CRYPTREC: ML-KEM Evaluation Report

PQShield

Thomas Espitau, Shuichi Katsumata, Niels Samwel, Thom Wiggers, Wessel van Woerden, Timo Zijlstra

Executive Summary

Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) is a lattice-based key encapsulation mechanism (KEM), standardized by the U.S. National Institute of Standards and Technology (NIST) as Federal Information Processing Standard (FIPS) 203 [Nat24b] in August, 2024. It is derived from CRYSTALS-KYBER [Ava+21] and designed to provide confidentiality against attackers equipped with large-scale quantum computers. The evaluation covers the scheme's design rationale, provable security guarantees, concrete hardness against cryptanalytic attacks, implementation security regarding side-channels, and its applicability in secure network protocols such as TLS 1.3. The following is a summary of ML-KEM.

Construction: The scheme is based on the Fujisaki-Okamoto transform, transforming a base INDCPA secure public key encryption scheme called KPKE into an IND-CCA secure KEM. It has a non-zero but negligible decryption failure rate (2^{-139} to 2^{-174}), which is deemed operationally safe and does not pose a security risk under correct implementation.

Provable Security: It's security is based on the *Module Learning with Errors* (MLWE) problem, a structured variant of the standard LWE problem that balances efficiency with security. Tight reductions exist in the classical random oracle model (ROM). In the quantum ROM, while reductions are asymptotically looser, they still provide sufficient confidence in the scheme' s resistance to quantum adversaries.

Security Margins: Concrete cost estimates for the best known attacks (using conservative cost models such as Core-SVP and MATZOV) indicate that all ML-KEM parameter sets meet or exceed their targeted NIST security levels.

Algebraic Structure: Attacks exploiting the algebraic structure of module lattices (e.g., ideal lattice attacks) were analyzed. The report concludes that for the specific module ranks used in ML-KEM ($k \in \{2, 3, 4\}$), these structured attacks do not outperform generic lattice reduction techniques.

Performance: ML-KEM demonstrates excellent performance in software and hardware, significantly outperforming traditional elliptic-curve cryptography (ECC) in computation time, although key and ciphertext sizes are larger.

Side-Channel Attacks: Unprotected implementations are vulnerable to side-channel attacks, including differential power analysis and timing attacks on decryption failures.

Countermeasures: Effective countermeasures such as masking (arithmetic and boolean) and shuffling are available. While these introduce performance overheads, they are necessary for high-assurance deployments in hostile environments.

Application to TLS 1.3: ML-KEM is fully viable for TLS 1.3. The larger key sizes (encapsulation keys and ciphertexts) increase handshake traffic but result in negligible impact on overall connection latency in most network scenarios. The report discuss the use of "hybrid" key exchange mechanisms, combining ML-KEM with traditional elliptic-curve Diffie-Hellman (ECDH), as a robust transition strategy to mitigate risks associated with new cryptographic primitives.

Based on current cryptanalytic knowledge, ML-KEM is a robust and secure post-quantum KEM. Its theoretical foundations are sound, and its parameter sets provide adequate security margins against known classical and quantum threats. While it offers faster processing speeds compared to traditional cryptographic schemes, the increased key and ciphertext sizes may present implementation challenges for memory-constrained devices. However, it is judged to be viable for use in other general-purpose applications without issues. Furthermore, provided that implementations are rigorously protected against side-channel attacks, it is suitable for widespread deployment in government and critical infrastructure systems.

エグゼクティブ・サマリー

Module-Lattice-Based Key-Encapsulation Mechanism（ML-KEM）は，モジュール格子に基づく鍵カプセル化メカニズム（KEM）であり，2024 年 8 月に，米国国立標準技術研究所（NIST）によって連邦情報処理標準 FIPS203 として標準化された．本方式は CRYSTALS-KYBER [Ava+21] から派生したものであり，大規模な量子計算機に対しても安全であるように設計されている．本評価は，方式の設計根拠，証明可能安全性，暗号解読攻撃に対する具体的な困難性，サイドチャネルに関する実装セキュリティ，および TLS 1.3 などのインターネット通信を保護する暗号化プロトコルへの適用可能性を網羅している．以下，評価内容の概要を述べる．

構成：本方式は藤崎・岡本変換に基づいており，IND-CPA 安全な公開鍵暗号方式である KPKE を，IND-CCA 安全な KEM に変換している．本方式はゼロではないが無視可能な復号失敗率（2^{-139} から 2^{-174}）を持つ．これは運用上安全であるとみなされ，正しく実装されている限り安全性を損ねるものではない．

証明可能安全性：安全性は，*Module Learning with Errors*（MLWE）問題に基づいている．これは，効率性と安全性のバランスをとるために，標準的な LWE 問題を構造化した変種である．古典的ランダムオラクルモデル（ROM）においては緊密な帰着が存在する．量子 ROM においては，帰着は漸近的に緩くなるものの，量子敵対者に対する耐性について十分な信頼性を与えるものである．

セキュリティマージン：既知の最良の攻撃に対する具体的なコスト見積もり（Core-SVP や MATZOV などの保守的なコストモデルを使用）は，すべての ML-KEM パラメータセットが，目標とする NIST 安全性レベルを満たしているか，あるいは上回っていることを示している．

代数的構造：モジュール格子の代数的構造を利用する攻撃（例：イデアル格子攻撃）についても分析を行った．本報告書では，ML-KEM で使用される特定のモジュールランク（$k \in \{2, 3, 4\}$）において，これらの構造を利用した攻撃は，構造を利用しない一般的な格子基底簡約アルゴリズムを上回るものではないと結論付ける．

性能：ML-KEM はソフトウェアおよびハードウェアにおいて優れたパフォーマンスを示し，鍵や暗号文のサイズは大きくなるものの，計算時間については従来の楕円曲線暗号（ECC）より大幅に高速である．

サイドチャネル攻撃：対策が施されていない実装は，電力差分解析や復号失敗に対するタイミング攻撃などのサイドチャネル攻撃に対して脆弱である．

対策：マスキング（算術およびブール）やシャフリングといった効果的な対策が利用可能である．これらは効率性を損なう対策だが，敵対的な環境における安全性を保証するために不可欠である．

TLS 1.3 への適用：ML-KEM は TLS 1.3 において十分に利用可能である．鍵長（カプセル化鍵および暗号文）の増大によりハンドシェイクのデータ送量は増加するが，多くのネットワークシナリオにおいて，全体的な接続遅延時間への影響は無視できる範囲である．本報告書では，新しい暗号プリミティブに伴うリスクを軽減するための堅牢な移行戦略として，ML-KEM と従来の楕円曲線 Diffie-Hellman（ECDH）を組み合わせたハイブリッド鍵交換メカニズムの性能についても言及する．

現在の暗号解析に関する知見に基づき，ML-KEM は堅牢かつ安全な耐量子 KEM であると考える．その理論的基盤は健全であり，各パラメータセットは既知の古典，および，量子の暗号解析アルゴリズムに対して十分なセキュリティマージンを提供している．計算時間は従来の暗号方式に比べより高速である一方で，鍵や暗号文長は増加するため，メモリ制約があるデバイス等においては実装上の課題が生じる可能性がある．しかし，それ以外の用途においては問題なく利用できると判断される．また，サイドチャネル攻撃に対して厳密に保護された実装が行われることを前提として，政府および重要インフラシステムへの広範な展開にも適している．

Contents

1 Introduction and Outline

As part of the Post-Quantum Cryptography (PQC) project initiated by the National Institute of Standards and Technology (NIST), the United States has started a multi-year effort to identify, optimize and eventually standardize cryptographic primitives that remain secure in the presence of large-scale quantum computers. NIST announced in 2022 its intention to standardize a lattice-based Key Encapsulation Mechanism (KEM), CRYSTALS-KYBER [Ava+21], as one of the first post-quantum public-key primitives. In 2024, this effort culminated in the publication of the *Module-Lattice-Based Key-Encapsulation Mechanism* (ML-KEM) as Federal Information Processing Standards (FIPS) 203 [Nat24b], which specifies a slightly modified but fully standardized version of CRYSTALS-KYBER. From the viewpoint of government agencies and operators of critical infrastructure, ML-KEM is intended to serve as a general-purpose building block for post-quantum key establishment in a broad range of protocols and deployment scenarios.

On the technical side, ML-KEM is a highly optimized module version [LS15] of the LPR encryption scheme, which is based on the Ring-LWE problem [LPR10]. The LPR construction itself rests on a long and steadily growing line of work on lattice-based encryption schemes [Reg05; LP11] that relate the security of practical cryptosystems to well-studied worst-case problems on lattices. ML-KEM inherits this foundation while making specific design choices — such as employing a module structure rather than fully unstructured lattices — that aim to balance performance, implementation simplicity, and confidence in the underlying hardness assumptions. In particular, ML-KEM has been designed to be efficient on a wide range of platforms, from general-purpose CPUs to constrained devices, while still admitting constant-time implementations and side-channel hardened variants suitable for high-assurance environments.

From a security perspective, ML-KEM is known to satisfy IND-CCA security, a strong notion of active security in which a ciphertext ct* leaks nothing about the encapsulated key, even to an adversary that is given access to an oracle decrypting any ciphertext other than ct*. This property is particularly important in realistic deployment settings where adversaries may interact with decryption endpoints via network protocols, inject or modify ciphertexts in transit, or exploit protocol error messages.

The purpose of the present report is to give a structured and self-contained overview of ML-KEM from both a theoretical and a practical standpoint.

Outline of Report.

Section 2. We begin by recalling the necessary background and notation, including basic concepts from lattice-based cryptography, security notions for public-key encryption and KEMs, and the role of the (quantum) random oracle model in the analysis of ML-KEM.

Section 3. We then provide a high-level description of ML-KEM itself, starting from the underlying IND-CPA secure lattice-based encryption scheme and explaining how the Fujisaki-Okamoto transform is used to obtain an IND-CCA secure KEM. In this part we also summarize the standardized parameter sets and briefly discuss their intended security levels and performance characteristics.

Section 4. Next, we review the provable security results for the IND-CCA security of ML-KEM, outlining the main reductions from standard lattice problems and clarifying in which models and under which assumptions these guarantees hold. Where relevant, we distinguish between classical and quantum adversaries. We further discuss other advanced security features ML-KEM is known to satisfy.

Section 5. We then turn to practical cryptanalysis and concrete security estimates. This includes an overview of the best known quantumu attacks on the underlying lattice problems, the impact of decryption failures and weak-key considerations, and the resulting security margins for the parameter sets selected in FIPS 203.

Section 6. We address implementation details, focusing on both performance and physical security. We describe the algorithmic building blocks and analyze vulnerabilities to side-channel attacks. We further discuss necessary countermeasures, such as masking and shuffling, and present performance results in hardware that illustrate the cost of these protections.

Section 7. Finally, we examine the application of ML-KEM in the Transport Layer Security (TLS) 1.3 protocol. We evaluate the integration of ML-KEM and hybrid (Post-Quantum/Traditional) key exchange mechanisms, comparing them to traditional ECDH in terms of computation time and bandwidth. The section also covers real-world deployment challenges and surveys the availability of ML-KEM in popular cryptographic libraries.

2 Preliminaries

To keep the report self-contained, we provide a minimal presentation of the preliminaries. More details are provided in the FIPS 203 publication [Nat24b] and references therein.

2.1 Notations

We write $A\|B$ for the concatenation of values A and B. Boldface lower- and upper-case symbols such as $\mathbf{a}$ and $\mathbf{M}$ denote column vectors and matrices, respectively. $\mathbf{a}^\top$ and $\mathbf{M}^\top$ denote their transposes. For vectors $\mathbf{a}$ and $\mathbf{b}$, we may sometimes use $\langle \mathbf{a}, \mathbf{b} \rangle$ to denote the inner product $\mathbf{a}^\top \cdot \mathbf{b}$. For a real $x \in \mathbb{R}$, $\lceil x \rfloor$ is the rounding of x to the nearest integer, where if $x = y + 1/2$ for some $y \in \mathbb{Z}$, then $\lceil x \rfloor = y + 1$. For a vector $\mathbf{v} \in \mathbb{R}^d$, $\|\mathbf{v}\|$ denotes the L_2 norm.

For a set $\mathcal{M}$, we write $m \xleftarrow{\$} \mathcal{M}$ for the process of sampling m uniformly at random from $\mathcal{M}$. For a randomized algorithm $f : \mathcal{X} \times \mathcal{R} \to \mathcal{Y}$, we write $y \xleftarrow{\$} f(x)$ for the process of executing f on x with randomness $r \xleftarrow{\$} \mathcal{R}$. $\Pr[E : X]$ denotes the probability that event E occurs over the randomness of variable X, typically used to express the advantage of an adversary winning some security game.

When we say an algorithm $\mathcal{A}$ is efficient, we mean that $\mathcal{A}$ is either a classical probabilistic polynomial-time (PPT) algorithm or a quantum polynomial-time (QPT) algorithm. A function $f : \mathbb{N} \to \mathbb{R}_{\geqslant 0}$ is said to be negligible if for all c, there exists λ_0 such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_0$, i.e., it decays faster than the inverse of any polynomial at infinity.

2.2 The NIST Security Level

In the NIST PQC standardization project, NIST asked submitters to categorize the instances of their submitted schemes into one of five security levels, which relate to the difficulty of breaking the symmetric schemes AES and SHA-2. We provide the definitions given by NIST in Table 1 [Nat16].

Table 1: NIST's categorization of security levels.

Level	Security description
1	At least as hard to break as AES128 (exhaustive key search)
2	At least as hard to break as SHA256 (collision search)
3	At least as hard to break as AES192 (exhaustive key search)
4	At least as hard to break as SHA384 (collision search)
5	At least as hard to break as AES256 (exhaustive key search)

NIST asked submitters to focus on levels 1–3, leaving levels 4 and 5 for high-security instances. As the project progressed, most submissions settled on providing parameter sets for security levels 1, 3, and 5. For ML-KEM, parameter sets have been defined at security levels 1, 3, and 5. These can be compared to AES128, AES192, and AES256. Compared to elliptic-curve-based schemes, these levels should (roughly) be equivalent to the security given by the NIST curves P-256, P-384, and P-521 against classical (non-quantum) adversaries.

It is worth noting that choosing an appropriate security level means balancing security with operational costs. ML-KEM at security levels 3 and 5 require (significantly) more computation time and memory, and has larger messages than security level 1. Based on the current state of the art

in cryptanalysis, which this report will cover in detail, breaking ML-KEM at level 1 should be out of reach of any conceivable adversary. Higher security levels can, however, provide some insurance against advances in cryptanalysis.

2.3 Lattices

2.3.1 Cyclotomic Rings

Let n be a power-of-two integer and q a prime. Let $R = \mathbb{Z}[X]/(X^n + 1)$ be the cyclotomic ring of degree n and denote $R_q = R \bmod q$. For ML-KEM we consider $n = 256$ unless otherwise stated.

2.3.2 (Module)-Lattices

For a matrix $\mathbf{B} \in R^{k\times\ell}$ with linearly independent columns, we define the (right) R-submodule

$$L := \mathbf{B} \cdot R^\ell = \{\mathbf{B} \cdot x : x \in R^\ell\} \subseteq R^k$$

and call L an (integer) R-module lattice of rank ℓ, with $\mathbf{B}$ a basis of L. Intuitively, L consists of all R-linear combinations of the columns of $\mathbf{B}$, so changing the basis $\mathbf{B}$ changes the way we describe vectors in L but not the underlying set.

When $qR^k \subseteq L \subseteq R^k$ for some integer modulus $q \geqslant 2$, we call L a *q-ary lattice*. In this case we can identify vectors in L with elements of R_q^k by reducing their coefficients modulo q. We will often tacitly work with this reduction and think of vectors of L as living in R_q^k.

In the special case $n = 1$ we have $R = \mathbb{Z}$, and we recover classical (unstructured) integer lattices $L \subseteq \mathbb{Z}^k$. For a basis $\mathbf{B}$ of such a lattice L, we write

$$\det(L) := \left|\sqrt{\det(\mathbf{B}^\top \cdot \mathbf{B})}\right|$$

for the lattice determinant (the Euclidean volume of a fundamental parallelepiped of L), and

$$\lambda_1(L) := \min_{\mathbf{y}\in L\setminus\{0\}} \|\mathbf{y}\|$$

for the *first minimum*, i.e., the length of a shortest nonzero lattice vector.

Given a basis $\mathbf{B} \in R^{k\times\ell}$ and its associated anti-circulant matrix $\mathbf{M} \in \mathbb{Z}^{nk\times n\ell}$ (obtained via the coefficient embedding $R^k \hookrightarrow \mathbb{Z}^{nk}$), we associate to the R-module lattice $L := \mathbf{B} \cdot R^\ell$ of rank ℓ the unstructured integer lattice

$$L' := \mathbf{M} \cdot \mathbb{Z}^{n\ell} \subseteq \mathbb{Z}^{nk}$$

of rank $n\ell$. We then *define* the determinant and first minimum of L via those of L' by setting

$$\det(L) := \det(L') \qquad \text{and} \qquad \lambda_1(L) := \lambda_1(L')\,.$$

To measure how "orthogonal" a basis of L' is, we will repeatedly use the Gram–Schmidt orthogonalization of the columns of $\mathbf{M}$. Writing $\mathbf{b}_1, \ldots, \mathbf{b}_{n\ell}$ for these column vectors, their Gram–Schmidt orthogonalization is the sequence $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_{n\ell}$ defined recursively by

$$\tilde{\mathbf{b}}_1 := \mathbf{b}_1, \qquad \tilde{\mathbf{b}}_i := \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\tilde{\mathbf{b}}_j \quad \text{for } i \geqslant 2,$$

where

$$\mu_{i,j} := \frac{\mathbf{b}_i^\top \cdot \tilde{\mathbf{b}}_j}{\tilde{\mathbf{b}}_j^\top \cdot \tilde{\mathbf{b}}_j}.$$

By construction, the vectors $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_{n\ell}$ are pairwise orthogonal in $\mathbb{R}^{nk}$, and each $\mathbf{b}_i$ lies in the span of $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_i$. We write $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_{n\ell}$ for this Gram–Schmidt family throughout, and we will often use the norms $\|\tilde{\mathbf{b}}_i\|$ as a convenient quantitative measure of the quality of the basis.

2.3.3 Rounding

For an even (resp. odd) positive integer q, we define $x' = x \bmod^{\pm} q$ to be the unique element x' in the range $-\frac{q}{2} < x' \leqslant \frac{q}{2}$ (resp. $-\frac{q-1}{2} < x' \leqslant \frac{q-1}{2}$) such that $x' = x \bmod q$. For any positive integer q, we define $x' = x \bmod^{+} q$ to be the unique element x' in the range $0 \leqslant x' < q$ such that $x' = x \bmod q$. We simply write $x \bmod q$ when the representation is not important. Also, for an element $x \in \mathbb{Q}$, $\lceil x \rfloor$ denotes rounding to the nearest integer, where in case of a tie, we take the larger integer.

2.3.4 Sizes of Elements

For an element $x \in \mathbb{Z}_q$, we write $\|x\|_\infty$ to mean $|x \bmod^{\pm} q|$. Using this, we define the L_∞ and L_2 norms for

$$x(X) = \sum_{i=0}^{n-1} x_i X^i \in R_q$$

as

$$\|x\|_\infty = \max_i \|x_i\|_\infty, \quad \|x\| = \sqrt{\|x_0\|_\infty^2 + \cdots + \|x_{n-1}\|_\infty^2}.$$

Similarly, for vectors of polynomial ring elements $\mathbf{x} = (x_0, \ldots, x_{k-1}) \in R_q^k$, we define

$$\|\mathbf{x}\|_\infty = \max_i \|x_i\|_\infty, \quad \|\mathbf{x}\| = \sqrt{\|x_0\|^2 + \cdots + \|x_{k-1}\|^2}.$$

2.3.5 Compression and Decompression

We define the following compression and decompression algorithms for positive integers d and q such that $d < \lceil \log_2(q) \rceil$:

$$\begin{aligned} \mathsf{Compress}_d : \mathbb{Z}_q &\longrightarrow \mathbb{Z}_{2^d} \\ x &\longmapsto \left\lceil \frac{2^d}{q} \cdot x \right\rfloor \bmod^{+} 2^d. \end{aligned} \tag{1}$$

$$\begin{aligned} \mathsf{Decompress}_d : \mathbb{Z}_{2^d} &\longrightarrow \mathbb{Z}_q \\ y &\longmapsto \left\lceil \frac{q}{2^d} \cdot y \right\rfloor. \end{aligned} \tag{2}$$

For these functions, we have the following:

Lemma 2.1. *Let d and q be positive integers such that $d < \lceil \log_2(q) \rceil$. Then, for any $x \in \mathbb{Z}_q$, we have*

$$\left| x' - x \bmod^{\pm} q \right| \leqslant \left\lceil \frac{q}{2^{d+1}} \right\rfloor,$$

where $x' = \mathsf{Decompress}_d(\mathsf{Compress}_d(x))$.

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

2.3.6 Hardness Assumption

The security of ML-KEM is based on the so-called Module Learning with Errors problem (MLWE).

Informal description of LWE. The (module) Learning with Errors (LWE/MLWE) problems ask us to recover a hidden linear relation from many *noisy* linear equations. In the simplest LWE setting, there is a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and one is given samples of the form

$$(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q, \qquad b_i = \mathbf{a}_i^\top \cdot \mathbf{s} + e_i \bmod q,$$

where each $\mathbf{a}_i$ is uniform in $\mathbb{Z}_q^n$ and each e_i is a small "error" drawn from a narrow distribution [Reg05]. The goal is either to distinguish such samples from uniformly random pairs, or to recover the secret $\mathbf{s}$. Thus, LWE can be viewed as the problem of solving a linear system whose right-hand side has been perturbed by small but unknown noise. Conceptually, the problem is hard because the noise forbids to use algebraci algorithms such as Gaussian elimination. In the modular setting small errors can wrap around modulo q and make the resulting distribution of the b_i statistically very close to uniform.

Module Learning with error. The Module Learning with Errors (MLWE) problem is a structured variant of LWE in which the vectors and matrices live in the polynomial residue ring $R_q = \mathbb{Z}_q[X]/(X^n+1)$ and one works with k-dimensional modules over R_q instead of plain $\mathbb{Z}_q$-vector spaces [LS15]. This allows more compact keys and efficient use of the number-theoretic transform, while preserving essentially the same hardness guarantees via reductions from worst-case module lattice problems. From a lattice perspective, MLWE interpolates between the highly structured ideal lattices arising from RLWE and the general lattices underlying plain LWE [LS15].

Definition 2.2 (MLWE). *Let m, k, q be integers and let χ be a probability distribution over R_q. The advantage of an adversary $\mathcal{A}$ against the* Module Learning with Errors $\mathsf{MLWE}_{q,m,k,\chi}$ *problem is defined as:*

$$\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{A}}(1^\lambda) = \left| \Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{A}\cdot\mathbf{s}+\mathbf{x}) = 1\right] - \Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{b}) = 1\right] \right|,$$

where $(\mathbf{A}, \mathbf{b}, \mathbf{s}, \mathbf{x}) \xleftarrow{\$} R_q^{m\times k} \times R_q^m \times \chi^k \times \chi^m$. The $\mathsf{MLWE}_{q,m,k,\chi}$ assumption states that any efficient adversary $\mathcal{A}$ has negligible advantage.

We consider a slight variant where the "secret" $\mathbf{s}$ and "noise" $\mathbf{x}$ are sampled from different distributions χ and χ', respectively. We denote this as the $\mathsf{MLWE}_{q,m,k,\chi,\chi'}$ assumption.

The parameters q, m, k are referred to as the modulus, number of samples, and the dimension (or module rank) of the MLWE problem. In the concrete instantiation underlying ML-KEM, one works over the ring $R = \mathbb{Z}[X]/(X^n+1)$ with $n = 256$ and $q = 3329$, and chooses $k \in \{2, 3, 4\}$ corresponding to the three parameter sets ML-KEM-512, ML-KEM-768, and ML-KEM-1024 [Nat24b]. The noise

and secret distributions are discrete, bounded, and very narrow: ML-KEM samples coefficients from centered binomial distributions B_{η_1}, B_{η_2} with small parameters $\eta_1, \eta_2 \in \{2, 3\}$.

Short secret and entropic LWE. When the secret is sampled from a small error distribution χ then this variant is also known as the small-secret MLWE assumption. This variant, used by ML-KEM, allows for some compression techniques (e.g., more compact public keys and seeds instead of explicit matrices) and is essentially equivalent to the MLWE assumption with a uniform secret and an increased number $m' = m + k$ of samples [App+09]. More generally, a number of works show that changing the secret distribution to another sufficiently entropic or short distribution does not substantially affect hardness: this is well understood for (ring-)LWE with secrets drawn from error-like or binary distributions [Bra+13; Mic18], and has recently been extended to the module setting for binary secrets as well [Bou+21; Bou+22]. These results support the use of small or bounded secrets as in ML-KEM.

Reductions. Definition 2.2 states the *decision* variant of MLWE. The *search* variant, given $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{x})$ sampled as in Definition 2.2, asks to recover $\mathbf{s}$. Up to differences in the number of samples m, search and decision are equivalent: this was proved for LWE by Regev [Reg05] and extended to more general settings [Pei09; App+09; MM11; MP12; Bra+13], including RLWE and MLWE[LPR10; LS15]. The main confidence in the hardness of the LWE problem comes from the existence of worst-case to average-case reductions that show that if one can solve LWE, then one can solve certain worst-case general lattice problems. Regev [Reg05] gives a *quantum* reduction from SIVP and GapSVP in dimension n with polynomial approximation to average-case LWE with polynomial modulus, while classical reductions from GapSVP to LWE use either exponential modulus [Pei09] or dimension n^2 with polynomial modulus [Bra+13]. These results extend to structured variants, relating them to the worst-case ideal/module lattice problems. Langlois–Stehlé [LS15] show that (search or decision) MLWE over a cyclotomic ring R and rank k is as hard as approximating *module*-SIVP and *module*-GapSVP on R-modules of rank k, up to polynomial factors; later work gives essentially tight converse reductions, so MLWE is asymptotically as hard as these module problems.

Concrete hardness and best known attacks. For the concrete parameters of ML-KEM, hardness is assessed by comparing against the cost of the best known lattice-based attacks, most notably the primal, dual, and hybrid variants of lattice-reduction attacks [dv25]. We provide a systematic and up-to-date presentation of these techniques in Section 5.

Error distribution. The hardness of the $\mathsf{MLWE}_{q,m,k,\chi}$ problem depends on the error distribution χ. Classically, the coefficients of χ follow continuous [Reg05] or discrete [GKV10] Gaussian distributions, which is convenient for theory. For efficiency, other distributions are used in practice (bounded small support, efficient constant-time sampling, or better bounds/analyses). Worst-case to average-case reductions extend to this regime, e.g., to uniform errors on small or even binary support [MP13; DM13], or to any distribution with sufficient min-entropy [Bra+13]. These work only under strong bounds on the number of samples, which is inherent given efficient attacks with many samples [AG11]. State-of-the-art cryptanalysis, see Section 5.3.5, similarly suggests that in the bounded-samples regime the exact distribution matters little as long as it has enough entropy and produces secret and error vectors of comparable size.

For ML-KEM, the secret and error follow centered binomial distributions with parameters η_1 and η_2, respectively. For ML-KEM-512 we have $\eta_2 < \eta_1$, so the error is smaller than the secret, but the compression function adds deterministic errors that heuristically rebalance this. This can be viewed as an LWE problem combined with a *Learning With Rounding* (LWR) problem. Heuristically, current cryptanalysis suggests that this extra rounding adds some security, but it is usually ignored in reductions; we will discuss its impact on concrete security in Section 5.1.

2.4 Cryptographic Primitives

2.4.1 Public Key Encryption

We review the syntax and minimal definition of a public key encryption (PKE) scheme. We only define IND-CPA security for PKE below since we will not require the stronger IND-CCA security for this report.

Definition 2.3 (PKE). *A public key encryption* (PKE) *scheme with message space $\mathcal{M}$ consists of the following three PPT algorithms:*

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$**:** *The key generation algorithm takes the security parameter as input and outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Encrypt}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$**:** *The (possibly randomized) encryption algorithm takes a public key* pk *and message* $\mathsf{m} \in \mathcal{M}$ *as input and outputs a ciphertext* ct.

$\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}$**:** *The (deterministic) decryption algorithm takes a secret key* sk *and a ciphertext* ct *as input and outputs a message* $\mathsf{m} \in \mathcal{M}$.

It is common in lattice-based cryptography to define correctness to hold with all but negligible probability. While it is possible to construct a scheme with perfect correctness as is standard in classical cryptography, this incurs an efficiency loss which we typically want to avoid. Moreover, as we see later in Section 3.3, computing the exact correctness failure rate in lattice-based cryptography can be challenging in practice, as it requires precise control over how "noise" accumulates.

Definition 2.4 (δ-Correctness). *We say a* PKE *is δ-correct if for all $\lambda \in \mathbb{N}$ we have*

$$\mathbb{E}_{(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)} \left[\max_{\mathsf{m} \in \mathcal{M}} \Pr\left[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) = \mathsf{m} : \mathsf{ct} \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, \mathsf{m}) \right] \right] \geqslant 1 - \delta,$$

where the probability is taken over the randomness of the encryption algorithm.

The following states that even if an adversary gets to choose two messages, the ciphertext does not reveal which message it encrypts. Since the adversary is not allowed access to the decryption oracle, this is sometimes called *passive* security.

Definition 2.5 (IND-CPA Security). *Let Π be a* PKE *scheme. For any adversary $\mathcal{A}$, we define the advantage for indistinguishability under chosen-plaintext attack (*IND-CPA*) security as follows:*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(1^\lambda) := \left| \Pr\left[b = b' : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda); \\ b \xleftarrow{\$} \{0,1\}; \\ (\mathsf{m}_0, \mathsf{m}_1, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(\mathsf{pk}); \\ \mathsf{ct}^* \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, \mathsf{m}_b); \\ b' \xleftarrow{\$} \mathcal{A}(\mathsf{pk}, \mathsf{ct}^*, \mathsf{state}) \end{array} \right] - \frac{1}{2} \right|,$$

We say a PKE *scheme* Π *is* IND-CPA *secure if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(1^\lambda)$ *is negligible for any efficient adversary* $\mathcal{A}$*.*

2.4.2 Key Encapsulation Mechanisms

A key encapsulation mechanism (KEM) is a simplified PKE where the message is randomly sampled. It allows a sender (i.e., the one that generates the ciphertext) and a receiver (i.e., the one holding the decapsulation key) to agree on a so-called *session key*, which is typically a random bit string.

Below, in order to differentiate from PKE, we use terms such as encapsulation and decapsulation as opposed to encryption and decryption.

Definition 2.6 (KEM). *A key encapsulation mechanism* (KEM) *scheme with key space* $\mathcal{K}$ *consists of the following three PPT algorithms:*

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{ek}, \mathsf{dk})$**:** *The key generation algorithm takes the security parameter as input and outputs a pair of keys* $(\mathsf{ek}, \mathsf{dk})$*.*

$\mathsf{Encaps}(\mathsf{ek}) \to (\mathsf{k}, \mathsf{ct})$**:** *The encapsulation algorithm takes an encapsulation key* ek *as input and outputs a session key* $\mathsf{k} \in \mathcal{K}$ *and a ciphertext* ct*.*

$\mathsf{Decaps}(\mathsf{dk}, \mathsf{ct}) \to \mathsf{k}$**:** *The (deterministic) decapsulation algorithm takes a decapsulation key* dk *and a ciphertext* ct *as input and outputs a session key* $\mathsf{k} \in \mathcal{K}$*.*

Similarly to a PKE, we consider a correctness definition where some decapsulation failure is allowed.

Definition 2.7 (δ-Correctness). *We say a* KEM *is* δ*-correct if for all* $\lambda \in \mathbb{N}$ *we have*

$$\Pr\left[\mathsf{Decaps}(\mathsf{dk}, \mathsf{ct}) = \mathsf{k} : \begin{array}{c}(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda); \\ (\mathsf{k}, \mathsf{ct}) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{ek})\end{array}\right] \geqslant 1 - \delta.$$

The following is the de-facto security standard for KEMs. It is defined analogously to IND-CPA security, except that the adversary is given access to a decapsulation oracle. On input a ciphertext, the oracle runs the decapsulation algorithm and gives back the output. In order not to trivialize the game, the decapsulation oracle will not take the challenge ciphertext as input. Since the adversary is allowed access to the decapsulation oracle, this is sometimes called *active* security.

Definition 2.8 (IND-CCA Security). *Let* Π *be a* KEM *scheme. For any adversary* $\mathcal{A}$*, we define the advantage for indistinguishability under chosen-ciphertext attack (*IND-CCA*) security as follows:*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\Pi,\mathcal{A}}(1^\lambda) := \left|\Pr\left[b = b' : \begin{array}{c}(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda); \\ b \xleftarrow{\$} \{0,1\}; \\ (\mathsf{k}_0^*, \mathsf{ct}^*) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{ek}); \\ \mathsf{k}_1^* \xleftarrow{\$} \mathcal{K}; \\ b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Decaps}}(\cdot)}(\mathsf{ek}, \mathsf{k}_b^*, \mathsf{ct}^*)\end{array}\right] - \frac{1}{2}\right|,$$

where $\mathcal{O}_{\mathsf{Decaps}}(\cdot)$ *is an oracle that takes as input a ciphertext* ct *and outputs* $\mathsf{Decaps}(\mathsf{dk}, \mathsf{ct})$ *if and only if* $\mathsf{ct} \neq \mathsf{ct}^*$*. We say a* KEM *scheme* Π *is* IND-CCA *secure if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\Pi,\mathcal{A}}(1^\lambda)$ *is negligible for any efficient adversary* $\mathcal{A}$*.*

In the above, if security only holds against an adversary $\mathcal{A}$ that is not given access to the decapsulation oracle $\mathcal{O}_{\mathsf{Decaps}}$, it is called indistinguishability under chosen-plaintext attack (IND-CPA) secure.

2.4.3 Pseudorandom Functions

Lastly, we recall the definition of a pseudorandom function.

