking pk at random. If there exists a PPT adversary $\mathcal{A}$ that can distinguish $\mathbf{G_1}$) from $\mathbf{G_0}$, then we can construct a PPT algorithm $\mathcal{B}$ solving the $\mathrm{MLWE}_{n,q,k,\ell,u_1,u_2}$ problem. Specifically, given an instance $(\mathbf{A}, \mathbf{t})$ of the MLWE problem, $\mathcal{B}$ aims to decide whether $(\mathbf{A}, \mathbf{t})$ is sampled from a uniform distribution of $R_q^{k\times k} \times R_q^k$. Formally, $\mathcal{B}$ behaves exactly as in game $\mathbf{G_0}$, except that it chooses a random $\rho \leftarrow \{0,1\}^n$, programs the random oracle Sam such that $Sam(\rho) = A$, and returns $pk = (\mathbf{t}, \rho)$ to $\mathcal{A}$. Since Sam is a random oracle, the probability that $Sam(\rho)$ has already been defined is negligible. If $(\mathbf{A}, \mathbf{t})$ is uniformly random in $R_q^{k\times k} \times R_q^k$, then $\mathcal{B}$ behaves as in game $\mathbf{G_1}$. Otherwise, $\mathcal{B}$ behaves as in game $\mathbf{G_0}$. In other words, if $\mathcal{A}$ can distinguish $\mathbf{G_0}$ and $\mathbf{G_1}$ with non-negligible probability, then $\mathcal{B}$ can solve the MLWE problem with non-negligible probability.

Game $\mathbf{G}_2$: This is the same as $\mathbf{G_1}$, except that using uniformly random values to replace $u = \mathbf{A}^T\mathbf{r} + \mathbf{e_1}$ and $v = \mathbf{t}^T\mathbf{r} + e_2$ used in the generation of the challenge ciphertext. If there exists a PPT adversary $\mathcal{A}$ who can distinguish $\mathbf{G_2}$ from $\mathbf{G_1}$, then we can construct a PPT algorithm $\mathcal{B}$ that solves the $\mathrm{MLWE}_{n,q,k+1,\ell,u_1,u_2}$ problem. Specifically, given an instance $(\mathbf{A}, \mathbf{t}, \mathbf{u}, v)$ of the MLWE problem, $\mathcal{B}$ aims to decide whether $(\mathbf{A}, \mathbf{t}, \mathbf{u}, v)$ is sampled from a uniform distribution in $R_q^{k\times k} \times R_q^k \times R_q^k \times R_q$. Formally, $\mathcal{B}$ chooses a random $\rho \leftarrow \{0,1\}^n$, programs the random oracle Sam such that $Sam(\rho) = A$, and returns $pk = (\mathbf{t}, \rho)$ to $\mathcal{A}$. After receiving two plaintexts m_0, m_1 from $\mathcal{A}$, $\mathcal{B}$ picks $b \leftarrow \{0,1\}$ at random, computes $c_1 = \mathrm{Compress}_q(\mathbf{u}, d_u), c_2 = \mathrm{Compress}_q(v + \lceil \frac{q}{2} \rceil \cdot m_b, d_v))$, and then gives the ciphertext $\mathbf{c} = (\mathbf{c_1}, c_2)$ to $\mathcal{A}$. If $(\mathbf{A}, \mathbf{t}, \mathbf{u}, v)$ is uniformly random in $R_q^{k\times k} \times R_q^k \times R_q^k \times R_q$, then $\mathcal{B}$ simulates as in game $\mathbf{G_1}$. Otherwise, $\mathcal{B}$ simulates as in game $\mathbf{G_0}$. Thus, if $\mathcal{A}$ can distinguish $\mathbf{G_2}$ and $\mathbf{G_1}$ with non-negligible probability, then $\mathcal{B}$ can solve the MLWE problem.

In the final game $\mathbf{G_2}$, m_b in the challenge ciphertext is perfectly hidden by uniformly random v. Therefore, the advantage of $\mathcal{A}$ is 0 in $\mathbf{G_2}$, which completes the proof of Theorem 5.

Appendix C.2 IND-CCA Security of uKyber.CCAKEM

Since uKyber.CCAKEM is obtained by applying a slightly tweaked FO transformation [29, 30], to the PKE scheme uKyber.CPAPKE, given the results in [29, 30] and Theorem 5, we have the following theorem.

Theorem C6. Suppose Sam, H and G are random oracles. If $\mathrm{MLWE}_{n,q,k,\ell,u_1,u_2}$ is hard, then the scheme uKyber.CCAKEM is CCA-secure.

Notice that the algorithm Decap will always return a random decapsulation key even if the checks fail (i.e., implicit rejection). Furthermore, the paper [46] showed that if the underlying PKE is CPA-secure, then the resulting KEM with implicit rejection obtained by using the FO transformation is also CCA-secure in the quantum random oracle model (QROM). Given the results in [46] and Theorem 5, we have the following theorem.

Theorem C7. Suppose Sam, H and G are random oracles. If $\mathrm{MLWE}_{n,q,k,\ell,u_1,u_2}$ is hard, then the scheme uKyber.CCAKEM is CCA-secure in the quantum random oracle model.