Definition 2.9 (Pseudorandom Function). *Let* $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *be an efficiently computable function. For any adversary* $\mathcal{A}$*, we define the advantage for the* pseudorandomness *of the* PRF *as follows:*

$$\mathsf{Adv}^{\mathsf{rand}}_{\mathsf{PRF},\mathcal{A}}(1^\lambda) := \left|\Pr\left[\mathcal{A}^{\mathsf{RF}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{PRF}(\mathsf{k},\cdot)}(1^\lambda) = 1\right]\right|,$$

where RF *is a random function sampled uniformly from the set of all functions from* $\mathcal{X}$ *to* $\mathcal{Y}$*, and* $\mathsf{k} \xleftarrow{\$} \mathcal{K}$*. We say a* PRF *is pseudorandom if the advantage* $\mathsf{Adv}^{\mathsf{rand}}_{\mathsf{PRF},\mathcal{A}}(1^\lambda)$ *is negligible for any efficient adversary* $\mathcal{A}$*.*

2.5 The Random Oracle Model

The security of many practical cryptographic primitives and protocols is analyzed in the *random oracle model* (ROM) [BR93]. This is an idealized framework used in cryptography to analyze the security of primitives and protocols, particularly those that rely on *cryptographic hash functions* (e.g., SHA-3). In this model, the hash function is replaced with a perfect random function that all entities, including adversaries, can only access as a black box through oracle queries. Concretely, if a hash function $\mathsf{H} : \{0,1\}^n \to \{0,1\}^m$ was used for a construction, this H will be replaced in the security proof by a function that is sampled uniformly at random from the set of all possible functions from $\{0,1\}^n$ to $\{0,1\}^m$. This abstracts our common belief that in practice, concrete cryptographic hash functions output random values, even if we know their full description.

Proving a scheme is secure in the ROM provides strong heuristic evidence that it will be secure in the real world when the oracle is instantiated with a well-designed cryptographic hash function (e.g., SHA-3). This is because the model captures the desired properties of a hash function, such as its output being unpredictable (pseudorandom) and deterministic. However, a proof in the ROM is not a guarantee of security in the standard model (where no such ideal oracles exist). Indeed, there are theoretical cryptographic schemes that are provably secure in the ROM but are demonstrably insecure no matter which real hash function is used to implement them. Despite this limitation though, the ROM remains an invaluable tool for designing and gaining confidence in the security of practical cryptographic schemes such as ML-KEM.

3 Overview of ML-KEM

In this section, we provide a minimal overview of ML-KEM necessary to understand its design principles. In particular, we ignore the implementation specifics such as how ring elements are represented as byte arrays or how ring elements are multiplied using so-called number-theoretic transforms (NTTs). These details do not impact its theoretical guarantees such as correctness, provable security, and hardness of the underlying mathematical problems. Implementation details and implementation-specific security are discussed in Section 6.

3.1 Design Principle

ML-KEM is based on the lattice-based cryptosystem from [LPR10; LP11]. While [LPR10] uses the highly structured *ring* LWE (RLWE) problem, leading to for instance NEWHOPE [Alk+16], [LP11] uses the unstructured standard LWE problem, leading to for instance Frodo [Bos+16]. The design of ML-KEM sits between these, using the somewhat structured *module* LWE (MLWE) problem.

The core cryptosystem. Let us go over the design principle of ML-KEM in a bottom-up manner. At its core, the encapsulation key is simply an MLWE instance:

$$(\mathsf{ek}, \mathsf{dk}) = \Big(\big(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}\big), \quad \mathbf{s} \Big) \in \Big(R_q^{k \times k} \times R_q^k \Big) \times R_q^k.$$

In general, $\mathbf{A}$ can be a non-square matrix but we only consider a square matrix for simplicity; ML-KEM also uses a square matrix.

A ciphertext is another MLWE instance:

$$\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1) = \big(\mathbf{A}^\top \cdot \mathbf{y} + \mathbf{e}_1, \quad \mathbf{b}^\top \cdot \mathbf{y} + e_2 + \mathsf{k} \cdot \lceil q/2 \rfloor\big) \in R_q^k \times R_q.$$

Here, the noises $(\mathbf{s}, \mathbf{e}, \mathbf{y}, \mathbf{e}_1, e_2)$ must be sampled from an appropriate distribution so that the MLWE problem is hard to solve, and the random key $\mathsf{k} \xleftarrow{\$} \{0,1\}^n$ is encoded as an element in the polynomial ring R_q whose i-th coefficient is k_i. Notice that since $\mathbf{b}, \mathsf{ct}_0$, and ct_1 are all MLWE instances, they are (informally) indistinguishable from random elements over R_q, establishing that the above construction achieves IND-CPA security.

To decapsulate, we first perform the following computation:

$$\begin{aligned} \mathsf{ct}_1 - \mathbf{s}^\top \cdot \mathsf{ct}_0 &= \big(\mathbf{b}^\top \cdot \mathbf{y} + e_2 + \mathsf{k} \cdot \lceil q/2 \rfloor\big) - \mathbf{s}^\top \cdot \big(\mathbf{A}^\top \cdot \mathbf{y} + \mathbf{e}_1\big) \\ &= \mathsf{k} \cdot \lceil q/2 \rfloor + \mathbf{e}^\top \cdot \mathbf{y} - \mathbf{s}^\top \cdot \mathbf{e}_1 + e_2 \mod q. \end{aligned}$$

Assuming the noises are small enough (i.e., $\|\mathbf{e}^\top \cdot \mathbf{y} - \mathbf{s}^\top \cdot \mathbf{e}_1 + e_2\|_\infty < q/4$), we can uniquely decode k by rounding each coefficient of the ring element $w = \mathsf{ct}_1 - \mathbf{s}^\top \cdot \mathsf{ct}_0$. That is, if the i-th coefficient of w is closer to 0 in absolute value compared to $\lceil q/2 \rfloor$, we decode to $\mathsf{k}_i = 0$, and otherwise we decode to $\mathsf{k}_i = 1$. Looking ahead, this assumption on the noise being small requires some care.

Notice that in the above, we obtain a KEM based on the unstructured LWE problem by setting $n = 1$, that is, $R_q = \mathbb{Z}_q$. While this results in a scheme based on a very conservative hardness assumption, the downside is that we can only encapsulate a one-bit key $\mathsf{k} \in \{0,1\}$. In contrast,

we obtain a KEM based on the highly structured RLWE problem by setting $k = 1$ and $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. We can encapsulate an n-bit key in one shot now but will have to rely on a more aggressive hardness assumption than the standard LWE assumption. Moreover, as n is required to be a power-of-two for implementation efficiency (i.e., allowing the use of NTT-based multiplication), the parameter choice for RLWE is quite sparse and hard to tune in practice.

The MLWE problem allows us to hit the right balance between these two cryptosystems lying at the opposite sides of the spectrum by tuning the values of n and k. In the specific case of the MLWE parameters used in ML-KEM, performance turns out to be very similar to the scheme based on RLWE since only a key of fixed size of 256 bits is necessary. This is because a 256-bit key will be used as a symmetric key to encrypt larger messages via the KEM-DEM framework (cf. [CS03]). Importantly, this allows ML-KEM to fix n to be 256 for all NIST security levels (cf. Table 3).

Upgrading it to IND-CCA security. As already mentioned, the above cryptosystem only offers weak IND-CPA security. Indeed, if an adversary is given access to a decapsulation oracle, it can trivially learn the key by submitting a ciphertext ct such that $\mathsf{ct} = \mathsf{ct}^* + (0, \Delta)$ for a small non-zero noise $\Delta \in R_q$, where ct^* is the challenge ciphertext.

The Fujisaki-Okamoto (FO) transform [FO99; FO13; HHK17] is used to upgrade such a weakly secure KEM to an IND-CCA-secure one. While there are numerous variations of the FO transform, we only informally explain one variant below. We will highlight the core parts of the FO transform that differ between variations, and provide the concrete FO transform used by ML-KEM in Section 3.2.

For convenience, let us view the above IND-CPA secure KEM as a PKE scheme, and explain how the FO transform upgrades this IND-CPA secure PKE into the desired IND-CCA secure KEM. At a high level, the FO transform modifies the encryption and decryption algorithm of the IND-CPA secure PKE as follows:

Encapsulation: $\mathsf{Encaps}(\mathsf{ek})$

- Sample a random message $\mathsf{m} \xleftarrow{\$} \{0,1\}^n$ for the PKE scheme.
- Derive a key k and randomness r for the PKE.Encrypt algorithm via $(\mathsf{k}, r) = \mathsf{G}(\mathsf{m})$, where G is a hash function.

  //* *Variations** // We can derive the key and randomness in a different manner such as by $\mathsf{G}(\mathsf{m}, \mathsf{ek})$ or $\mathsf{G}(\mathsf{m}, \mathsf{H}(\mathsf{ek}))$, where H is another hash function. This provides slightly stronger security than IND-CCA security and will be discussed in Section 4. Indeed, ML-KEM performs $\mathsf{G}(\mathsf{m}, \mathsf{H}(\mathsf{ek}))$. Another variation is to first derive only the randomness $r = \mathsf{G}(\mathsf{m})$, compute the PKE ciphertext ct, and then derive the final key as $\mathsf{k} = \mathsf{G}(\mathsf{m}, \mathsf{ct})$.
- Run $\mathsf{ct} = \mathsf{PKE.Encrypt}(\mathsf{pk}, \mathsf{m}; r)$, where ek is defined as the public key pk of the PKE scheme.
- Output $(\mathsf{k}, \mathsf{ct})$.

Decapsulation: $\mathsf{Decaps}(\mathsf{dk}, \mathsf{ct})$

- Run $\mathsf{m}' = \mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct})$, where dk is defined as the secret key sk of the PKE scheme.
- Derive $(\mathsf{k}, r') = \mathsf{G}(\mathsf{m}')$.
- Re-encrypt and check if $\mathsf{PKE.Encrypt}(\mathsf{pk}, \mathsf{m}'; r') = \mathsf{ct}$.

- If the check passes, output k.

  //* *Variations** // We intentionally keep it agnostic in the case the check does *not* pass. We can output a special symbol $\perp$ indicating decapsulation failure, leading to the so-called *explicit* rejection approach. In contrast, we can output a random key k so as not to indicate whether a decapsulation failure occurred, leading to the so-called *implicit* rejection approach. ML-KEM uses implicit rejection.

Regardless of which variations are used, the core idea of the FO-transform is to recover the randomness used to run the encryption algorithm of the underlying IND-CPA secure PKE, and perform a re-encryption check during decapsulation. This intuitively guarantees that if an adversary outputs a ciphertext that decapsulates to a valid key, it must have known how the ciphertext was generated. To put it differently, we can prove IND-CCA security of the KEM relying only on the IND-CPA security of the PKE since the decapsulation oracle does not give away new information about the submitted ciphertext to the adversary. Formal details about the IND-CCA security of ML-KEM are given in Sections 4.1.1 and 4.1.2.

Notable optimizations. We wrap up this section by explaining some of the notable optimizations of ML-KEM to better understand the design choice made by ML-KEM in the next section. As mentioned earlier, we will not discuss implementation-specific optimizations such as how polynomial ring elements are stored in memory to minimize the number of NTT operations.

(i) Encapsulation key compression. ML-KEM adopts the approach taken by Alkim et al. [Alk+16] and generates the public matrix $\mathbf{A} \in R_q^{k \times k}$ in the encapsulation key ek via a hash function $\mathsf{H}_{\mathbf{A}}$. Concretely, $\mathbf{A}$ is replaced by a 256-bit seed $\rho \in \{0,1\}^{256}$ such that $\mathbf{A} = \mathsf{H}_{\mathbf{A}}(\rho)$. This effectively reduces the size of the encapsulation key almost for free.

(ii) Bit-dropping. As decryption only looks at the upper bits of the ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ (i.e., the lower bits are simply "noise"), we can safely round off the lower bits to compress the size of ct. This is a common technique used in LWE-based schemes (cf. [Pei09; PG14]). In ML-KEM, this is implemented using the functions $\mathsf{Compress}_d$ and $\mathsf{Decompress}_d$ from Section 2.3.5. Moreover, the rate of compression is different for the first and second halves of the ciphertext ct_0 and ct_1, respectively. These choices are made to balance between ciphertext size, security, and decapsulation failure probability.

It is worth mentioning that one can also perform bit-dropping on the vector $\mathbf{b}$ included in the encapsulation key. This has the effect of compressing the encapsulation key size while increasing the decapsulation failure probability. However, as analyzed in [Bos+18], it is unclear how to prove such a scheme secure from the MLWE problem, and this optimization is not implemented by ML-KEM.

(iii) Binomial noise. The LWE problem, including RLWE and MLWE, typically considers Gaussian noise, either rounded Gaussian [Reg05] or discrete Gaussian [Bra+13]. This is because these are the distributions for which we have theoretical guarantees on the hardness of the LWE problem under worst-case lattice problems. However, these noise distributions turn out to be either inefficient to implement (cf. [Bos+15]) or difficult to securely implement against side-channel attacks (cf. [Bru+16; Esp+17; PBY17]).

For practical lattice-based cryptosystems, state-of-the-art lattice cryptanalysis shows that the concrete hardness of the LWE problem does not depend on the exact distribution of the noise, but rather on the standard deviation of it (see Section 5 for the details). Therefore, a more pragmatic

option is to assume the hardness of the LWE problem with a noise distribution that can be easily, efficiently, and securely sampled. ML-KEM chooses the centered binomial distribution (cf. Section 2.3.6) used in [Alk+16].

Recall that ML-KEM relies on several MLWE instances, $\mathbf{b}, \mathsf{ct}_0$, and ct_1, where different rates of bit-dropping are performed on ct_0 and ct_1. As bit-dropping adds implicit noise created via the function $\mathsf{Compress}_d$, it can be interpreted as increasing the noise of the MLWE instances ct_0 and ct_1. ML-KEM utilizes this observation exclusively for the scheme with NIST level 1 security (i.e., ML-KEM-512) and samples the noise e_2 in ct_1 from a slightly smaller centered binomial distribution.

(iv) Decapsulation failure. Lastly, ML-KEM allows for a negligible decapsulation failure probability. While a parameter with a zero chance of decapsulation failure (i.e., perfect correctness) is appealing from a theoretical standpoint and mitigates specific attacks exploiting decapsulation failures, it will have a negative effect on either the concrete hardness of the MLWE problem (by significantly decreasing the noise) or the performance (by increasing the lattice dimension k to compensate for the loss in security).

In ML-KEM, the parameters are set such that the decapsulation failure probabilities are negligibly small; $2^{-138.8}, 2^{-164.8}$, and $2^{-174.8}$ for the NIST levels 1, 3, and 5 parameters. While there are attacks that can exploit these decapsulation failures, these are typically a much smaller threat than directly attacking ML-KEM. See Section 5.6 for more details.

3.2 Description of ML-KEM

Following the FIPS-203 publication [Nat24b], we first describe an IND-CPA secure PKE scheme called K-PKE (see Section 3.2.1, Figure 2), and then describe the IND-CCA secure ML-KEM obtained by performing the FO-transform on K-PKE (see Section 3.2.2, Figure 3). For reference, we provide an overview in Figure 1.

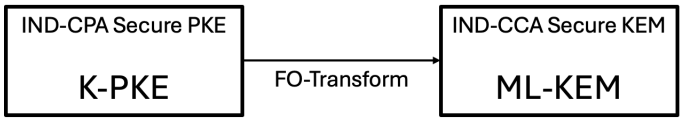

Figure 1: Constructing ML-KEM from K-PKE.

ML-KEM comes equipped with three parameter sets as given in Table 2. As explained in the previous section, the length of the shared key k is set to 32 bytes, or equivalently 256 bits, for all security levels. This means that the shared secret can be used directly in AES-256, though note that the *security* level will match the ML-KEM parameter set used and not the security level provided by AES-256.

For each of the security levels, the concrete parameters used by ML-KEM (and the underlying K-PKE) are listed in Table 3. We provide a short rationale for how each parameter was chosen:

- n is chosen to be 256 across all security levels. This relates to the fact that we only need to encapsulate a shared key of size 256 bits. Smaller values of n will require encoding more bits into one polynomial coefficient, which requires lowering noise levels for correctness, and therefore lower security. Larger values of n make it hard to fine tune the other parameters as we require n to be a power-of-two for efficient implementation using NTT-based multiplication.

Table 2: Overview of the sizes (in bytes) of keys and ciphertexts of ML-KEM and its decapsulation failure rate. NIST levels 1, 3, and 5 security correspond to ML-KEM-512, ML-KEM-768, and ML-KEM-1024, respectively.

	encapsulation key (ek)	decapsulation key (dk)	ciphertext (ct)	shared key (k)	decapsulation failure rate δ
NIST-1	800	1632	768	32	$2^{-138.8}$
NIST-3	1184	2400	1088	32	$2^{-164.8}$
NIST-5	1568	3168	1568	32	$2^{-174.8}$

Table 3: Overview of the parameters used by ML-KEM (and K-PKE). " - " indicates that it takes the same value as the value in its left cell. NIST levels 1, 3, and 5 security correspond to ML-KEM-512, ML-KEM-768, and ML-KEM-1024, respectively.

Notations	Explanation	Concrete Parameters		
		NIST-1	NIST-3	NIST-5
q	Modulus size	3329	-	-
R_q	Polynomial ring $\mathbb{Z}[X]/(X^n+1)$ of degree n	256	-	-
k	Module rank	2	3	4
d_u	Amount of bit-dropping on the first half of ciphertext	10	-	11
d_v	Amount of bit-dropping on the second half of ciphertext	4	-	5
B_{η_1}	Centered binomial distribution with parameter η_1	3	2	-
B_{η_2}	Centered binomial distribution with parameter η_2	2	-	-

- q is chosen to be a small prime satisfying $n \mid (q-1)$. This is required to enable fast NTT-based multiplication. While $q = 257$ and 769 satisfy this condition, the next smallest prime 3329 was chosen since the former two primes are too small to achieve negligible decapsulation failure probability.
- k is chosen to fix the lattice dimension as a multiple of n. This defines the hardness of the underlying MLWE problem. Put differently, changing k is the main mechanism allowing us to tune ML-KEM to different security levels.
- The remaining parameters $(d_u, d_v, B_{\eta_1}, B_{\eta_2})$ were chosen to balance between security, ciphertext size, and decapsulation failure probability.

3.2.1 Algorithms for K-PKE

K-PKE is the IND-CPA secure PKE used by ML-KEM. As explained in Section 3.1, K-PKE is similar to the PKE scheme that was introduced in [LPR10; LP11]. The pseudocode for K-PKE is given in Figure 2. The internal functions used by K-PKE are listed in Table 4.

3.2.2 Algorithms for ML-KEM

The pseudocode for ML-KEM is given in Figure 3, where it runs K-PKE internally and performs the FO transform. The internal functions used by K-PKE are listed in Table 4. As we explained

Table 4: Overview of the functions used by ML-KEM and K-PKE. The functions above the dashed line are used internally in ML-KEM, while those below are used internally in K-PKE.

Functions	Explanation
G	A function $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{256} \times \{0,1\}^{256}$
J	A function $\mathsf{J} : \{0,1\}^* \to \{0,1\}^{256}$ for implicit rejection
H	A function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{256}$ to compress encapsulation key ek
$\mathsf{H}_{\mathbf{A}}$	A function $\mathsf{H}_{\mathbf{A}} : \{0,1\}^* \to R_q^{k \times k}$ to generate $\mathbf{A}$
$\mathsf{H}_{\mathbf{s},\mathbf{e}}$	A function $\mathsf{H}_{\mathbf{s},\mathbf{e}} : \{0,1\}^* \to R_q^k \times R_q^k$ to generate MLWE secrets $(\mathbf{s},\mathbf{e})$ from B_{η_1}
$\mathsf{H}_{\mathbf{y},\mathbf{e}_1,e_2}$	A function $\mathsf{H}_{\mathbf{y},\mathbf{e}_1,e_2} : \{0,1\}^* \to R_q^k \times R_q^k \times R_q$ to generate MLWE secrets $\mathbf{y}$ from B_{η_1} and noises $(\mathbf{e}_1, e_2)$ from B_{η_2}

$\mathsf{K\text{-}PKE.KeyGen}(; d)$

Input: randomness $d \in \{0,1\}^{256}$
Output: encryption key $\mathsf{ek}_{\mathsf{PKE}} \in R_q^k \times \{0,1\}^{256}$
Output: decryption key $\mathsf{dk}_{\mathsf{PKE}} \in R_q^k$

$(\rho, \sigma) := \mathsf{G}(d)$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\rho)$ // $\mathbf{A} \in R_q^{k \times k}$
$(\mathbf{s}, \mathbf{e}) := \mathsf{H}_{\mathbf{s},\mathbf{e}}(\sigma)$ // Sample $\mathbf{s}, \mathbf{e} \in R_q^k$ from $B_{\eta_1}^k$
$\mathbf{t} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
$\mathsf{ek}_{\mathsf{PKE}} := (\mathbf{t} \| \rho)$ // append $\mathbf{A}$ seed
$\mathsf{dk}_{\mathsf{PKE}} := \mathbf{s}$
return $(\mathsf{ek}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}})$

$\mathsf{K\text{-}PKE.Encrypt}(\mathsf{ek}_{\mathsf{PKE}}, m; r)$

Input: encryption key $\mathsf{ek}_{\mathsf{PKE}} \in R_q^k \times \{0,1\}^{256}$
Input: message $m \in \{0,1\}^{256} \subset R_q$
Input: randomness $r \in \{0,1\}^{256}$
Output: ciphertext $\mathsf{ct} \in R_{2^{d_u}}^k \times R_{2^{d_v}}$

parse $(\mathbf{t} \| \rho) \leftarrow \mathsf{ek}_{\mathsf{PKE}}$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\rho)$ // regenerate $\mathbf{A} \in R_q^{k \times k}$
$\mathbf{y} := \mathsf{H}_{\mathbf{y},\mathbf{e}_1,e_2}(r)$ // Sample $\mathbf{y}, \mathbf{e}_1 \in R_q^k$ from $B_{\eta_1}^k$ // and $B_{\eta_2}^k$, respectively, and $e_2 \in R_q$ from B_{η_2}
$\mathbf{u} := \mathbf{A}^\top \cdot \mathbf{y} + \mathbf{e}_1$
$\mu := \mathsf{Decompress}_1(m)$
$v := \mathbf{t}^\top \cdot \mathbf{y} + e_2 + \mu$
$\mathsf{ct}_1 := \mathsf{Compress}_{d_u}(\mathbf{u})$
$\mathsf{ct}_2 := \mathsf{Compress}_{d_v}(v)$
return $\mathsf{ct} := (\mathsf{ct}_1 \| \mathsf{ct}_2)$

$\mathsf{K\text{-}PKE.Decrypt}(\mathsf{dk}_{\mathsf{PKE}}, \mathsf{ct})$

Input: decryption key $\mathsf{dk}_{\mathsf{PKE}} \in R_q^k$
Input: ciphertext $\mathsf{ct} \in R_{2^{d_u}}^k \times R_{2^{d_v}}$
Output: message $m \in \{0,1\}^{256} \subset R_q$

parse $(\mathsf{ct}_1 \| \mathsf{ct}_2) \leftarrow \mathsf{ct}$
$\mathbf{u}' := \mathsf{Decompress}_{d_u}(\mathsf{ct}_1)$
$v' := \mathsf{Decompress}_{d_v}(\mathsf{ct}_2)$
parse $\mathbf{s} \leftarrow \mathsf{dk}_{\mathsf{PKE}}$
$w := v' - \mathbf{s}^\top \cdot \mathbf{u}'$
$m := \mathsf{Compress}_1(w)$
return m

Figure 2: Simplified algorithms for the IND-CPA secure K-PKE: K-PKE.KeyGen, K-PKE.Encrypt, and K-PKE.Decrypt. Above, we implicitly assume $\mathsf{m} \in \{0,1\}^{256}$ is encoded as a polynomial over R_q. See Figure 8 for the concrete specification.

ML-KEM.KeyGen()

Output: encapsulation key $\mathsf{ek} \in R_q^k$

Output: decapsulation key $\mathsf{dk} \in (R_q^k)^2 \times (\{0,1\}^{256})^2$

1 : $d \xleftarrow{\$} \{0,1\}^{256}$

2 : $z \xleftarrow{\$} \{0,1\}^{256}$

3 : $(\mathsf{ek}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}}) \xleftarrow{\$} \mathsf{K\text{-}PKE.KeyGen}(\ ; d)$

// run key generation for K-PKE using randomness d

4 : $\mathsf{ek} := \mathsf{ek}_{\mathsf{PKE}}$

5 : $\mathsf{dk} := (\mathsf{dk}_{\mathsf{PKE}} \| \mathsf{ek} \| \mathsf{H}(\mathsf{ek}) \| z)$

6 : **return** $(\mathsf{ek}, \mathsf{dk})$

ML-KEM.Encaps(ek)

Input: encapsulation key $\mathsf{ek} \in R_q^k$

Output: shared secret key $\mathsf{k} \in \{0,1\}^{256}$

Output: ciphertext $\mathsf{ct} \in R_{2^{d_u}}^k \times R_{2^{d_v}}$

1 : $m \xleftarrow{\$} \{0,1\}^{256}$

2 : $(\mathsf{k}, r) := \mathsf{G}(m \| \mathsf{H}(\mathsf{ek}))$

3 : $\mathsf{ct} := \mathsf{K\text{-}PKE.Encrypt}(\mathsf{ek}, m; r)$

// encrypt m using K-PKE with randomness r

4 : **return** $(\mathsf{k}, \mathsf{ct})$

ML-KEM.Decaps(dk, ct)

Input: decapsulation key $\mathsf{dk} \in (R_q^k)^2 \times (\{0,1\}^{256})^2$

Input: ciphertext $\mathsf{ct} \in R_{2^{d_u}}^k \times R_{2^{d_v}}$

Output: shared secret key $\mathsf{k} \in \{0,1\}^{256}$

1 : **parse** $(\mathsf{dk}_{\mathsf{PKE}} \| \mathsf{ek} \| h \| z) \leftarrow \mathsf{dk}$

2 : $m' := \mathsf{K\text{-}PKE.Decrypt}(\mathsf{dk}_{\mathsf{PKE}}, \mathsf{ct})$

3 : $(\mathsf{k}', r') := \mathsf{G}(m' \| h)$

4 : $\bar{\mathsf{k}} := \mathsf{J}(z \| \mathsf{ct})$

5 : $\mathsf{ct}' := \mathsf{K\text{-}PKE.Encrypt}(\mathsf{ek}_{\mathsf{PKE}}, m'; r')$ // re-encrypt m' using derived randomness r'

6 : **if** $(\mathsf{ct} \neq \mathsf{ct}')$ **then**

7 : $\quad \mathsf{k}' \leftarrow \bar{\mathsf{k}}$

8 : **return** k'

Figure 3: Simplified algorithms for ML-KEM: ML-KEM.KeyGen, ML-KEM.Encaps, and ML-KEM.Decaps running K-PKE in Figure 2 as a sub-routine. See Figure 9 for the concrete specification.

in Section 3.1, there are small variations of the FO transform. Notice that ML-KEM.Encaps first hashes the encapsulation key ek by H, and then generates the session key and encryption randomness as $\mathsf{G}(m \| \mathsf{H}(\mathsf{ek}))$. Hashing the ek can be useful in situations where the message is not yet known as it can be precomputed; if not for the hash, then $\mathsf{G}(m \| \mathsf{ek})$ can only be computed once m is defined. Moreover, notice that $\mathsf{H}(\mathsf{ek})$ is included as part of the decapsulation key dk. This allows for a faster decapsulation process since the decryptor does not need to compute $\mathsf{H}(\mathsf{ek})$.

Another notable design choice is that ML-KEM performs *implicit* rejection. This is done by including a random seed $z \in \{0,1\}^{256}$ in the decapsulation key. In algorithm ML-KEM.Decaps, if the re-encryption check fails (cf. line 6), it outputs a "fake" session key $\bar{\mathsf{k}} := \mathsf{J}(z \| \mathsf{ct}) \in \{0,1\}^{256}$ as

opposed to a special symbol $\perp$ explicitly indicating a decapsulation failure. As z is hidden to the outside world, $\bar{\mathsf{k}}$ looks random assuming J is a pseudorandom function.

Remark 3.1 (Implicit vs Explicit Rejection). Alternatively to above, one can consider a variant of ML-KEM that performs *explicit* rejection which outputs a special symbol $\perp$ explicitly indicating a decapsulation failure. It is clear that if a KEM that performs explicit rejection is secure, it is also secure if it performs implicit rejection (cf. [Bin+19]). This is intuitive by noticing that we can replace the explicit rejection symbol $\perp$ by any value without deteriorating its security. On the other hand, while there are some implications, e.g., [HK25], we do not know whether the other direction holds unconditionally. Even to this date, there are only a handful of research on KEMs with explicit rejections [JZM19a; Don+22; HHM22; HM24; HK25]. This is one of the historical reasons why researchers focused on ML-KEM with implicit rejection; this was easier to formally prove the security compared to its explicitly rejecting counterpart. Moreover, from an implementation perspective, one can argue that implicit rejection is a better design as it "hides" whether a decapsulation failure occurred — this gives less attack surface to the adversary. However, we note that this argument is limited when using KEMs as a building block (e.g., key exchange). This is because if the protocol using the KEM prematurely terminates, this indirectly indicates that the KEM did not decapsulate properly.

3.3 Correctness

The correctness of ML-KEM has been theoretically and analytically analyzed in [Bos+18]. Theoretically, the decryption failure rate δ of ML-KEM is expressed as follows. Below, [Bos+18, Theorem 1] provides the decryption failure rate δ of the underlying K-PKE, and [HHK17] shows that the same δ is inherited to ML-KEM in the (quantum) random oracle model.

Theorem 3.2 ([Bos+18, Theorem 1] and [HHK17]). *Let $\mathbf{s}, \mathbf{e}, \mathbf{y} \in R_q^k$ be sampled from $B_{\eta_1}^k$, $\mathbf{e}_1 \in R_q^k$ from $B_{\eta_2}^k$, and $e_2 \in R_q$ from B_{η_2}. Also, let $\mathbf{c}_u \xleftarrow{\$} F_{d_u}^k$ and $c_v \xleftarrow{\$} F_{d_v}$ be distributed according to the distribution F_d defined as follows:*

Let F_d be the following distribution:

- *Choose uniformly random $y \xleftarrow{\$} R$.*
- *Output $(y - \mathsf{Decompress}_d(\mathsf{Compress}_d(y))) \bmod^{\pm} q$.*

Denote the decryption failure rate δ as

$$\delta := \Pr\left[\|\mathbf{e}^\top \cdot \mathbf{y} + e_2 + c_v - \mathbf{s}^\top \cdot \mathbf{e}_1 - \mathbf{s}^\top \cdot \mathbf{c}_u\|_\infty \geqslant \lceil q/4 \rfloor\right].$$

Then ML-KEM is δ-correct.

Notice that the above provides only an asymptotic guarantee on the decryption failure rate δ, and it is non-trivial to exactly compute δ in a closed form. As such, [Bos+18] also provides a Python script (`Kyber.py`) to compute a tight upper bound on δ, available online at https://github.com/pq-crystals/security-estimates. The parameters in Table 2 correspond to those upper bounds.

Remark 3.3 (Heuristic Arguments). It is worth highlighting that the formal proof of Theorem 3.2 relies on a heuristic argument. More concretely, in the proof of [Bos+18, Theorem 1], they assume $\mathbf{u}$ and v generated in K-PKE.Encrypt are distributed uniformly random — this is reflected in the

distribution F_d in Theorem 3.2. While this is informally argued by relying on the hardness of the MLWE assumption, there is no formal reduction for this claim. This issue has been explicitly pointed out in [Alm+24; Kre24; Bar+25]. While we can formally prove ML-KEM to be correct assuming MLWE as shown in [Bar+25], this results in a significantly worse bound than the one computed heuristically. For instance, for ML-KEM with NIST security level 3 (i.e., ML-KEM-768), the heuristic decapsulation failure rate is $2^{-164.8}$, while the provable rate is 2^{-80} (see [Bar+25, Table 1] for more details). Formally proving the heuristic decapsulation failure rate for ML-KEM is considered an open problem.

4 Provable Security of ML-KEM

In this section, we review the theoretical results establishing IND-CCA security of ML-KEM, and survey results covering more advanced form of security such as anonymity and KEM binding properties.

4.1 IND-CCA Security

There have been more than a dozen of papers on the IND-CCA security of ML-KEM (or variants thereof). One of the main reasons for so many papers on this topic is due to the complication caused when proving IND-CCA security of ML-KEM in the *quantum* random oracle model (QROM) [Bon+11] — indeed, if we only consider *classical* results, then there are only a few papers we need to cover. QROM is a type of random oracle model (cf. Section 2.5) where the adversary is able to perform quantum queries to the random oracle. This corresponds to the real-world setting where a quantum adversary can run the cryptographic hash function such as SHA-3 in superposition. While many natural properties in the classical ROM are expected to hold in the QROM, there are contrived examples proving otherwise, e.g., [Bon+11; YZ21]. Considering that the main motivation of ML-KEM was to prepare a KEM secure against quantum adversaries, the (theoretical side of the) cryptographic community naturally tended towards proving the more ambitious IND-CCA security in the QROM.

The other reason is caused by the numerous variants of the FO transform as explained in Section 3.1. While these variants can be proven secure in a similar manner in the classical ROM, this is not the case in the QROM. As such, many incomparable proof techniques have been developed over the past years to overcome the difficulty arising in QROM proofs. Adding further to this complication is the fact that ML-KEM performs a different type of FO transform compared to CRYSTALS-KYBER [Ava+21], the KEM that was submitted to the NIST PQC project. As such, while there are papers explicitly explaining the security of CRYSTALS-KYBER, they do not translate directly to ML-KEM. This has lead non-experts unsure of what the exact reference establishing IND-CCA security of ML-KEM is.

Below, we first review the classical result on IND-CCA security and then review the more nuanced post-quantum results.

4.1.1 In the Classical ROM

For any security proofs on the FO transform, be it in the classical or quantum ROM, we first need to establish the security of the base PKE scheme K-PKE. The following theorem is a refinement of [Ava+21, Theorem 1] which holds both in the classical and quantum ROM. We note that we can rely on the result from [Ava+21] since while the FO transform used by ML-KEM and CRYSTALS-KYBER differ, the underlying K-PKE is identical.

Theorem 4.1 (IND-CPA Security of K-PKE, [Ava+21, Theorem 1]). *Assume* $\mathsf{H}_{\mathbf{A}}$ *is modeled as a random oracle and* $\mathsf{H}_{\mathbf{s},\mathbf{e}}$ *and* $\mathsf{H}_{\mathbf{y},\mathbf{e}_1,e_2}$ *are pseudorandom functions. Then, for any adversary* $\mathcal{A}$ *against the* IND-CPA *security of* K-PKE *as in Figure 2, there exist adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *against the* $\mathsf{MLWE}_{q,k,k,B_{\eta_1}}$ *and* $\mathsf{MLWE}_{q,k+1,k,B_{\eta_1},B_{\eta_2}}$ *problems, respectively, and an adversary* $\mathcal{C}$ *against the pseudorandomness of* $\mathsf{H}_{\mathbf{s},\mathbf{e}}$ *and* $\mathsf{H}_{\mathbf{y},\mathbf{e}_1,e_2}$ *such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{K\text{-}PKE},\mathcal{A}}(1^\lambda) \leqslant \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2}(1^\lambda) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{C}}(1^\lambda).$$

Above, the first $\mathsf{MLWE}_{q,k,k,B_{\eta_1}}$ (resp. second $\mathsf{MLWE}_{q,k+1,k,B_{\eta_1},B_{\eta_2}}$) assumption in the theorem statement guarantees that the public key (resp. ciphertext) is pseudorandom. Moreover, $\mathsf{H}_{\mathbf{A}}$ is

required to be modeled as a random oracle since the reduction needs to program $\mathsf{H}_{\mathbf{A}}$ to output the MLWE challenge matrix $\mathbf{A} \in R_q^{k \times k}$ on input the public seed ρ; this can be done efficiently both in the classical and quantum ROM.

The next step is identifying the specific type of FO-transform used by ML-KEM. Once this is done, we can invoke the correct theorem establishing the IND-CCA security of the given FO-transformed KEM. As discussed in [MX23], while ML-KEM relies on a slightly different description, it is syntactically equivalent to the standard $\mathsf{FO}_m^{\not\perp}$-transform in [HHK17]. More concretely, we have the following:

- The FO-transform used by ML-KEM uses a single hash function G to compute both the session key k and the encryption randomness r. In contrast, the $\mathsf{FO}_m^{\not\perp}$-transform in [HHK17] uses two separate hash functions. However, these two computations are equivalent when the corresponding hash functions are modeled as independent random oracles with appropriate outputs lengths.
- Another difference is that the FO-transform used by ML-KEM uses the hash $\mathsf{H}(\mathsf{ek})$ to compute (k, r). In contrast, the $\mathsf{FO}_m^{\not\perp}$-transform ignores this input. Since the IND-CCA security notion only assumes a single-user setting (rather than a multi-user setting where the game considers many encapsulation keys ek), this difference can be trivially incorporated into the proof of the $\mathsf{FO}_m^{\not\perp}$-transform KEM without any notable changes.

To summarize, we can rely on the results in [HHK17; Hö20] regarding the $\mathsf{FO}_m^{\not\perp}$-transform to establish the IND-CCA security of ML-KEM in the classical ROM. Formally, we have the following theorem. Here, we note [Hö20] is a refined version of [HHK17].

Theorem 4.2 (IND-CCA Security of ML-KEM in Classical ROM, [HHK17; Hö20]). *Assume* K-PKE *is δ-correct and* IND-CPA *secure. Then, for any adversary $\mathcal{A}$ against the* IND-CCA *security of* ML-KEM *that makes at most q_{RO} classical queries to the random oracle, there exists an adversary $\mathcal{B}$ against the* IND-CPA *security of* K-PKE *such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{ML\text{-}KEM},\mathcal{A}}(1^\lambda) \leqslant 3 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{K\text{-}PKE},\mathcal{B}}(1^\lambda) + (q_{\mathsf{RO}} + 1) \cdot \delta + \frac{q_{\mathsf{RO}}}{2^{256}}.$$

Notice the result is *tight* in the sense that the IND-CCA security of ML-KEM is tightly bounded by the IND-CPA security of K-PKE. I.e., if there exists an adversary that breaks the IND-CCA security of ML-KEM with advantage ϵ, then there exists another adversary that breaks the IND-CPA security of K-PKE with a similar advantage. As we see in the next section, this is no longer the case in QROM.

Further notice that the decryption failure rate δ has an explicit implication in the IND-CCA security guarantee. Put differently, while $\delta \approx 2^{64}$ may be an acceptable decryption failure rate from a usability standpoint, the above theorem would not establish IND-CCA security anymore since the number of random oracle queries q_{RO} is typically expected to be larger than 2^{64}. That is, we arrive at a trivial upper bound $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{ML\text{-}KEM},\mathcal{A}}(1^\lambda) \leqslant 1$. In fact, this is not an artifact of the security proof, but as we will see in Section 5.6, there is a concrete attack exploiting such high decryption failure rate.

4.1.2 In the Quantum ROM

As mentioned above, there have been many numerous works on the IND-CCA security of the FO-transform in the QROM, e.g., [TU16; SXY18; Jia+18; Höv+20; Zha19; JZM19a; JZM19b; Bin+19; Kuc+20; Don+22; HHM22; Che+23; HM24; GLX24; HK25]. We do not plan to go over the details of these works, and refer to survey papers such as [ZLJ25] for a general overview on the topic.

At a high level, the reasons why there are so many different papers on this topic can be summarized as follows:

(i) Tightness of the security reduction. As we saw in Theorem 4.2 in the classical ROM, we know that ML-KEM is as secure as K-PKE with almost no reduction loss. Since K-PKE is also as secure as MLWE (cf. Theorem 4.1), this means we can simply focus on how hard the MLWE problem is when assessing the hardness of ML-KEM.

While this is something we would like to have in the quantum setting, we currently do not know how to do this. For instance, in some known proofs, the upper bound for $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{ML\text{-}KEM},\mathcal{A}}(1^\lambda)$ may look something like $\sqrt{q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{K\text{-}PKE},\mathcal{B}}(1^\lambda)} + q_{\mathsf{RO}} \cdot \sqrt{\delta}$, e.g., [JZM19b]. More concretely, even if there exists an adversary that breaks the IND-CCA security of ML-KEM with probability ϵ, we can only assume an adversary that breaks the IND-CPA security of K-PKE (and hence break the MLWE problem) with probability $\epsilon^2/q_{\mathsf{RO}}$. For a small (yet non-negligible) ϵ and large q_{RO}, this is far less optimal compared to the guarantees we get from the classical proofs, and there have been numerous papers trying to bring the tightness loss of the quantum proof as close to the classical proof.

That said, we would like to emphasize that this does not imply that ML-KEM is less secure when considering quantum adversaries; these loose bounds may simply by an artifact of our proof methodology, and the parameters of ML-KEM are set under the common belief that similar bounds in Theorem 4.2 hold in the quantum setting.

(ii) Additional assumptions on K-PKE. In the classical setting, the only thing we assumed from K-PKE was that it is correct and IND-CPA secure. In the quantum setting, we may assume a bit more properties from K-PKE to make the security proof tighter, e.g., spreadness [FO99; FO13; HHK17], disjoint simulatable [SXY18], injective [Bin+19; Kuc+20]. While some are known to be easily implied from the design of K-PKE, some are far less obvious, requiring numerical analysis [Din+22].

(iii) Implicit or explicit rejection. As we already mentioned in Remark 3.1, in the classical setting, whether ML-KEM performs implicit or explicit rejection has almost no consequence to the security proof. However, in the quantum setting, due to our limited proof technique in the QROM, this makes a notable impact. Indeed, most of the proofs in the QROM critically uses the fact that the KEM performs implicit rejection (as in ML-KEM), and only a few work with explicit rejection exists [JZM19a; Don+22; HHM22; HM24; HK25].

In summary, it is not always clear what the *best* quantum proof of ML-KEM is since there are several tradeoffs. Below, we present two representative and incomparable security proofs of ML-KEM in the QROM. We note that the following results hold generally for any KEM derived from standard $\mathsf{FO}^{\not\perp}_m$-transform in [HHK17]; as explained in the previous section, the FO-transform performed by ML-KEM can be thought of as the $\mathsf{FO}^{\not\perp}_m$-transform.

Theorem 4.3 (IND-CCA **Security of ML-KEM in QROM, [GLX24, Corollary 1]).** *Assume* K-PKE *is δ-correct and* IND-CPA *secure. Then, for any adversary $\mathcal{A}$ against the* IND-CCA *security of* ML-KEM *that makes at most q_{RO} quantum queries to the random oracle, there exists an adversary $\mathcal{B}$ against the* IND-CPA *security of* K-PKE *and an adversary $\mathcal{C}$ against the pseudorandomness of* J *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{ML\text{-}KEM},\mathcal{A}}(1^\lambda) \leqslant \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{C}}(1^\lambda) + 2\cdot\sqrt{q_{\mathsf{RO}}}\cdot(8\cdot q_{\mathsf{RO}}+3)\cdot\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{K\text{-}PKE},\mathcal{B}}(1^\lambda)\\ + 16\cdot(4\cdot q_{\mathsf{RO}}+1)\cdot\delta + 4\cdot\sqrt{q_{\mathsf{RO}}}\cdot\delta + 16\cdot\sqrt{q_{\mathsf{RO}}}\cdot(8\cdot q_{\mathsf{RO}}+3)\cdot\frac{10\cdot q_{\mathsf{RO}}+1}{2^{256}}.\end{aligned}$$

Theorem 4.4 (IND-CCA **Security of ML-KEM in QROM, [JZM19b, Theorem 1]).** *Assume* K-PKE *is δ-correct and* IND-CPA *secure. Then, for any adversary $\mathcal{A}$ against the* IND-CCA *security of* ML-KEM *that makes at most q_{RO} quantum queries to the random oracle, there exists an adversary $\mathcal{B}$ against the* IND-CPA *security of* K-PKE *and an adversary $\mathcal{C}$ against the pseudorandomness of* J *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{ML\text{-}KEM},\mathcal{A}}(1^\lambda) \leqslant \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{C}}(1^\lambda) + 4\cdot q_{\mathsf{RO}}\cdot\sqrt{\delta}\\ + 2\cdot\sqrt{(2\cdot q_{\mathsf{RO}}+1)\cdot\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{K\text{-}PKE},\mathcal{B}}(1^\lambda) + 2\cdot\frac{(2\cdot q_{\mathsf{RO}}+1)^2}{2^{256}}}.\end{aligned}$$

It is clear that both theorems establish the IND-CCA security of ML-KEM relying on the δ-correctness and IND-CPA security of K-PKE. Here, we note that unlike in the classical setting, there is a bound concerning the pseudorandomness of J. The only reason why this does not show up in the classical setting is because we can simply assume J is implemented by a (classical) random oracle — while we can similarly do this in the quantum setting, it is much easier to assume J is a PRF.

While both theorems provide the same asymptotic guarantee, the qualitative meaning differs. Denoting $\epsilon_{\mathsf{cca}} = \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{ML\text{-}KEM},\mathcal{A}}(1^\lambda)$ and $\epsilon_{\mathsf{cpa}} = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{K\text{-}PKE},\mathcal{B}}(1^\lambda)$, and ignoring all other components, we get $\epsilon_{\mathsf{cpa}} \gtrsim \epsilon_{\mathsf{cca}}\cdot q_{\mathsf{RO}}^{-1.5}$ and $\epsilon_{\mathsf{cpa}} \gtrsim \epsilon_{\mathsf{cca}}^2\cdot q_{\mathsf{RO}}^{-1}$ from the first and second theorems, respectively. If $\epsilon_{\mathsf{cca}} \approx 1$, i.e., an adversary breaks the IND-CCA security of ML-KEM with very high advantage, then the second theorem gives us a better proof as it translates to an adversary that breaks the MLWE problem with higher advantage. In contrast, if $\epsilon_{\mathsf{cca}} \approx q_{\mathsf{RO}}^{-1}$, then the first theorem gives us a better proof. We refer to the papers we cited at the beginning of this section for more details on the different types of asymptotic guarantees.

4.2 Other Security Properties

While IND-CCA security is the most important security property for any KEM, there are other notions of security that may be important in other contexts. We survey several additional security properties ML-KEM is known to satisfy.

Multi-user security. Standard security definitions (and concrete security analysis) only considers the security of KEMs with a single encapsulation key, i.e., single-user setting. In particular, it does not make any claims of the security of KEMs where an adversary can observe many encapsulation key, i.e., the multi-user setting. In the design of ML-KEM, two decisions were made to aim at improving security against attackers targeting multiple users (cf. [Ava+21, Section 4.5.3]).

- The matrix $\mathbf{A}$ is re-generated for each encapsulation key. This is in contrast to viewing $\mathbf{A}$ as a public parameter like the modulus q or degree n, used by all the users in the system. This protects against an attacker attempting to break many keys at the cost of breaking one key, by finding a vulnerability in the matrix $\mathbf{A}$.
- The FO-transform used by ML-KEM hashes the encapsulation key ek to generate the session key k and encryption randomness r. Making the randomness r dependent of the encapsulation key protects agains pre-computation attacks that attempt to break one out of many keys.

Anonymity and robustness. One common way to use a KEM is to use it as a PKE via the KEM-DEM paradigm [CS03]. For some applications of PKEs (and of KEMs for some degree), one may consider *anonymity* [Bel+01] and *robustness* [ABN10]. Roughly, the former guarantees that a ciphertext hides the receiver's information (i.e., the encapsulation key), a useful property for privacy-enhancing technologies such as anonymous credential systems [CL01] and anonymous authenticated key exchanges [Boy+09; Fuj+13; Fuj+15; SSW20]. In contrast, the later is an orthogonal property roughly stating that only the intended receiver can obtain a meaningful message from a ciphertext, a property having applications to searchable encryption [Abd+05] and auctions [Sak00].

There have been several works studying the properties needed by the underlying KEM for the PKE to satisfy anonymity and robustness [Moh10; Xag22; GMP22; MX23]. Since the FO-transform performed by ML-KEM can be thought of as the $\mathsf{FO}^{\not\perp}_m$-transform in [HHK17], the results of [Xag22; GMP22] indicate that if ML-KEM is *strong collision-free* CCA secure and K-PKE is *strongly disjoint-simulatable* and δ-correct for a negligible δ, then the resulting PKE obtained via the KEM-DEM paradigm with an appropriate symmetric key encryption scheme (i.e., DEM) is anonymous and robust. These properties are proven to hold even in the QROM; see also [Xag22, Section 3.1] and [GMP22, Section 5.4], where note that while these results focus on CRYSTALS-KYBER, the same arguments hold for ML-KEM.

KEM binding. [CDM24] systematically explores the possible *binding* properties of KEMs. The above mentioned robustness is one type of the binding properties explored. Slightly more formally, robustness in [Xag22; GMP22] guarantees that an adversary given two honestly generated encapsulation keys cannot find a ciphertext such that it decapsulates to a valid session key under the two decapsulation keys. While robustness guarantees that a ciphertext is bound to an honest KEM key, we can think of numerous other types of binding properties. For instance, say an adversary is given an honestly generated ciphertext under an encapsulation key of Alice. We may not want the adversary to be able to produce another ciphertext that decapsulates to the same session key under a decapsulation key of Bob. Such a re-encapsulation attack can occur regardless of whether the KEM is robust since the issue is that the session key is not bound to the KEM key (and ciphertext). This typically manifests as an explicit attack when using the KEM at the protocol level where two parties compute the same session key despite disagreeing on their respective partners, e.g., [KS23]. Since ML-KEM hashes the encapsulation key when deriving the session key, such re-encapsulation attacks are known to be mitigated. We refer to [CDM24; Sch24] for a more detailed discussion on all the different types of the binding properties and which properties ML-KEM is expected to satisfy.

5 Practical Cryptanalysis of ML-KEM

This section evaluates the concrete security of ML-KEM by analyzing the hardness of the underlying MLWE problem against all known classical and quantum attack families. We model the induced error distribution (including bit-dropping), map MLWE to standard lattice problems, and review primal, dual, hybrid, and structure-exploiting attacks. The official ML-KEM security estimates confirm that all standardized parameter sets (ML-KEM-512, -768, -1024) comfortably meet or exceed their targeted NIST levels 1, 3, and 5 under both classical and quantum cost models.

5.1 (Practical) Error Modeling of ML-KEM

We consider an $\mathsf{ML\text{-}KEM}_{q,m,k,\chi_1,\chi_2}$ instance $(\mathbf{A}, \mathbf{b}, \mathbf{s}, \mathbf{x})$ where $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{x}$ with short secret $\mathbf{s} \xleftarrow{\$} \chi_1^k$ and error $\mathbf{x} \xleftarrow{\$} \chi_2^m$. In the IND-CPA security proof in Theorem 4.1 of K-PKE, underlying the final IND-CCA security proof of ML-KEM, we saw that we have to consider two parametrizations: $m = k$ samples with $\chi_1 = \chi_2 = B_{\eta_1}$, and $m = k + 1$ samples with $\chi_1 = B_{\eta_1}$ and $\chi_2 = B_{\eta_2}$ respectively. As the influence of the number of samples is small, we will conservatively assume that $m = k + 1$ for our security estimates. This leaves the difference in the error distribution χ_2 with parameter η_1 or η_2 for the keygen and encapsulation respectively.

For ML-KEM-512 we have that $\eta_2 < \eta_1$, while for the other two instantiations we have $\eta_1 = \eta_2$. For ML-KEM-512 we would thus have to consider the easier instance $\mathsf{ML\text{-}KEM}_{q,k+1,k,B_{\eta_1},B_{\eta_2}}$ coming from the encapsulation procedure. The reduction to this instance is not entirely tight, however, in particular it ignores the additional errors coming from bit-dropping. For the concrete security estimate of ML-KEM-512 we therefore consider two situations that could be seen as conservative or reasonable respectively.

Conservative model (referred to as ML-KEM-512$^{\text{no drop}}$): we ignore the additional errors coming from the bit-dropping and derive our security estimate based on the parameters above.

Reasonable model (referred to simply as ML-KEM-512): We will see that the efficiency of the best attacks depends on the norms of $\mathbf{s}$, and of the induced error $\mathbf{x}'$ after bit-dropping. The former depends on the variance of B_{η_1} which is $\eta_1/2$, while the latter depends on the variance $\eta_2/2$ of B_{η_2} plus the increase due to the bit-dropping. The bit-dropping is a deterministic procedure, but due to the randomness of $\mathbf{A}, \mathbf{s}, \mathbf{x}$ it is still reasonable to consider the impact on the error as a random procedure with some variance per coordinate. In fact, only applying this deterministic error is known as the *Learning With Rounding assumption*. Taking account of this variance, the variance of the norms of the coordinates $\mathbf{x}'$ is in fact always significantly larger than $\eta_1/2$. The reasonable assumption therefore, which is also followed by the authors in the Kyber proposal, is to simply assume that $\chi_2 = \chi_1$.

Note that for ML-KEM-768 and ML-KEM-1024 this was already the case and thus the conservative and reasonable regimes are already equivalent; we therefore only consider this single regime in the following.

5.2 MLWE as a Lattice Problem

The search variant of the MLWE problem can be interpreted as a Bounded Distance Decoding (BDD) problem in a random family of q-ary R-lattices.

Definition 5.1 (BDD). *Let $\mathbf{B} \in R^{d\times d}$ be a basis of a full-rank lattice L, let $\mathbf{e} \in R^d$ be a small error such that $\|\mathbf{e}\| < \frac{1}{2}\lambda_1(L)$, and let $\mathbf{t} \in \mathbf{e} + L$ be a target. Given $(\mathbf{B}, \mathbf{t})$, compute the error $\mathbf{e}$.*

The BDD error $\mathbf{e} \in \mathbf{t} + L$ is the unique error satisfying $\|\mathbf{e}\| \leqslant \frac{1}{2}\lambda_1(L)$, as $\lambda_1(L)$ is the minimum distance between two distinct lattice points. To see that a $\mathsf{MLWE}_{q,m,k,\chi_1,\chi_2}$ instance $(\mathbf{A}, \mathbf{b} = \mathbf{A}\cdot\mathbf{s} + \mathbf{x}, \mathbf{s}, \mathbf{x})$ is indeed a BDD instance we first consider the following q-ary lattice

$$\Lambda_q(\mathbf{A}) := \{(\mathbf{z}, \mathbf{y}) \in R^m \times R^k : \mathbf{z} \equiv \mathbf{A}\mathbf{y} \bmod q\} \subset R^{m+k} \text{ with basis } \mathbf{B} := \begin{pmatrix} qI_m & \mathbf{A} \\ 0 & I_k \end{pmatrix}.$$

Due to $\Lambda_q(\mathbf{A})$ being q-ary we have that $qR^{m+k} \subset \Lambda_q(\mathbf{A})$ and thus we can simply consider its coefficients as being elements of R_q. For the target $\mathbf{t} := (\mathbf{b}, 0) \in R_q^{m+k}$ we can write

$$\mathbf{t} = (\mathbf{b}, 0) = (\mathbf{A}\cdot\mathbf{s} + \mathbf{x}, 0) = (\mathbf{x}, -\mathbf{s}) + (\mathbf{A}\cdot\mathbf{s}, \mathbf{s}) \in (\mathbf{x}, -\mathbf{s}) + \Lambda_q(\mathbf{A}),$$

where $\mathbf{e} := (\mathbf{x}, -\mathbf{s}) \leftarrow \chi_2^m \times \chi_1^k$ is a small error for appropriate distributions χ_1, χ_2. $\|(\mathbf{x}, -\mathbf{s})\| < \frac{1}{2}\lambda_1(\Lambda_q(\mathbf{A}))$ we thus indeed obtain a BDD instance $(\mathbf{B}, \mathbf{t})$ with unique error $(\mathbf{x}, -\mathbf{s})$. To understand how small the error $(\mathbf{x}, -\mathbf{s})$ is compared to the first minimum $\lambda_1(\Lambda_q(\mathbf{A}))$ we consider $\Lambda_q(\mathbf{A})$ as an unstructured $\mathbb{Z}$-lattice of dimension $d = (m+k)n$. The random lattice $\Lambda_q(\mathbf{A})$ typically follows the Gaussian Heuristic, which says that a lattice L of dimension d typically satisfies

$$\lambda_1(L) \approx \mathrm{gh}(L) := \frac{\det(L)^{1/d}}{\mathrm{vol}(\mathcal{B}_d)^{1/d}} \approx \sqrt{d/2\pi e}\cdot\det(L)^{1/d},$$

where $\mathrm{vol}(\mathcal{B}_d)$ is the volume of a d-dimensional unit ball. For $\Lambda_q(\mathbf{A})$ we have $\det(\Lambda_q(\mathbf{A})) = q^{mn}$ and thus we expect that

$$\lambda_1(\Lambda_q(\mathbf{A})) = \mathrm{gh}(\Lambda_q(\mathbf{A})) \approx \sqrt{(m+k)n/2\pi e}\cdot q^{m/(m+k)}.$$

The length $\|(\mathbf{x}, -\mathbf{s})\|$ of the error $(\mathbf{x}, -\mathbf{s}) \xleftarrow{\$} \chi_1 \times \chi_2$ depends on their precise distributions. Assuming $\chi_1 = \chi_2$ with a variance of σ^2 per coefficient we expect that

$$\mathbb{E}\left[\|(\mathbf{x}, -\mathbf{s})\|^2\right] = n(k+m)\cdot\sigma^2,$$

and thus $\|(\mathbf{x}, -\mathbf{s})\|/\lambda_1(\Lambda_q(\mathbf{A})) \approx \sqrt{2\pi e\sigma^2}\cdot q^{-m/(m+k)}$. For ML-KEM σ^2 is a small constant, and $m/(m+k) \geqslant \frac{1}{2}$, so the error is a factor at least $\Omega(\sqrt{q})$ smaller than the first minimum of the lattice.

This gap, along with the lattice dimension, will be the most important factor for the final security estimates. Now that we have rephrased the underlying MLWE problem as a purely geometrical problem over a lattice, we review the known techniques used to tackle it, using lattice reduction algorithms.

5.3 Lattice Attacks and Estimates

In this section we will consider the current best attacks on the *unstructured* lattice problems underlying the security of ML-KEM. The concrete estimate of the cost of these attacks determines the bit security of the scheme. We will discuss the exploitation of the algebraic structure coming from the module later in Section 5.4.

5.3.1 Lattice Reduction and SVP Solvers

Lattice reduction. The currently best known attacks on the BDD problem underlying ML-KEM all reduce to the computation of short lattice vectors, also known as the *Shortest Vector Problem* (SVP), and of a good basis of the lattice, known as *lattice reduction*. Lattice reduction algorithms reduce the problem of finding a good basis of a lattice of dimension d to the computation of short vectors in lattices of a lower dimension $\beta \ll d$. They give a trade-off between the shortness of the basis and the dimension β. The history of lattice reduction is quite old and can be traced back to the early work of Hermite in the 18th century. While there exists a plethora of variants of reduction algorithms, in the following we only consider the Block-Korkine-Zolotarev (BKZ) reduction algorithm [Sch87], which gives the most effective trade-off in practice.

On the output quality and its estimation. The following lemma gives a good estimation of the geometry of the lattice basis it outputs.

Lemma 5.2 (Profile under the Geometric Series Assumption (GSA)). *Let* $\mathbf{B}$ *be a BKZ-*β *reduced basis of dimension* d*, then under the Geometric Series Assumption and the Gaussian heuristic the Gram-Schmidt vectors* $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_d$ *of* $\mathbf{B}$ *satisfy*

$$\log(\|\tilde{\mathbf{b}}_i\|) \approx \frac{d+1-2i}{2} \cdot \log(\alpha_\beta) + \frac{\log(\det(\mathbf{B}^T\mathbf{B}))}{2d},$$

for any $1 \leqslant i \leqslant d$ *and where* $\alpha_\beta = \mathrm{vol}(\mathcal{B}_d)^{-\frac{2}{\beta(\beta-1)}}$.

The Geometric Series Assumption in Lemma 5.2 gives a heuristic indication of the behavior of the lattice reduction algorithm BKZ on the basis of the lattice. One can observe that by increasing the dimension β, also referred to as the *blocksize*, we decrease the norm $\|\mathbf{b}_1\| = \|\tilde{\mathbf{b}}_1\|$ of the first basis vector. For $\beta = 2$ we obtain an exponential approximation with $\|\mathbf{b}_1\| \leqslant 2^{O(d)} \cdot \mathrm{gh}(L(\mathbf{B}))$, while for $\beta = d$ we obtain $\|\mathbf{b}_1\| \approx \mathrm{gh}(L)$. However, BKZ furthermore gives guarantees on the Gram-Schmidt norms of the rest of the basis vectors, which will be important for some attacks.

The GSA is a good first-order approximation which is often sufficient for asymptotic estimates. For more precise concrete estimates one can use simulators which, on input a starting basis, try to emulate the BKZ reduction process [CN11; YD17; BSW18]. This captures the more precise, sometimes probabilistic, nature of BKZ, and has in particular a big effect on the first and last few Gram-Schmidt norms. Additional care has to be taken in the case of q-ary lattices, as the initial basis has non-standard Gram-Schmidt norms $q, \ldots, q, 1, \ldots, 1$ that form a so-called "Z-shape", see [AD21] for more details. In certain regimes, this can lead to additional weaknesses, for example when q is large for NTRU [ABD16; KF17; DW21], or when q is small [DEP23].

Progressive reduction. In practice, BKZ is almost never run with a fixed large blocksize from scratch. Instead, implementations use *progressive* BKZ [Aon+16; Xia+22], where the blocksize is increased gradually, e.g. $\beta_1 < \beta_2 < \cdots < \beta_t$. At each stage one performs several *rounds* (also called *tours*) of BKZ-β_j until the shape of the Gram–Schmidt norms stabilizes, before moving on to the next larger blocksize. This strategy amortizes the cost of the expensive high-dimensional SVP calls and leads to noticeably better reduced bases for a given total running time. In concrete security estimates, one typically accounts for this by charging the cost of all intermediate rounds towards the final target blocksize.

The SVP routine in BKZ. To compute a BKZ reduced basis we have to consider the exact SVP problem, that of computing the shortest vector of length $\lambda_1(L)$ in a lattice $L \subset \mathbb{R}^\beta$ of dimension β.

Definition 5.3 (SVP). *Let* $\mathbf{B} \in \mathbb{R}^{\beta \times \beta}$ *be a basis of a full-rank lattice* L*. Given* $\mathbf{B}$*, compute a shortest non-zero vector* $\mathbf{y} \in L$ *such that* $\|\mathbf{y}\| = \lambda_1(L)$*.*

Typically, one considers the case of random lattices, known to be the hardest instances. Here we have $\lambda_1(L) \approx \mathrm{gh}(L)$. There are two broad categories of heuristic algorithms to solve SVP in such random lattices: enumeration algorithms which take super-exponential time $\beta^{O(\beta)}$ and polynomial memory, and sieving algorithms which take single-exponential $2^{O(\beta)}$ time and memory.

Enumeration algorithms are branch-and-bound algorithms which recursively reduce the enumeration of short vectors in a lattice L to the enumeration of many short vectors in a lower-dimensional projected lattice $\pi(L)$. The precise projections and bounds used can have a significant impact on the total size of the enumeration tree, and therefore on the time complexity. The asymptotic best algorithm by Kannan [Kan83] takes time $\beta^{\beta/2e+o(\beta)}$ in the worst case [HS07; HS08]. Later, this was improved by [Alb+20a] to $\beta^{\beta/8+o(\beta)}$ in the setting of lattice reduction, as was already heuristically indicated as a lower bound in [Ngu10; HS10]. Further exponential $2^{O(\beta)}$ improvements followed from pruning of the search tree [SH95; GNR10], and from better parametrization within lattice reduction [Aon+16; Agg+20; LN25; Alb+21]. Overall, the currently best concrete estimation of the enumeration cost inside lattice reduction is given by $2^{0.125\beta\log(\beta)-0.654\beta+25.84}$ [Alb+21]. Enumeration algorithms benefit directly from the quantum speed-up by Grover's algorithm, leading to an asymptotic time cost of $\beta^{\beta/16+o(\beta)}$.

Sieving algorithms run in single-exponential time at the cost of a single-exponential memory usage. The core idea is to keep track of a long list of somewhat short lattice vectors $S \subset L$, while trying to find pairs $x, y \in S$ of distinct close lattice vectors such that $x - y$ is shorter than x or y. This new shorter vector is then inserted into the list S, replacing a longer vector. Heuristically, this process continues until the shortest vector is found whenever $|S| \geqslant (4/3)^{\beta/2+o(\beta)} = 2^{0.2075\beta+o(\beta)}$ [NV08]. As one needs to check all pairs of vectors in the list, the time complexity is at least $|S|^2 \geqslant (4/3)^{\beta+o(\beta)} = 2^{0.415\beta+o(\beta)}$. A long line of research on nearest-neighbour search methods decreased the time complexity further to $(3/2)^{\beta/2+o(\beta)} = 2^{0.292\beta+o(\beta)}$ [Laa15; BGJ15; BL16; Bec+16], which was shown to be optimal in this context [KL21]. High overheads in practice initially made sieving algorithms mostly of theoretical interest. This changed, however, after several (heuristic) sub-exponential [Duc18] and polynomial improvements [LM18; Alb+19]. The idea of [Duc18] is that lattice sieving does not only give a single shortest vector, but $|S| = (4/3)^{\beta/2+o(\beta)}$ of them. By finding these many short vectors in a lower-dimensional projected lattice $\pi(L)$ we can hope that at least one of them lifts to the shortest vector in L, where $\pi(L)$ can heuristically be of dimension $\beta - f$ for $f \approx \frac{\ln(4/3)\beta}{\ln(\beta/2\pi e)}$, essentially obtaining f "dimensions for free". The works [LM18; Duc18] furthermore propose applying lattice sieving progressively, by starting with finding short vectors in a smaller dimensional lattice, while increasing the lattice dimension step by step. As a result, the vectors in the set S are already somewhat short once the highest dimensions are reached, reducing the number of iterations needed there. These and other implementation improvements combined now make sieving algorithms superior to enumeration algorithms, both in theory and in practice already from dimension 90 or higher [Alb+19]. All current record SVP computations are achieved by sieving algorithms [Alb+19; DSW21; ZDY25]. While heuristic sieving algorithms are performing well in practice now, their sub-exponential factors in the time complexity, hidden in the $o(\beta)$ in the exponent, are still not entirely understood [Duc22b]. Furthermore, the influence of memory access costs when using exponential memory makes it unclear what the cost will be in cryptographic dimensions. For example, due to these memory access costs, the asymptotic best algorithm [Bec+16]

is currently beaten in practice by asymptotically worse algorithms [DSW21; ZDY25]. Still, for estimating the security of ML-KEM, one assumes the complexity $2^{0.292\beta+o(\beta)}$ of the asymptotically best algorithm, which can be seen as a conservative estimate.

Low memory and quantum sieving. Another line of research tries to reduce the memory complexity while increasing the time complexity, by looking at differences of k-tuples of vectors for $k \geqslant 3$ [BLS16; HKL18]. For $k = 3$ this leads, for example, to a time complexity of $2^{0.338\beta+o(\beta)}$ using $2^{0.1887\beta+o(\beta)}$ memory [CL23]. Grover-like algorithms can similarly be used to improve the time complexity of sieving algorithms. However, contrary to enumeration, they achieve far from the possible quadratic speed-up. A line of works [LMP15; Laa16; CL21; Hei21; Bon+23] only improved the classical time complexity of $2^{0.292d+o(d)}$ down to $2^{0.2563+o(d)}$. More concrete estimates seem to indicate that, even in unit cost quantum memory models, these exponential speed-ups are tenuous at best for cryptographic dimensions [Alb+20b; Dor+24]. With all the expected overhead of quantum computations, quantum sieving is so far not a threat. Furthermore, all sieving algorithms based on pairs, and thus also quantum algorithms, have a fundamental lower bound of $2^{0.2075d+o(d)}$ on the number of vectors that need to be stored.

We now move back to solving the MLWE problem.

5.3.2 Primal Attack

The primal attack solves the BDD problem underlying MLWE by reducing it to a unique Shortest Vector Problem (uSVP), which in turn is resolved using BKZ. It does so by embedding the BDD instance $\mathbf{t} \in \mathbf{e} + L$ in a lattice L of dimension d into a lattice L' of dimension $d+1$ for which the unique shortest vector $\mathbf{v} \in L'$ corresponds to the BDD error $\mathbf{e}$.

Concretely, let $\mathbf{B}$ be a basis of the lattice $L \subset \mathbb{R}^d$ and let $\mathbf{t} = \mathbf{B} \cdot \mathbf{y} + \mathbf{e} \in \mathbb{R}^d$ be a BDD instance with $\|\mathbf{e}\| < \frac{1}{2}\lambda_1(L)$. We now consider the lattice $L' \subset \mathbb{R}^{d+1}$ with the following basis:

$$\mathbf{B}' := \begin{pmatrix} \mathbf{B} & \mathbf{t} \\ 0 & c \end{pmatrix} \quad \text{for some constant } c > 0.$$

Note that we can construct $\mathbf{B}'$ using the public information $\mathbf{B}$ and $\mathbf{t}$, and that $\det(L') = c \cdot \det(L)$ has a similar determinant. We do have that

$$\mathbf{B}' \cdot \begin{pmatrix} -\mathbf{y} \\ 1 \end{pmatrix} = \begin{pmatrix} -\mathbf{B} \cdot \mathbf{y} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{t} \\ c \end{pmatrix} = \begin{pmatrix} -\mathbf{B} \cdot \mathbf{y} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{B} \cdot \mathbf{y} + \mathbf{e} \\ c \end{pmatrix} = \begin{pmatrix} \mathbf{e} \\ c \end{pmatrix} \in L',$$

and thus L' contains the vector $(\mathbf{e}, c)$. As $\mathbf{e}$ is small, and for suitably chosen c, the BDD error $\mathbf{e}$ thus corresponds to the short vector $(\mathbf{e}, c) \in L'$. As $\|\mathbf{e}\| < \frac{1}{2}\lambda_1(L)$, and for suitable $c > 0$, we actually expect that $(\mathbf{e}, c)$ is the unique shortest vector in L'.

We now proceed to recover the shortest vector $(\mathbf{e}, c) \in L'$. Recall that typically BKZ requires blocksize $\beta = d + 1$ to recover the shortest vector of a lattice of dimension $d + 1$. Under the premise that the shortest vector is unusually short $\|\mathbf{e}, c\| \ll \mathrm{gh}(L')$, BKZ however recovers it already with a much lower blocksize [GN08; AFG14; Alk+16].

Lemma 5.4 (BKZ for uSVP, [Alk+16]). *Let L be a lattice of dimension d and let $\mathbf{v} \in L$ be a shortest vector satisfying $\|\mathbf{v}\| \ll \mathrm{gh}(L)$. Then, under the GSA (Lemma 5.2), BKZ with blocksize β heuristically recovers $\mathbf{v}$ if*

$$\sqrt{\frac{\beta}{d}} \cdot \|\mathbf{v}\| < \mathrm{vol}(\mathcal{B}_d)^{\frac{d+1-2\beta}{\beta(\beta-1)}} \cdot \det(L)^{1/d}.$$

For example, asymptotically we get that if $\|\mathbf{v}\| = \mathrm{gh}(L)/\Theta(\sqrt{d})$, as will be the case for ML-KEM parameters, we only require a blocksize of $\beta = \frac{d}{2} + o(d)$. This (roughly) explains why the underlying BDD problem related to the key generation of ML-KEM corresponds to a lattice problem in dimension $(k+m)n = 1024, 1536$ and 2048 for ML-KEM-512, ML-KEM-768 and ML-KEM-1024 respectively.

Beyond these asymptotics Lemma 5.4 gives much more precise estimates that one can use for concrete estimates of the required blocksize β, and therefore of the cost of the primal attack. These estimates can be refined in several additional ways. First, the term $\sqrt{\frac{\beta}{d}} \cdot \|\mathbf{v}\|$ relates to the expected length of the shortest vector $\mathbf{v}$ after projecting it away from the first $d - \beta$ basis vectors. Instead of using the expectation one can model this length as a random variable under certain heuristic distributions on the vector [Dac+20; PV21]. Secondly, the right part $\mathrm{vol}(\mathcal{B}_d)^{\frac{d+1-2\beta}{\beta(\beta-1)}} \cdot \det(L)^{1/d}$ directly relates to the expectation of the Gram-Schmidt norm $\|\tilde{\mathbf{b}}_{d-\beta+1}\|$ of the basis after BKZ-β reduction following Lemma 5.2. One could replace this rough estimate by the already mentioned BKZ simulators [CN11; YD17; BSW18]. Thirdly, for the case of LWE, to obtain the lowest blocksize estimate β from Lemma 5.4, it can be beneficial to not use all the samples. For example, for ML-KEM parameters the optimal number of samples is often slightly below $m = k$, and therefore an optional $k+1$-th sample does not influence the security estimate.

Further improvements in the runtime can be made by a two-step approach [Xia+24], where the basis is first reduced using several SVP calls in dimension β, after which a slightly larger SVP call in dimension $\beta' > \beta$ is made to recover the final solution. This compensates for the fact that BKZ makes many calls, compared to only a single SVP call that is needed for the last step.

5.3.3 Dual Attack

The dual attack uses vectors in the dual lattice to distinguish between BDD targets that lie close to the lattice, and targets that lie far away from the lattice [MR09; Alk+16; Alb17]. For a lattice $L \subset \mathbb{R}^d$ its dual lattice L^* is given by those vectors that have an integer inner product with all lattice vectors, i.e.,

$$L^* := \{\mathbf{y} \in \mathbb{R}^d : \forall \mathbf{v} \in L, \langle \mathbf{y}, \mathbf{v} \rangle \equiv 0 \bmod 1\}.$$

As a result, if we look at a BDD instance $\mathbf{t} = \mathbf{v} + \mathbf{e} \in \mathbb{R}^d, \mathbf{v} \in L$ and a dual vector $\mathbf{y} \in L^*$, then

$$\langle \mathbf{t}, \mathbf{y} \rangle \equiv \langle \mathbf{v}, \mathbf{y} \rangle + \langle \mathbf{e}, \mathbf{y} \rangle \equiv \langle \mathbf{e}, \mathbf{y} \rangle \bmod 1.$$

Now if both $\mathbf{e}$ and $\mathbf{y}$ are short, then $|\langle \mathbf{e}, \mathbf{y} \rangle| \leqslant \|\mathbf{e}\| \cdot \|\mathbf{y}\|$ is biased towards 0, or equivalently, $\langle \mathbf{t}, \mathbf{y} \rangle$ is expected to be close to an integer. Otherwise, if $\mathbf{t}$ is a uniform target, then $\langle \mathbf{t}, \mathbf{y} \rangle \bmod 1$ is also uniform. The differences in these distributions can be used to distinguish if $\mathbf{t}$ is a BDD instance or a uniform target.

The *dual attack* generally proceeds as follows: first a list $S \subset L^*$ of short dual vectors is computed, after which $\langle \mathbf{t}, \mathbf{y} \rangle \bmod 1$ is computed for all $\mathbf{y} \in S$, and a statistical test is performed to decide if $\mathbf{t}$ is most likely a BDD instance or a uniform target. Note that the computation of the list $S \subset L^*$ can be seen as a preprocessing step with time complexity $> |S|$, after which computing the inner

products only has a cost of $|S|$. It is therefore common that one performs the preprocessing once, to then solve multiple decisional BDD instances. We will discuss how these multiple decisional BDD instances are created in the next section on hybrid attacks.

In the last years there has been a significant increase in attention for the dual attack. This was especially the case due to claims of strong improvements to the concrete costs of dual attacks [EJK20; GJ21; Li+21; MAT22], indicating improvements over the primal attack. In particular, the report by MATZOV [MAT22] made a large impact, claiming to push KYBER (now ML-KEM) below the NIST security levels. These attacks relied on the assumption that the vectors $\mathbf{y} \in S$, and in particular their inner products $\langle \mathbf{t}, \mathbf{y} \rangle \bmod 1$, all behave independently and can be analysed as such. In [DP23b; BW25] this *independence heuristic* was questioned and shown to be false in certain regimes, and in particular in the regime of MATZOV's claims [MAT22]. Several follow-up works improved the analysis without the independence heuristic, either provable or backed up by experiments [DP23a; PS24]. Furthermore, the impact on the MATZOV attack was investigated further in [Car+25].

5.3.4 Hybrid Attacks

Hybrid attacks are a combination of standard lattice attacks and combinatorial guessing attacks: part of the secret is guessed, reducing one instance of the problem to multiple easier instances in a *lower-dimensional* or *sparser* lattice. In certain circumstances, especially when the secret has low entropy, this can give a good trade-off.

Hybrid attacks were first considered in the context of NTRU [How07; Hir+09], but were later extended to LWE [Alb17]. Concretely, consider a lattice L, a BDD instance $\mathbf{t} = \mathbf{v} + \mathbf{e}$, and a dual vector $\mathbf{w} \in L^*$ for which we know that $\langle \mathbf{e}, \mathbf{w} \rangle \in T$ for some small set T. Now, if we know that $\langle \mathbf{e}, \mathbf{w} \rangle = c \in T$, then we obtain an equation $\langle \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{t} - \mathbf{e}, \mathbf{w} \rangle = \langle \mathbf{t}, \mathbf{w} \rangle - c = c'$ on $\langle \mathbf{v}, \mathbf{w} \rangle$, where c' is known. Note that $\langle \mathbf{v}, \mathbf{w} \rangle = c'$ defines some hyperplane H in which the solution lies, i.e., $\mathbf{v} \in H \cap L$. Now for any $\mathbf{x} \in H \cap L$ we obtain a lattice $L' = -\mathbf{x} + (H \cap L)$ of dimension $d - 1$ and volume $\det(L') = \|\mathbf{w}\| \cdot \det(L)$. Furthermore, the original BDD instance is translated into a new BDD instance $\pi_H(\mathbf{t}) - \mathbf{x}$ on L' with solution $\mathbf{v} - \mathbf{x}$, from which $\mathbf{v}$ can be derived. By guessing the value of $c \in T$, we can thus turn the d-dimensional BDD instance into $|T|^t$ BDD instances over the $d - t$-dimensional lattice L'. One can repeat this step t times, leading to $|T|^t$ instances over a $d - t$ dimensional lattice. Alternatively, if $T = T' + q\mathbb{Z}$, we can instead consider a modular constraint $\langle \mathbf{e}, \mathbf{w} \rangle \equiv c \bmod q$, leading to T' BDD instances in a d-dimensional but sparser lattice L'.

For LWE as a BDD instance in the q-ary lattice $\Lambda_q(\mathbf{A})$, one can, for example, always consider a dual unit vector $\mathbf{w}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$, such that $\langle \mathbf{w}_i, \mathbf{e} \rangle = e_i$ is a coefficient of the BDD error. The number of guesses depends on the precise error or secret distribution used, e.g. for binary LWE we have $|T|^t = 2^t$, while for ML-KEM we have $|T|^t \geqslant 5^t$ to reduce the dimension by t. Note that, depending on the distribution, one does not necessarily have to consider all guesses; for example, for very sparse distributions it can be sufficient to only guess errors with a few non-zero coefficients [Alb17; ACW19].

What remains is to solve the $|T|^t$ BDD instances in the $d - t$ dimensional lattice L'. This is also known as Batch-BDD or the Bounded Distance Problem with Preprocessing (BDDP), as one can perform some pre-computation on L' that can be amortized over all BDD instances. Here one can again consider both a primal or a dual attack approach.

For the hybrid primal attack one would typically first reduce the lattice L', after which Babai's nearest plane algorithm [ACW19; Kar+25b; PV25], or more advanced BDDP algorithms [EK20; DLW20; Ber22; Ber23; Kar+25a], are used to solve the BDD instances. Especially interesting is the

combination of this approach with the *randomized iterative slicer*, which solves each BDDP instance with a relatively small amortized cost, obtaining an exponential asymptotic speed-up [Ber23] if $|T|$ is finite. More precisely, asymptotically the bit security is reduced by a factor $(1-\mathcal{K})$ [Ber23; Kar+25a] where $\mathcal{K} = (1+\frac{\mathcal{H}(\chi)}{0.058})^{-1}$ and $\mathcal{H}(\chi)$ is the Shannon entropy of the error distribution χ on a single coordinate. For ML-KEM we have, for example, $\mathcal{H}(B_3) = 2.333$ and $\mathcal{K} \approx 0.024$, which would amount to roughly 3.5, 5 and 6.5 bits for the three security levels respectively. This hybrid attack was also shown to outperform the standard primal attack in practice [Kar+25a] by a small factor. At this point, however, it remains unclear whether this is a result of implementation details or of an asymptotic speed-up, and therefore also what the concrete impact will be on cryptographic dimensions.

For the hybrid dual attack one typically proceeds by pre-computing a list $S \subset (L')^*$ of short dual vectors. The inner products with the targets and the dual vectors are then used to distinguish which ones are real BDD targets, i.e., lie close to the lattice, and which ones are not. Note that one only uses the dual attack as a distinguisher here, to determine which of the guesses was correct. Note that such a hybrid version is almost always better than a standard dual attack as it amortizes the cost of computing the many dual vectors over several targets. After the first hybrid dual attacks [Alb17; EJK20; Kar+25b] several improvements followed. Firstly, note that one has to compute the inner product between a large list of $|S|$ vectors and a list of $|T|^t$ targets, leading to a cost of $O(|S| \cdot |T|^t)$ inner products. Typically, these targets are highly structured; for example, they might form an additive group modulo the lattice L. In these cases one can use FFT techniques to improve the cost down to $\tilde{O}(\max\{|S|, |T|^t\})$ [GJ21]. While these group structures often do not appear in the case of small or sparse errors or secrets, one can additionally either only guess their most significant bits or perform modulus switching techniques to still fall into this case and decrease the size of $|T|$ [GJ21; MAT22]. In [Car+25] this was further improved by even better coding-theoretic techniques to replace the modulus switching step.

5.3.5 Aurora-Ge and BKW

We shortly discuss the Aurora-Ge [AG11] and BKW [BKW00; KF15] algorithms. These generally require many samples or an error distribution with a very small support. For ML-KEM, after converting to plain LWE, we obtain at most $(m+k)d \leqslant (2k+1)d$ samples, which, along with the error support of size at least 5, is far from sufficient to make these attacks competitive.

The Aurora-Ge [AG11] attack proceeds by modelling the LWE problem as a system of polynomial equations, which is then solved using standard linearization or Gröbner bases techniques. The idea is that if the LWE errors e_i take at most k values $C = \{c_1, \dots, c_k\}$, then $f_i(X) := f_i(X_1, \dots, X_n) = b_i - \sum_{j=1}^{n} A_{ij} \cdot X_j = e_i \in C$, and thus for each sample we obtain a polynomial equation $\prod_{l=1}^{k}(f_i(X) - c_i) = 0$ of degree k in n variables. As such a system typically has $\Omega(n^k)$ distinct monomials, one requires at least that number of samples and thus equations to linearise and solve the system. Gröbner basis techniques [Alb+14; Ste24] allow for further trade-offs between the time cost and the number of samples. For ML-KEM we have $k \geqslant 5$ and only on the order of $O(n)$ samples, making linearization not applicable and the Gröbner basis attack far from competitive, both asymptotically and concretely.

The BKW attack, initially developed by Blum, Kalai and Wasserman for LPN [BKW00], is a combinatorial attack on LWE that runs in sub-exponential time when many samples are available [KF15]. The idea is to find many pairs of LWE samples $(b_i, a_i), (b_j, a_j) \in \mathbb{Z} \times \mathbb{Z}^n$ such that the first $k > 0$

coefficients of $a_i - a_j$ are 0. Removing these coefficients, we obtain a new LWE sample $(b_i - b_j, a_i - a_j)$ of LWE dimension $n - k$, but with the same (truncated) secret. We can repeat this process until the LWE problem becomes feasible. Such an attack has two problems: firstly, the error $e_i - e_j$ of the new LWE sample is larger than the original one, and secondly, we need about $m = \Theta(\sqrt{q^k})$ samples to find even a single pair that collides on the first k coefficients. To use this technique recursively we would constantly need about $m = \Theta(q^k)$ samples. The parameter k cannot be too small, as otherwise the error would increase too much. All in all, while some further trade-offs are possible, for typical parameters the BKW algorithm requires a large number of samples to perform well, and is therefore not applicable to ML-KEM.

5.4 Structured Attacks

Structured attacks aim to exploit the algebraic structure of ML-KEM's underlying module-LWE $\mathsf{MLWE}_{q,m,k}$ instance – namely that the public-key lattice is an R-module for the 2-power cyclotomic ring $R = \mathbb{Z}[x]/(x^{256}+1)$ of degree $n = 256$ with small module rank $m+k \in \{4, \ldots, 9\}$. When dealing with such structured lattices, two families of attacks and algorithms are relevant.

(i) ***Algebraic attacks on ideal lattices.*** Several results exploit cyclotomic-ideal structure – e.g., recovering short generators of principal ideals and finding "mildly short" vectors, sometimes in quantum polynomial time for related problems – showing *quantum* speedups for *ideal-SVP/PIP*-type tasks [Cra+16; CDW17]. However, these do not translate into attacks on the average-case *module-LWE* instances underlying ML-KEM, and no concrete parameter break or asymptotic improvement against ML-KEM's choices is known.

(ii) ***Module-/ideal-aware lattice reduction.*** Dedicated reductions for module and ideal lattices (e.g., module-LLL) preserve the R-module structure and can yield constant-factor savings (better preprocessing, slightly altered slope predictions) compared to treating the lattice as unstructured, but they do not provide an asymptotic advantage over state-of-the-art BKZ/sieving at ML-KEM dimensions; concrete estimates for ML-KEM therefore still model cost essentially as generic lattice reduction in dimension $n(m+k)$ after coefficient embedding [Lee+19; KEF20; MS20; DEP25].

We give more details about these attacks in Sections 5.4.1 to 5.4.3.

Scope and caveats. Known "subfield/weak-ring" attacks target non-cyclotomic or specially structured rings, or overstretched-modulus NTRU variants, and do not apply to ML-KEM's power-of-two cyclotomic ring and small module rank; indeed ML-KEM' s specification selects parameters precisely to avoid such pathologies [ABD16; KF17; DW21]. In short, while structure enables efficient implementation (NTT, automorphisms) and motivates specialized algorithms for some *ideal*-lattice problems, there is currently no structured attack that asymptotically outperforms generic lattice attacks on ML-KEM' s module-LWE; security estimates therefore continue to rely on generic BKZ/sieving models with, at most, modest constant-factor allowances for structure.

5.4.1 Ideal Lattices

The special case of rank 1 module lattices, also known as *ideal lattices*, can be seen as the most structured case among all module lattices. While this makes them interesting in terms of efficiency for cryptographic purposes and in terms of structure for, e.g., Fully Homomorphic Encryption

schemes [Gen09], it also could introduce vulnerabilities. Indeed, for ideal lattices there is a gap between the best classical and the best quantum algorithms: while classical algorithms can efficiently achieve an approximation ratio of $2^{O(n)}$ using LLL, quantum algorithms achieve a ratio of $2^{O(\sqrt{n})}$ for ideal lattices over cyclotomic fields [Cra+16; CDW17; DPW19; CDW21].

The same quantum algorithm for ideal lattices also applies to general number fields [PHS19] although only heuristically, with up to a $2^{O(n)}$ preprocessing cost depending on the number field, and with a potentially worse approximation ratio depending on the discriminant of the number field.

A special case of the above algorithms occurs in the setting when one considers a principal ideal $I \subset R$ generated by an unusually short principal generator $g \in R$. In this case this short generator g can be recovered in quantum polynomial time [Cra+16].

While these quantum algorithms on general ideal lattices do not reach an approximation ratio that would threaten the security of basic cryptographic primitives, the existence of this quantum advantage has mostly stopped the adoption of lattice-based cryptography based on ideal lattices. For example, ML-KEM uses module lattices with rank at least 4 to fall out of the scope of this line of attacks.

5.4.2 Class and (S-)Unit Groups

The quantum algorithms for ideal-SVP rely on the class and (S-)unit group of the underlying number ring R. Indeed, while computing these groups classically takes sub-exponential time [BF14; Bia+17], they can quantumly be computed in polynomial time [Eis+14; BS15; BDF20; BF25]. All these quantum algorithms rely on the same *continuous hidden subgroup* problem framework. For a group G of the form $G = \mathbb{R}^s \times \mathbb{Z}^t$, a subgroup $H \subset G$ and query access to a (sufficiently non-flat) H-periodic function $f : G \to \mathbb{C}$ satisfying $G(g+h) = G(x)$ for all $g \in G, h \in H$, the continuous hidden subgroup problem asks to recover (generators of) H. The problem of computing the class group, (S-)unit group and the recovery of a principal generator can all be reduced to an appropriate instance of this period-finding problem, which can be solved efficiently by a quantum computer. For more technical details about the continuous hidden subgroup problem we refer to [BF25].

5.4.3 Module-LLL

Recall that the lattice reduction algorithms like LLL and BKZ reduce the problem of computing approximately short vectors to several exact SVP instances in a lower dimension $\beta \geqslant 2$. *Module lattice reduction* algorithms proceed similarly, but they make use of the module structure, for example such that the lower-dimensional exact SVP instances are still acting on module lattices [Lee+19; MS20; DEP25], or to use the algebraic structure to improve the efficiency [KEF20].

Module-LLL or Module-BKZ [Lee+19; MS20; DEP25] indeed reduce the problem of lattice reduction of a rank $r > 2$ R-module lattice to that of a rank $2 \leqslant \beta < r$ R-module lattice. One obtains a trade-off between r, β, the discriminant of R, and the reached approximation radius. If one solves the rank β R-module SVP problem using a classical SVP algorithm in $\mathbb{Z}$-dimension βn, then this trade-off is worse than the classical BKZ algorithm for the power-of-two cyclotomic field used by ML-KEM. However, for fields with smaller discriminants the trade-off can lead to subexponential speed-ups [DEP25].

Note that module-LLL and module-BKZ only reduce to the module-SVP problem with rank at least 2, so the earlier mentioned ideal-SVP attacks are not relevant. This seems to indicate a hardness gap between rank 1 and rank at least 2 module lattice problems.

Another approach by [KEF20] is to use the algebraic structure to improve the runtime of the classical LLL (and in principle BKZ) algorithms. This can reduce the cost by polynomial factors, which especially for the already polynomial-time LLL algorithm can be beneficial. While such improvements can give small practical speed-ups for attacks on ML-KEM, they are insignificant compared to the large cost of the SVP calls.

5.5 Concrete Security Estimates

From these attack baselines we can estimate numerically the security given by the different parameter sets. To generate these figures, we relied on the state-of-the-art estimator from Albrecht et al. [APS15], available at https://github.com/malb/lattice-estimator.

5.5.1 Most Conservative Estimates via CORE-SVP Costing

To be as conservative as possible, we first rely on the so-called *CORE-SVP* costing model for lattice reduction, which forgets all polynomial factors beyond the main sieving cost subroutine in dimension β, which is thus costed as 0.292β bits following the asymptotically fastest sieve [Bec+16]. These costs are reported in Table 5. For consistency with the literature—and in particular with the original KYBER submission document [Ava+21], we used the *reasonable error modelling* as described in Section 5.1, but we also showcase the effect of the *no bit dropping* model (labeled as ML-KEM-512$^{\text{no drop}}$ in the table) as it provides the most conservative estimates. As mentioned, for ML-KEM-768 and ML-KEM-1024, there is no difference in these models.

Table 5: ML-KEM concrete security estimates in bits under CORE-SVP cost model. Best attack cost is in bold and corresponds to the dual attack, hybridized with some enumeration as described in Section 5.3.4.

Param set	NIST cat.	Primal attack	Dual attack (hybrid)	Combinatorial attacks
ML-KEM-512$^{\text{no drop}}$	1	115	120 (**112**)	167 (coded BKW)
ML-KEM-512	1	119	124 (**115**)	179 (coded BKW)
ML-KEM-768	3	182	189 (**174**)	238 (coded BKW)
ML-KEM-1024	5	255	264 (**242**)	310 (coded BKW)

5.5.2 Refined Estimates

We also consider a more refined cost model following [MAT22]. This model tries to estimate the costs of the progressive BKZ algorithms, the concrete cost of the BDGL sieve [Bec+16], and the probabilistic nature of the error distribution. Additionally, for a more refined estimate, matching the cost model, one should also use a BKZ simulator instead of the Geometric Series Assumption; however, this is currently not supported for the (hybrid) dual attacks in the estimator. These more refined bit-cost estimates are shown in Table 6. Here too, we use the reasonable error modeling as the base case, but also show the no bit dropping model for conservative costing.

Comparing the estimator to the literature, [Car+25] reports 0.2 to 2.5 bits lower for the dual hybrid attack after some additional improvements under the same cost model. Furthermore, while

in the current estimator the primal hybrid attacks are not shown to be effective, the works [Ber23; Kar+25a] indicate that a small speed-up is possible over the regular primal attack.

While the security estimates reported in Table 6 are slightly below the NIST bit-security levels of 143, 207 and 272 bits respectively, it is important to note that these more refined numbers should be seen as lower bounds for the current best known attacks. For example, they do not account for the basis quality loss coming from progressive sieving (+2.5 bits [Duc22a]), or for the overheads in the BDGL sieve [Duc22b] (+5 at security level 1), and some of the constants in the cost model are chosen optimistically. Additionally, they do not account for the cost of memory storage or access, which can turn out to be the largest cost in practice.

Table 6: ML-KEM concrete security estimates in bits under GSA and MATZOV cost model. These estimates should be interpreted as more refined lower-bounds. Best attack cost is in bold, corresponding to a dual attack hybridized with enumeration Section 5.3.4.

Param set	NIST cat.	Primal attack	Dual attack (hybrid)
ML-KEM-512$^{\text{no drop}}$	1	136.8	146.2 (**136.0**)
ML-KEM-512	1	140.2	149.9 (**139.7**)
ML-KEM-768	3	201.0	214.3 (**196.4**)
ML-KEM-1024	5	270.7	288.5 (**262.3**)

Taking this into account, all standardized ML-KEM parameter sets remain well within their target security categories even when the most optimistic attack parameters are assumed.

5.6 Decryption-Failure Attacks and Weak Keys in ML-KEM

A *decryption-failure (DF) attack* exploits the rare event that Decaps produces a key that does not match the encapsulator's key. Concretely, letting $\mathsf{ct} = (\mathbf{u}, v)$ be a ciphertext and $\mathbf{s}$ the secret, decapsulation essentially forms (modulo the technicalities yielded by the decompression) $w = v - \mathbf{s}^\top \cdot \mathbf{u} \bmod q$ and recovers bits by rounding the coefficients of w; a failure occurs only if noise pushes some coefficient across the decision boundary. If there is an observable distinction between such "success" and "failure" (different return values, timing, logs, or protocol branching, for instance), the adversary obtains a *binary oracle* [D'A+19]. The idea of decryption failure attacks is then to *boost* the failure rate by crafting ciphertexts whose induced error is near the boundary, and use the oracle's bit to run statistical tests on linear forms in $\mathbf{s}$; "directional" variants iteratively nudge (u, v) along locally more failure-prone directions to accelerate information extraction [D'A+19; DB22]. In a multi-target setting, *weak keys* are those secrets for which a fixed family of crafted ciphertexts happens to sit closer to the boundary—yielding a slightly higher failure probability and allowing an attacker to select weak users (or to amortize precomputation across many public keys) [DB22]. DF attacks can also be *amplified* operationally if the decapsulator uses a public key that is not the one matching its secret (e.g. due to storage/PK-substitution issues), which perturbs the effective noise and raises the failure rate [Flu+25]. In ML-KEM, however, properly designed systems are resilient: (i) parameters make the baseline failure probability δ negligible, so harvesting enough genuine failures is computationally infeasible [Nat24b][1]; (ii) the Fujisaki–Okamoto (FO) transform

[1]Remark that the standardized API uses implicit rejection so that a correct and secure implementation exposes no success/failure bit at all. However, as mentioned in Remark 3.1, in a more involved protocol, an implicit failure

binds the coins to a hash of the public key, which blocks multi-target precomputation and mitigates weak-key leverage [Ava+21; Sch22b; Sch22a]. With ML-KEM' s parameters and FO binding even accidental leakage leaves little room for practical failure exploitation without an oracle specifically *boosting* the failure probability; the remaining burden is mostly implementational: constant-time Decaps, identical code paths and logging across outcomes, storing/verifying the public key alongside the secret, and conservative query/rate limits [Flu+25]. We provide more details in Section 6.2.3.

5.7 Summary

Across all examined attack classes—lattice, hybrid, structural, and decryption-failure—no practically exploitable weakness is known for ML-KEM at any parameter level. Concrete estimates confirm that the schemes maintain security margins exceeding NIST's target categories even under aggressive classical and quantum cost assumptions.

might abort the protocol, giving de facto access to this bit, making the need for δ to be negligible in any case.

6 Implementation Details: Performance and Security

This section discusses subjects related to the implementation of ML-KEM in practice. In particular, Section 6.1 provides details about the algorithms used to implement ML-KEM. This includes low level functions implementing polynomial arithmetic, binomial sampling and the symmetric primitives that are used for cryptographic hashing and deterministic randomness generation.

In Section 6.2 the side channel security of ML-KEM implementations is studied. It provides background on side channel analysis and explains how side channel attacks on ML-KEM are performed. The focus is on the decapsulation, which is the operation in ML-KEM that is most vulnerable to physical attacks. Several countermeasures to protect against side channel attacks are also discussed.

Section 6.4 contains a comparison of results of hardware implementations of ML-KEM. The goal of this section is to provide some intuition regarding the computation time and area usage of practical implementations. It is also shown that the cost of protecting against side channel analysis can be considerable.

6.1 A Closer Look at the Algorithms of ML-KEM

This subsection provides background on the computation of the operations in ML-KEM that are particularly relevant to implementation security.

6.1.1 Building Blocks

Message encoding and decoding. The random 32-byte message generated during Encrypt in Encaps determines the shared secret. Since the confidentiality of the shared secret must be guaranteed, the operations that process the message are therefore of particular interest.

During encapsulation, a random message of 32 bytes is encoded to be hidden in the plaintext. The encoding process first converts the 32 bytes to 256 bits using the algorithms in Figure 4. Bits equal to 0 have to be mapped to 0 in $\mathbb{Z}_q$ and bits equal to 1 are mapped to $\frac{q}{2}$ in $\mathbb{Z}_q$. This is done using the $\mathsf{Decompress}_d$ function described in Section 2.3.5, with $d = 1$. The result is a message polynomial μ with coefficients in $\{0, \frac{q}{2}\}$ that is hidden in ciphertext part v.

During Decaps the original message bytes must be recovered from the ciphertext. This is done by applying the $\mathsf{Compress}_1$ function (explained in Section 2.3.5) to the intermediate result w in line 5 of Decrypt in Figure 8. This function maps elements from $\mathbb{Z}_q$ that are closer to 0 mod q than to $\frac{q}{2}$ to 0, and elements that are closer to $\frac{q}{2}$ are mapped to 1. The resulting bits are converted back to bytes using the BitsToBytes function from Figure 4.

Number Theoretic Transform (NTT). The NTT is a variant of the Fast Fourier Transform over the finite field $\mathbb{Z}_q$. It takes as input a polynomial in R_q and evaluates it in the powers of a primitive 512-th root of unity in $\mathbb{Z}_q$. The result is a vector of length 256 with coefficients in $\mathbb{Z}_q$, in the *NTT domain*. Polynomial multiplication between two polynomials a and b in the NTT domain is computed by multiplying their coefficients point-wise:

$$a \cdot b = (a_0 \cdot b_0 \mod q, a_1 \cdot b_1 \mod q, \ldots, a_{n-1} \cdot b_{n-1} \mod q). \tag{3}$$

This means that polynomial multiplication in the NTT domain can be computed in only $n = 256$ multiplications in $\mathbb{Z}_q$, whereas the Schoolbook polynomial multiplication method would require n^2

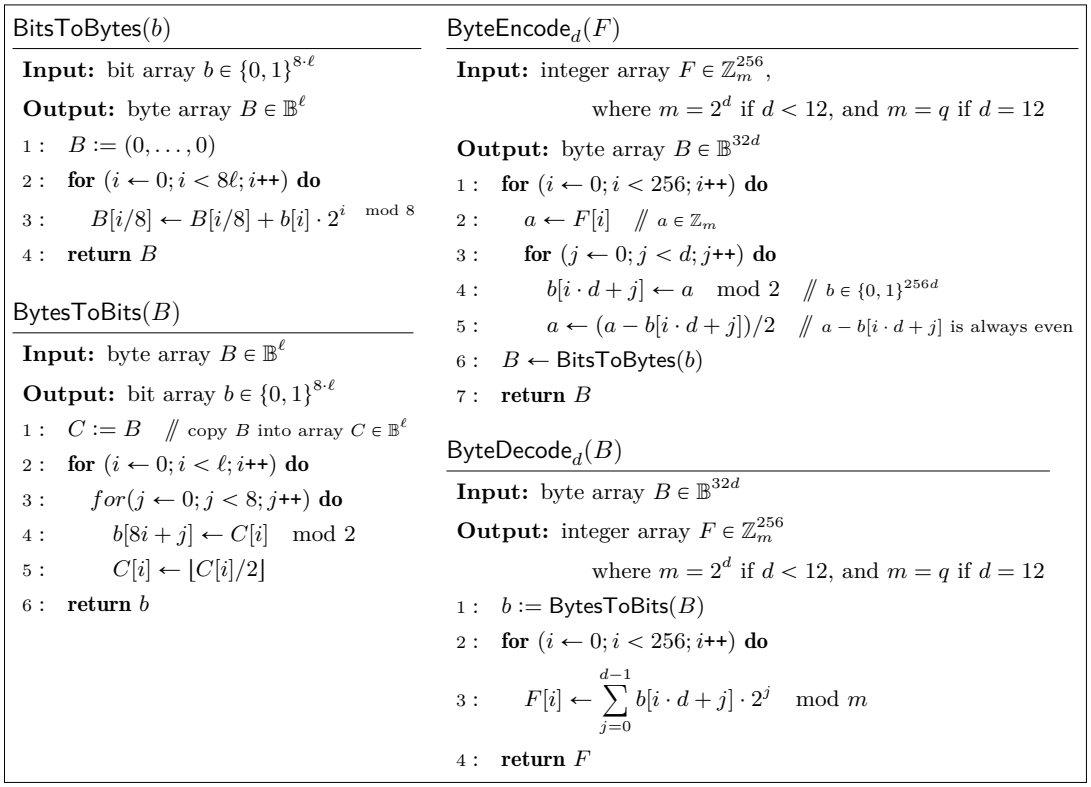

Figure 4: Algorithms BitsToBytes, BytesToBits, ByteEncode, and ByteDecode.

multiplications in $\mathbb{Z}_q$. Since the NTT itself is a linear operation, polynomial addition can simply be computed in the same way as in the *time* domain:

$$a + b = (a_0 + b_0 \mod q, a_1 + b_1 \mod q, \ldots, a_{n-1} + b_{n-1} \mod q). \tag{4}$$

Therefore polynomial multiplication, which can be a major performance bottleneck, can be sped up using the NTT. Once all polynomial arithmetic has been computed, the results must be transformed back into the time domain before applying non-linear operations. The inverse transform follows a similar structure and is also shown in Figure 5.

In ML-KEM, an *incomplete* NTT is used. That is, a polynomial in the time domain is evaluated by substituting x^2 by the powers of the 256-th root of unity ζ. The result is a vector of 128 polynomials of degree 1. The algorithm that computes the incomplete NTT is shown in Figure 5. In order to multiply two polynomials that are transformed by the incomplete NTT, the MultiplyNTTs algorithm from Figure 6 is computed. The product of two polynomials in the incomplete NTT domain can be computed by pair-wise multiplying the degree 1 polynomials of the two vectors. The pair-wise multiplication is defined by BaseCaseMultiply in Figure 6.

$\mathsf{NTT}(f)$

Input: array $f \in \mathbb{Z}_q^{256}$ // the coefficients of the input polynomial

Output: array $\hat{f} \in \mathbb{Z}_q^{256}$ // the coefficients of the NTT of the input polynomial

1 : $\hat{f} := f$ // will compute in place on a copy of input array
2 : $i := 1$
3 : **for** $(\mathsf{len} \leftarrow 128; \mathsf{len} \geqslant 2; \mathsf{len} \leftarrow \mathsf{len}/2)$ **do**
4 : **for** $(\mathsf{start} \leftarrow 0; \mathsf{start} < 256; \mathsf{start} \leftarrow \mathsf{start} + 2 \cdot \mathsf{len})$ **do**
5 : $\mathsf{zeta} \leftarrow \zeta^{\mathsf{BitRev}_7(i)} \mod q$
6 : $i \leftarrow i + 1$
7 : **for** $(j \leftarrow \mathsf{start}; j < \mathsf{start} + \mathsf{len}; j\mathsf{++})$ **do**
8 : $t \leftarrow \mathsf{zeta} \cdot \hat{f}[j + \mathsf{len}]$ // steps 8-10 done modulo q
9 : $\hat{f}[j + \mathsf{len}] \leftarrow \hat{f}[j] - t$
10 : $\hat{f}[j] \leftarrow \hat{f}[j] + t$
11 : **return** $\hat{f}$

$\mathsf{NTT}^{-1}(\hat{f})$

Input: array $\hat{f} \in \mathbb{Z}_q^{256}$ // the coefficients of input NTT representation

Output: array $f \in \mathbb{Z}_q^{256}$ // the coefficients of the inverse NTT of the input polynomial

1 : $f := \hat{f}$ // will compute in place on a copy of input array
2 : $i \leftarrow 127$
3 : **for** $(\mathsf{len} \leftarrow 2; \mathsf{len} \leqslant 128; \mathsf{len} \leftarrow \mathsf{len} \cdot 2)$ **do**
4 : **for** $(\mathsf{start} \leftarrow 0; \mathsf{start} < 256; \mathsf{start} \leftarrow \mathsf{start} + 2 \cdot \mathsf{len})$ **do**
5 : $\mathsf{zeta} \leftarrow \zeta_{2 \cdot \mathsf{len}}^{-\mathsf{BitRev}_7(i)} \mod q$
6 : $i \leftarrow i - 1$
7 : **for** $(j \leftarrow \mathsf{start}; j < \mathsf{start} + \mathsf{len}; j\mathsf{++})$ **do**
8 : $t \leftarrow f[j]$
9 : $f[j] \leftarrow t + f[j + \mathsf{len}]$ // steps 9-10 done modulo q
10 : $f[j + \mathsf{len}] \leftarrow \mathsf{zeta} \cdot (t - f[j + \mathsf{len}])$
11 : $f \leftarrow f \cdot 3303 \mod q$ // multiply every entry by $3303 = 128^{-1} \mod q$
12 : **return** f

Figure 5: Algorithms NTT and NTT^{-1}.

Binomial sampling. Both key generation and $\mathsf{Encrypt}$ require the sampling of polynomials whose coefficients follow the binomial distribution for a fixed parameter η. To sample one single coefficient from the binomial distribution, two uniformly random bit vectors $x = (x_0, \ldots, x_{\eta-1})$ and $y = (y_0, \ldots, y_{\eta-1})$ in $\mathbb{Z}_2^\eta$ are generated, and the sample

MultiplyNTTs$(\hat{f}, \hat{g})$

Input: Two arrays $\hat{f} \in \mathbb{Z}_q^{256}$ and $\hat{g} \in \mathbb{Z}_q^{256}$ // the coefficients of two NTT representations

Output: An array $\hat{h} \in \mathbb{Z}_q^{256}$ // the coefficients of the product of the inputs

1 : **for** $(i \leftarrow 0; i < 128; i$++$)$ **do**

2 : $(\hat{h}[2i], \hat{h}[2i+1]) \leftarrow \mathsf{BaseCaseMultiply}(\hat{f}[2i], \hat{f}[2i+1], \hat{g}[2i], \hat{g}[2i+1], \zeta^{2\mathsf{BitRev}_7(i)+1})$

3 : **return** $\hat{h}$

BaseCaseMultiply$(a_0, a_1, b_0, b_1, \gamma)$

Input: $a_0, a_1, b_0, b_1 \in \mathbb{Z}_q$ // the coefficients of $a_0 + a_1 X$ and $b_0 + b_1 X$

Input: $\gamma \in \mathbb{Z}_q$ // the modulus is $X^2 - \gamma$

Output: $c_0, c_1 \in \mathbb{Z}_q$ // the coefficients of the product of the two polynomials

1 : $c_0 := a_0 \cdot b_0 + a_1 \cdot b_1 \cdot \gamma$ // steps 1-2 done modulo q

2 : $c_1 := a_0 \cdot b_1 + a_1 \cdot b_0$

3 : **return** (c_0, c_1)

Figure 6: Algorithms MultiplyNTTs and BaseCaseMultiply.

$$(x_0 + \cdots + x_{\eta-1}) - (y_0 + \cdots + y_{\eta-1}) \mod q \tag{5}$$

is computed. This is done for each of the 256 coefficients in order to obtain the polynomial, as shown in SamplePolyCBD in Figure 7. In Encrypt, the uniformly random bits must be generated in a deterministic way to ensure that the encryption of a fixed message always returns the same ciphertext. Therefore the SHAKE128 algorithm is used to generate random bits. Encrypt is used in both Encaps and Decaps, so the total time spent for the generation of binomial samples is considerable.

Primitives using the Keccak permutation. Several cryptographic hash and extendible output functions are used in the ML-KEM description. In Section 3.2.1 they are called G, J, H, $\mathsf{H}_{\mathbf{A}}$, $\mathsf{H}_{\mathbf{s},\mathbf{e}}$, and $\mathsf{H}_{\mathbf{y},\mathbf{e}_1,e_2}$. They are all instantiated with primitives based on the Keccak permutation. SHA3-512 is used for function G, SHA3-256 is used for functions H and J, SHAKE256 is used as PRF for $\mathsf{H}_{\mathbf{s},\mathbf{e}}$ and $\mathsf{H}_{\mathbf{y},\mathbf{e}_1,e_2}$, and SHAKE128 is used as XOF for $\mathsf{H}_{\mathbf{A}}$.

Given the total number of Keccak permutations in ML-KEM, the computation of this permutation constitutes the other main performance bottleneck.

6.1.2 Building K-PKE and ML-KEM

Putting the building blocks together, the K-PKE is described in details in Figure 8. The following paragraphs specify how the building block functions from the previous paragraphs are used to implement the K-PKE scheme.

Key generation. The KeyGen takes as input a random seed which is used to derive seeds for the sampling of the secret key part $\mathbf{A}$ and the secret parts $\mathbf{s}$ and $\mathbf{e}$ respectively. The pseudorandom

SampleNTT(B)

Input: byte array $B \in \mathbb{B}^{32}$ // a 32-byte seed along with two indices

Output: array $\hat{a} \in \mathbb{Z}_q^{256}$ // the coefficients of the NTT of a polynomial

1 : ctx := XOF.Init()
2 : ctx ← XOF.Absorb(ctx, B) // input the given byte array into XOF
3 : $j := 0$
4 : **while** $(j < 256)$ **do**
5 : $(\text{ctx}, C) \leftarrow \text{XOF.Squeeze}(\text{ctx}, 3)$ // get a fresh 3-byte array C from XOF
6 : $d_1 \leftarrow C[0] + 256 \cdot (C[1] \mod 16)$ // $0 \leqslant d_1 < 2^{12}$
7 : $d_2 \leftarrow \lfloor C[1]/16 \rfloor + 16 \cdot C[2]$ // $0 \leqslant d_2 < 2^{12}$
8 : **if** $(d_1 < q)$ **then**
9 : $\hat{a}[j] \leftarrow d_1$ // $\hat{a} \in \mathbb{Z}_q^{256}$
10 : $j \leftarrow j + 1$
11 : **if** $(d_2 < q) \wedge (j < 256)$ **then**
12 : $\hat{a}[j] \leftarrow d_2$
13 : $j \leftarrow j + 1$
14 : **return** $\hat{a}$

SamplePolyCBD(B)

Input: byte array $B \in \mathbb{B}^{64\eta}$

Output: array $f \in \mathbb{Z}^{256}$ // the coefficients of the sampled polynomial

1 : $b := \text{BytesToBits}(B)$
2 : **for** $(i \leftarrow 0; i < 256; i\text{++})$ **do**
3 : $x \leftarrow \sum_{j=0}^{\eta-1} b[2i\eta + j]$ // $0 \leqslant x \leqslant \eta$
4 : $y \leftarrow \sum_{j=0}^{\eta-1} b[2i\eta + \eta + j]$ // $0 \leqslant y \leqslant \eta$
5 : $f[i] \leftarrow x - y \mod q$ // $0 \leqslant f[i] \leqslant \eta$ or $q - \eta \leqslant f[i] \leqslant q - 1$
6 : **return** f

Figure 7: Algorithm SampleNTT and SamplePolyCBD.

coefficients for matrix $\mathbf{A}$ are directly sampled in the NTT domain as shown in SampleNTT in Figure 7. The binomial distributed secret vectors are sampled using SamplePolyCBD from Figure 7. The input randomness required for binomial sampling is generated by the PRF. The pseudorandom matrix and the secret vectors are used to compute k MLWE samples in a vector $\mathbf{t}$. These MLWE samples constitute the public key, while the secret vector $\mathbf{s}$ is kept as secret key. Arithmetic computations are sped up using the NTT. The polynomial parts of the keys can be stored in the NTT domain, such that they can be multiplied directly when they are used during Encrypt or Decrypt. The

K-PKE.KeyGen$(;d)$

Input: randomness $d \in \mathbb{B}^{32}$

Output: encryption key $\mathsf{ek}_{\mathsf{PKE}} \in \mathbb{B}^{384k+32}$

Output: decryption key $\mathsf{dk}_{\mathsf{PKE}} \in \mathbb{B}^{384k}$

$(\rho, \sigma) := \mathsf{G}(d\|k)$

// expand 32+1 bytes to two pseudorandom 32-byte seeds

$N := 0$

for $(i \leftarrow 0; i < k; i\texttt{++})$ **do** // generate $\hat{\mathbf{A}} \in (\mathbb{Z}_q^{256})^{k \times k}$

  for $(j \leftarrow 0; i < k; j\texttt{++})$ **do**

    $\hat{\mathbf{A}}[i, j] := \mathsf{SampleNTT}(\rho\|j\|i)$

for $(i \leftarrow 0; i < k; i\texttt{++})$ **do** // generate $\mathbf{s} \in (\mathbb{Z}_q^{256})^k$

  $\mathbf{s}[i] := \mathsf{SamplePolyCBD}_{\eta_1}(\mathsf{PRF}_{\eta_1}(\sigma, N))$

  $N \leftarrow N + 1$

for $(i \leftarrow 0; i < k; i\texttt{++})$ **do** // generate $\mathbf{e} \in (\mathbb{Z}_q^{256})^k$

  $\mathbf{e}[i] := \mathsf{SamplePolyCBD}_{\eta_1}(\mathsf{PRF}_{\eta_1}(\sigma, N))$

  $N \leftarrow N + 1$

$\hat{\mathbf{s}} := \mathsf{NTT}(\mathbf{s})$

$\hat{\mathbf{e}} := \mathsf{NTT}(\mathbf{e})$

$\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$

$\mathsf{ek}_{\mathsf{PKE}} := (\mathsf{ByteEncode}_{12}(\hat{\mathbf{t}})\|\rho)$ // append $\hat{\mathbf{A}}$ seed

$\mathsf{dk}_{\mathsf{PKE}} := \mathsf{ByteEncode}_{12}(\hat{\mathbf{s}})$

return $(\mathsf{ek}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}})$

K-PKE.Decrypt$(\mathsf{dk}_{\mathsf{PKE}}, \mathsf{ct})$

Input: decryption key $\mathsf{dk}_{\mathsf{PKE}} \in \mathbb{B}^{384k}$

Input: ciphertext $\mathsf{ct} \in \mathbb{B}^{32(d_u k + d_v)}$

Output: message $m \in \mathbb{B}^{32}$

$\mathsf{ct}_1 := \mathsf{ct}[0 : 32d_u k]$

$\mathbf{u}' := \mathsf{Decompress}_{d_u}(\mathsf{ByteDecode}_{d_u}(\mathsf{ct}_1))$

$v' := \mathsf{Decompress}_{d_v}(\mathsf{ByteDecode}_{d_v}(\mathsf{ct}_2))$

$\hat{\mathbf{s}} := \mathsf{ByteDecode}_{12}(\mathsf{dk}_{\mathsf{PKE}})$

$w := v' - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^\top \circ \mathsf{NTT}(\mathbf{u}'))$

$m := \mathsf{ByteEncode}_1(\mathsf{Compress}_1(w))$

// decode plaintext m from polynomial w

return m

K-PKE.Encrypt$(\mathsf{ek}_{\mathsf{PKE}}, m; r)$

Input: encryption key $\mathsf{ek}_{\mathsf{PKE}} \in \mathbb{B}^{384k+32}$

Input: message $m \in \mathbb{B}^{32}$

Input: randomness $r \in \mathbb{B}^{32}$

Output: ciphertext $\mathsf{ct} \in \mathbb{B}^{32(d_u k + d_v)}$

$N := 0$

$\hat{\mathbf{t}} := \mathsf{ByteDecode}_{12}(\mathsf{ek}_{\mathsf{PKE}}[0 : 384k])$

$\rho := \mathsf{ek}_{\mathsf{PKE}}[384k : 384k + 32]$

// extract 32-byte seed from $\mathsf{ek}_{\mathsf{PKE}}$

for $(i \leftarrow 0; i < k; i\texttt{++})$ **do** // regenerate $\hat{\mathbf{A}} \in (\mathbb{Z}_q^{256})^{k \times k}$

  for $(j \leftarrow 0; i < k; j\texttt{++})$ **do**

    $\hat{\mathbf{A}}[i, j] \xleftarrow{\$} \mathsf{SampleNTT}(\rho\|j\|i)$

for $(i \leftarrow 0; i < k; i\texttt{++})$ **do** // generate $\mathbf{y} \in (\mathbb{Z}_q^{256})^k$

  $\mathbf{y}[i] \xleftarrow{\$} \mathsf{SamplePolyCBD}_{\eta_1}(\mathsf{PRF}_{\eta_1}(r, N))$

  $N \leftarrow N + 1$

for $(i \leftarrow 0; i < k; i\texttt{++})$ **do** // generate $\mathbf{e}_1 \in (\mathbb{Z}_q^{256})^k$

  $\mathbf{e}_1[i] \xleftarrow{\$} \mathsf{SamplePolyCBD}_{\eta_2}(\mathsf{PRF}_{\eta_2}(r, N))$

  $N \leftarrow N + 1$

$e_2 \xleftarrow{\$} \mathsf{SamplePolyCBD}_{\eta_2}(\mathsf{PRF}_{\eta_2}(r, N))$

$\hat{\mathbf{y}} := \mathsf{NTT}(\mathbf{y})$

$\mathbf{u} := \mathsf{NTT}^{-1}(\hat{\mathbf{A}}^\top \circ \hat{\mathbf{y}}) + \mathbf{e}_1$

$\mu := \mathsf{Decompress}_1(\mathsf{ByteDecode}_1(m))$

$v := \mathsf{NTT}^{-1}(\hat{\mathbf{t}}^\top \circ \hat{\mathbf{y}}) + e_2 + \mu$

// encode plaintext m into polynomial v

$\mathsf{ct}_1 := \mathsf{ByteEncode}_{d_u}(\mathsf{Compress}_{d_u}(\mathbf{u}))$

$\mathsf{ct}_2 := \mathsf{ByteEncode}_{d_v}(\mathsf{Compress}_{d_v}(v))$

return $\mathsf{ct} := (\mathsf{ct}_1\|\mathsf{ct}_2)$

Figure 8: Specification of algorithms K-PKE.KeyGen, K-PKE.Encrypt, and K-PKE.Decrypt. See Figure 2 for the high level pseudocode.

pseudorandom matrix is not stored as a key, instead the seed that allows to re-compute this matrix is stored.

Encryption. In Encrypt the function SampleNTT is used to expand the public key seed, and SamplePolyCBD to sample from the binomial distribution. Similarly to KeyGen, k MLWE samples are computed for the pseudorandom matrix, using polynomial arithmetic in the NTT domain. A separate MLWE sample is computed for part $\mathbf{t}$ of the public key. The random 32-byte message is encoded using BytesToBits and Decompress as described in the paragraphs above. The encoded message is then added to the MLWE sample. All MLWE samples are then compressed using Compress in order to reduce the size, before being returned as ciphertext.

Decryption. Since the ciphertext was compressed during Encrypt, the first step of Decrypt is decompressing the two ciphertexts parts using Decompress. They are in the normal domain, so the polynomials of ciphertext part $\mathbf{u}'$ must be mapped to the NTT domain before multiplying them with the secret key polynomials which are already in the NTT domain. The product is mapped back to the time domain using the inverse NTT, before subtracting the result from ciphertext part v'. As described in Section 3.1, the result is a polynomial whose coefficients are close to 0 or close to $\frac{q}{2}$. The $\mathsf{Compress}_d$ function with $d = 1$ is applied to obtain a bit vector, which is then encoded to bytes using ByteEncode from Figure 4.

ML-KEM. The functions described above, including Encrypt, Decrypt, KeyGen, G, H and J, together define the Key Generation, Encaps and Decaps from the IND-CCA scheme. Figure 9 shows in detail how these functions are used. The design principle of ML-KEM was discussed in Section 3.1.

6.2 Side Channel Attacks on ML-KEM

6.2.1 Introduction

Side Channel Attacks (SCA) exploit leakage of information through *side channels* during the execution of a cryptographic algorithm on a device in order to recover secret information. Any physical observable can be used as side channel, and the most widely used side channels are the computation time, power consumption and electromagnetic emanation (EM) of the device that performs the cryptographic computations. The background provided here uses power analysis as example, but EM measurements can be used in an equivalent manner.

Simple power analysis. Power analysis on the RSA signature generation scheme is a well known example of a side channel attack. The exponentiation in RSA is computed using the square-and-multiply algorithm, where squaring operations and multiplications are computed in an order depending on the bits of the secret exponent. In unprotected devices, the power consumption of the device observed during a squaring operation might be slightly different from the power consumption during a multiplication. Therefore, by measuring and analyzing the power consumption of the device during the exponentiation, a SCA attacker may recover the operations sequence of squarings and multiplications. This sequence is uniquely determined by the value of the secret exponent, which can thus be recovered by a SCA attacker. This attack is an example of *Simple Power Analysis* (SPA), where the power consumption is measured during one single execution of the cryptographic algorithm. The power *trace* (measurement) is analyzed in order to recover the secret key directly.

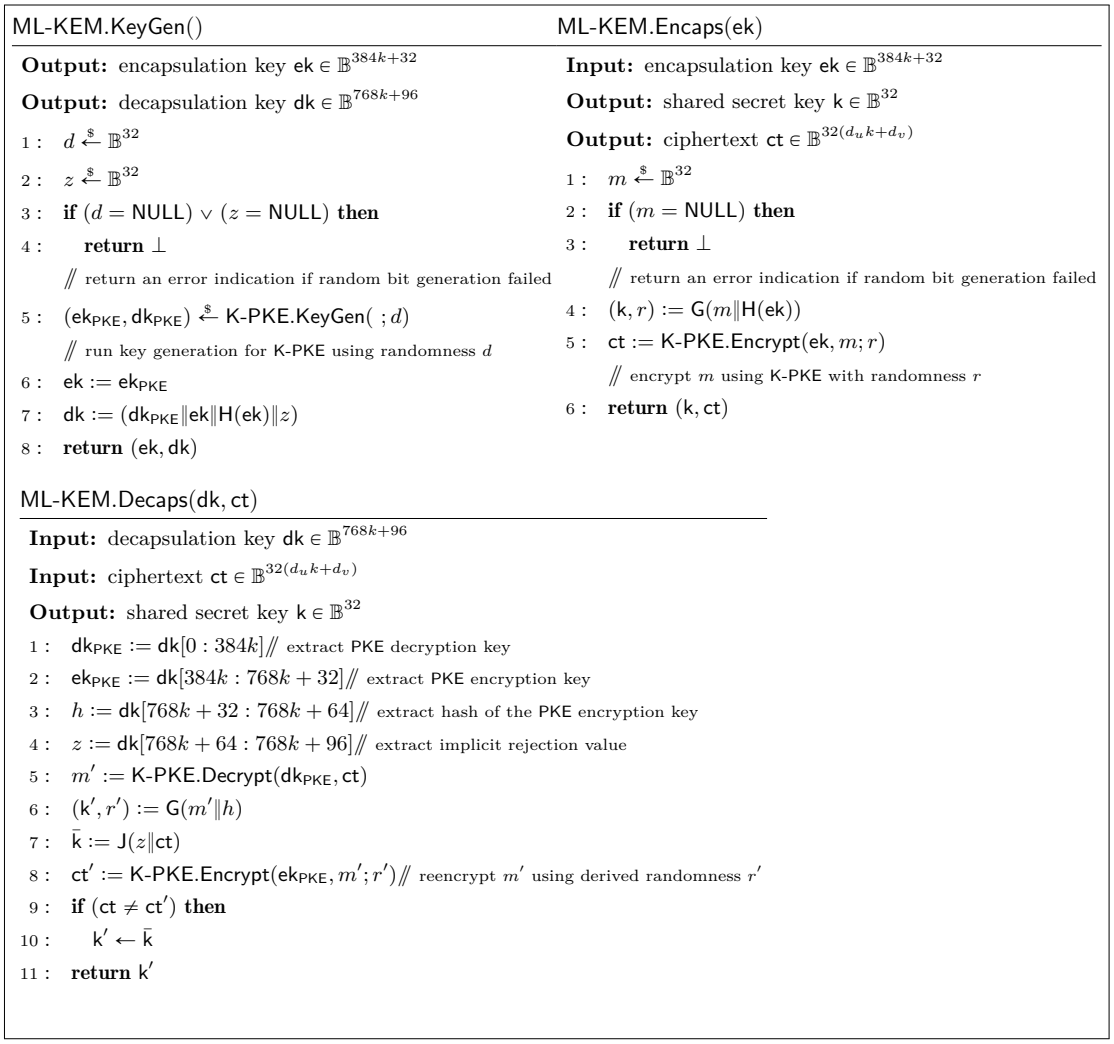

Figure 9: Specification of algorithms ML-KEM.KeyGen, ML-KEM.Encaps, and ML-KEM.Decaps running K-PKEas a sub-routine. See Figure 3 for the high level pseudocode.

Differential power analysis. Depending on the target device, the information leakage from a single power measurement might not be sufficient for full key recovery. In the Hamming weight model, it is assumed that an attacker can obtain (an approximation of) the Hamming weight of the bytes (or words) that are processed by the device. A power trace of the device computing an AES [Aes] may for example leak the Hamming weights of the bytes of the key. This is not sufficient to directly recover the value of the secret key.

In this case the attacker may try a divide-and-conquer strategy using multiple power measurements and targeting the key bytes one by one. Suppose that the attacker can trigger encryptions

for known plaintext on the target device which uses a fixed unknown key. By combining knowledge of the plaintext and the recovered Hamming weight of the intermediate state bytes during the first round, information about the key bytes can be obtained. The Hamming weight of the intermediate state bytes right after the AES' SubBytes operation (see [Aes]), which computes the S-box permutation on the bytes of the state of the first round provides valuable information. For each of the 256 possible guesses of the first byte of the first round key, the first byte of the intermediate state can be computed by XOR'ing the plaintext byte with the key guess and applying the S-Box permutation. Only a subset of those 256 possible intermediate state bytes will have the same Hamming weight as was observed by analyzing the power trace. Those key guesses that result in intermediate state bytes with a different Hamming weight are eliminated, so that the number of possible guesses for the first byte of the secret key is reduced. Repeating the same procedure for a different input plaintext will, with high probability, even further reduce the number of remaining possibilities for the first key byte. After analyzing a certain number of traces, only one single guess will remain for the first key byte. The same can be done to recover all other 15 byte positions so that the first round key is completely recovered. This divide-and-conquer strategy in which the key bytes are targeted separately by exploiting new information from each trace is referred to as *Differential Power Analysis* (DPA).

In practice, the power measurements may be noisy such that only a rough approximation of the Hamming weight of the processed bytes can be obtained from trace analysis. In that case, the attacker will have to increase the number of traces to be measured. For each trace and for each of the 256 possible key byte guesses, the Hamming weight of the intermediate state bytes is computed in the same way as described above. In *Correlation Power Analysis* (CPA), the attacker computes the correlation between the power measurements and the computed intermediate state bytes for each of the 256 key byte guesses. If a sufficient amount of traces is used (or if the noise level is sufficiently low), the highest correlation is obtained for the correct key guess.

Template attacks. In some cases, an attacker might be able to acquire a device similar to the target device. On this *clone* device, the attacker has full control over the keys, and has therefore knowledge of all the intermediate values during the computations on the clone device. This ability can be used to create a model of the clone device, which consists of templates that describe the power consumption as a function of the intermediate values during computation. If the power consumption of the clone device is sufficiently similar to the power consumption of the target device, then the power model can be used to predict the values of intermediates given power traces of the target device. Templates derived from a clone device can be used to enhance SPA, DPA or CPA attacks.

6.2.2 Side Channel Attacks on Decapsulation

During the Decrypt part of Decaps, part $\mathbf{s} = (\mathbf{s}_0, \ldots, \mathbf{s}_{k-1})$ of the decapsulation key dk is multiplied by part $\mathbf{u} = (\mathbf{u}_0, \ldots, \mathbf{u}_{k-1})$ of the input ciphertext. This section will focus on the first polynomial multiplication between $s := \mathbf{s}_0$ and $u := \mathbf{u}_0$ only, as the others are treated the same way and can therefore be attacked using the same methods.

Leakage of intermediate products during the multiplication can be exploited by a CPA attacker in order to recover s. The multiplication is computed in the NTT domain, such that the polynomial multiplication $s \cdot u$ consists of 128 multiplications of degree 1 polynomials, as can be seen in the MultiplyNTTs description in Figure 6.

Multiplications between degree 1 polynomials $u_0 + u_1 \cdot x$ and $s_0 + s_1 \cdot x$ are computed using the BaseCaseMultiply algorithm from Figure 6 and include the computation of the product $u_0 \cdot s_0 \mod q$. If the target device leaks the Hamming weight of $u_0 \cdot s_0 \mod q$ through side channels during this computation, then a DPA attacker can exploit this to recover s_0 by following the steps of this subroutine:

1. Generate a list of N_{traces} random ciphertexts and save the u_0 of each ciphertext in a list $U := u_0^{(0)}, \ldots, u_0^{(N_{\text{traces}}-1)}$.
2. Execute the decapsulation on the target device for each of the ciphertexts in the list and measure the power traces $T^{(0)}, \ldots, T^{(N_{\text{traces}}-1)}$, where each trace $T^{(i)}$ consists of N_{samples} measured samples: $T^{(i)} = T_0^{(i)}, \ldots, T_{N_{\text{samples}}-1}^{(i)}$.
3. For each candidate $\hat{s} \in \mathbb{Z}_q$ compute the *prediction vector* $P_{\hat{s}}$, which is computed as the Hamming weight $\text{HW}(\cdot)$ of the products:

$$P_{\hat{s}} = \text{HW}(U \cdot \hat{s}) = \text{HW}(u_0^{(0)} \cdot \hat{s} \mod q), \ldots, \text{HW}(u_0^{(N_{\text{traces}}-1)} \cdot \hat{s} \mod q) \tag{6}$$

4. If the traces leak the Hamming weight of the $u_0 \cdot s_0 \mod q$, then P_{s_0} must have high correlation with the vector $T_i^{(0)}, \ldots, T_i^{(N_{\text{traces}}-1)}$ for some trace sample i and the correct key guess s_0. Therefore the attacker computes the correlation $\rho_{\hat{s},j}$ between $P_{\hat{s}}$ and $T_j^{(0)}, \ldots, T_j^{(N_{\text{traces}}-1)}$ for each candidate $\hat{s} \in \mathbb{Z}_q$ and each trace sample $j = 0, 1, \ldots, N_{\text{samples}} - 1$.
5. The $\hat{s}$ for which $\rho_{\hat{s},j}$ is maximum is the correct key guess.

If the device leaks the Hamming weight of $u_i \cdot s_i \mod q$ for all $0 \leqslant i < 256$, then the complete secret key polynomial s can be recovered by repeating the process of each coefficient. The subroutines that recover the 256 coefficients are independent from one another and can therefore be performed in parallel.

Related work. The first CPA on lattice-based schemes was presented by [Rep+15]. The same framework was adapted by [Muj+24] for various polynomial multiplication algorithms, and also applied to the first version of KYBER, a predecessor of ML-KEM. The use of an incomplete NTT in ML-KEM makes the CPA slightly more complicated [Alp+24] because coefficients at odd indices are treated differently from those at even indices.

6.2.3 Side Channel Assisted Chosen Ciphertext Attacks

The re-encryption and ciphertext comparison steps in lines 5 and 6 of the Decaps algorithm in Figure 3 make sure that the decapsulation output does not reveal any information about the decrypted message m in line 6 of the Decrypt algorithm in Figure 2. This is necessary because a chosen ciphertext attacker may craft specific ciphertexts for which the bits of the decrypted message m reveal information about the secret key. However, by exploiting side channel leakage during the decapsulation, it may still be possible to perform chosen ciphertext attacks. The following paragraphs discuss the various side channel assisted chosen ciphertext attacks that are applicable to ML-KEM.

Implementing a plaintext-checking oracle with SCA. A Plaintext-Checking (PC) oracle tells whether the decrypted message is equal to the zero message $m = 000\ldots00$ or to the non-zero message $m = 100\ldots00$. In order to use a PC oracle for secret key recovery, specific input ciphertexts must be crafted. Let CT be the ciphertext for which $\mathbf{u}'_0 = k_u$, $v' = k_v$ for some constants $k_u, k_v \in \mathbb{Z}_q$, and $\mathbf{u}'_i = 0$ for $1 \leqslant i < k$. Then line 5 in Decrypt computes

$$\begin{aligned} w &= v' - \mathbf{s}^\top \cdot \mathbf{u}' \\ &= k_v - k_u \cdot \mathbf{s}_0. \end{aligned}$$

Writing $s := \mathbf{s}_0$, the first coefficient of w is equal to $w_0 = k_v - k_u \cdot s_0$ while the other coefficients $w_i = -k_u \cdot s_i$ for $i = 1, \ldots, 255$.

The next step in Decrypt is $\mathsf{Compress}_1$ which maps elements from $\mathbb{Z}_q$ to 0 or 1 depending on whether they are closer to 0 mod q or to $\frac{q}{2}$. Since all the coefficients of s are within the small interval of $\{-\eta, \ldots, \eta\}$, it is possible to choose constants k_u and k_v such that Decrypt computes $\mathsf{Compress}_1(w_i) = 0$ for all $i > 0$, and $\mathsf{Compress}_1(w_0)$ is equal to 0 or 1 depending on the value of s_0 only. For instance, choosing k_v close to $\frac{q}{4}$ and k_u a positive constant slightly smaller than $\frac{q}{4\eta}$, then for $i > 0$, $|w_i| = |k_u \cdot s_i| < \frac{q}{4}$ for any $s_i \in \{-\eta, \ldots, \eta\}$, such that it compresses to 0 independently of s_i. For the first coefficient, if $s_0 > 0$ then it holds that $w_0 = k_v - k_u \cdot s_0 > \frac{q}{4}$ such that it compresses to 1, and if $s_0 < 0$ then $w_0 < \frac{q}{4}$ such that it compresses to 0. Therefore knowing whether $m = 000\ldots00$ or $m = 100\ldots00$ can enable an attacker to recover 1 bit of information about s_0.

The whole coefficient s_0 can be recovered by using the PC oracle for various different pairs of constants k_u, k_v. Other coefficients of s can be targeted by exploiting the cyclic nature of multiplication by x in R_q and repeating the same procedure with $\mathbf{u}'_0 = k_u \cdot x^i$ for $i = 1, \ldots, 255$.

SCA can be used in order to instantiate a PC oracle. As can be seen in line 5 of Decaps in Figure 3, the inputs to Encrypt depend only on the public key and the decrypted message m'. When using a constant public key, the ciphertext computed by Encrypt is completely determined by m'. If $m = 000\ldots00$ then the output of Encrypt and almost all of the intermediate values computed during Encrypt are completely different from those that are computed in the case where $m = 100\ldots00$. In other words, many intermediate values computed by Encrypt depend on a single bit of information, which is the same bit that a PC oracle should return. The PC oracle can be instantiated in the following way:

1. Craft two ciphertexts CT_0 and CT_1, where CT_0 decrypts to $000\ldots00$ and CT_1 decrypts to $100\ldots00$. This can be done using the Encrypt method, and knowledge of the secret key used during decryption is not required as long as the correct public key is used.

2. Perform a number of decapsulations using inputs CT_0 and CT_1, and record the power traces during the computation, such that two sets S_0 and S_1 of power traces are obtained. Each set S_i for $i \in \{0, 1\}$ contains power traces of decapsulations of CT_i only.

3. Use S_i to create models M_i for $i \in \{0, 1\}$ that describe the power consumption of the device during decapsulation of CT_i. One method could be for instance to let M_i be the average power trace in set S_i. The models can be used as a PC oracle by taking as input a power trace and using models M_i to check whether the power trace is more likely to correspond to a decapsulation of $000\ldots00$ or $100\ldots00$.

During the attack phase of a PC oracle SCA, a ciphertext of a special form is crafted using constants k_u and k_v as described earlier. A power trace is measured during the decapsulation of the ciphertext, and given to the PC oracle which returns one bit of information about the secret key. This process is repeated until all coefficients of the secret key are recovered.

The re-encryption is not the only part of Decaps that can be targeted by PC oracle SCA. Lines 3 and 6 of Decaps in Figure 3 are also uniquely determined by the decrypted message. Therefore, all the intermediates during the computation of hash function G depend on the same secret key dependent bit of information. The re-computed ciphertext is compared to the input ciphertext in line 6, which means that the comparison operation is also a potential target.

Related work. One of the first PC oracle attacks on ML-KEM was presented by [Rav+20b], in which they show that the attack is generic and applies to many lattice-based KEMs that use the F-O transform. It was shown by [Uen+22] that a similar SCA framework is also applicable to several code-based KEMs.

Even if the leakage is insufficient to instantiate a perfect PC oracle, [She+23] showed that it is still possible to recover the ML-KEM secret key using probabilistic PC oracles. One way to use a PC oracle that is correct with some probability < 1 is to repeatedly query it by using multiple power traces.

The work by [Raj+23] reduces the number of traces required for key recovery by attacking multiple secret key coefficients simultaneously. Instead of a binary distinguisher, they use leakage throughout the decapsulation to create a multiple-bit distinguisher.

Implementing a full decryption oracle with SCA. A Full Decryption (FD) oracle returns all the bits of the decrypted message that is computed during Decrypt. A successful a message recovery attack allows to re-compute the shared secret by concatenating it with the public key hash digest h and computing the hash function G, as shown in line 3 in Decaps. Moreover, message recovery can also be used to recover the secret key. Given a specifically crafted input ciphertext similar to those used in PC oracle attacks, the bits of the decrypted message contain information about the secret key. While in PC oracle attacks both ciphertext parts $\mathbf{u}_0$ and v are set to constants (degree 0 polynomials), in FD oracle attacks the input ciphertext part v is set to $k_v \sum_{i=0}^{n-1} x^i$. All 256 coefficients of intermediate $w = v' - \mathbf{s}^\top \cdot \mathbf{u}'$ of the decrypted ciphertext then have the same behaviour as decribed in the previous section for the first coefficient in the context of PC oracle attacks. Each coefficient is compressed to either 0 or 1 depending on the value of the corresponding secret key coefficient. Therefore, for all 256 bits of the decrypted message, the i-th bit reveals one bit of information about the i-th secret key coefficient.

Only the operations that process all bits of the decrypted message can be targeted by FD oracle SCA. These operations are ByteEncode and Compress in line 6 of Decrypt in Figure 9 and ByteDecode and Decompress in line 16 of Encrypt during the re-encryption in Decaps.

Related work. In [Xu+22] a Kyber-512 secret key is recovered using only 8 power traces, by recovering the complete decrypted message for each trace.

Subsequent works have focused on the same target operations in implementations that use countermeasures to protect against SCA. A method to defeat the shuffling countermeasure was presented by [Rav+22]. An implementation that used both shuffling and masking was shown to be still vulnerable against FD oracle attacks by [Bac+23; Jen+23]. An attack on the same target operation protected by higher order masking was presented by [Dub+23].

Implementing a decryption failure oracle with SCA. A decryption failure occurs if during the decapsulation of some ciphertext, the decrypted m' in line 2 of Figure 3 is different from the m that was encrypted in line 3 of Encaps when the ciphertext was generated. In case of Decryption Failure, the ciphertext that is re-computed during Decaps will be different from the input ciphertext. Then Decaps returns a bogus shared secret that is different from the one obtained during Encaps.

Decryption Failure (DF) oracles tell whether or not a decryption failure occurred during Decaps in line 2 of Figure 3.

The occurance of a decryption failure for a valid ciphertext can leak information about the secret key. The decryption computes

$$\begin{aligned} w &= v' - \mathbf{s}^\top \cdot \mathbf{u}' + \delta_{\text{compress}} \\ &= \mathbf{t}^\top \cdot \mathbf{y} + e_2 + \mu - \mathbf{s}^\top \cdot (\mathbf{A}^\top \cdot \mathbf{y} + \mathbf{e}_1) + \delta_{\text{compress}} \\ &= (\mathbf{A} \cdot \mathbf{s} + \mathbf{e})^\top \cdot \mathbf{y} + e_2 + \mu - \mathbf{s}^\top \cdot (\mathbf{A}^\top \cdot \mathbf{y} + \mathbf{e}_1) + \delta_{\text{compress}} \\ &= \mathbf{e}^\top \cdot \mathbf{y} + e_2 + \mu - \mathbf{s}^\top \cdot \mathbf{e}_1 + \delta_{\text{compress}}, \end{aligned}$$

where δ_{compress} denotes the small errors introduced by the compression and decompression, which can be computed by the attacker who created the ciphertext using the encapsulation algorithm. The error terms $\mathbf{e}_1$, $\mathbf{y}$ and e_2, and the message μ are also part of the encapsulation and therefore known to the attacker. If the decryption succeeds, then the sum of all the error terms must be smaller than $\frac{q}{4}$ in absolute value, such that $\mathsf{Compress}_1(w)$ computes exactly the same message bits as those used during encapsulation. If, however, the decryption fails, then that means that the absolute value of the sum of all error terms exceeds $\frac{q}{4}$ in at least one coefficient index i. The attacker obtains a linear inequality for unknown variables $\mathbf{s}$ and $\mathbf{e}$:

$$|(\mathbf{e}^\top \cdot \mathbf{y})_i + e_{2,i} - (\mathbf{s}^\top \cdot \mathbf{e}_1)_i - \delta_{\text{compress},i}| > \frac{q}{4} \tag{7}$$

This provides some information about secret key parts $\mathbf{s}$ and $\mathbf{e}$. Given many inequalities of this form, a system of inequalities is obtained. This system may be solved for $\mathbf{s}$ and $\mathbf{e}$, so that the secret key can be obtained.

Since the probability of such a decryption failure occurring for a valid ciphertext is negligible, the attacker may add $\frac{q}{4}$ to one of the coefficients of v', such that a decryption failure is triggered with a non-negligible probability.

Related work. The impact of decryption failures on the security of lattice-based crypto schemes was studied by [DVV18]. Decryption failures occur with negligible probability for valid ciphertexts, but this can be improved by crafting (invalid) chosen ciphertexts and using SCA leakage to detect decryption failures. It was shown by [GJN20] that side channel leakage could be used to instantiate a DF oracle for lattice-based schemes. Multiple works [Bha+21; Uen+22] have shown that practical implementations of ML-KEM can be successfully targeted by SCA-based DF oracle attacks. The number of traces required for key recovery in the case of imperfect SCA-based DF oracles was studied by [Her+23].

6.3 Countermeasures Against Side Channel Attacks

In order to protect against side channel attacks, countermeasures must be implemented. Algorithmic countermeasures aim to reduce or remove the statistical dependencies between processed data and

static secret keys.

Masking. *Masking* is a particularly effective technique in which each secret key dependent variable x is decomposed in random *shares* before computing the cryptographic algorithm. To protect a multiplication $x \cdot y \mod q$, where x is a secret key and y is public, first a random mask $r \xleftarrow{\$} \mathbb{Z}_q$ is generated, and the decomposition $x = x_0 + x_1 \mod q$ is computed as $x_0 = r$ and $x_1 = x - x_0 \mod q$. Note that the mask must be a fresh uniformly random value for each execution of the algorithm. The multiplication is computed on the two shares separately $x_0 \cdot y$ and $x_1 \cdot y$. All the operands during the masked multiplication are statistically independent of the secret key x. Therefore, side channel leakage of the operands cannot be used to recover information about the secret key. After computing the complete cryptographic algorithm in multiple shares, the correct output can be obtained by combining the two output shares, in this particular example by addition of the shares in $\mathbb{Z}_q$. The example uses only two shares, but the masking order can be increased to protect against higher order side channel attacks.

Depending on the type of operation to protect, either arithmetic masking (as in the example above) can be used or boolean masking. Boolean masking is suitable for protecting boolean operations. ML-KEM uses both arithmetic and boolean operations. In order to protect both types of operations, *A2B* (Arithmetic to Boolean) and *B2A* (Boolean to Arithmetic) mask conversion algorithms must be used to switch between boolean and arithmetic masking without unmasking any intermediate values. The first masked implementations of predecessors of ML-KEM were created by [Ode+18] and [Rep+15]. Masking gadgets for all operations in ML-KEM can be found in [Bos+21] and [Hei+22].

The inconvenience of masking is that the computation time of the cryptographic algorithm is multiplied by the number of shares. In practice the performance overhead is even greater due to mask conversions and other non-linear operations, which are harder to protect.

Multiplicative masking is a technique that randomizes inputs or secret keys by multiplying them with a random scalar at the start of the computation. After processing all arithmetic operations, the randomization is undone by multiplying the result with its multiplicative inverse in $\mathbb{Z}_q$. This technique is sometimes referred to as *blinding*, and was first described for ML-KEM's predecessors by [Saa18]. In [Rav+20a] the NTT in ML-KEM is protected by using blinding. The advantage of multiplicative masking is the limited computation overhead: all arithmetic operations are still only computed once, unlike for additive masking where the number of computations to be performed is doubled. A drawback of this technique is that is only protects operations that are linear over $\mathbb{Z}_q$. Many non-linear operations such as the Keccak-based primitives or the binomial sampling cannot be protected in this manner and must rely on boolean masking.

Hiding. The goal of hiding countermeasures is to dissimulate the true location of the target operation in the power trace. There are many different ways to obtain such an effect. For instance, the *shuffling* countermeasure consists of processing several independent operations in random order. The 128 calls to the BaseCaseMultiply routine for example, are all independent from one another. It does not matter if the first call processes the first degree 1 polynomials or any of the 127 others. The polynomial multiplication remains correct as long as each of the degree 1 polynomials is processed exactly once during MultiplyNTTs. By randomizing the processing order, the SCA attacker is unable to locate with certainty the exact samples of the power trace that should be targeted, thus complicating the attack. Shuffling was used in [Rav+20a] to randomize the computation order of operations

inside the NTT. The hardware implementation by [XWT25] also increased SCA resistance by using shuffling.

Other hiding countermeasures aim to degrade the quality of the power trace, by for instance causing misalignment between different power traces, or increasing the noise level of the measurements. Trace misalignment complicates SCA because the exact location of the trace samples that must be targeted varies from one trace to another. This complicates the detection of statistical dependencies between power traces and processed data.

6.4 Performance Results in Hardware

This section contains a selection of performance results reported in publications in the state of the art. Table 7 shows the cycle counts, computation time and area usage for several works. The comparison focuses on pure hardware implementations only, therefore benchmark results on CPUs are not included. NIST asked the submitters to include two benchmarks using Intel Haswell CPUs and ARM Cortex-M4 CPUs, which can be found in the round 3 KYBER documentation [Ava+21]. They are not included in the comparison of Table 7 because memory usage in a CPU is not comparable to physical chip area on FPGA. The the number of cycles on a FPGA depends on the level of parellelization, while CPUs compute instruction sequentially.

The performance numbers, both in terms of speed and area, depend strongly on the type of FPGA that is used. An implementation synthesized for higher end FPGAs such as the Virtex-7 will have lower area usage and higher speed than when synthesized for a lower end FPGA such as the Artix-7. This is because not all FPGAs implement look-up tables (LUT), flipflops (FF), BRAM (block RAM) and DSP (digital signal processors) in the same way. Even implementations for the same FPGA family are difficult to compare. In [DMG23] the XC7A200 device is used, while [XL21] used the XA7A12 variant, which are slightly different. The fastest implementation was made by [DMG23], while the most compact implementation, i.e. with the smallest area footprint, is the one by [XL21].

When adding SCA countermeasures there is a performance impact to be expected, both on speed and area usage. The two entries in the table for the work by [Kam+22] show that the impact is indeed considerable, even though they used a high end FPGA.

Table 7: Comparison of FPGA Implementations of ML-KEM (Kyber)

Reference	Parameter	Cycles (×1000) (K / E / D)	Freq (MHz)	Time (μs) (K / E / D)	Area				FPGA
					LUT (×1000)	FF (×1000)	DSP	BRAM	
[XL21]	Kyber-512 Kyber-768 Kyber-1024	3.8 / 5.1 / 6.7 6.3 / 7.9 / 10.0 9.4 / 11.3 / 13.9	161	23.4 / 31.5 / 41.4 39.2 / 49.2 / 62.4 58.3 / 70.3 / 86.4	7.4	4.6	2	3	Artix-7
[Hua+20]	Kyber-512 Kyber-768 Kyber-1024	- / 48.0 / 68.8 - / 77.5 / 102.1 - / 107.1 / 135.6	155 192	- / 366 / 444 - / 564 / 686 - / 802 / 975	88.9 110.2 132.9	152.9 167.3 172.5	354 292 548	202 202 202	Artix-7
[DMG23]	Kyber-512 Kyber-768 Kyber-1024	2.2 / 3.2 / 4.5 2.6 / 3.7 / 4.9 3.6 / 4.8 / 5.8	220	10.0 / 14.7 / 20.5 12.0 / 17.0 / 22.2 16.2 / 21.7 / 26.4	9.5 10.5 11.6	8.5 9.8 11.1	4 6 8	4.5 6.5 8.5	Artix-7
(Hiding-only) [Kam+22]	Kyber-512	- / 882 / 1266	100	- / 88.2 / 126.6	153.9	-	60	294	Virtex-7
(Hiding+Masking) [Kam+22]	Kyber-512	- / 881 / 1377	100	- / 88.1 / 137.7	163.6	-	76	489.5	Virtex-7

Note: K = KeyGen, E = Encapsulation, D = Decapsulation. Kamucheka et al. figures are for Kyber-512 on Virtex-7 (VC707).

7 Application on ML-KEM: Transport Layer Security

In this section, we will discuss the Transport Layer Security (TLS) protocol, how we can integrate ML-KEM to provide post-quantum confidentiality, and what the performance impact of this change is. We will first explain TLS, before discussing how ML-KEM is integrated into TLS and the performance impacts. We will also discuss post-quantum/traditional (PQ/T) "hybrids" of classic and post-quantum cryptography, and discuss the ongoing deployment of TLS in the web. Note that although we will briefly mention post-quantum authentication using post-quantum digital signatures, we will not go in detail. Parts of this section are based on [Wig24].

7.1 Transport Layer Security Version 1.3

The Transport Layer Security protocol, the current version of which is TLS 1.3, is defined by the Internet Engineering Task Force (IETF) standard RFC 8446 [Res18]. TLS, also known as SSL (which is the name of its original versions developed by Netscape), is well known for being the 'S' component in 'HTTPS', used in secure web browsing [Res00]. However, TLS is widely used in many contexts that require a secure channel, including secure email [New99; Hof02], file transfer [FH05], and VPN connections [Opeb].

TLS consists of two sub-protocols. For the actual encrypted transmission of application data, the *record layer* protocol uses symmetric-key authenticated encryption algorithms. Because these are quantum-secure, we will not further discuss the record layer. The keys that the record layer uses are computed in the *handshake* protocol. This is an authenticated key exchange algorithm which performs an ephemeral key exchange to compute encryption keys, and authenticates the server using digital signatures. TLS 1.3 optionally supports authentication of the client, but this is not particularly relevant for this document as we are chiefly concerned with ML-KEM, a key exchange algorithm.

A high-level overview of TLS 1.3 is shown in Figure 10. In the initial message by the client, it samples a new Elliptic Curve Diffie–Hellman (ECDH) private key x and sends the corresponding public value xG to the server. The server uses this value with its sampled ECDH private key y to compute a shared secret key ss. Traffic encryption keys are derived from ss using a Key Derivation Function (KDF). The server responds to the client's message by sending its ECDH public value yG. This allows the client to also compute ss and the traffic encryption keys. The server also sends its identity and signature public key in a *certificate*, plus a signature over the exchanged messages (i.e., the *transcript*). This signature proves that the server owns the private key corresponding to the authenticated public key in the certificate. Finally, it sends a key confirmation message, after which the server can start sending encrypted application data. The client will confirm its view on the handshake by also sending a key confirmation message, after which the handshake is completed.

7.2 TLS with Post-Quantum Confidentiality

To provide security against quantum adversaries in TLS 1.3, we can, in principle, straightforwardly replace all pre-quantum algorithms by post-quantum primitives.

We can replace the pre-quantum elliptic-curve Diffie–Hellman (DH) key exchange algorithms by ML-KEM, to provide post-quantum confidentiality. This is particularly important when considering "harvest-now-decrypt-later" attacks.

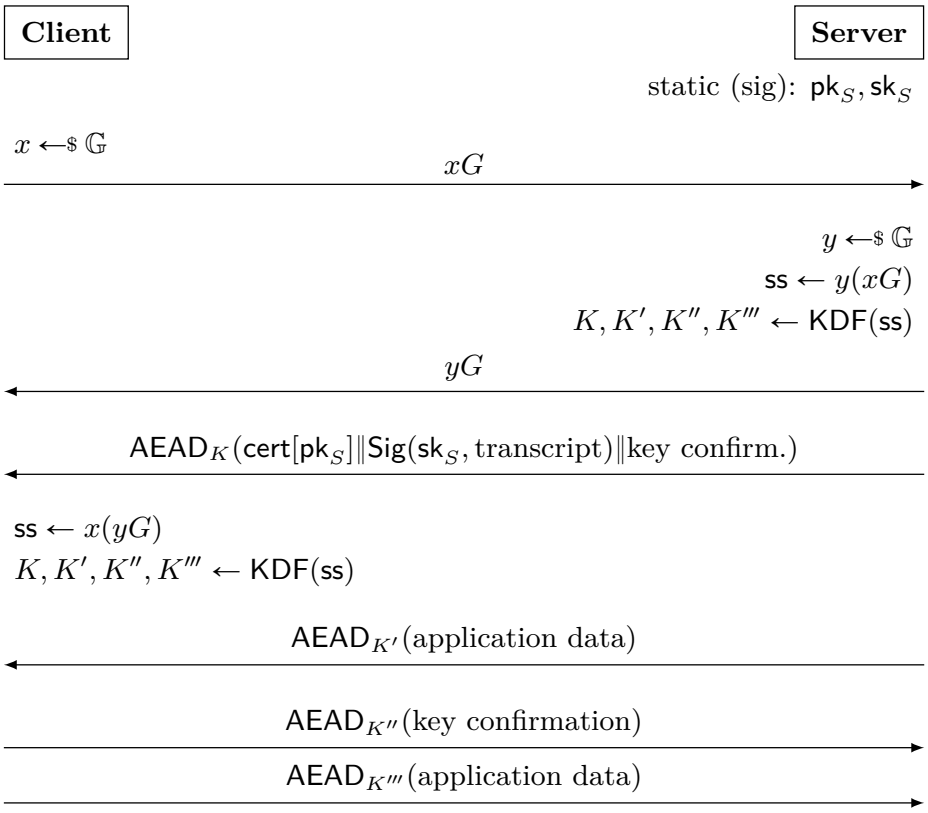

Figure 10: High-level overview of the TLS 1.3 handshake.

The way we replace DH by ML-KEM in TLS 1.3 is straightforward.[2] The client still sends an encapsulation key to the server, though now it is generated using ML-KEM's key generation functionality. The server's behavior changes a bit more: instead of generating an encapsulation key, it calls the KEM.Encapsulate function of ML-KEM and transmits the generated ciphertext.

The signature algorithms used in the handshake and certificates, which currently are based on RSA or elliptic-curve signatures, can simply be replaced by post-quantum signature algorithms, as they provide the same functionality.[3] As this document focuses on ML-KEM, a post-quantum *key exchange* algorithm, we will not further treat this issue, and in all measurements we are assuming *classical authentication* algorithms.

A high-level overview of TLS 1.3 using post-quantum ML-KEM in place of DH is shown in Figure 11.

[2]Though many uses of DH key exchange can be replaced by ML-KEM, this is not true in general. In particular, when DH is used for authentication, ML-KEM may not be suitable. There are unfortunately no practical, truly drop-in post-quantum replacements for Diffie–Hellman, so each protocol will need to be evaluated for compatibility.

[3]Practically, there are significant concerns on the transition to post-quantum authentication due to the large number of signatures involved, combined with the large sizes of post-quantum signature schemes.

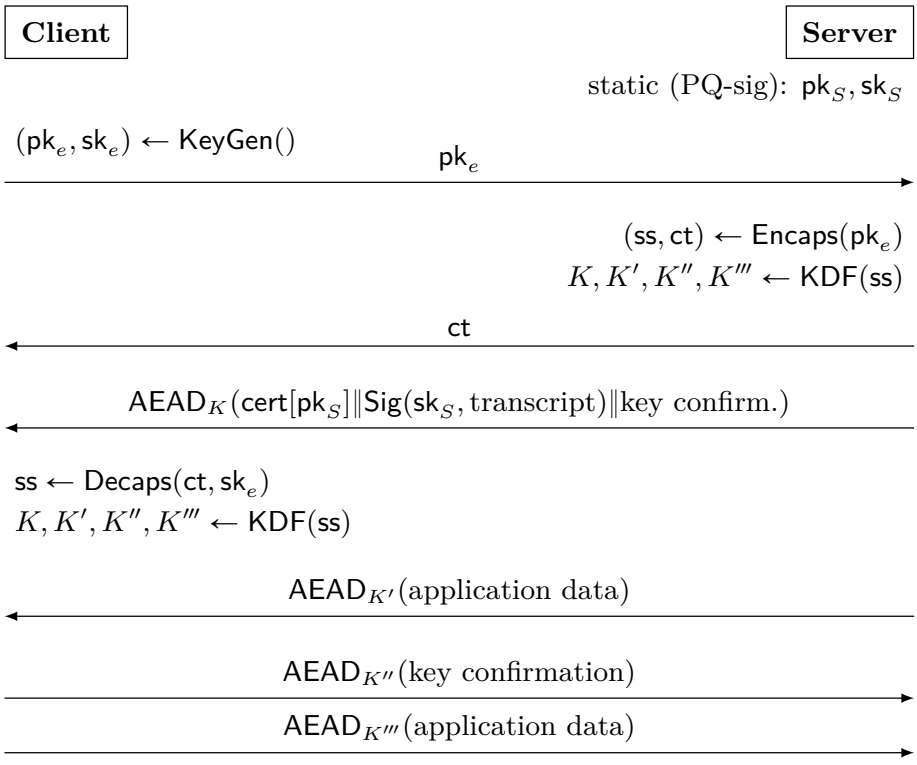

Figure 11: A high-level overview of TLS 1.3 with post-quantum ML-KEM. The ECDH ephemeral key exchange is replaced by ML-KEM operations KeyGen, Encaps, and Decaps. Note that it is possible to use post-quantum KEM without using post-quantum signatures, which provides post-quantum confidentiality against harvest-now-decrypt-later attacks. If post-quantum signatures are used in place of the classical signatures, this protocol is fully post-quantum secure against active quantum attackers.

7.3 Comparing ML-KEM to ECDH Key Exchange Algorithms

Although there are no theoretical obstacles to integrating ML-KEM into TLS 1.3, there are other considerations that affect practical use, notably overhead. In particular, almost all use cases are concerned with runtime performance and bandwidth usage.[4] In this section, we compare these characteristics for key exchange algorithms as used in TLS. This includes the currently-used classic or "traditional" algorithms and "pure" post-quantum key exchange methods (i.e., ML-KEM), but also combinations of traditional and post-quantum algorithms (also called hybrid algorithms, see Section 7.4).

[4]Some platforms may also be concerned with implementation size, but it is difficult to make general statements about the implementation size of any algorithm, including ML-KEM, without making assumptions on the target platform and technology. However, some representative benchmarks can be found at the pqm4 project [Kan+].

Table 8: Performance of cryptographic primitive operations (in milliseconds) for ECDH and ML-KEM.

Primitive	**PQ security level**	KeyGen	Encaps	Decaps
Classic key exchange algorithms				
EC X25519	1†	0.027	0.058	0.029
EC P-256	1†	0.008	0.058	0.047
EC P-384	3†	0.088	0.327	0.229
EC P-521	5†	0.098	0.341	0.226
Post-Quantum key exchange algorithms				
ML-KEM-512	1	0.020	0.014	0.023
ML-KEM-768	3	0.031	0.020	0.032
ML-KEM-1024	5	0.047	0.028	0.043
Post-Quantum/Traditional "hybrid" key exchange algorithms				
X25519 + ML-KEM-768	1† + 3	0.061	0.076	0.060
P-256 + ML-KEM-768	1† + 3	0.044	0.076	0.076
P-384 + ML-KEM-1024	3† + 5	0.143	0.344	0.256

†: (approximate) security level versus classic adversaries, no post-quantum security.
Benchmarks obtained using `openssl speed`, using OpenSSL 3.6.0 on a Macbook Pro with M2 Pro, running MacOS 26.0.

7.3.1 Computation Time

The time it takes to compute the KeyGen, Encaps, and Decaps operations is very relevant to TLS, as these operations are directly contributing to the time it takes to set up a connection. Fortunately, ML-KEM computation time is generally very good. In Table 8, we compare the performance of ML-KEM to ECDH algorithms currently used in TLS. For the ECDH algorithms, "Encapsulate" measures the generation of a new ECDH key, plus a group operation; "Decapsulate" is a single group operation. ML-KEM-512 is about as fast or much faster than all algorithms in the list, except for P-256's keygen operation. At higher security levels, ML-KEM is much faster than the NIST P-384 and P-521 elliptic curves.

It is worth pointing out, however, that we generally measure transit time on the network in (tens of) milliseconds, so all of these algorithms contribute only fractionally to the time it takes to set up a single connection: they are "fast enough". Runtime may still be relevant to applications that handle many connections at the same time, such as web servers or firewalls that do TLS connection inspection, though such applications may also want to consider offloading public-key computations from the main CPU to dedicated hardware implementations.

7.3.2 Bandwidth Usage

Another factor that determines the practicality of TLS is bandwidth usage. This both concerns the time that is needed to transmit messages, but also the direct cost of bandwidth for e.g. users on metered connections. Bandwidth is significantly affected by integrating ML-KEM. This is in

Table 9: Comparing encapsulation key and ciphertext sizes (in bytes) for ECDH and ML-KEM.

Primitive	PQ security level	Encapsulation key	Ciphertext
Classic key exchange algorithms			
EC X25519	$1^\dagger$	32	32
EC P-256	$1^\dagger$	65	65
EC P-384	$3^\dagger$	97	97
EC P-521	$5^\dagger$	123	123
Post-Quantum key exchange algorithms			
ML-KEM-512	1	800	768
ML-KEM-768	3	1184	1088
ML-KEM-1024	5	1568	1568
Post-Quantum/Traditional "hybrid" key exchange algorithms			
X25519 + ML-KEM-768	$1^\dagger + 3$	1216	1120
P-256 + ML-KEM-768	$1^\dagger + 3$	1249	1153
P-384 + ML-KEM-1024	$3^\dagger + 5$	1665	1617

†: (approximate) security level versus classic adversaries, no post-quantum security.

large part due to the fact that ML-KEM encapsulation keys and ciphertexts are much larger. In Table 9, we compare the encapsulation key and ciphertext sizes for ECDH algorithm with ML-KEM. It is clear that ML-KEM encapsulation keys and ciphertexts are a factor 12–25× larger than the ECDH equivalents. In particular, X25519 is currently the most popular key exchange algorithm on the web, and ML-KEM-512 encapsulation keys are 25× larger. In the benchmark results and in the deployment of ML-KEM on the web, both of which we will discuss below, the increase in size has proven manageable and do not hinder the usage of ML-KEM on the internet. However, in Section 7.5.1 we will discuss how this increase in sizes has uncovered some bugs in implementations.

7.3.3 Laboratory Experiments with ML-KEM in TLS 1.3

To explore the end-to-end effects of the integration of ML-KEM-based key exchange algorithms in TLS on connection setup times, we integrated ML-KEM-512, ML-KEM-768, and ML-KEM-1024, as well as the PQ/T hybrids X25519 + ML-KEM-768, P-256 + ML-KEM-768, and P-384 + ML-KEM-1024, into TLS 1.3. In particular, we have integrated the ML-KEM implementations from `liboqs` [Ope25] 0.14.0 into Rustls [BP]. The benchmark setup is based on the work by Wiggers; a more detailed description can be found in [Wig24, Ch. 10].

Note that we are only measuring the impact of changing the authentication algorithms. For all reported measurements, we have fixed the signature algorithms for the server's identity certificate, the intermediate CA certificate and the root CA certificate to RSA2048.

We report performance in two emulated network environments. In the first, the network latency is 31ms and the network bandwidth is 1000mbps. This represents a high-bandwidth, relatively low-latency connection; for example, between two servers across a continent. The second network environment sets the network latency to 195ms, and the network bandwidth to 10 mbps. This

represents a low-bandwidth with a latency representing a transatlantic connection. Experiments were run on a `m8a.24xlarge` virtual machine from Amazon AWS, using 96 cores on an AMD EPYC 9R45 CPU, running Amazon Linux 2023 with Linux 6.12. The reported numbers are the average of 24 000 runs. The benchmarking software and collected results are archived with DOI 10.5281/zenodo.17607131.

Table 10: Comparing TLS 1.3 connection setup times.

	Client handshake time	
Key exchange algorithm	1000mbps, 31.4ms latency	10mbps, 196.1ms latency
X25519	62.6 ms	394.8 ms
P-256	62.6 ms	394.9 ms
P-384	63.2 ms	396.1 ms
ML-KEM-512	62.6 ms	395.8 ms
ML-KEM-768	62.6 ms	396.5 ms
ML-KEM-1024	62.6 ms	397.2 ms
X25519 + ML-KEM-768	62.7 ms	396.6 ms
P-256 + ML-KEM-768	62.6 ms	396.7 ms
P-384 + ML-KEM-1024	63.3 ms	398.1 ms

RSA-2048 was used for all server signatures.

The results of the experiments are shown in Table 10. The client handshake times are dominated by the round-trips necessary for the handshake: one to establish the TCP connection over which the TLS protocol is run (the TCP `SYN`/`ACK`), and one to do the message exchange for TLS 1.3. Only on the low-bandwidth connection there appears to be any effect of using ML-KEM, but the incurred additional handshake latency is very minimal.

7.4 Post-Quantum/Traditional "Hybrid" Algorithms

The currently-used classic key exchange algorithms in TLS, and their software or hardware implementations, are well-understood and well-tested. Some consider replacing these implementations by "new" ML-KEM implementations risky. This is one of the reasons that many choose to use Post-Quantum/Traditional (PQ/T) KEM algorithms, also known as "hybrid" KEMs. These algorithms combine a quantum-vulnerable traditional (typically elliptic-curve) algorithm with ML-KEM. They are executed in parallel in such a way that both algorithms would need to be compromised in order to compromise the combination. This provides strong assurances against algorithm or implementation flaws, though at the cost of (a small amount) of additional bandwidth used and having to compute both key exchanges. Using a PQ/T algorithm instead of "pure", stand-alone ML-KEM also implies more code size or area (as both algorithms need to be implemented), which may be prohibitive in embedded settings. Finally, using PQ/T algorithms implies migrating twice: once to the PQ/T scheme, and once quantum computers are available or trust in ML-KEM and its implementation is sufficient, away from the hybrid to the "pure" ML-KEM. Choosing between PQ/T algorithms or "pure" ML-KEM requires balancing these concerns.

In Tables 8 and 9, we already listed the characteristics of currently popular PQ/T KEMs built from ML-KEM. It is notable that they use ML-KEM-768, which targets a 192-bit (PQ) security level,

together with X25519 and P-256, which provide 128 bits of classical security. The choice for a higher security level of ML-KEM hedges against advances in cryptanalysis of ML-KEM. The combination of P-384 and ML-KEM-1024 is a more conservative option.

7.4.1 Regulator Positions on Hybrids

In Europe, the German federal regulator BSI [ISB25], and French regulator ANSSI [ANS23] recommend or require (depending on context) the use of hybrid schemes. This position is mirrored in a European Commission recommendation to EU member states [Gro25]. In, for example, Australia, Canada, the United Kingdom, and the United States of America, national cybersecurity bodies are taking a more neutral position, allowing the use of hybrids as a stepping stone, but acknowledging that they add complexity along the way to a "fully post-quantum end-state" [Aus25; Mat25; Nat24a; Moo+24]. In a set of requirements (CNSA 2.0) for the intelligence systems overseen by the United States National Security Agency, meanwhile, hybrids are generally not permitted, as they seemingly are trying to avoid the complexity of a diversified cryptographic algorithm landscape and managing multiple migrations (in general, CNSA 2.0 only permits a very limited set of algorithms) [Nat24c].

7.5 Experiments with TLS with Post-Quantum Confidentiality

Because of many practical, large-scale experiments going back to 2016, we are able to provide an accurate view on the practicality of TLS 1.3 with ML-KEM on the public internet. The Google Chrome browser started the CECPQ1 ("combined elliptic-curve and post-quantum 1") experiment in 2016 [Lan16], combining X25519 [Ber06] with the NewHope lattice-based KEM key exchange [Alk+16] (a predecessor to ML-KEM) in the TLS 1.2 handshake. The follow-up CECPQ2 experiment based on TLS 1.3 was announced in late 2018 [Lan18; KV19], using a combination of X25519 and the lattice-based scheme NTRU-HRSS [Hül+17; Sch+17],[5] and X25519 with the isogeny-based scheme SIKE [Jao+22]. The first results from this experiment are presented in [Kwi+19].

The academic Open Quantum Safe (OQS) initiative [SM16] provides prototype integrations of post-quantum and hybrid key exchange in TLS 1.2 and TLS 1.3 to the OpenSSL library [Opea]. First results in terms of feasibility of migration and performance using OQS were presented in [CPS19]; more detailed benchmarks are presented in [PST20]. Schwabe, Stebila, and Wiggers [SSW20; SSW21], Celi et al. [Cel+21], and Wiggers [Wig24] present results using post-quantum TLS with different key exchange and authentication algorithms through experiments with Rustls [BP], though the focus of their work is on comparing post-quantum signature-based authentication in TLS 1.3 to KEMTLS, an alternative TLS protocol which uses authentication based on KEMs. On embedded platforms, experimentation with TLS was done by [GW22; BS+20; Tas+22].

Meanwhile, draft specifications for PQ/T key exchange had started the discussion on standardization of post-quantum cryptography for TLS [SWZ17; CC21; KK18; Why+17; SS17; SFG25; HW20]. This has progressed with three drafts that are almost finalized to RFC, and that are rapidly getting implemented and deployed [Con25; Kwi+25; SFG25]

Based on the results from academic, CECPQ1 and CECPQ2 experiments, Google Chrome and Cloudflare were able to start the deployment of the `X25519MLKEM768` key exchange algorithm that combines X25519 with ML-KEM-768.[6] This started with in August 2023 with a small percentage rollout on desktop [O'B23], but has now been fully enabled in Google Chrome (100% since

[5] NTRU-HRSS was merged into the NTRU [Che+20] submission which was eventually not selected.
[6] Though initially using draft standards with minor differences to the final ML-KEM-768 in FIPS 203.

April 2024 [Adr+24]), and in Chrome-derived browers like Microsoft Edge [Mic25]. Apple has enabled PQC hybrid key exchange in Safari since September 2025 [App25].

As of November 2025, Cloudflare reports that around 50% of their HTTPS traffic is using `X25519MLKEM768` [Clo]

7.5.1 Compatibility Problems Found During the Deployment of `X25519MLKEM768`

As Google was rolling out the PQ/T algorithm `X25519MLKEM768`, they found that this was causing some connections to fail. This was caused by so-called "middleboxes". These are network appliances like firewalls, typically found in enterprise networks, that may inspect or intercept TLS traffic. They were found to sometimes have implementations that were not prepared to handle initial TLS messages that span more than a single network packet. This was not encountered before, as all key exchange methods (plus protocol overhead) typically easily fit in single network packets. As using `X25519MLKEM768` pushes the initial message over the common approximately 1200 byte limit for network packets, the packet needs to be fragmented. On the website `https://tldr.fail`, Google further explains this issue and tracks the status of fixes.

What this issue indicates is that implementations may make (invalid) assumptions on what a typical TLS connection should look like. These assumptions may be stricter than or even go against protocol specifications. Making a large change to the "shape" of the connection, including adding larger-than-before key exchange methods such as ML-KEM, may invalidate those assumptions. This issue is sometimes referred to as "protocol ossification". Similar issues plagued the deployment of TLS 1.3 [Sul17].

When deploying ML-KEM, one should be mindful of protocol ossification problems.

7.6 Availability of ML-KEM Support in Popular TLS Libraries

In this section, we survey some popular TLS libraries and note the availability of ML-KEM support. Table 11 show for each library if algorithms are not supported (✗); supported, but currently disabled by default (✓); supported, and enabled by default, but not the preferred algorithm in notation (✓✓); and if an algorithm is supported, enabled, and preferred in algorithm negotiation (★). It also lists the version number in which (any) ML-KEM support first became available,[7] though the supported features are for the most recent version as of mid-November 2025.

As Table 11 shows, TLS libraries are rapidly adopting support for ML-KEM, in particular the X25519 + ML-KEM-768 PQ/T algorithm. Note that Table 11 only refers to functionality for TLS; many libraries also offer access to cryptographic primitives directly, and they may support using additional parameter sets of ML-KEM that way even if they are not supported in TLS.

Lastly, note that most libraries come with defaults, but these may be overridden by the users of the library or, e.g., configurations shipped in Linux distributions. Only checking library versions is not sufficient to verify if an application uses ML-KEM and provides post-quantum confidentiality.

7.7 Discussion

The results discussed in this section show that ML-KEM is suitable for use in replacing elliptic-curve based key exchange in TLS for "normal" web browsing settings. Even when using "hybrid" PQ/T algorithms in which the cost of ML-KEM is added "on top" of the current connection overhead

[7] Though we do not consider support for draft versions of ML-KEM.

Table 11: Support for ML-KEM and ML-KEM-based PQ/T algorithms in popular TLS libraries.

		First support		Support in TLS					
Library	**Website**	**Version**	**Date**	**ML-KEM-512**	**ML-KEM-768**	**ML-KEM-1024**	**X25519 + ML-KEM-768**	**P-256 + ML-KEM-768**	**P-384 + ML-KEM-1024**
OpenSSL	[Opea]	3.5.0	2025-04-08	✓	✓	✓	★	✓✓	✓✓
mbed TLS	[ARM]	–	–	✗	✗	✗	✗	✗	✗
Rustls	[BP]	0.23.22	2025-01-31	✗	✓	✗	★	✓	✗
BoringSSL	[Goo]	N/A	2024-08-27	✗	✗	✓	★	✗	✗
Bouncy Castle Java	[Theb]	1.81	2025-06-04	✓	✓	✓	✓	✓	✓
Bouncy Castle C#	[Theb]	2.6.1	2025-06-04	✓	✓	✓	✓	✓	✓
NSS	[Moz]	0.105.0	2024-09-26	✗	✗	✗	★	✓✓	✓✓
Go	[Thea]	1.24.0	2025-02-11	✗	✗	✗	★	✗	✗
Microsoft SChannel	[Mic]	–	–	✗	✗	✗	✗	✗	✗
Apple Network.framework	[App]	26.0	2025-09-15	✗	✗	✗	★	✗	✗

Support legend: ✗: Not supported; ✓: Supported, but not enabled by default; ✓✓: Enabled by default; ★: Enabled and preferred algorithm in negotiation. Version listed indicates first availability of any support; checkmarks indicate support as of writing (November 2025).
Note: applications may override the default preferences of the libraries they use.

posed by cryptography, ML-KEM adds negligible additional latency to TLS connection setup. For embedded settings, the additional bandwidth costs may be more prohibitive, see e.g., [GW22; BS+20; Tas+22]. Additionally, when integrating ML-KEM into protocols that are currently used, protocol ossification risks may exist that can hinder deployment. Such concerns will need to be addressed on a case-by-case basis. Finally, many TLS libraries are already updated for ML-KEM support, indicating broad support and availability.

References

[Abd+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. "Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions". In: *CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. LNCS. Springer, Berlin, Heidelberg, Aug. 2005, pp. 205–222. DOI: 10.1007/11535218_13.

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. "A Subfield Lattice Attack on Overstretched NTRU Assumptions - Cryptanalysis of Some FHE and Graded Encoding Schemes". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Berlin, Heidelberg, Aug. 2016, pp. 153–178. DOI: 10.1007/978-3-662-53018-4_6.

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. "Robust Encryption". In: *TCC 2010*. Ed. by Daniele Micciancio. Vol. 5978. LNCS. Springer, Berlin, Heidelberg, Feb. 2010, pp. 480–497. DOI: 10.1007/978-3-642-11799-2_28.

[ACW19] Martin R. Albrecht, Benjamin R. Curtis, and Thomas Wunderer. "Exploring Trade-offs in Batch Bounded Distance Decoding". In: *SAC 2019*. Ed. by Kenneth G. Paterson and Douglas Stebila. Vol. 11959. LNCS. Springer, Cham, Aug. 2019, pp. 467–491. DOI: 10.1007/978-3-030-38471-5_19.

[AD21] Martin Albrecht and Léo Ducas. *Lattice Attacks on NTRU and LWE: A History of Refinements*. Cryptology ePrint Archive, Report 2021/799. 2021. URL: https://eprint.iacr.org/2021/799.

[Adr+24] David Adrian, Bob Beck, David Benjamin, and Devon O'Brien. *Advancing Our Amazing Bet on Asymmetric Cryptography*. May 23, 2024. URL: https://blog.chromium.org/2024/05/advancing-our-amazing-bet-on-asymmetric.html.

[Aes] *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology, NIST FIPS PUB 197, U.S. Department of Commerce. Nov. 2001.

[AFG14] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. "On the Efficacy of Solving LWE by Reduction to Unique-SVP". In: *ICISC 13*. Ed. by Hyang-Sook Lee and Dong-Guk Han. Vol. 8565. LNCS. Springer, Cham, Nov. 2014, pp. 293–310. DOI: 10.1007/978-3-319-12160-4_18.

[AG11] Sanjeev Arora and Rong Ge. "New Algorithms for Learning in Presence of Errors". In: *ICALP 2011, Part I*. Ed. by Luca Aceto, Monika Henzinger, and Jiri Sgall. Vol. 6755. LNCS. Springer, Berlin, Heidelberg, July 2011, pp. 403–415. DOI: 10.1007/978-3-642-22006-7_34.

[Agg+20] Divesh Aggarwal, Jianwei Li, Phong Q. Nguyen, and Noah Stephens-Davidowitz. "Slide Reduction, Revisited - Filling the Gaps in SVP Approximation". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 274–295. DOI: 10.1007/978-3-030-56880-1_10.

[Alb+14] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. *Algebraic Algorithms for LWE*. Cryptology ePrint Archive, Report 2014/1018. 2014. URL: https://eprint.iacr.org/2014/1018.

[Alb17] Martin R. Albrecht. "On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL". In: *EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. Springer, Cham, 2017, pp. 103–129. DOI: 10.1007/978-3-319-56614-6_4.

[Alb+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. "The General Sieve Kernel and New Records in Lattice Reduction". In: *EUROCRYPT 2019, Part II*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11477. LNCS. Springer, Cham, May 2019, pp. 717–746. DOI: 10.1007/978-3-030-17656-3_25.

[Alb+20a] Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. "Faster Enumeration-Based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ Time $k^{k/8+o(k)}$". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 186–212. DOI: 10.1007/978-3-030-56880-1_7.

[Alb+20b] Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck. "Estimating Quantum Speedups for Lattice Sieves". In: *ASIACRYPT 2020, Part II*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer, Cham, Dec. 2020, pp. 583–613. DOI: 10.1007/978-3-030-64834-3_20.

[Alb+21] Martin R. Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. "Lattice Reduction with Approximate Enumeration Oracles - Practical Algorithms and Concrete Performance". In: *CRYPTO 2021, Part II*. Ed. by Tal Malkin and Chris Peikert. Vol. 12826. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 732–759. DOI: 10.1007/978-3-030-84245-1_25.

[Alk+16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. "Post-quantum Key Exchange - A New Hope". In: *USENIX Security 2016*. Ed. by Thorsten Holz and Stefan Savage. USENIX Association, Aug. 2016, pp. 327–343. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim.

[Alm+24] José Bacelar Almeida, Santiago Arranz Olmos, Manuel Barbosa, Gilles Barthe, François Dupressoir, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Cameron Low, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Peter Schwabe, and Pierre-Yves Strub. "Formally Verifying Kyber - Episode V: Machine-Checked IND-CCA Security and Correctness of ML-KEM in EasyCrypt". In: *CRYPTO 2024, Part II*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14921. LNCS. Springer, Cham, Aug. 2024, pp. 384–421. DOI: 10.1007/978-3-031-68379-4_12.

[Alp+24] Estuardo Alpirez Bock, Gustavo Banegas, Chris Brzuska, Lukasz Chmielewski, Kirthivaasan Puniamurthy, and Milan Sorf. "Breaking DPA-Protected Kyber via the Pair-Pointwise Multiplication". In: *ACNS 2024, Part II*. Ed. by Christina Pöpper and Lejla Batina. Vol. 14584. LNCS. Springer, Cham, Mar. 2024, pp. 101–130. DOI: 10.1007/978-3-031-54773-7_5.

[ANS23] ANSSI. *ANSSI views on the Post-Quantum Cryptography transition (2023 follow up)*. Dec. 21, 2023. URL: https://messervices.cyber.gouv.fr/documents-guides/follow_up_position_paper_on_post_quantum_cryptography.pdf.

[Aon+16] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. "Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator". In: *EUROCRYPT 2016, Part I*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 789–819. DOI: 10.1007/978-3-662-49890-3_30.

[App] Apple. *Network.framework*. URL: https://developer.apple.com/documentation/network/.

[App+09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Berlin, Heidelberg, Aug. 2009, pp. 595–618. DOI: 10.1007/978-3-642-03356-8_35.

[App25] Apple. *Prepare your network for quantum-secure encryption in TLS*. July 23, 2025. URL: https://support.apple.com/en-me/122756.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors". In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203. DOI: doi:10.1515/jmc-2015-0016. URL: https://doi.org/10.1515/jmc-2015-0016.

[ARM] ARM Limited. *mbed TLS*. URL: https://tls.mbed.org/.

[Aus25] Australian Signals Directorate. *Guidelines for Cryptography*. Information Security Manual (ISM), first published 4 December 2025, last updated 4 December 2025. Dec. 2025. URL: https://www.cyber.gov.au/business-government/asds-cyber-security-frameworks/ism/cyber-security-guidelines/guidelines-for-cryptography (visited on 01/13/2026).

[Ava+21] Roberto Avanzi, Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation (version 3.02)*. Submission to NIST PQC Round 3. Version 3.02. Aug. 2021. URL: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

[Bac+23] Linus Backlund, Kalle Ngo, Joel Gärtner, and Elena Dubrova. "Secret Key Recovery Attack on Masked and Shuffled Implementations of CRYSTALS-Kyber and Saber". In: *Applied Cryptography and Network Security Workshops: ACNS 2023 Satellite Workshops, ADSC, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, SiMLA, Kyoto, Japan, June 19–22, 2023, Proceedings*. Kyoto, Japan: Springer-Verlag, 2023, pp. 159–177. DOI: 10.1007/978-3-031-41181-6_9.

[Bar+25] Manuel Barbosa, Matthias J Kannwischer, Thing-han Lim, Peter Schwabe, and Pierre-Yves Strub. *Formally Verified Correctness Bounds for Lattice-Based Cryptography*. To Appear in *ACM CCS 2025*. 2025. URL: https://eprint.iacr.org/2025/1562.

[BDF20] Koen de Boer, Léo Ducas, and Serge Fehr. "On the Quantum Complexity of the Continuous Hidden Subgroup Problem". In: *EUROCRYPT 2020, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. LNCS. Springer, Cham, May 2020, pp. 341–370. DOI: 10.1007/978-3-030-45724-2_12.

[Bec+16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. "New directions in nearest neighbor searching with applications to lattice sieving". In: *27th SODA*. Ed. by Robert Krauthgamer. ACM-SIAM, Jan. 2016, pp. 10–24. DOI: 10.1137/1.9781611974331.ch2.

[Bel+01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. "Key-Privacy in Public-Key Encryption". In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Springer, Berlin, Heidelberg, Dec. 2001, pp. 566–582. DOI: 10.1007/3-540-45682-1_33.

[Ber06] Daniel J. Bernstein. "Curve25519: New Diffie-Hellman Speed Records". In: *PKC 2006*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Vol. 3958. LNCS. Springer, Berlin, Heidelberg, Apr. 2006, pp. 207–228. DOI: 10.1007/11745853_14.

[Ber22] Daniel J. Bernstein. *Multi-ciphertext security degradation for lattices*. Cryptology ePrint Archive, Report 2022/1580. 2022. URL: https://eprint.iacr.org/2022/1580.

[Ber23] Daniel J. Bernstein. *Asymptotics of hybrid primal lattice attacks*. Cryptology ePrint Archive, Report 2023/1892. 2023. URL: https://eprint.iacr.org/2023/1892.

[BF14] Jean-François Biasse and Claus Fieker. "Subexponential class group and unit group computation in large degree number fields". In: *LMS Journal of Computation and Mathematics* 17.A (2014), pp. 385–403. DOI: 10.1112/S1461157014000345.

[BF25] Koen de Boer and Joël Felderhoff. "Quantumly Computing S-unit Groups in Quantified Polynomial Time and Space". In: *Cryptology ePrint Archive* (2025).

[BGJ15] Anja Becker, Nicolas Gama, and Antoine Joux. *Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search*. Cryptology ePrint Archive, Report 2015/522. 2015. URL: https://eprint.iacr.org/2015/522.

[Bha+21] Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. "Attacking and Defending Masked Polynomial Comparison". In: *IACR TCHES* 2021.3 (2021), pp. 334–359. ISSN: 2569-2925. DOI: 10.46586/tches.v2021.i3.334-359. URL: https://tches.iacr.org/index.php/TCHES/article/view/8977.

[Bia+17] Jean-François Biasse, Thomas Espitau, Pierre-Alain Fouque, Alexandre Gélin, and Paul Kirchner. "Computing Generator in Cyclotomic Integer Rings - A Subfield Algorithm for the Principal Ideal Problem in $L_{|\Delta_{\mathbb{K}}|}(\frac{1}{2})$ and Application to the Cryptanalysis of a FHE Scheme". In: *EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. LNCS. Springer, Cham, 2017, pp. 60–88. DOI: 10.1007/978-3-319-56620-7_3.

[Bin+19] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. "Tighter Proofs of CCA Security in the Quantum Random Oracle Model". In: *TCC 2019, Part II*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. LNCS. Springer, Cham, Dec. 2019, pp. 61–90. DOI: 10.1007/978-3-030-36033-7_3.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. "Noise-tolerant learning, the parity problem, and the statistical query model". In: *32nd ACM STOC*. ACM Press, May 2000, pp. 435–440. DOI: 10.1145/335305.335355.

[BL16] Anja Becker and Thijs Laarhoven. "Efficient (Ideal) Lattice Sieving Using Cross-Polytope LSH". In: *AFRICACRYPT 16*. Ed. by David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi. Vol. 9646. LNCS. Springer, Cham, Apr. 2016, pp. 3–23. DOI: 10.1007/978-3-319-31517-1_1.

[BLS16] Shi Bai, Thijs Laarhoven, and Damien Stehlé. "Tuple lattice sieving". In: *LMS Journal of Computation and Mathematics* 19.A (2016), pp. 146–162. DOI: 10.1112/S1461157016000292.

[Bon+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random Oracles in a Quantum World". In: *ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, Berlin, Heidelberg, Dec. 2011, pp. 41–69. DOI: 10.1007/978-3-642-25385-0_3.

[Bon+23] Xavier Bonnetain, André Chailloux, André Schrottenloher, and Yixin Shen. "Finding Many Collisions via Reusable Quantum Walks: Application to Lattice Sieving". In: *EUROCRYPT 2023, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. LNCS. Springer, Cham, Apr. 2023, pp. 221–251. DOI: 10.1007/978-3-031-30589-4_8.

[Bos+15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. "Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem". In: *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2015, pp. 553–570. DOI: 10.1109/SP.2015.40.

[Bos+16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. "Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE". In: *ACM CCS 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM Press, Oct. 2016, pp. 1006–1018. DOI: 10.1145/2976749.2978425.

[Bos+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *2018 IEEE European Symposium on Security and Privacy*. IEEE Computer Society Press, Apr. 2018, pp. 353–367. DOI: 10.1109/EuroSP.2018.00032.

[Bos+21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. "Masking Kyber: First- and Higher-Order Implementations". In: *IACR TCHES* 2021.4 (2021), pp. 173–214. ISSN: 2569-2925. DOI: 10.46586/tches.v2021.i4.173-214. URL: https://tches.iacr.org/index.php/TCHES/article/view/9064.

[Bou+21] Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen. "On the Hardness of Module-LWE with Binary Secret". In: *CT-RSA 2021*. Ed. by Kenneth G. Paterson. Vol. 12704. LNCS. Springer, Cham, May 2021, pp. 503–526. DOI: 10.1007/978-3-030-75539-3_21.

[Bou+22] Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen. "Entropic Hardness of Module-LWE from Module-NTRU". In: *INDOCRYPT 2022*. Ed. by Takanori Isobe and Santanu Sarkar. Vol. 13774. LNCS. Springer, Cham, Dec. 2022, pp. 78–99. DOI: 10.1007/978-3-031-22912-1_4.

[Boy+09] Colin Boyd, Yvonne Cliff, Juan M. Gonzalez Nieto, and Kenneth G. Paterson. "One-round key exchange in the standard model". In: *International Journal of Applied Cryptography* 1.3 (Feb. 2009), pp. 181–199. DOI: 10.1504/IJACT.2009.023466.

[BP] Joseph Birr-Pixton. *Rustls: A modern TLS library in Rust*. URL: https://github.com/rustls/rustls (visited on 12/22/2022).

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *ACM CCS 93*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[Bra+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical hardness of learning with errors". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 575–584. DOI: 10.1145/2488608.2488680.

[Bru+16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. "Flush, Gauss, and Reload - A Cache Attack on the BLISS Lattice-Based Signature Scheme". In: *CHES 2016*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. LNCS. Springer, Berlin, Heidelberg, Aug. 2016, pp. 323–345. DOI: 10.1007/978-3-662-53140-2_16.

[BS15] J.-F. Biasse and F. Song. *A note on the quantum attacks against schemes relying on the hardness of finding a short generator of an ideal in* $\mathbb{Q}(\zeta_{2^n})$. Tech. rep. 2015-12. Revision of September 28th 2015. The University of Waterloo, 2015.

[BS+20] Kevin Bürstinghaus-Steinbach, Christoph Krauß, Ruben Niederhagen, and Michael Schneider. "Post-Quantum TLS on Embedded Systems: Integrating and Evaluating Kyber and SPHINCS+ with mbed TLS". In: *ASIACCS 20*. Ed. by Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese. ACM Press, Oct. 2020, pp. 841–852. DOI: 10.1145/3320269.3384725.

[BSW18] Shi Bai, Damien Stehlé, and Weiqiang Wen. "Measuring, Simulating and Exploiting the Head Concavity Phenomenon in BKZ". In: *ASIACRYPT 2018, Part I*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11272. LNCS. Springer, Cham, Dec. 2018, pp. 369–404. DOI: 10.1007/978-3-030-03326-2_13.

[BW25] Kaveh Bashiri and Andreas Wiemers. "On the independence heuristic in the dual attack". In: *Journal of Mathematical Cryptology* 19.1 (2025), p. 20240028. DOI: doi:10.1515/jmc-2024-0028. URL: https://doi.org/10.1515/jmc-2024-0028.

[Car+25] Kevin Carrier, Charles Meyer-Hilfiger, Yixin Shen, and Jean-Pierre Tillich. "Assessing the Impact of a Variant of MATZOV's Dual Attack on Kyber". In: *Advances in Cryptology – CRYPTO 2025*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Cham: Springer Nature Switzerland, 2025, pp. 444–476. ISBN: 978-3-032-01855-7.

[CC21] Matt Campagna and Eric Crockett. *Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS)*. Internet-Draft draft-campagna-tls-bike-sike-hybrid-07. Work in Progress. Internet Engineering Task Force, Sept. 2, 2021. 17 pp. URL: https://datatracker.ietf.org/doc/html/draft-campagna-tls-bike-sike-hybrid-07.

[CDM24] Cas Cremers, Alexander Dax, and Niklas Medinger. "Keeping Up with the KEMs: Stronger Security Notions for KEMs and Automated Analysis of KEM-based Protocols". In: *ACM CCS 2024*. Ed. by Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie. ACM Press, Oct. 2024, pp. 1046–1060. DOI: 10.1145/3658644.3670283.

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. "Short Stickelberger Class Relations and Application to Ideal-SVP". In: *EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. LNCS. Springer, Cham, 2017, pp. 324–348. DOI: 10.1007/978-3-319-56620-7_12.

[CDW21] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. "Mildly Short Vectors in Cyclotomic Ideal Lattices in Quantum Polynomial Time". In: *J. ACM* 68.2 (Jan. 2021). DOI: 10.1145/3431725.

[Cel+21] Sofía Celi, Armando Faz-Hernández, Nick Sullivan, Goutam Tamvada, Luke Valenta, Thom Wiggers, Bas Westerbaan, and Christopher A. Wood. "Implementing and Measuring KEMTLS". In: *LATINCRYPT 2021*. Ed. by Patrick Longa and Carla Ràfols. Vol. 12912. LNCS. Springer, Cham, Oct. 2021, pp. 88–107. DOI: 10.1007/978-3-030-88238-9_5.

[Che+20] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. *NTRU*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. National Institute of Standards and Technology, 2020.

[Che+23] Zhao Chen, Xianhui Lu, Dingding Jia, and Bao Li. "IND-CCA Security of Kyber in the Quantum Random Oracle Model, Revisited". In: *Information Security and Cryptology*. Springer Nature Switzerland, 2023, pp. 148–166. DOI: 10.1007/978-3-031-26553-2_8.

[CL01] Jan Camenisch and Anna Lysyanskaya. "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation". In: *EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. LNCS. Springer, Berlin, Heidelberg, May 2001, pp. 93–118. DOI: 10.1007/3-540-44987-6_7.

[CL21] André Chailloux and Johanna Loyer. "Lattice Sieving via Quantum Random Walks". In: *ASIACRYPT 2021, Part IV*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13093. LNCS. Springer, Cham, Dec. 2021, pp. 63–91. DOI: 10.1007/978-3-030-92068-5_3.

[CL23] André Chailloux and Johanna Loyer. "Classical and Quantum 3 and 4-Sieves to Solve SVP with Low Memory". In: *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023*. Ed. by Thomas Johansson and Daniel Smith-Tone. Springer, Cham, Aug. 2023, pp. 225–255. DOI: 10.1007/978-3-031-40003-2_9.

[Clo] Cloudflare. *Cloudflare Radar – Adoption & Usage – Post-quantum encryption*. URL: https://radar.cloudflare.com/adoption-and-usagexpost-quantum-encryption (visited on 11/11/2025).

[CN11] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, Berlin, Heidelberg, Dec. 2011, pp. 1–20. DOI: 10.1007/978-3-642-25385-0_1.

[Con25] Deirdre Connolly. *ML-KEM Post-Quantum Key Agreement for TLS 1.3*. Internet-Draft draft-ietf-tls-mlkem-05. Work in Progress. Internet Engineering Task Force, Nov. 2025. 10 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-tls-mlkem/05/.

[CPS19] Eric Crockett, Christian Paquin, and Douglas Stebila. *Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH*. Workshop Record of the Second PQC Standardization Conference. 2019. iacr: 2019/858. URL: https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/stebila-prototyping-post-quantum.pdf.

[Cra+16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. "Recovering Short Generators of Principal Ideals in Cyclotomic Rings". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 559–585. DOI: 10.1007/978-3-662-49896-5_20.

[CS03] Ronald Cramer and Victor Shoup. "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack". In: *SIAM Journal on Computing* 33.1 (2003), pp. 167–226. DOI: 10.1137/S0097539702403773.

[D'A+19] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. "Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes". In: *PKC 2019, Part II*. Ed. by Dongdai Lin and Kazue Sako. Vol. 11443. LNCS. Springer, Cham, Apr. 2019, pp. 565–598. DOI: 10.1007/978-3-030-17259-6_19.

[Dac+20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. "LWE with Side Information: Attacks and Concrete Security Estimation". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 329–358. DOI: 10.1007/978-3-030-56880-1_12.

[DB22] Jan-Pieter D'Anvers and Senne Batsleer. "Multitarget Decryption Failure Attacks and Their Application to Saber and Kyber". In: *PKC 2022, Part I*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Vol. 13177. LNCS. Springer, Cham, Mar. 2022, pp. 3–33. DOI: 10.1007/978-3-030-97121-2_1.

[DEP23] Léo Ducas, Thomas Espitau, and Eamonn W. Postlethwaite. "Finding Short Integer Solutions When the Modulus Is Small". In: *CRYPTO 2023, Part III*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14083. LNCS. Springer, Cham, Aug. 2023, pp. 150–176. DOI: 10.1007/978-3-031-38548-3_6.

[DEP25] Léo Ducas, Lynn Engelberts, and Paola de Perthuis. "Predicting Module-Lattice Reduction". In: *Cryptology ePrint Archive* (2025).

[Din+22] Xiaohui Ding, Muhammed F. Esgin, Amin Sakzad, and Ron Steinfeld. "An Injectivity Analysis of Crystals-Kyber and Implications on Quantum Security". In: *ACISP 22*. Ed. by Khoa Nguyen, Guomin Yang, Fuchun Guo, and Willy Susilo. Vol. 13494. LNCS. Springer, Cham, Nov. 2022, pp. 332–351. DOI: 10.1007/978-3-031-22301-3_17.

[DLW20] Léo Ducas, Thijs Laarhoven, and Wessel P. J. van Woerden. "The Randomized Slicer for CVPP: Sharper, Faster, Smaller, Batchier". In: *PKC 2020, Part II*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12111. LNCS. Springer, Cham, May 2020, pp. 3–36. DOI: 10.1007/978-3-030-45388-6_1.

[DM13] Nico Döttling and Jörn Müller-Quade. "Lossy Codes and a New Variant of the Learning-With-Errors Problem". In: *EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, Berlin, Heidelberg, May 2013, pp. 18–34. DOI: 10.1007/978-3-642-38348-9_2.

[DMG23] Viet Ba Dang, Kamyar Mohajerani, and Kris Gaj. "High-Speed Hardware Architectures and FPGA Benchmarking of CRYSTALS-Kyber, NTRU, and Saber". In: *IEEE Transactions on Computers* 72.2 (2023), pp. 306–320. DOI: 10.1109/TC.2022.3222954.

[Don+22] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. "Online-Extractability in the Quantum Random-Oracle Model". In: *EUROCRYPT 2022, Part III*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. LNCS. Springer, Cham, 2022, pp. 677–706. DOI: 10.1007/978-3-031-07082-2_24.

[Dor+24] Joao F. Doriguello, George Giapitzakis, Alessandro Luongo, and Aditya Morolia. *On the practicality of quantum sieving algorithms for the shortest vector problem*. Cryptology ePrint Archive, Report 2024/1692. 2024. URL: https://eprint.iacr.org/2024/1692.

[DP23a] Léo Ducas and Ludo N. Pulles. *Accurate Score Prediction for Dual-Sieve Attacks*. Cryptology ePrint Archive, Report 2023/1850. 2023. URL: https://eprint.iacr.org/2023/1850.

[DP23b] Léo Ducas and Ludo N. Pulles. "Does the Dual-Sieve Attack on Learning with Errors Even Work?" In: *CRYPTO 2023, Part III*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14083. LNCS. Springer, Cham, Aug. 2023, pp. 37–69. DOI: 10.1007/978-3-031-38548-3_2.

[DPW19] Léo Ducas, Maxime Plançon, and Benjamin Wesolowski. "On the Shortness of Vectors to Be Found by the Ideal-SVP Quantum Algorithm". In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. LNCS. Springer, Cham, Aug. 2019, pp. 322–351. DOI: 10.1007/978-3-030-26948-7_12.

[DSW21] Léo Ducas, Marc Stevens, and Wessel P. J. van Woerden. "Advanced Lattice Sieving on GPUs, with Tensor Cores". In: *EUROCRYPT 2021, Part II*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. LNCS. Springer, Cham, Oct. 2021, pp. 249–279. DOI: 10.1007/978-3-030-77886-6_9.

[Dub+23] Elena Dubrova, Kalle Ngo, Joel Gärtner, and Ruize Wang. "Breaking a Fifth-Order Masked Implementation of CRYSTALS-Kyber by Copy-Paste". In: *Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop*. APKC '23. Melbourne, VIC, Australia: Association for Computing Machinery, 2023, pp. 10–20. ISBN: 9798400701832. DOI: 10.1145/3591866.3593072. URL: https://doi.org/10.1145/3591866.3593072.

[Duc18] Léo Ducas. "Shortest Vector from Lattice Sieving: A Few Dimensions for Free". In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Cham, 2018, pp. 125–145. DOI: 10.1007/978-3-319-78381-9_5.

[Duc22a] Léo Ducas. *Comment on "Improved Dual Lattice Attack"*. NIST PQC Forum. May 2022. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/Fm4cDfsx65s/m/BZFRC8hiAAAJ.

[Duc22b] Léo Ducas. "Estimating the Hidden Overheads in the BDGL Lattice Sieving Algorithm". In: *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022*. Ed. by Jung Hee Cheon and Thomas Johansson. Springer, Cham, Sept. 2022, pp. 480–497. DOI: 10.1007/978-3-031-17234-2_22.

[dv25] Koen de Boer and Wessel P. J. van Woerden. *Lattice-based Cryptography: A survey on the security of the lattice-based NIST finalists*. Cryptology ePrint Archive, Report 2025/304. 2025. URL: https://eprint.iacr.org/2025/304.

[DVV18] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. *On the impact of decryption failures on the security of LWE/LWR based schemes*. Cryptology ePrint Archive, Report 2018/1089. 2018. URL: https://eprint.iacr.org/2018/1089.

[DW21] Léo Ducas and Wessel P. J. van Woerden. "NTRU Fatigue: How Stretched is Overstretched?" In: *ASIACRYPT 2021, Part IV*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13093. LNCS. Springer, Cham, Dec. 2021, pp. 3–32. DOI: 10.1007/978-3-030-92068-5_1.

[Eis+14] Kirsten Eisenträger, Sean Hallgren, Alexei Kitaev, and Fang Song. "A quantum algorithm for computing the unit group of an arbitrary degree number field". In: *46th ACM STOC*. Ed. by David B. Shmoys. ACM Press, 2014, pp. 293–302. DOI: 10.1145/2591796.2591860.

[EJK20] Thomas Espitau, Antoine Joux, and Natalia Kharchenko. "On a Dual/Hybrid Approach to Small Secret LWE - A Dual/Enumeration Technique for Learning with Errors and Application to Security Estimates of FHE Schemes". In: *INDOCRYPT 2020*. Ed. by Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran. Vol. 12578. LNCS. Springer, Cham, Dec. 2020, pp. 440–462. DOI: 10.1007/978-3-030-65277-7_20.

[EK20] Thomas Espitau and Paul Kirchner. *The nearest-colattice algorithm*. Cryptology ePrint Archive, Report 2020/694. 2020. URL: https://eprint.iacr.org/2020/694.

[Esp+17] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. "Side-Channel Attacks on BLISS Lattice-Based Signatures: Exploiting Branch Tracing against strongSwan and Electromagnetic Emanations in Microcontrollers". In: *ACM CCS 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 1857–1874. DOI: 10.1145/3133956.3134028.

[FH05] Paul Ford-Hutchinson. *Securing FTP with TLS*. RFC 4217. Oct. 2005. DOI: 10.17487/RFC4217.

[Flu+25] Scott Fluhrer, Quynh Dang, John Preuß Mattsson, Kevin Milner, and Daniel Shiu. *ML-KEM Security Considerations*. Internet-Draft draft-sfluhrer-cfrg-ml-kem-security-considerations-03. Work in Progress. IETF, May 2025. URL: https://datatracker.ietf.org/doc/draft-sfluhrer-cfrg-ml-kem-security-considerations/.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Journal of Cryptology* 26.1 (Jan. 2013), pp. 80–101. DOI: 10.1007/s00145-011-9114-1.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Berlin, Heidelberg, Aug. 1999, pp. 537–554. DOI: 10.1007/3-540-48405-1_34.

[Fuj+13] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. "Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism". In: *ASIACCS 13*. Ed. by Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng. ACM Press, May 2013, pp. 83–94. DOI: 10.1145/2484313.2484323.

[Fuj+15] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. "Strongly secure authenticated key exchange from factoring, codes, and lattices". In: *DCC* 76.3 (2015), pp. 469–504. DOI: 10.1007/s10623-014-9972-2.

[Gen09] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *41st ACM STOC*. Ed. by Michael Mitzenmacher. ACM Press, 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[GJ21] Qian Guo and Thomas Johansson. "Faster Dual Lattice Attacks for Solving LWE with Applications to CRYSTALS". In: *ASIACRYPT 2021, Part IV*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13093. LNCS. Springer, Cham, Dec. 2021, pp. 33–62. DOI: 10.1007/978-3-030-92068-5_2.

[GJN20] Qian Guo, Thomas Johansson, and Alexander Nilsson. "A Key-Recovery Timing Attack on Post-quantum Primitives Using the Fujisaki-Okamoto Transformation and Its Application on FrodoKEM". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 359–386. DOI: 10.1007/978-3-030-56880-1_13.

[GKV10] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. "A Group Signature Scheme from Lattice Assumptions". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Berlin, Heidelberg, Dec. 2010, pp. 395–412. DOI: 10.1007/978-3-642-17373-8_23.

[GLX24] Jiangxia Ge, Heming Liao, and Rui Xue. *Measure-Rewind-Extract: Tighter Proofs of One-Way to Hiding and CCA Security in the Quantum Random Oracle Model*. Cryptology ePrint Archive, Report 2024/777. 2024. URL: https://eprint.iacr.org/2024/777.

[GMP22] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. "Anonymous, Robust Post-quantum Public Key Encryption". In: *EUROCRYPT 2022, Part III*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. LNCS. Springer, Cham, 2022, pp. 402–432. DOI: 10.1007/978-3-031-07082-2_15.

[GN08] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Berlin, Heidelberg, Apr. 2008, pp. 31–51. DOI: 10.1007/978-3-540-78967-3_3.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. "Lattice Enumeration Using Extreme Pruning". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Berlin, Heidelberg, 2010, pp. 257–278. DOI: 10.1007/978-3-642-13190-5_13.

[Goo] Google Inc. *BoringSSL*. URL: https://boringssl.googlesource.com/boringssl/.

[Gro25] NIS Cooperation Group. *Recommendation on a Coordinated Implementation Roadmap for the transition to Post-Quantum Cryptography*. June 11, 2025. URL: https://digital-strategy.ec.europa.eu/en/library/recommendation-coordinated-implementation-roadmap-transition-post-quantum-cryptography.

[GW22] Ruben Gonzalez and Thom Wiggers. "KEMTLS vs. Post-quantum TLS: Performance on Embedded Systems". In: *Security, Privacy, and Applied Cryptography Engineering*. Ed. by Lejla Batina, Stjepan Picek, and Mainack Mondal. Cham: Springer Nature Switzerland, 2022, pp. 99–117. ISBN: 978-3-031-22829-2. DOI: 10.1007/978-3-031-22829-2. URL: https://thomwiggers.nl/publication/kemtls-embedded/.

[Hei21] Max Heiser. *Improved Quantum Hypercone Locality Sensitive Filtering in Lattice Sieving*. Cryptology ePrint Archive, Report 2021/1295. 2021. URL: https://eprint.iacr.org/2021/1295.

[Hei+22] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Amber Sprenkels. *First-Order Masked Kyber on ARM Cortex-M4*. Cryptology ePrint Archive, Report 2022/058. 2022. URL: https://eprint.iacr.org/2022/058.

[Her+23] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. "Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors". In: *IACR TCHES* 2023.4 (2023), pp. 287–317. DOI: 10.46586/tches.v2023.i4.287-317.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: *TCC 2017, Part I*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10677. LNCS. Springer, Cham, Nov. 2017, pp. 341–371. DOI: 10.1007/978-3-319-70500-2_12.

[HHM22] Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. "Failing Gracefully: Decryption Failures and the Fujisaki-Okamoto Transform". In: *ASIACRYPT 2022, Part IV*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13794. LNCS. Springer, Cham, Dec. 2022, pp. 414–443. DOI: 10.1007/978-3-031-22972-5_15.

[Hir+09] Philip S. Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. "Choosing NTRUEncrypt Parameters in Light of Combined Lattice Reduction and MITM Approaches". In: *ACNS 2009*. Ed. by Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud. Vol. 5536. LNCS. Springer, Berlin, Heidelberg, June 2009, pp. 437–455. DOI: 10.1007/978-3-642-01957-9_27.

[HK25] Kathrin Hövelmanns and Mikhail A. Kudinov. "Treating Dishonest Ciphertexts in Post-quantum KEMs - Explicit vs. Implicit Rejection in the FO Transform". In: *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part II*. Ed. by Ruben Niederhagen and Markku-Juhani O. Saarinen. Springer, Cham, Apr. 2025, pp. 325–350. DOI: 10.1007/978-3-031-86602-9_12.

[HKL18] Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven. "Speed-Ups and Time-Memory Trade-Offs for Tuple Lattice Sieving". In: *PKC 2018, Part I*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10769. LNCS. Springer, Cham, Mar. 2018, pp. 407–436. DOI: 10.1007/978-3-319-76578-5_14.

[HM24] Kathrin Hövelmanns and Christian Majenz. "A Note on Failing Gracefully: Completing the Picture for Explicitly Rejecting Fujisaki-Okamoto Transforms Using Worst-Case Correctness". In: *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*. Ed. by Markku-Juhani Saarinen and Daniel Smith-Tone. Springer, Cham, June 2024, pp. 245–265. DOI: 10.1007/978-3-031-62746-0_11.

[Hof02] Paul E. Hoffman. *SMTP Service Extension for Secure SMTP over Transport Layer Security*. RFC 3207. Feb. 2002. DOI: 10.17487/RFC3207.

[Höv+20] Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. "Generic Authenticated Key Exchange in the Quantum Random Oracle Model". In: *PKC 2020, Part II*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12111. LNCS. Springer, Cham, May 2020, pp. 389–422. DOI: 10.1007/978-3-030-45388-6_14.

[How07] Nick Howgrave-Graham. "A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU". In: *CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. LNCS. Springer, Berlin, Heidelberg, Aug. 2007, pp. 150–169. DOI: 10.1007/978-3-540-74143-5_9.

[HS07] Guillaume Hanrot and Damien Stehlé. "Improved Analysis of Kannan's Shortest Lattice Vector Algorithm". In: *CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. LNCS. Springer, Berlin, Heidelberg, Aug. 2007, pp. 170–186. DOI: 10.1007/978-3-540-74143-5_10.

[HS08] Guillaume Hanrot and Damien Stehlé. "Worst-case Hermite-Korkine-Zolotarev reduced lattice bases". In: *arXiv preprint arXiv:0801.3331* (2008).

[HS10] Guillaume Hanrot and Damien Stehlé. "A complete worst-case analysis of Kannan' s shortest lattice vector algorithm". In: *Manuscript* (2010). Full version of [HS07; HS08], pp. 1–34.

[Hua+20] Yiming Huang, Miaoqing Huang, Zhongkui Lei, and Jiaxuan Wu. "A Pure Hardware Implementation of CRYSTALS-KYBER PQC Algorithm through Resource Reuse". In: *IEICE Electronics Express* advpub (2020), p. 17.20200234. DOI: 10.1587/elex.17.20200234.

[Hül+17] Andreas Hülsing, Joost Rijneveld, John M. Schanck, and Peter Schwabe. "High-Speed Key Encapsulation from NTRU". In: *CHES 2017*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. LNCS. Springer, Cham, Sept. 2017, pp. 232–252. DOI: 10.1007/978-3-319-66787-4_12.

[HW20] Jonathan Hoyland and Christopher Wood. *TLS 1.3 Extended Key Schedule*. Internet-Draft draft-jhoyla-tls-extended-key-schedule-03. Work in Progress. Internet Engineering Task Force, Dec. 2020. 1–7. URL: https://datatracker.ietf.org/doc/html/draft-jhoyla-tls-extended-key-schedule-03.

[Hö20] Kathrin Hövelmanns. "Generic constructions of quantum-resistant cryptosystems". PhD Thesis. PhD thesis. Bochum: Ruhr-Universität Bochum, 2020. URL: https://research.tue.nl/en/publications/generic-constructions-of-quantum-resistant-cryptosystems.

[ISB25] German Federal Office for Information Security (BSI). *Cryptographic Mechanisms: Recommendations and Key Lengths*. Tech. rep. BSI TR-02102-1. BSI, Jan. 2025. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.html.

[Jao+22] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. *SIKE*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions. National Institute of Standards and Technology, 2022.

[Jen+23] Sönke Jendral, Kalle Ngo, Ruize Wang, and Elena Dubrova. *A Single-Trace Message Recovery Attack on a Masked and Shuffled Implementation of CRYSTALS-Kyber*. Cryptology ePrint Archive, Report 2023/1587. 2023. URL: https://eprint.iacr.org/2023/1587.

[Jia+18] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. "IND-CCA-Secure Key Encapsulation Mechanism in the Quantum Random Oracle Model, Revisited". In: *CRYPTO 2018, Part III*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10993. LNCS. Springer, Cham, Aug. 2018, pp. 96–125. DOI: 10.1007/978-3-319-96878-0_4.

[JZM19a] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. "Key Encapsulation Mechanism with Explicit Rejection in the Quantum Random Oracle Model". In: *PKC 2019, Part II*. Ed. by Dongdai Lin and Kazue Sako. Vol. 11443. LNCS. Springer, Cham, Apr. 2019, pp. 618–645. DOI: 10.1007/978-3-030-17259-6_21.

[JZM19b] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. "Tighter Security Proofs for Generic Key Encapsulation Mechanism in the Quantum Random Oracle Model". In: *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*. Ed. by Jintai Ding and Rainer Steinwandt. Springer, Cham, 2019, pp. 227–248. DOI: 10.1007/978-3-030-25510-7_13.

[Kam+22] Tendayi Kamucheka, Alexander Nelson, David Andrews, and Miaoqing Huang. "A Masked Pure-Hardware Implementation of Kyber Cryptographic Algorithm". In: *2022 International Conference on Field-Programmable Technology (ICFPT)*. 2022, pp. 1–1. DOI: 10.1109/ICFPT56656.2022.9974404.

[Kan+] Matthias J. Kannwischer, Richard Petri, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. *pqm4: Post-quantum crypto library for the ARM Cortex-M4*. URL: https://github.com/mupq/pqm4.

[Kan83] Ravi Kannan. "Improved Algorithms for Integer Programming and Related Lattice Problems". In: *15th ACM STOC*. ACM Press, Apr. 1983, pp. 193–206. DOI: 10.1145/800061.808749.

[Kar+25a] Alexander Karenin, Elena Kirshanova, Alexander May, and Julian Nowakowski. "Fast Slicer for Batch-CVP: Making Lattice Hybrid Attacks Practical". In: *Cryptology ePrint Archive* (2025).

[Kar+25b] Alexandr Karenin, Elena Kirshanova, Julian Nowakowski, Eamonn W Postlethwaite, and Fernando Virdia. "Cool+Cruel=Dual". In: *Cryptology ePrint Archive* (2025).

[KEF20] Paul Kirchner, Thomas Espitau, and Pierre-Alain Fouque. "Fast Reduction of Algebraic Lattices over Cyclotomic Fields". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 155–185. DOI: 10.1007/978-3-030-56880-1_6.

[KF15] Paul Kirchner and Pierre-Alain Fouque. "An Improved BKW Algorithm for LWE with Applications to Cryptography and Lattices". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Berlin, Heidelberg, Aug. 2015, pp. 43–62. DOI: 10.1007/978-3-662-47989-6_3.

[KF17] Paul Kirchner and Pierre-Alain Fouque. "Revisiting Lattice Attacks on Overstretched NTRU Parameters". In: *EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. LNCS. Springer, Cham, 2017, pp. 3–26. DOI: 10.1007/978-3-319-56620-7_1.

[KK18] Franziskus Kiefer and Krzysztof Kwiatkowski. *Hybrid ECDHE-SIDH Key Exchange for TLS*. Internet-Draft draft-kiefer-tls-ecdhe-sidh-00. Work in Progress. Internet Engineering Task Force, Nov. 2018. 13 pp. URL: https://datatracker.ietf.org/doc/html/draft-kiefer-tls-ecdhe-sidh-00.

[KL21] Elena Kirshanova and Thijs Laarhoven. "Lower Bounds on Lattice Sieving and Information Set Decoding". In: *CRYPTO 2021, Part II*. Ed. by Tal Malkin and Chris Peikert. Vol. 12826. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 791–820. DOI: 10.1007/978-3-030-84245-1_27.

[Kre24] Katharina Kreuzer. "Verification of Correctness and Security Properties for CRYSTALS-KYBER". In: *CSF 2024 Computer Security Foundations Symposium*. IEEE Computer Society Press, July 2024, pp. 511–526. DOI: 10.1109/CSF61375.2024.00016.

[KS23] Ehren Kret and Rolfe Schmidt. *The PQXDH Key Agreement Protocol*. Protocol documentation. Oct. 18, 2023. URL: https://signal.org/docs/specifications/pqxdh/.

[Kuc+20] Veronika Kuchta, Amin Sakzad, Damien Stehlé, Ron Steinfeld, and Shifeng Sun. "Measure-Rewind-Measure: Tighter Quantum Random Oracle Model Proofs for One-Way to Hiding and CCA Security". In: *EUROCRYPT 2020, Part III*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12107. LNCS. Springer, Cham, May 2020, pp. 703–728. DOI: 10.1007/978-3-030-45727-3_24.

[KV19] Kris Kwiatkowski and Luke Valenta. *The TLS Post-Quantum Experiment*. Post on the Cloudflare blog. Cloudflare, 2019. URL: https://blog.cloudflare.com/the-tls-post-quantum-experiment/ (visited on 12/22/2022).

[Kwi+19] Krzysztof Kwiatkowski, Nick Sullivan, Adam Langley, Dave Levin, and Alan Mislove. *Measuring TLS key exchange with post-quantum KEM*. Workshop Record of the Second PQC Standardization Conference. 2019. URL: https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/kwiatkowski-measuring-tls.pdf.

[Kwi+25] Kris Kwiatkowski, Panos Kampanakis, Bas Westerbaan, and Douglas Stebila. *Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3*. Internet-Draft draft-ietf-tls-ecdhe-mlkem-01. Work in Progress. Internet Engineering Task Force, Sept. 2025. 10 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-tls-ecdhe-mlkem/01/.

[Laa15] Thijs Laarhoven. "Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Berlin, Heidelberg, Aug. 2015, pp. 3–22. DOI: 10.1007/978-3-662-47989-6_1.

[Laa16] Thijs Laarhoven. "Search problems in cryptography: from fingerprinting to lattice sieving". English. PhD Thesis. PhD thesis. Mathematics and Computer Science, Feb. 2016. ISBN: 978-90-386-4021-1.

[Lan16] Adam Langley. *CECPQ1 results*. Blog post. 2016. URL: https://www.imperialviolet.org/2016/11/28/cecpq1.html.

[Lan18] Adam Langley. *CECPQ2*. Blog post. 2018. URL: https://www.imperialviolet.org/2018/12/12/cecpq2.html.

[Lee+19] Changmin Lee, Alice Pellet-Mary, Damien Stehlé, and Alexandre Wallet. "An LLL Algorithm for Module Lattices". In: *ASIACRYPT 2019, Part II*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11922. LNCS. Springer, Cham, Dec. 2019, pp. 59–90. DOI: 10.1007/978-3-030-34621-8_3.

[Li+21] Shuaigang Li, Xianhui Lu, Jiang Zhang, Bao Li, and Lei Bi. "Predicting the Concrete Security of LWE Against the Dual Attack Using Binary Search". In: *ICICS 21, Part I*. Ed. by Debin Gao, Qi Li, Xiaohong Guan, and Xiaofeng Liao. Vol. 12919. LNCS. Springer, Cham, Nov. 2021, pp. 265–282. DOI: 10.1007/978-3-030-88052-1_16.

[LM18] Thijs Laarhoven and Artur Mariano. "Progressive Lattice Sieving". In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*. Ed. by Tanja Lange and Rainer Steinwandt. Springer, Cham, 2018, pp. 292–311. DOI: 10.1007/978-3-319-79063-3_14.

[LMP15] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. "Finding shortest lattice vectors faster using quantum search". In: *DCC* 77.2-3 (2015), pp. 375–400. DOI: 10.1007/s10623-015-0067-5.

[LN25] Jianwei Li and Phong Q Nguyen. "A complete analysis of the BKZ lattice reduction algorithm". In: *Journal of Cryptology* 38.1 (2025), p. 12. DOI: 10.1007/s00145-024-09527-0.

[LP11] Richard Lindner and Chris Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". In: *CT-RSA 2011*. Ed. by Aggelos Kiayias. Vol. 6558. LNCS. Springer, Berlin, Heidelberg, Feb. 2011, pp. 319–339. DOI: 10.1007/978-3-642-19074-2_21.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Berlin, Heidelberg, 2010, pp. 1–23. DOI: 10.1007/978-3-642-13190-5_1.

[LS15] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices". In: *DCC* 75.3 (2015), pp. 565–599. DOI: 10.1007/s10623-014-9938-4.

[MAT22] MATZOV. *Report on the Security of LWE: Improved Dual Lattice Attack*. Tech. rep. 2022. DOI: https://doi.org/10.5281/zenodo.6493704.

[Mat25] John Preuss Mattsson. *PQC Dialogue with Government Stakeholders*. Transcript of an IETF side-meeting posted on a mailing list. May 13, 2025. URL: https://mailarchive.ietf.org/arch/msg/pqc/14f3hjUlpiwSbusornAjRN98QLc/.

[Mic] Microsoft. *Schannel (Security Support Provider)*. URL: https://learn.microsoft.com/en-us/windows/win32/secauthn/schannel.

[Mic18] Daniele Micciancio. "On the Hardness of Learning With Errors with Binary Secrets". In: *Theory of Computing* 14.13 (2018), pp. 1–17. DOI: 10.4086/toc.2018.v014a013.

[Mic25] Microsoft. *Microsoft Edge browser policy PostQuantumKeyAgreementEnabled*. Sept. 18, 2025. URL: https://learn.microsoft.com/en-us/deployedge/microsoft-edge-browser-policies/postquantumkeyagreementenabled.

[MM11] Daniele Micciancio and Petros Mol. "Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions". In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Berlin, Heidelberg, Aug. 2011, pp. 465–484. DOI: 10.1007/978-3-642-22792-9_26.

[Moh10] Payman Mohassel. "A Closer Look at Anonymity and Robustness in Encryption Schemes". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Berlin, Heidelberg, Dec. 2010, pp. 501–518. DOI: 10.1007/978-3-642-17373-8_29.

[Moo+24] Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. *Transition to Post-Quantum Cryptography Standards*. NIST Interagency Report 8547. Initial Public Draft. National Institute of Standards and Technology, Nov. 2024. DOI: 10.6028/NIST.IR.8547.ipd. URL: https://csrc.nist.gov/pubs/ir/8547/ipd.

[Moz] Mozilla Foundation. *Network Security Services (NSS)*. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS.

[MP12] Daniele Micciancio and Chris Peikert. "Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Berlin, Heidelberg, Apr. 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41.

[MP13] Daniele Micciancio and Chris Peikert. "Hardness of SIS and LWE with Small Parameters". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Berlin, Heidelberg, Aug. 2013, pp. 21–39. DOI: 10.1007/978-3-642-40041-4_2.

[MR09] Daniele Micciancio and Oded Regev. "Lattice-based Cryptography". In: *Post-Quantum Cryptography*. Ed. by Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 147–191. DOI: 10.1007/978-3-540-88702-7_5.

[MS20] Tamalika Mukherjee and Noah Stephens-Davidowitz. "Lattice Reduction for Modules, or How to Reduce ModuleSVP to ModuleSVP". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 213–242. DOI: 10.1007/978-3-030-56880-1_8.

[Muj+24] Catinca Mujdei, Lennert Wouters, Angshuman Karmakar, Arthur Beckers, Jose Maria Bermudo Mera, and Ingrid Verbauwhede. "Side-channel Analysis of Lattice-based Post-quantum Cryptography: Exploiting Polynomial Multiplication". In: *ACM Transactions on Embedded Computing Syststems* 23.2 (Mar. 2024). DOI: 10.1145/3569420. URL: https://doi.org/10.1145/3569420.

[MX23] Varun Maram and Keita Xagawa. "Post-quantum Anonymity of Kyber". In: *PKC 2023, Part I*. Ed. by Alexandra Boldyreva and Vladimir Kolesnikov. Vol. 13940. LNCS. Springer, Cham, May 2023, pp. 3–35. DOI: 10.1007/978-3-031-31368-4_1.

[Nat16] National Institute of Standards and Technology. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. Dec. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf (visited on 05/09/2023).

[Nat24a] National Cyber Security Centre. *Next steps in preparing for post-quantum cryptography. How system owners can begin planning for the migration to post-quantum cryptography (PQC)*. Originally published November 2023; updated August 2024. National Cyber Security Centre (NCSC). Aug. 14, 2024. URL: https://www.ncsc.gov.uk/whitepaper/next-steps-preparing-for-post-quantum-cryptography (visited on 01/13/2026).

[Nat24b] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Federal Information Processing Standards Publication (FIPS) FIPS 203. NIST has assigned NIST FIPS 203 as the publication identifier for this FIPS. Department of Commerce, Washington, D.C., 2024.

[Nat24c] National Security Agency. *The Commercial National Security Algorithm Suite 2.0 and Quantum Computing FAQ*. Dec. 2024. URL: https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF.

[New99] Chris Newman. *Using TLS with IMAP, POP3 and ACAP*. RFC 2595. June 1999. DOI: 10.17487/RFC2595. URL: https://www.rfc-editor.org/info/rfc2595.

[Ngu10] Phong Q. Nguyen. "Hermite's Constant and Lattice Algorithms". In: ed. by Phong Q. Nguyen and Brigitte Vallée. ISC. Springer, 2010, pp. 19–69. ISBN: 978-3-642-02294-4. DOI: 10.1007/978-3-642-02295-1.

[NV08] Phong Q Nguyen and Thomas Vidick. "Sieve algorithms for the shortest vector problem are practical". In: *Journal of Mathematical Cryptology* 2.2 (2008), pp. 181–207.

[O'B23] Devon O'Brien. *Protecting Chrome Traffic with Hybrid Kyber KEM*. Aug. 10, 2023. URL: https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html.

[Ode+18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. "Practical CCA2-Secure Masked Ring-LWE Implementations". In: *IACR TCHES* 2018.1 (2018), pp. 142–174. ISSN: 2569-2925. DOI: 10.13154/tches.v2018.i1.142-174. URL: https://tches.iacr.org/index.php/TCHES/article/view/836.

[Opea] *OpenSSL: The Open Source toolkit for SSL/TLS*. OpenSSL project. URL: https://www.openssl.org/ (visited on 05/09/2023).

[Opeb] *OpenVPN Protocol*. URL: https://openvpn.net/community-resources/openvpn-protocol/ (visited on 11/10/2025).

[Ope25] Open Quantum Safe. *liboqs: An open-source C library for quantum-safe cryptography*. Version 0.14.0. July 11, 2025. URL: https://github.com/open-quantum-safe/liboqs.

[PBY17] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. "To BLISS-B or not to be: Attacking strongSwan's Implementation of Post-Quantum Signatures". In: *ACM CCS 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 1843–1855. DOI: 10.1145/3133956.3134023.

[Pei09] Chris Peikert. "Public-key cryptosystems from the worst-case shortest vector problem: extended abstract". In: *41st ACM STOC*. Ed. by Michael Mitzenmacher. ACM Press, 2009, pp. 333–342. DOI: 10.1145/1536414.1536461.

[PG14] Thomas Pöppelmann and Tim Güneysu. "Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware". In: *SAC 2013*. Ed. by Tanja Lange, Kristin Lauter, and Petr Lisonek. Vol. 8282. LNCS. Springer, Berlin, Heidelberg, Aug. 2014, pp. 68–85. DOI: 10.1007/978-3-662-43414-7_4.

[PHS19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. "Approx-SVP in Ideal Lattices with Pre-processing". In: *EUROCRYPT 2019, Part II*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11477. LNCS. Springer, Cham, May 2019, pp. 685–716. DOI: 10.1007/978-3-030-17656-3_24.

[PS24] Amaury Pouly and Yixin Shen. "Provable Dual Attacks on Learning with Errors". In: *EUROCRYPT 2024, Part VII*. Ed. by Marc Joye and Gregor Leander. Vol. 14657. LNCS. Springer, Cham, May 2024, pp. 256–285. DOI: 10.1007/978-3-031-58754-2_10.

[PST20] Christian Paquin, Douglas Stebila, and Goutam Tamvada. "Benchmarking Post-quantum Cryptography in TLS". In: *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*. Ed. by Jintai Ding and Jean-Pierre Tillich. Springer, Cham, 2020, pp. 72–91. DOI: 10.1007/978-3-030-44223-1_5.

[PV21] Eamonn W. Postlethwaite and Fernando Virdia. "On the Success Probability of Solving Unique SVP via BKZ". In: *PKC 2021, Part I*. Ed. by Juan Garay. Vol. 12710. LNCS. Springer, Cham, May 2021, pp. 68–98. DOI: 10.1007/978-3-030-75245-3_4.

[PV25] Ludo N Pulles and Paul Vié. "Accelerating the Primal Hybrid Attack against Sparse LWE using GPUs". In: *Cryptology ePrint Archive* (2025).

[Raj+23] Gokulnath Rajendran, Prasanna Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. "Pushing the Limits of Generic Side-Channel Attacks on LWE-based KEMs - Parallel PC Oracle Attacks on Kyber KEM and Beyond". In: *IACR TCHES* 2023.2 (2023), pp. 418–446. DOI: 10.46586/tches.v2023.i2.418-446.

[Rav+20a] Prasanna Ravi, Romain Poussier, Shivam Bhasin, and Anupam Chattopadhyay. "On Configurable SCA Countermeasures Against Single Trace Attacks for the NTT: A Performance Evaluation Study over Kyber and Dilithium on the ARM Cortex-M4". In: *Security, Privacy, and Applied Cryptography Engineering: 10th International Conference, SPACE 2020, Kolkata, India, December 17–21, 2020, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2020, pp. 123–146. DOI: 10.1007/978-3-030-66626-2_7.

[Rav+20b] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. "Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs". In: *IACR TCHES* 2020.3 (2020), pp. 307–335. ISSN: 2569-2925. DOI: 10.13154/tches.v2020.i3.307-335. URL: https://tches.iacr.org/index.php/TCHES/article/view/8592.

[Rav+22] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. "On Exploiting Message Leakage in (Few) NIST PQC Candidates for Practical Message Recovery Attacks". In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 684–699. DOI: 10.1109/TIFS.2021.3139268.

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[Rep+15] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. "A Masked Ring-LWE Implementation". In: *CHES 2015*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. LNCS. Springer, Berlin, Heidelberg, Sept. 2015, pp. 683–702. DOI: 10.1007/978-3-662-48324-4_34.

[Res00] Eric Rescorla. *HTTP Over TLS*. RFC 2818. May 2000. DOI: 10.17487/RFC2818.

[Res18] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Aug. 2018. DOI: 10.17487/RFC8446.

[Saa18] Markku-Juhani O. Saarinen. "Arithmetic coding and blinding countermeasures for lattice signatures - Engineering a side-channel resistant post-quantum signature scheme with compact signatures". In: *Journal of Cryptographic Engineering* 8.1 (Apr. 2018), pp. 71–84. DOI: 10.1007/s13389-017-0149-6.

[Sak00] Kazue Sako. "An Auction Protocol Which Hides Bids of Losers". In: *PKC 2000*. Ed. by Hideki Imai and Yuliang Zheng. Vol. 1751. LNCS. Springer, Berlin, Heidelberg, Jan. 2000, pp. 422–432. DOI: 10.1007/978-3-540-46588-1_28.

[Sch+17] John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. *NTRU-HRSS-KEM*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. National Institute of Standards and Technology, 2017.

[Sch22a] Peter Schwabe. *CRYSTALS-Kyber Update*. Talk at the Fourth NIST PQC Standardization Conference. Slides: https://csrc.nist.gov/csrc/media/Presentations/2022/crystals-kyber-update/images-media/session-1-schwabe-crystals-kyber-pqc2022.pdf. Nov. 2022. URL: https://csrc.nist.gov/Presentations/2022/crystals-kyber-update.

[Sch22b] Peter Schwabe. *Kyber decisions, part 2: FO transform*. Message to the `pqc-forum@list.nist.gov` mailing list. Explains hashing of (hash of) the public key into coins/shared key and its robustness rationale. Dec. 2022. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/C0D3W1KoINY.

[Sch24] Sophie Schmieg. *Unbindable Kemmy Schmidt: ML-KEM is neither MAL-BIND-K-CT nor MAL-BIND-K-PK*. Cryptology ePrint Archive, Report 2024/523. 2024. URL: https://eprint.iacr.org/2024/523.

[Sch87] Claus-Peter Schnorr. "A hierarchy of polynomial time lattice basis reduction algorithms". In: *Theoretical computer science* 53.2-3 (1987), pp. 201–224. DOI: https://doi.org/10.1016/0304-3975(87)90064-8.

[SFG25] Douglas Stebila, Scott Fluhrer, and Shay Gueron. *Hybrid key exchange in TLS 1.3*. Internet-Draft draft-ietf-tls-hybrid-design-16. Work in Progress. Internet Engineering Task Force, Sept. 2025. 23 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/16/.

[SH95] Claus-Peter Schnorr and Horst Helmut Hörner. "Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction". In: *EUROCRYPT'95*. Ed. by Louis C. Guillou and Jean-Jacques Quisquater. Vol. 921. LNCS. Springer, Berlin, Heidelberg, May 1995, pp. 1–12. DOI: 10.1007/3-540-49264-X_1.

[She+23] Muyan Shen, Chi Cheng, Xiaohan Zhang, Qian Guo, and Tao Jiang. "Find the Bad Apples: An efficient method for perfect key recovery under imperfect SCA oracles - A case study of Kyber". In: *IACR TCHES* 2023.1 (2023), pp. 89–112. DOI: 10.46586/tches.v2023.i1.89-112.

[SM16] Douglas Stebila and Michele Mosca. "Post-quantum Key Exchange for the Internet and the Open Quantum Safe Project". In: *SAC 2016*. Ed. by Roberto Avanzi and Howard M. Heys. Vol. 10532. LNCS. Springer, Cham, Aug. 2016, pp. 14–37. DOI: 10.1007/978-3-319-69453-5_2.

[SS17] John M. Schanck and Douglas Stebila. *A Transport Layer Security (TLS) Extension For Establishing An Additional Shared Secret*. Internet-Draft draft-schanck-tls-additional-keyshare-00. Work in Progress. Internet Engineering Task Force, Apr. 2017. 1–10. URL: https://datatracker.ietf.org/doc/html/draft-schanck-tls-additional-keyshare-00.

[SSW20] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Post-Quantum TLS Without Handshake Signatures". In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1461–1480. DOI: 10.1145/3372297.3423350.

[SSW21] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "More Efficient Post-quantum KEMTLS with Pre-distributed Public Keys". In: *ESORICS 2021, Part I*. Ed. by Elisa Bertino, Haya Shulman, and Michael Waidner. Vol. 12972. LNCS. Springer, Cham, Oct. 2021, pp. 3–22. DOI: 10.1007/978-3-030-88418-5_1.

[Ste24] Matthias Johann Steiner. "The Complexity of Algebraic Algorithms for LWE". In: *EUROCRYPT 2024, Part III*. Ed. by Marc Joye and Gregor Leander. Vol. 14653. LNCS. Springer, Cham, May 2024, pp. 375–403. DOI: 10.1007/978-3-031-58734-4_13.

[Sul17] Nick Sullivan. *Why TLS 1.3 isn't in browsers yet*. Cloudflare Blog. Dec. 26, 2017. URL: https://blog.cloudflare.com/why-tls-1-3-isnt-in-browsers-yet/.

[SWZ17] John M. Schanck, William Whyte, and Zhenfei Zhang. *Quantum-Safe Hybrid (QSH) Ciphersuite for Transport Layer Security (TLS) version 1.2*. Internet-Draft draft-whyte-qsh-tls12-02. Work in Progress. Internet Engineering Task Force, Jan. 2017. 1–19. URL: https://datatracker.ietf.org/doc/html/draft-whyte-qsh-tls12-02.

[SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. "Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model". In: *EUROCRYPT 2018, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. LNCS. Springer, Cham, 2018, pp. 520–551. DOI: 10.1007/978-3-319-78372-7_17.

[Tas+22] George Tasopoulos, Jinhui Li, Apostolos P. Fournaris, Raymond K. Zhao, Amin Sakzad, and Ron Steinfeld. "Performance Evaluation of Post-Quantum TLS 1.3 on Resource-Constrained Embedded Systems". In: *Information Security Practice and Experience: 17th International Conference, ISPEC 2022, Taipei, Taiwan, November 23–25, 2022, Proceedings*. Taipei, Taiwan: Springer-Verlag, 2022, pp. 432–451. ISBN: 978-3-031-21279-6. DOI: 10.1007/978-3-031-21280-2_24. URL: https://doi.org/10.1007/978-3-031-21280-2_24.

[Thea] The Go Authors. *crypto/tls package*. URL: https://pkg.go.dev/crypto/tls.

[Theb] The Legion of the Bouncy Castle. *Bouncy Castle Crypto APIs*. URL: https://www.bouncycastle.org/.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. "Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms". In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Berlin, Heidelberg, 2016, pp. 192–216. DOI: 10.1007/978-3-662-53644-5_8.

[Uen+22] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. "Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs". In: *IACR TCHES* 2022.1 (2022), pp. 296–322. DOI: 10.46586/tches.v2022.i1.296-322.

[Why+17] William Whyte, Zhenfei Zhang, Scott Fluhrer, and Oscar Garcia-Morchon. *Quantum-Safe Hybrid (QSH) Key Exchange for Transport Layer Security (TLS) version 1.3*. Internet-Draft draft-whyte-qsh-tls13-06. Work in Progress. Internet Engineering Task Force, Oct. 2017. 19 pp. URL: https://datatracker.ietf.org/doc/html/draft-whyte-qsh-tls13-06.

[Wig24] Thom Wiggers. "Post-Quantum TLS". PhD thesis. Nijmegen, The Netherlands: Radboud University, Jan. 9, 2024. URL: https://thomwiggers.nl/publication/thesis/.

[Xag22] Keita Xagawa. "Anonymity of NIST PQC Round 3 KEMs". In: *EUROCRYPT 2022, Part III*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. LNCS. Springer, Cham, 2022, pp. 551–581. DOI: 10.1007/978-3-031-07082-2_20.

[Xia+22] Wenwen Xia, Leizhang Wang, GengWang, Dawu Gu, and Baocang Wang. *Improved Progressive BKZ with Lattice Sieving*. Cryptology ePrint Archive, Report 2022/1343. 2022. URL: https://eprint.iacr.org/2022/1343.

[Xia+24] Wenwen Xia, Leizhang Wang, Geng Wang, Dawu Gu, and Baocang Wang. "A Refined Hardness Estimation of LWE in Two-Step Mode". In: *PKC 2024, Part II*. Ed. by Qiang Tang and Vanessa Teague. Vol. 14603. LNCS. Springer, Cham, Apr. 2024, pp. 3–35. DOI: 10.1007/978-3-031-57725-3_1.

[XL21] Yufei Xing and Shuguo Li. "A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA". In: *IACR TCHES* 2021.2 (2021), pp. 328–356. ISSN: 2569-2925. DOI: 10.46586/tches.v2021.i2.328-356. URL: https://tches.iacr.org/index.php/TCHES/article/view/8797.

[Xu+22] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. "Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems With Chosen Ciphertexts: The Case Study of Kyber". In: *IEEE Transactions on Computers* 71.9 (2022), pp. 2163–2176. DOI: 10.1109/TC.2021.3122997.

[XWT25] Dejun Xu, Kai Wang, and Jing Tian. "A Hardware-Friendly Shuffling Countermeasure Against Side-Channel Attacks for Kyber". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 72.3 (2025), pp. 504–508. DOI: 10.1109/TCSII.2025.3528751.

[YD17] Yang Yu and Léo Ducas. "Second Order Statistical Behavior of LLL and BKZ". In: *SAC 2017*. Ed. by Carlisle Adams and Jan Camenisch. Vol. 10719. LNCS. Springer, Cham, Aug. 2017, pp. 3–22. DOI: 10.1007/978-3-319-72565-9_1.

[YZ21] Takashi Yamakawa and Mark Zhandry. "Classical vs Quantum Random Oracles". In: *EUROCRYPT 2021, Part II*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. LNCS. Springer, Cham, Oct. 2021, pp. 568–597. DOI: 10.1007/978-3-030-77886-6_20.

[ZDY25] Ziyu Zhao, Jintai Ding, and Bo-Yin Yang. "Sieving with Streaming Memory Access". In: *IACR TCHES* 2025.2 (2025), pp. 362–384. DOI: 10.46586/tches.v2025.i2.362-384.

[Zha19] Mark Zhandry. "How to Record Quantum Queries, and Applications to Quantum Indifferentiability". In: *CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. LNCS. Springer, Cham, Aug. 2019, pp. 239–268. DOI: 10.1007/978-3-030-26951-7_9.

[ZLJ25] Biming Zhou, Yiting Liu, and Haodong Jiang. "SoK: Post-Quantum Key Encapsulation Mechanisms—Security Definitions and Proof Techniques". In: *Security Standardisation Research*. Springer Nature Switzerland, 2025. DOI: 10.1007/978-3-031-87541-0_6.